Stable Dinner Party Seating Arrangements

Damien Berriaud, Andrei Constantinescu and Roger Wattenhofer

Abstract

A group of n agents with numerical preferences for each other are to be assigned to the n seats of a dining table. We study two natural topologies: circular (cycle) tables and panel (path) tables. For a given seating arrangement, an agent's utility is the sum of its preference values towards its (at most two) direct neighbors. An arrangement is envy-free if no agent strictly prefers someone else's seat, and it is stable if no two agents strictly prefer each other's seats. We show that it is NP-complete to decide whether an envy-free arrangement exists for both paths and cycles, even with binary preferences. In contrast, under the assumption that agents come from a bounded number of classes, for both topologies, we present polynomial-time algorithms computing envy-free and stable arrangements, working even for general preferences. Proving the hardness of computing stable arrangements seems more difficult, as even constructing unstable instances can be challenging. To this end, we propose a characterization of the existence of stable arrangements based on the number of distinct values in the preference matrix and the number of classes of agents. For two classes of agents, we show that stability can always be ensured, for both paths and cycles. For cycles, we moreover show that binary preferences with four classes of agents, as well as three-valued preferences with three classes of agents, are sufficient to prevent the existence of a stable arrangement. For paths, the latter still holds, while we argue that a path-stable arrangement always exists in the binary case under the additional constraint that agents can only swap seats when sitting at most two positions away. We moreover consider the swap dynamics and exhibit instances where they do not converge, despite a stable arrangement existing.

1 Introduction

Your festive dinner table is ready, and the guests are arriving. As soon as your guests take their assigned seats, two of them are unhappy about their neighbors and rather want to switch seats. Alas, right after the switch, two other guests become upset, and then pandemonium ensues! Could you have prevented all the social awkwardness by seating your guests "correctly" from the get-go?

In this paper, we study the difficulty of finding a *stable* seating arrangement; i.e. one where no two guests would switch seats. We focus on two natural seating situations: a round table (cycle), and an expert panel (path). In either case, we assume guests only care about having the best possible set of direct left and right neighbors. In certain cases, not even a stable arrangement might make the cut, as a single guest envying the seat of another could potentially lead to trouble. Therefore, we are also interested in finding *envy-free* arrangements.

Formally, n guests have to be assigned bijectively to the n seats of a dining table: either a path or a cycle graph. Guests express their preferences for the other guests numerically, with higher numbers corresponding to a greater desire to sit next to the respective guest. The utility of a guest g for a given seating arrangement is the sum of g's preference values towards g's neighbors. A guest g envies another guest if g's utility would strictly increase if they swapped places. Two guests want to swap places whenever they envy each other. Our goal is to compute a stable (no two guests want to swap) respectively envy-free (no guest envies another) seating arrangement.

Besides the table topology, we conduct our study in terms of two natural parameters. The

	No. of Classes	
	Bounded	Unbounded
EF	Poly	NP-hard
STA	Poly	?

(a) Complexity results for cycles. The hardness result is shown in Theorem 1 and the polynomial results in Theorem 5.

	No. of Classes	
	Bounded	Unbounded
EF	Poly	NP-hard
STA	Poly	?

(b) Complexity results for paths. The hardness result is shown in Theorem 2 and the polynomial results in Theorem 4.

Table 1: Summary of our computational results on computing envy-free (EF) and stable (STA) arrangements. We distinguish between two cases, depending on whether the total number of guest classes is bounded by a constant or not. Our hardness results are for the decision variants "does such an arrangement exist?" and hold even for binary preferences, while our polynomial-time algorithms recover an envy-free/stable arrangement whenever any exist, and work with no constraints on the preference values.

Values \ Classes	≤ 2	3	4	≥ 5
≤ 2	S	?	**U**	U
3	S	**U**	U	U
≥ 4	**S**	U	U	U

(a) Characterization results for cycles. The bold entries, from left to right, are proven in Theorem 6, Theorem 8 and Theorem 10.

Values \ Classes	≤ 2	3	≥ 4
≤ 2	S	?	?
3	S	**U**	U
≥ 4	**S**	U	U

(b) Characterization results for paths. The bold entries, from left to right, are proven in Theorem 7 and Theorem 9.

Table 2: Summary of our results characterizing the emergence of instability for different combinations of constraints on the number of preference values and classes of guests. Our stability (S) results mean that all instances satisfying the constraints admit a stable arrangement, and hold for arbitrary preference values. Our constructions with no stable arrangements (U) only use small non-negative values, often $0, 1, 2, \ldots$, and work for any large enough number of agents. For the remaining unknown (?) entries, we do not know whether any unstable instances exist.

first parameter is the number of numerical values guests can choose from when expressing their preferences for other guests. An example of two-valued preferences would be when all preference values are either zero or one (i.e., *binary*, also known as *approval* preferences), in which case every guest has a list of "favorite" guests they want to sit next to, and is indifferent towards the others. In contrast, if the values used are ± 1; i.e., every guest either likes or dislikes every other guest; then the preferences are still two-valued, but no longer binary. Increasing the number of allowed values allows for finer-grained preferences.

Our second parameter is the number of different guest classes. In particular, dinner party guests can often be put into certain categories, e.g., charmer, entertainer, diva, politico, introvert, outsider. Each class has its own preferences towards other classes, e.g., outsiders would prefer to sit next to a charmer, but not next to an introvert.

Our Contribution. We study the existence and computational complexity of finding stable/envy-free arrangements on paths and cycles. Some of our results are quite surprising. For instance, six people with binary preferences can always be stably seated at a round table, while for five (or seven) guests some preferences are inherently unstable, and we better invite (or uninvite) another guest. However, even for six people with binary preferences, for which a stable arrangement always exists, the swap dynamics might still never converge to one.

Our computational results are summarized in Table 1. Notably, we prove that it is

NP-hard to decide whether an envy-free arrangement exists, for both paths and cycles, even under binary preferences. For stability, on the other hand, even constructing unstable instances can be rather challenging. Hoping to gain insight into the complexity of stability, we study for which combinations of our two parameters, i.e., number of preference values and guest classes, do stable arrangements always exist and for which combinations this is not the case. Our results on this front are summarized in Table 2. For the case of paths, we do not know whether unstable two-valued preferences exist, but an exhaustive search could not produce unstable binary instances for $n \leq 7$. However, we prove that binary preferences always admit a stable arrangement on a path assuming that guests are only willing to swap places with other guests that they are separated from by at most one seat.

Appendix. In the appendix, we supply the proofs omitted from the main text, as well as supporting material referenced throughout. Moreover, we show that stability is a highly fragile notion, being non-monotonic with respect to adding/removing guests. We also give evidence that knowledge about stability on paths is unlikely to transfer to computing cycle-stable arrangements. Finally, we use probabilistic tools to study the expected number of stable arrangements of random binary preferences sampled from the Erdős-Rényi model.

Concurrent Work. In their recent work, to appear in IJCAI 2023, Ceylan, Chen and Roy [10] tackle a list of problems similar to ours, also building on the model of Bodlaender et al. [7, 6]. They prove that deciding the existence of envy-free arrangements is NP-hard under binary preferences for both paths and cycles. Their reduction is however different from ours, showing hardness even for symmetric preferences, at the expense of making the reduction more complex. Moreover, they consider the complexity of deciding the existence of stable arrangements for paths and cycles. While they do not provide an answer for the binary case, they show that it is NP-complete for cycles with 4-valued non-negative preferences and for paths with 6-valued unrestricted preferences. To the best of our knowledge, the other results from their paper do not overlap with our findings.

2 Related Work

The algorithmic study of stability in collective decision-making has its roots in the seminal paper of Gale and Shapley [12], introducing the now well-known *Stable Marriage* and *Stable Roommates* problems. Classically, the former is presented as follows: an equal number of men and women want to form couples such that no man and woman from different couples strictly prefer each other over their current partners, in which case the matching is called *stable*. The authors give the celebrated Gale-Shapley deferred acceptance algorithm showing that a stable matching always exists and can be computed in linear time. Irving [14] later extended the algorithm to also handle preferences with ties. The *Stable Roommates* problem is the non-bipartite analog of Stable Marriage: an even number of students want to allocate themselves into identical two-person rooms in a dormitory. A matching is stable if no two students allocated to different rooms prefer each other over their current roommates. In this setting, stable matchings might no longer exist, but a polynomial-time algorithm for computing one if any exist is known [13]. However, when ties are also allowed, the problem becomes NP-hard [16].

The seating arrangement problem that we study is, in fact, well-connected with Stable Roommates. Instead of one table with n seats, the latter considers $n/2$ tables with two seats each. However, there is another more subtle difference: in Stable Roommates, two people unhappy with their current roommates can choose to move into any free room. This is not possible if there are exactly $n/2$ rooms. Instead, the stability notion that we study corresponds to the distinct notion of *exchange-stability* in the Stable Roommates model, where unhappy students can agree to exchange roommates. Surprisingly, under exchange-stability, finding a stable roommate allocation becomes NP-hard even without ties [9].

One can also see our problem through the lens of coalition formation. In particular, *hedonic games* [2] consider the formation of coalitions under the assumption that individuals only care about members in their own coalition. Then, fixing the sizes of the coalitions allows one to generalize from tables of size two and study stability more generally. Bilò et al. [5] successfully employ this approach to show a number of attractive computational results concerning exchange-stability. The main drawback of this approach, however, is that it assumes that any two people sitting at the same table can communicate, which is not the case for larger tables. Our approach takes the topology of the dining table into account.

Some previous works have also considered the topology of the dining table. Perhaps closest to our paper is the model of Bodlaender et al. [7, 6], in which n individuals are to be assigned to the n vertices of an undirected seating graph. The authors prove a number of computational results regarding both envy-freeness and exchange-stability, among other notions. However, we found some of the table topologies considered to be rather unnatural, especially in hardness proofs (e.g., trees or unions of cliques and independent sets). Bullinger and Suksompong [8] also conduct an algorithmic study of a similar problem, but with a few key differences: (i) individuals are seated in the nodes of a graph, but there may be more seats than people; (ii) for the stability notion, they principally consider *jump-stability*, where unhappy people can choose to move to a free seat; (iii) individuals now contribute to everyone's utility, although their contribution decreases with distance.

Last but not least, studying stability in the context of *Schelling games* has recently been a popular area of research [11, 1, 4, 15, 3]. In Schelling games, individuals belong to a fixed number of classes. However, unlike in our model, agents from one class only care about sitting next to others of their own class. This additional assumption often allows for stronger results; e.g., in [3] the authors prove the existence of exchange-stable arrangements on regular and almost regular topological graphs such as cycles and paths, and show that the swap dynamics are guaranteed to converge in polynomial time on such topologies.

Overall, it seems that exchange-stability has been studied in the frameworks of both hedonic and Schelling games. However, both approaches present some shortcomings: on the one hand, Schelling games inherently consider a topology on which agents evolve, but, being historically motivated by the study of segregation (e.g., ethnic, racial), they usually restrict themselves to very simple preferences. On the other hand, works on hedonic games are accustomed to considering diverse preferences. However, while multiple works have introduced topological considerations, their analysis is usually restrained to graphs that can be interpreted as non-overlapping coalitions, e.g., with multiple fully connected components.

3 Preliminaries

We write $[n] = \{1, \dots, n\}$. Given an undirected graph $G = (V(G), E(G))$, we write $N_G(v)$ for the set of neighbors of vertex $v \in V(G)$. When clear from context, oftentimes we will simply write V, E and $N(v)$ respectively.

The model we describe next is similar to the one in [7]. A group of n agents (guests) $\mathcal{A}$ has to be seated at a dining table represented by an undirected graph $G = (V, E)$, where vertices correspond to seats. We will be interested in the cases of G being a cycle or a path. We assume that $|V| = n$ and that no two agents can be seated in the same place, from which it also follows that all the seats have to be occupied. Agents have numerical preferences over each other, corresponding to how much utility they gain from being seated next to other agents. In particular, each agent $i \in \mathcal{A}$ has a *preference* over the other agents expressed as a function $p_i : \mathcal{A} \setminus \{i\} \to \mathbb{R}$, where $p_i(j)$ denotes the utility gained by agent i when seating next to j. Note that we do not assume symmetry; i.e., it might be that $p_i(j) \neq p_j(i)$. We denote by $\mathcal{P} = (p_i)_{i \in \mathcal{A}}$ the collection of agent preferences, or *preference profile*, of the agents. A number of different interpretations can be associated to $\mathcal{P}$. In particular, we will usually see $\mathcal{P}$ as a matrix $\mathcal{P} = (p_{ij})_{i,j \in \mathcal{A}}$, where $p_{ij} = p_i(j)$. Note that the diagonal entries are not defined, so we will usually use the convention that $p_{ii} = 0$. Using the matrix notation, we say that the preferences in $\mathcal{P}$ are *binary* when $\mathcal{P} \in \{0,1\}^{n \times n}$ and *k-valued* if there exists $\Gamma \subseteq \mathbb{R}$, $|\Gamma| = k$, such that $\mathcal{P} \in \Gamma^{n \times n}$. Note that binary preferences are two-valued, but two-valued preferences are not necessarily binary. Moreover, we will often represent binary preferences as a directed graph, where a directed edge between two agents signifies that the first agent approves of the second.

We define a *class of agents* to be a subset of indistinguishable agents $\mathcal{C} \subseteq \mathcal{A}$. More formally, all agents in $\mathcal{C}$ share a common preference function $p_{\mathcal{C}} : \mathcal{A} \to \mathbb{R}$ and no agent in $\mathcal{A}$ discriminates between two agents in $\mathcal{C}$. Note that this implies that the lines and columns of the preference matrix corresponding to agents in $\mathcal{C}$ are identical, if we adopt the convention that diagonal terms inside a class are all equal but not necessarily null. We say that preference profile $\mathcal{P}$ has *k-classes*, or is a *k-class preference*, if $\mathcal{A}$ can be partitioned into k classes $\mathcal{C}_1, \ldots, \mathcal{C}_k \subseteq \mathcal{A}$.

We define an *arrangement* of the agents on G to be a bijection $\pi : \mathcal{A} \to V(G)$, i.e., an assignment of each agent to a unique vertex of the seating graph (and vice-versa). For a given arrangement π, we define for each agent $i \in \mathcal{A}$ its *utility* $U_i(\pi) = \sum_{v \in N_G(\pi(i))} p_i(\pi^{-1}(v))$ to be the sum of agent i's preferences towards its graph neighbors in the arrangement. We say that agent i *envies* agent j whenever $U_i(\pi) < U_i(\pi')$, where π' is π with $\pi(i)$ and $\pi(j)$ swapped. We further say that (i,j) is a *blocking pair* if both i envies j and j envies i; i.e., they would both strictly increase their utility by exchanging seats. An arrangement is *envy-free* if no two agents envy each other, and it is *stable* if it induces no blocking pairs. Note that envy-freeness implies stability, but the converse is not necessarily true. By extension, we call preference profile $\mathcal{P}$ stable (respectively envy-free) on G if there exists a stable (respectively envy-free) arrangement π on G.

4 Envy-Freeness

It turns out to be relatively easy to construct preferences with no envy-free arrangements: for paths, even if all agents like each other, the agents sitting at the endpoints will envy the others; for cycles, add an agent despised by everyone, agents sitting next to it will envy their peers. We now show that, furthermore, it is NP-hard to decide whether envy-free arrangements exist, for both paths and cycles, even under binary preferences. We begin with the case of cycles.

Theorem 1. *For binary preferences, deciding whether an envy-free arrangement on a cycle exists is NP-hard.*

Proof. We proceed by reduction from Hamiltonian Cycle on directed graphs. Let $G = (V, E)$ be a directed graph such that, without loss of generality, $V = [n]$. If G has any vertices with no outgoing edges, then map the input instance to a canonical no-instance, unless $n = 1$, in which case we map to a canonical yes-instance. Hence, from now on assume that all vertices have outgoing edges. For each vertex $v \in V$ introduce three agents x_v, y_v, z_v such that agent x_v only likes y_v and dislikes everyone else, agent y_v only likes z_v and dislikes everyone else, and agent z_v likes agent x_u for all $u \in V$ such that $(v,u) \in E$, and dislikes everyone else. We claim that the so-constructed preference profile $\mathcal{P}$ has an envy-free arrangement on a cycle precisely when G has a Hamiltonian cycle. To show this, first assume without loss of generality that $1 \to 2 \to \ldots \to n \to 1$ is a Hamiltonian cycle in G. Then, arranging agents around the cycle in the order $x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n$ is an envy-free arrangement. To see why, notice that in this arrangement all agents get utility 1, so envy could only potentially stem from an agent being able to swap places with another agent to get utility 2. To prove this is not possible, first notice that agents $(x_i)_{i\in[n]}$ and $(y_i)_{i\in[n]}$ each only like one other agent, so they can never get a utility of more than 1 in any arrangement. Moreover, no agent z_i can get to a utility of 2 by a single swap because any two agents they like are seated at least three positions away on the cycle. Conversely, assume that an envy-free arrangement π exists. First, if x_i is not seating next to y_i in π, then x_i could improve by swapping to a place next to y_i (also similarly for y_i and z_i). Therefore, in arrangement π agent x_i is seated next to y_i and y_i is seated next to z_i. Moreover, consider the other neighbor of z_i in π. Since z_i does not like y_i, it follows that if z_i also does not like their other neighbor, then z_i could strictly improve their utility by swapping next to some agent they like, which is always possible because all vertices in G have outgoing edges. Therefore, the other neighbor of z_i has to be some agent that they like, hence being of the form x_j, where $j \neq i$. By construction, this means that $(i, j) \in E$. Putting together what we know, we get that under π the agents are arranged around in the cycle in some order $x_{\sigma_1}, y_{\sigma_1}, z_{\sigma_1}, x_{\sigma_2}, y_{\sigma_2}, z_{\sigma_2}, \ldots, x_{\sigma_n}, y_{\sigma_n}, z_{\sigma_n}$, where σ is a permutation of the n agents such that $(\sigma_i, \sigma_{i+1}) \in E$ holds for $i \in [n]$, assuming that addition is performed with wrap-around such that $n + 1 = 1$. Therefore, a Hamiltonian cycle $\sigma_1 \to \sigma_2 \to \ldots \to \sigma_n \to \sigma_1$ exists in G. This completes the reduction. □

A similar proof can be used to show hardness for the case of paths. We outline the changes required in the appendix.

Theorem 2. *For binary preferences, deciding whether an envy-free arrangement on a path exists is NP-hard.*

Proving similar hardness results for stability would be highly desirable; we discuss some steps towards achieving this in Section 6. Moreover, one might ask how does the number of agent classes affect the computational complexity of our problems. In the next section, we address this question precisely, showing that a bounded number of agent classes renders the problems we consider computable in polynomial time, even for non-binary preferences.

5 Polynomial Solvability for k-Class Preferences

In this section, we show that deciding whether envy-free and stable arrangements exist for a given preference profile can be achieved in polynomial time, for both paths and cycles, assuming that the number of agent classes is bounded by a number k. Note that preferences in this case are not constrained to being binary. By extension, our algorithms can also be used to construct such arrangements whenever they exist.

We begin with the case of paths. For simplicity, we assume that $n \geq 3$, as for $n \leq 2$ any arrangement is both stable and envy-free. Assume that the agent classes are identified by the numbers $1, \ldots, k$ and that $n_1, n_2, \ldots, n_k$ are the number of agents of each class in our preference profile, where $n_1 + \ldots + n_k = n$. For ease of writing, we will see arrangements as sequences $s = (s_i)_{i\in[n]}$, where $s_i \in [k]$ and for any agent class $j \in [k]$ the number of values j in s is n_j. Moreover, for brevity, we lift agent preferences to class preferences, in order to give meaning to statements such as "class a likes class b." To simplify the treatment of agents sitting at the ends of the path, we introduce two agents of a dummy class 0 with preference values 0 from and towards the other agents. We require the dummy agents to sit at the two ends of the path; i.e., $s_0 = s_{n+1} = 0$. In order to use a common framework for stability and envy-freeness, we will define the concept of *compatible triples* of agent classes, as follows. First, for envy-freeness, let a, b, c, d, e, f be agent classes, then we say that triples (a, b, c) and (d, e, f) are *long-range compatible* if $p_b(a) + p_b(c) \geq p_b(d) + p_b(f)$ and $p_e(d)+p_e(f) \geq p_e(a)+p_e(c)$; intuitively, neither b wants to swap with e, nor vice-versa. Furthermore, for a, b, c, d agent classes, we say that triples (a, b, c) and (b, c, d) are *short-range compatible* if $p_b(a) \geq p_b(d)$ and $p_c(d) \geq p_c(a)$; intuitively, if a, b, c, d are consecutive in the arrangement, then neither b wants to swap with c, nor vice-versa. For stability, we keep the same definitions but use "or" instead of "and." Note that long-range and short-range compatibility do not imply each other. We call an arrangement s *compatible* if for all $1 \leq i < j \leq n$ the triplets (s_{i-1}, s_i, s_{i+1}) and (s_{j-1}, s_j, s_{j+1}) are long-range compatible when $j - i > 1$ and short-range compatible when $j - i = 1$. Note that arrangement s is envy-free (resp. stable) if and only if it is compatible. In the following, we explain how to decide the existence of a compatible arrangement.

Lemma 3. *Deciding whether compatible arrangements exist can be achieved in poly-time.*

Proof. We first present a nondeterministic algorithm (i.e., with guessing) that solves the problem in polynomial time. The algorithm builds a compatible arrangement s one element at a time. Initially, the algorithm sets $s_0 \leftarrow 0$ and guesses the values of s_1 and s_2. Then, at step i, for $3 \leq i \leq n + 1$, the algorithm will guess s_i (except for $i = n + 1$, where we enforce that $s_i \leftarrow 0$) and check whether (s_{i-2}, s_{i-1}, s_i) is short-range conflicting with $(s_{i-3}, s_{i-2}, s_{i-1})$, rejecting if so. Moreover, the algorithm will check whether (s_{i-2}, s_{i-1}, s_i) is long-range conflicting with any (s_{j-2}, s_{j-1}, s_j) for $2 \leq j \leq i - 2$, again rejecting if so. At the end, the algorithm checks whether for each $i \in [k]$ value i occurs in s exactly n_i times, accepting if so, and rejecting otherwise.

Alone, this algorithm only shows containment in NP, which is not a very attractive result. Next, we show how the same algorithm can be implemented with only a constant number of variables, explaining afterward why this implies our result. First, to simulate the check at the end of the algorithm without requiring knowledge of the whole of s, it is enough to maintain throughout the execution counts $(x_j)_{j\in[k]}$ such that at step i in the algorithm x_j gives the number of positions $1 \leq \ell \leq i$ such that $s_\ell = j$. To simulate the short-range compatibility check, it is enough that at step i we have knowledge of $s_{i-3}, \ldots, s_i$. Finally, for the long-range compatibility check, a more insightful idea is required. In particular, we make the algorithm maintain throughout the execution counts $m_{a,b,c}$ for each triple (a, b, c) of agent classes, such that at step i value $m_{a,b,c}$ gives the number of positions $2 \leq \ell \leq i$ such that $(s_{\ell-2}, s_{\ell-1}, s_\ell) = (a, b, c)$. Using this information, to check at step i whether (s_{i-2}, s_{i-1}, s_i) long range conflicts with any (s_{j-2}, s_{j-1}, s_j) for $2 \leq j \leq i - 2$, it is enough to temporarily decrease by one the values $m_{s_{i-3},s_{i-2},s_{i-1}}$ and m_{s_{i-2},s_{i-1},s_i} and then check whether there exists a triple (a, b, c) of agent classes such that $m_{a,b,c} > 0$ and (a, b, c) long-range conflicts with (s_{i-2}, s_{i-1}, s_i). In total, at step i, the algorithm only needs to know the values $s_{i-3}, \ldots, s_i$, as well as $(x_j)_{j\in[k]}$ and the counts $m_{a,b,c}$ for all triples (a, b, c) of agent classes. Since k bounds the total number of agent classes, this is only a constant number of variables. As each variable can be represented with $O(\log n)$ bits, it follows that our nondeterministic algorithm uses only logarithmic space, implying containment in the corresponding complexity class NL. It is well known that NL $\subseteq$ P, from which our conclusion follows. For readers less familiar with this result, we give a short overview of how our algorithm can be converted into a deterministic polynomial-time algorithm, as follows. Since our NL algorithm uses only logarithmic space, it follows that the space of algorithm states that can be reached depending on the nondeterministic choices is of polynomial size, as $2^{O(\log n)}$ is polynomial. Therefore, building a graph with vertices being states and oriented edges corresponding to transitions between states, the problem reduces to deciding whether an accepting state can be reached from the initial state, which can be done with any graph search algorithm. □

Theorem 4. *Fix k in $\mathbb{N}$. For k-class preferences, there are polynomial-time algorithms computing an envy-free/stable arrangement on a path or reporting the nonexistence thereof.*

For the case of cycles, a similar approach can be used, although with rather tedious, yet minor tweaks, presented in the appendix.

Theorem 5. *Fix k in $\mathbb{N}$. For k-class preferences, there are polynomial-time algorithms computing an envy-free/stable arrangement on a cycle or reporting the nonexistence thereof.*

6 Stability

Previously, we showed that deciding whether envy-free arrangements exist is NP-hard. However, finding stable arrangements might be easier, as the requirement of stability is much weaker than that of envy-freeness. A first step towards understanding the difficulty of the problem is being able to construct instances where no stable arrangements exist; after all, a problem where the answer is always "yes" cannot be NP-hard. In this section, we show how to construct such unstable preferences. In fact, our analysis is more granular, considering how different constraints on the number of agent classes as well as the number of different values allowed in the preferences influence the existence of preferences making every arrangement unstable on a path or a cycle. By doing so, we hope to develop a better understanding of the necessary conditions for the disappearance of stability. In particular, we show that two-class preferences always induce a stable arrangement, whereas three-valued three-class preferences are sufficient to break stability on both paths and cycles for any n large enough. For cycles, we moreover show that binary four-class preferences are also enough to break stability for any n large enough, this construction turning out to be the trickiest. Table 2 summarises our results. Afterwards, we venture into the leftover case, which is stability on paths under binary preferences. For this case, we were unable to construct any unstable instances, and have proven their nonexistence for $n \leq 7$. We conjecture this to be true in general and prove a weaker version of this result, namely that a stable arrangement is guaranteed to exist if agents can only swap seats when sitting at most two positions away on the path.

6.1 Preferences with Two Classes of Agents

As a warm-up, note that when all agents come from a single class, any arrangement on any given seating graph is stable. In the following, we extend this result to two classes of agents for cycles and paths. We begin with cycles:

Theorem 6. *Two-class preferences always induce a stable arrangement on a cycle.*

Proof. Without loss of generality, the preferences can be assumed to be binary in the case of two-class preferences on a cycle. Suppose there are two classes of agents, say Blues and Reds. First, note that any blocking pair must consist of one Blue and one Red. Moreover, note that any arrangement is stable whenever one of the classes likes the two classes equally. Now, suppose this is not the case, meaning that each class has a preferred class to sit next to. There are only two cases to consider:

If one class, say Blue, prefers its own class, then sit all Blues together and give the remaining seats to Reds: all Blues but two, say B_1 and B_2, get maximum utility, and neither B_1 nor B_2 can improve since no Red has more than one Blue neighbor. Hence no Blue is part of a blocking pair, and the arrangement is stable.

If both classes prefer the opposite class, we may assume there are at least as many Reds as Blues. Then we alternate between Reds and Blues for as long as there are Blues without a

n	3	4	5	6	7
Cycle	0	0	1	0	2
Path	0	0	0	0	0

Table 3: Number of non-isomorphic families of unstable binary preferences.

seat, then seat all the remaining Reds next to each other. Every Blue has maximum utility, and cannot be part of a blocking pair, hence the arrangement is stable. □

The following extends our result to the case of paths. The proof is largely similar, but the case analysis becomes more involved, so we present it in the appendix.

Theorem 7. *Two-class preferences always induce a stable arrangement on a path.*

Note that a path of size n is equivalent to a cycle of size $n+1$ where an agent with null preferences is added. This explains why the case of paths is harder to study than the one of cycles, as it corresponds to having one more class of agents and potentially one more value (zero). A similar approach could perhaps be used to handle the case of three classes under binary preferences, but the number of cases to consider would be noticeably larger.

6.2 Three-Valued Preferences with Three Classes of Agents

We now consider the case of three-valued preferences using only three classes. In particular, we show the existence of three-class three-valued preferences such that no arrangement on a cycle or a path is stable.

Theorem 8. *For $n \geq 4$, there exist three-class three-valued preferences such that all arrangements on a cycle are unstable.*

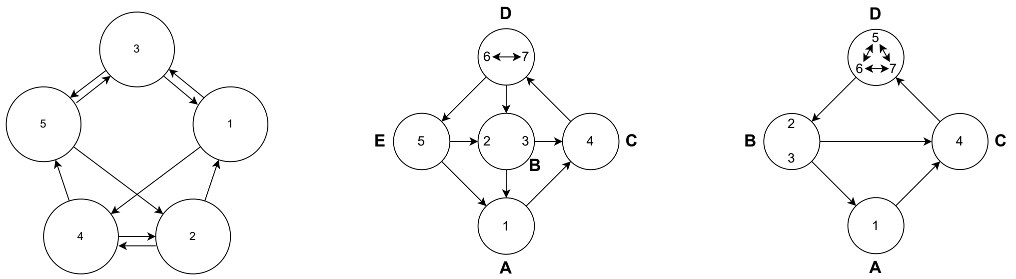
Proof. Consider three classes of agents: Alice, Bob, and $n-2$ of Bob's friends. The story goes as follows: Alice and Bob broke up. Alice does not want to hear about Bob and would hence prefer to sit next to any of his friends rather than Bob. On the other hand, Bob wants to win her back, so he would above all want to sit next to Alice. Finally, Bob's friends prefer first Bob, then the other friends, and finally Alice. To show that these preferences are unstable on a cycle, there are two cases to consider: either sit Alice and Bob next to each other, or separately.

In the first case, Alice and her second neighbor, who is one of Bob's friends, would exchange seats. After the switch, Alice is better as she no longer sits next to Bob, and the friend is better because he sits next to Bob.

In the second case, Bob and one of Alice's neighbors would exchange seats. Bob is better because he now sits next to Alice. To see that the neighbor, who is one of Bob's friends, is also better, distinguish two sub-cases: if the friend sits right between Alice and Bob, then he is better because he now no longer sits next to Alice, while if this is not the case, he is better because before he was sitting next to Alice and a friend, while now he is sitting next to two friends.

□

Unfortunately, the proof of Theorem 8 does not directly transfer to paths; e.g., for $n = 5$, path arrangements (F, F, B, A, F) or (B, F, F, A, F) are stable. It is possible to use the same construction for paths by introducing negative values in the preferences. Negative preferences are however not necessary to make every arrangement unstable: we show that three-valued non-negative preferences are enough for n large enough. We summarize both of these results in the following.



(a) $\mathcal{P}_5$ as a directed graph. (b) $\mathcal{P}_7^{(1)}$ as a directed graph. (c) $\mathcal{P}_7^{(2)}$ as a directed graph.

Figure 1: The three non-isomorphic families of binary unstable preferences for $n \leq 7$.

Theorem 9. *For $n \geq 4$, there exist three-valued three-class preferences such that all arrangements on a path are unstable. For $n \geq 12$, this holds even for non-negative values.*

6.3 Two-Valued Preferences

It remains to study what happens for two-valued preferences with three or more classes of agents. We will actually study binary preferences, i.e., values zero and one, but, at least for cycles, this is without loss of generality (on paths, extremities artificially introduce a comparison with zero, changing the behavior for negative preferences). We exhausted all binary preferences for $n \leq 7$ using a Z3 Python solver (see Appendix G), hoping to find unstable ones and generalize from them. Table 3 summarizes our findings. For paths, no unstable preferences were found, while for cycles, unstable preferences were found for $n = 5$ and $n = 7$, with one and, respectively, two non-isomorphic families of unstable preferences. Examples of such preferences from each family $\mathcal{P}_5, \mathcal{P}_7^{(1)}$ and $\mathcal{P}_7^{(2)}$ are illustrated in Figure 1.

Analyzing why $\mathcal{P}_5$ is unstable turns out to be quite complex (see Appendix B), and, as of our current understanding, its instability seems to be more of a "small size artefact" than anything else. In contrast, with their highly regular structure, the two instances with $n = 7$ seem more promising. In particular, profile $\mathcal{P}_7^{(2)}$ exhibits only four classes, denoted by A, B, C, D in Figure 1c. In the following, we show that $\mathcal{P}_7^{(2)}$ can be extended to unstable preferences of any size $n \geq 7$.

Theorem 10. *For $n \geq 7$, there exist binary preferences using four classes such that no arrangement on a cycle is stable.*

Proof. For $n \geq 7$, we consider four classes A, B, C and D, as well as their respective members a, b_1, b_2, c and $d_1, \ldots, d_{n-4}$. Similarly to Figure 1c, we suppose that: (i) a only likes c; (ii) b_1 and b_2 both only like a and c; disliking each other; (iii) c only likes members of D; (iv) members of D all like each other, as well as b_1 and b_2, only disliking a and c. We show in the appendix why such preferences induce no stable arrangement on a cycle. □

For the case of paths, on the other hand, we could not find any unstable preferences with $n \leq 7$. In the next section we show that, in fact, binary preferences cannot be unstable on paths under the additional assumption that agents only trade places with other agents sitting at most two positions away on the path.

6.3.1 Stability of Binary Preferences on Paths

In this section, we work under the additional assumption that two agents can never swap seats when they are more than two positions away on the path, no matter how much it would increase their utilities. This can be thought of as a practical constraint: once the agents are seated, each agent knows which other agents they envy, but finding out whether envy is reciprocal would be too cumbersome if the other agent is seated very far away. For this setup, we prove that the swap dynamics always converge, so a stable arrangement can be found by starting with an arbitrary arrangement and swapping blocking pairs until the arrangement becomes stable. This can be seen as a generalization of a result from [4], where agents only have preference for others of their kind and swaps are only with adjacent agents. To begin, for any arrangement π define the utilitarian social welfare $W(\pi) = \sum_{i \in \mathcal{A}} U_i(\pi)$. Moreover, to each arrangement π we associate a sequence $S(\pi)$ of length $n-1$ with elements in $\{0, 1, 2, 3\}$, constructed as follows. Let x_i and x_{i+1} be the agents sitting at positions i and $i+1$ on the path: if they do not like each other, then $S(\pi)_i = 0$, if they both like each other, then $S(\pi)_i = 3$, if only x_i likes x_{i+1}, then $S(\pi)_i = 1$, otherwise $S(\pi)_i = 2$. To prove that the swap dynamics converge, we define the potential $\Phi(\pi) = (W(\pi), S(\pi))$, where sequences are compared lexicographically, and prove that swaps always strictly increase the potential. The following two lemmas show this for swaps at distances one and two, respectively.

Lemma 11. *Let π be an arrangement where agents a and b form a blocking pair and are seated in adjacent seats. Let π' be π with a and b's seats swapped. Then, $\Phi(\pi) > \Phi(\pi')$.*

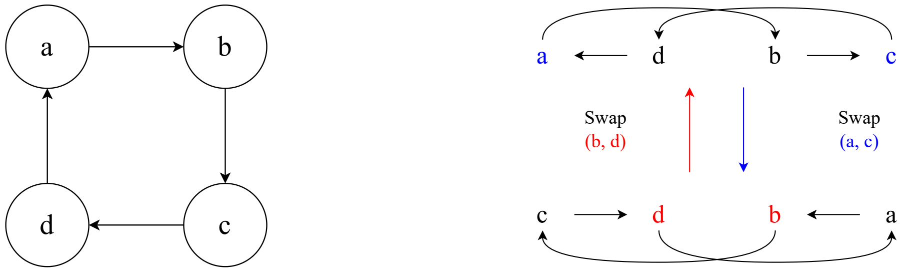
Proof. When $n \leq 2$, there are no blocking pairs, so assume $n \geq 3$. First, notice that swapping the places of a and b keeps a and b adjacent, from which the swap changes the utility of any agent by at most one. Since a and b's utilities have to increase, they have to each change by exactly one. Moreover, note that neither a nor b can be seated at the ends of the table, as otherwise swapping would make one of them lose a neighbor while keeping the other one, hence not increasing their utility. Hence, assume that x is the other neighbor of a and y is the other neighbor of b; i.e., x, a, b, y are seated in order consecutively on the path, at positions say $i, \ldots, i+3$. If either $U_x(\pi') \geq U_x(\pi)$ or $U_y(\pi') \geq U_y(\pi)$, it follows that $W(\pi') - W(\pi) = U_x(\pi') - U_x(\pi) + U_y(\pi') - U_y(\pi) + 2 \geq 1$, so $\Phi(\pi') > \Phi(\pi)$. Otherwise, we know that $U_x(\pi') - U_x(\pi) = U_y(\pi') - U_y(\pi) = -1$, from which $W(\pi) = W(\pi')$. Together with $U_a(\pi') - U_a(\pi) = U_b(\pi') - U_b(\pi) = 1$, this means that preferences satisfy $a \to y \to b \to x \to a$, where an arrow $u \to v$ means that agent u likes v but not the other way around. Therefore, $S(\pi)_i = 1$ and $S(\pi')_i = 2$. Since $S(\pi)$ and $S(\pi')$ only differ at positions $i, \ldots, i+2$, this means that $S(\pi') > S(\pi)$, so $\Phi(\pi') > \Phi(\pi)$, as required. □

Lemma 12. *Let π be an arrangement where agents a and b form a blocking pair and are seated two seats away. Let π' be π with a and b's seats swapped. Then, $\Phi(\pi) > \Phi(\pi')$.*

Proof. The same argument works, except that now we consider five agents x, a, z, b, y seated at positions $i, \ldots, i+4$. This is because agent z remains a common neighbor to a and b when swapping places, and can, essentially, be ignored. □

Therefore, since the potential is upper-bounded, we get that the swap dynamics have to converge. One might now rightfully ask whether convergence is guaranteed to take polynomial time. While we could neither prove nor disprove this, in Appendix C we give evidence of why exponential time might be required. Moreover, note that for cycles convergence is not guaranteed even for swaps at distance at most two; e.g., $\mathcal{P}_5$ in Figure 1a, where any two agents are seated at most two seats away anyway. For paths, on the other hand, one could hope that the result generalizes beyond distance at most two. However, this is not the case, even when stable arrangements exist, as we show next.

Lemma 13. *Consider the four-agent preference profile $\mathcal{P}_4$ in Figure 2a. A path stable arrangement exists, yet swap dynamics started from certain arrangements cannot converge.*



(a) $\mathcal{P}_4$ as a directed graph.

(b) Looping swap dynamics.

Figure 2: Binary four-agent preferences $\mathcal{P}_4$ with path stable arrangement $\pi^* = (a, b, c, d)$ but where the swap dynamics necessarily alternate between $\pi_1 = (a, d, b, c)$ and $\pi_2 = (c, d, b, a)$ when started in either of them.

Note that this result generalizes to an arbitrary number of agents $n \geq 4$ by adding $n - 4$ dummy agents to $\mathcal{P}_4$ liking nobody and being liked by nobody and seating them at positions $5, \ldots, n$ on the path. Hence, non-convergence for distance ≤ 3 is not a small-n artifact.

7 Conclusions and Future Work

We studied envy-freeness and exchange-stability on paths and cycles. For both topologies, we showed that finding envy-free/stable arrangements can be achieved in polynomial time when the number of agent classes is bounded, while for envy-freeness the problem becomes NP-hard without this restriction, even for binary preferences. For stability, it would be interesting to see if such an NP-hardness result can still be proven; we believe this to be the case, but were unable to do so. In part, this is because of the difficulty of constructing unstable instances. As a step towards understanding such preferences, we proved that instability may emerge from simple preferences with few different values and classes of agents. In particular, the construction of binary preferences unstable on a cycle turned out to be the trickiest. We are still unsure whether binary preferences unstable on a path exist. We, however, partially answer this in the negative by showing that the swap dynamics are guaranteed to converge if agents can only swap places with other agents seated close enough to them. Without this assumption, convergence might not be ensured even when stable arrangements exist, so a different approach would be required to prove existence. It would also be interesting to know if unstable preferences are exceptions or the norm. We give a probabilistic treatment of this question for random preference digraphs of average degree $O(\sqrt{n})$ in the appendix. As an avenue for future research, it would be attractive to consider other kinds of tables commonly used in practice, the most relevant being the one shaped as a $2 \times n$ grid, with guests on either side facing each other.

References

[1] Aishwarya Agarwal, Edith Elkind, Jiarui Gan, Ayumi Igarashi, Warut Suksompong, and Alexandros A. Voudouris. Schelling games on graphs. *Artificial Intelligence*, 301: 103576, 2021. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103576.

[2] Haris Aziz and Rahul Savani. Hedonic games. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational*

Social Choice, chapter 15. Cambridge University Press, USA, 1st edition, 2016. ISBN 1107060435.

[3] Davide Bilò, Vittorio Bilò, Pascal Lenzner, and Louise Molitor. Topological influence and locality in swap schelling games. *Autonomous Agents and Multi-Agent Systems*, 36, 08 2022. doi: 10.1007/s10458-022-09573-7.

[4] Vittorio Bilò, Angelo Fanelli, Michele Flammini, Gianpiero Monaco, and Luca Moscardelli. Nash Stable Outcomes in Fractional Hedonic Games: Existence, Efficiency and Computation. *Journal of Artificial Intelligence Research*, 62:315–371, June 2018. ISSN 1076-9757. doi: 10.1613/jair.1.11211.

[5] Vittorio Bilò, Gianpiero Monaco, and Luca Moscardelli. Hedonic Games with Fixed-Size Coalitions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9287–9295, June 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i9.21156. Number: 9.

[6] Hans L. Bodlaender, Tesshu Hanaka, Lars Jaffke, Hirotaka Ono, Yota Otachi, and Tom C. van der Zanden. Hedonic Seat Arrangement Problems. *arXiv*, February 2020. doi: 10.48550/arXiv.2002.10898.

[7] Hans L. Bodlaender, Tesshu Hanaka, Lars Jaffke, Hirotaka Ono, Yota Otachi, and Tom C. van der Zanden. Hedonic seat arrangement problems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS'20, pages 1777–1779, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

[8] Martin Bullinger and Warut Suksompong. Topological Distance Games. *arXiv*, November 2022. doi: 10.48550/arXiv.2211.11000. To appear in AAAI'23.

[9] Katarína Cechlárová. On the complexity of exchange-stable roommates. *Discrete Applied Mathematics*, 116(3):279–287, February 2002. ISSN 0166-218X. doi: 10.1016/S0166-218X(01)00230-X.

[10] Esra Ceylan, Jiehua Chen, and Sanjukta Roy. Optimal seat arrangement: What are the hard and easy cases? *arXiv 2305.10381*, May 2023. To appear in IJCAI'23.

[11] Ankit Chauhan, Pascal Lenzner, and Louise Molitor. Schelling segregation with strategic agents. In Xiaotie Deng, editor, *Algorithmic Game Theory*, pages 137–149, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99660-8.

[12] D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, January 1962. ISSN 0002-9890, 1930-0972. doi: 10.1080/00029890.1962.11989827.

[13] Robert W Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595, December 1985. ISSN 01966774. doi: 10.1016/0196-6774(85)90033-1.

[14] Robert W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48 (3):261–272, 1994. ISSN 0166-218X. doi: 10.1016/0166-218X(92)00179-P.

[15] Luca Kreisel, Niclas Boehmer, Vincent Froese, and Rolf Niedermeier. Equilibria in schelling games: Computational hardness and robustness. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS'22, pages 761–769, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

[16] Eytan Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11(2): 285–304, June 1990. ISSN 0196-6774. doi: 10.1016/0196-6774(90)90007-2.

Damien Berriaud
ETH Zürich
Zürich, Switzerland
Email: dberriaud@student.ethz.ch

Andrei Constantinescu
ETH Zürich
Zürich, Switzerland
Email: aconstantine@ethz.ch

Roger Wattenhofer
ETH Zürich
Zürich, Switzerland
Email: wattenhofer@ethz.ch

A Omitted Proofs

In this appendix, we provide the proofs omitted in the main text of the paper.

A.1 Proofs Omitted From Section 4

In this section, we prove Theorem 2, which we restate below for convenience.

Theorem 2. *For binary preferences, deciding whether an envy-free arrangement on a path exists is NP-hard.*

Proof. We proceed similarly as for Theorem 1, this time reducing from Hamiltonian Path on directed graphs. To make the reduction work, we require the additional stipulation that the input graph G has a vertex with no outgoing edges, which we assume without loss of generality to be vertex n. Note that this preserves NP-hardness. Moreover, we will now only check that vertices $v \in V \setminus \{n\}$ have outgoing edges. Otherwise, the construction of preference profile $\mathcal{P}$ stays the same. To show that if G has a Hamiltonian path, then there exists an envy-free arrangement on a path, the same argument as above can be used. To show that an envy-free arrangement on a path implies the existence of a Hamiltonian path, the argument stays similar to the previous one but requires some minor tweaks. In particular, for agent z_n, and only for them, it holds that they do not necessarily need a second neighbor other than y_n, because they like no other agents, so they will be happy with a utility of 0, obtained by sitting at one of the ends of the path. The analysis for the other agents stays the same, as they do require a second neighbor to get a utility of 1, and hence cannot sit at the ends. □

A.2 Proofs Omitted From Section 5

In this section, we prove Theorem 5, which we restate below for convenience.

Theorem 5. *For k-class preferences, there are polynomial-time algorithms computing an envy-free/stable arrangement on a cycle or reporting the nonexistence thereof.*

Proof. The proof idea stays similar to that for paths, first designing a nondeterministic logarithmic space algorithm deciding whether a compatible arrangement exists and then lifting this to one running in deterministic polynomial time. Since cycles no longer have endpoints, the definition of compatible arrangements needs adjusting. First, instead of introducing a dummy agent class 0 and placing it at positions 0 and $n+1$ in s, we now make s_0 stand for s_n, and s_{n+1} stand for s_1; i.e., $s_0 = s_n$ and $s_{n+1} = s_1$. Moreover, arrangement s is now called compatible if for all $1 \leq i < j \leq n$ the triplets (s_{i-1}, s_i, s_{i+1}) and (s_{j-1}, s_j, s_{j+1}) are short-range compatible when i and j are adjacent on the cycle, i.e., $(j - i) \in \{1, n-1\}$, and long-range compatible otherwise. Note, therefore, that the pair $(i, j) = (1, n)$ is the only pair for which the required check differs from the path case: previously the check was for long-range compatibility, while now it is for short-range compatibility. This time, instead of beginning the algorithm by guessing the values of s_1 and s_2, we instead begin it by guessing the values of s_0, s_1 and s_2. The algorithm then proceeds as before. However, when the value $i = n$ is reached, naturally, no guessing takes place, as s_n has already been guessed, but all other computations execute as before. Moreover, when $i = n + 1$ is reached, it is even trickier; not only do we not have to guess $s_{n+1} = s_1$, but also the check performed has to be altered. In particular, instead of checking whether (s_{n-1}, s_n, s_1) is long-range compatible with (s_n, s_1, s_2), the check now has to be for short-range compatibility. Checking for short-range compatibility can easily be incorporated, so it remains to show how to ensure that the two triplets are not also tested for long-range compatibility like in the previous implementation. This is done as follows: instead of temporarily decreasing the values $m_{s_{i-3},s_{i-2},s_{i-1}}$ and m_{s_{i-2},s_{i-1},s_i} by one and then checking (s_{i-2}, s_{i-1}, s_i) against the counts in m, we now do the same but also decrease m_{s_n,s_1,s_2} by one. We stress that these rather tedious modifications are only applied for $i = n + 1$. The modified algorithm needs to store s_1 and s_2 throughout its execution in addition to the state it already stored, but this does not impact the logarithmic space bound, completing the proof. □

A.3 Proofs Omitted From Section 6.1

In this section, we prove Theorem 7, which we restate below for convenience.

Theorem 7. *Two-class preferences always induce a stable arrangement on a path.*

Proof. The introduction of agents sitting at the two endpoints calls for a more careful analysis:

- If one class, say Blue, likes everyone equally.

  Then Blues either prefer to be on the edge or in the middle of the path (depending on the sign of their constant preference). If they prefer edges, seating two Blues on the edges makes the arrangement stable; otherwise, sitting all Blues in the middle ensures stability.

  Those two solutions are not available if and only if there is a single agent in one of the two classes: in that case, ensuring that this agent gets maximal utility stabilizes the whole arrangement.

- If Blues and Reds prefer the same class, say Blues.

  The reasoning from the previous proof still holds, as long as no Blue sits at an extremity. This would happen if and only if there is a unique Red: ensuring maximum utility for that one agent would then give stability.

- If Blues and Reds both strictly prefer the opposite kind, alternate Blues and Reds starting from the most numerous class, say Blue. Suppose:

  – Preferences for the opposite kind are positive for both. Whenever there is strictly more of one class than the other, the reasoning from the previous proof still holds. In case of equality, only the Blue and Red at both extremities could want to swap: it is not the case, as they would exchange a “different colour” neighbor for a “same colour” one.

  – Preferences for the opposite kind are negative for both. Now at most two agents have maximum utility: the Blue extremal one, and the Red extremal one if there are as many Reds as Blues. If both have maximum utility, the others cannot improve, and the arrangement is stable. The arrangement is still stable if there are strictly more Blues than Reds, as the extremal Blues could only agree to swap between two Reds. As only Blues are sitting between two Reds, this cannot happen.

  – Preference of Blues for Reds is negative, but Reds for Blues is positive. Then non-extremal Blues would envy the extremal Red, which would envy them back if and only if the preference of Reds towards Reds is strictly positive. In this latter case, proceed to the exchange: Blues in the middle can only improve by switching with an extremal Blue, which would never be accepted. This new arrangement is therefore stable.

- If Blues and Reds both strictly prefer their own kind, sit all Blues on one side and all Reds on the other. Let B_{out} be the extremal Blue, B_{in} be the only Blue with both a Blue and a Red neighbor R_{in}, and let R_{out} be the extremal Red. As previously, (B_{in}, R_{in}) is never a blocking pair. Suppose:
  - Preferences for their own kind are positive for both. We verify that (B_{in}, R_{out}) is not a blocking pair, as B_{in} would lose his only Blue neighbor. By symmetry, the only remaining possible blocking pair is (B_{out}, R_{out}): it is also not a blocking pair since both would only gain an "opposite colour" neighbor.
  - Preferences for their own kind are negative for both. Both B_{out} and R_{out} have maximum utility, hence B_{out} an R_{out} are not part of a blocking pair. Other Blues except B_{in} can only improve their utility by moving to an extremity, i.e., by switching with R_{out} or B_{out}, both being impossible. The only remaining possibility (B_{in}, R_{in}) is also not a blocking pair. Therefore, the arrangement is stable.
  - Preference of Blues for Blues is negative, but that of Reds for Reds is positive. Note that all Reds but R_{in} and R_{out} have maximum utility, hence cannot be part of a blocking pair. Moreover, since B_{out} has maximum utility, R_{in} could only switch with Blues having at least one blue neighbor, so it cannot improve. If Reds dislike Blues, R_{out} is also unable to improve, and the arrangement is stable. On the contrary, suppose Reds strictly like Blues, and consider the arrangement obtained after exchanging R_{out} and B_{in}: now both extremal Reds have the second highest utility, and cannot improve since no Blue is sitting between two Reds, so the arrangement is stable.

Hence, it is always possible to sit two classes in a stable manner on a path. □

A.4 Proofs Omitted From Section 6.2

In this section, we prove Theorem 9, which we restate below for convenience.

Theorem 9. *For $n \geq 4$, there exist three-valued three-class preferences such that all arrangements on a path are unstable. For $n \geq 12$, this can be achieved with non-negative values.*

We will prove the two parts as separate lemmas, as follows. We begin with the simpler construction, which uses negative preference values, but is otherwise essentially identical to that used in the proof of Theorem 8.

Lemma 14. *For $n \geq 4$, there exist three-valued three-class preferences such that all arrangements on a path are unstable.*

Proof. We consider the same instance of preferences as in the proof of Theorem 8, with minor modifications: now Bob's friends strictly dislike Alice, and Alice strictly dislikes Bob (strictly negative preferences). One can also make Bob strictly dislike everyone but Alice to achieve three-valued preferences, it is however not necessary for instability. See Figure 3 for an instance of such preferences. The analysis from Theorem 8 still stands, so we only need to study edge cases. Suppose Alice and Bob are sitting next to each other: since Alice now dislikes Bob, she would rather sit anywhere else than next to Bob, even if it is at one of the two extremities. Besides, even if Alice herself is at an extremity, there would always be someone agreeing to exchange with her: Bob's friend at the other extremity of the table (not a neighbor of Bob since $n \geq 4$). Now, suppose Alice and Bob are sitting apart. Bob would still want to sit next to Alice, even if it means being at the extremity of the table.

$$\mathcal{P} = \left[\begin{array}{cc|ccc} * & -1 & 1 & \dots & 1 \\ 1 & * & -1 & \dots & -1 \\ \hline -1 & 2 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ -1 & 2 & 1 & \dots & 1 \end{array}\right]$$

Figure 3: Possible instance of preferences for the proof of Lemma 14, with the ordering (Alice, Bob, Friends ...).

$$\mathcal{P} = \left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & 1 & \dots & 1 \\ 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ \hline 0 & 3 & 3 & 3 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 3 & 3 & 3 & 1 & \dots & 1 \end{array}\right]$$

Figure 4: Possible instance of preferences for the proof of Lemma 15, with the ordering (Alice, 1^{st} Bob, 2^{nd} Bob, 3^{rd} Bob, Friends ...).

Moreover, since his friends now dislike Alice, they would rather sit anywhere else than next to her, even if this means sitting at one end of the table. Alice's neighbor would therefore always agree to swap seats with Bob. □

To prove the version with non-negative preference values, the idea from the proof of Theorem 8 can still be used, but this time more insight is required. Most notably, we will make two copies of Bob. We explain the details below.

Lemma 15. *For $n \geq 12$, there exist three-class three-valued non-negative preferences such that all arrangements on a path are unstable.*

Proof. We consider the same instance of preferences as in the proof of Theorem 8, with the following modifications: we make two copies of Bob, and we suppose there are at least eight friends. Alice likes everyone but the Bobs, the Bobs only like Alice, the friends like the Bobs the most and Alice the least. We furthermore suppose that friends would rather be next to one Bob than between two other friends. Figure 4 displays a possible instance of such preferences.

Suppose one of the Bobs is sitting next to Alice. Since there are at least eight different friends, at least one of them is neither sitting beside a Bob nor at an extremity of the table. Indeed, at most five friends are sitting next to Bobs (since Alice sits next to one of them), and two more friends can be sitting at an extremity. Hence, Alice and this friend would both agree to switch places since it is always worth it for a friend to move beside a Bob, even if this means sitting at an extremity.

Now, suppose no Bob is sitting next to Alice. Since there are three Bobs, at least one of them is not sitting at an extremity, say B_1. The only case where one of Alice's neighbors would not agree to switch with B_1 is if it was already sitting next to another Bob, say B_2; moreover, the only reason for him not to switch with B_2 is if B_2 is sitting at the extremity of the table. Hence, the seating arrangement is of the form $(B_2,\ F,\ A,\ F,\ \dots,\ B_1,\ \dots)$. Alice is not sitting at an extremity, therefore possesses a second friend as a neighbor, and that second friend would agree to swap with at least one of the two remaining Bobs. □

A.5 Proofs Omitted From Section 6.3

In this section, we complete the proof of Theorem 10 (restated below for convenience) by proving that the constructed preferences induce no arrangement stable on a cycle.

Theorem 10. *For $n \geq 7$, there exist binary preferences using four classes such that no arrangement on a cycle is stable.*

Proof (continued). We now show that all arrangements on a cycle are unstable for our preference profile, by considering every possible local arrangement around c and showing that they all induce a blocking pair.

- If the local arrangement around c consists of $(b_1,\ c,\ b_2)$, then c has utility 0 and would agree to swap with anyone having a neighbor in D. Since D contains strictly more than two agents, one of the d_i neighbor of a has another member of D as a neighbor. Performing a swap with c would increase its utility from 1 to 2, while c would improve to utility 1: it is a blocking pair.
- If it consists of $(b_1,\ c,\ a)$, c has again utility 0 and would switch with whoever has a neighbor in D. In particular, if the local arrangement is $(b_1,\ c,\ a,\ b_2)$, then (b_1, c) is a blocking pair, as b_1 second neighbor is in D. If it is $(b_2,\ b_1,\ c,\ a)$, then (b_2, c) is a blocking pair, as b_2's second neighbor is in D. Otherwise, the local arrangement must be $(d_i,\ b_1,\ c,\ a,\ d_j)$, and (b_2, c) forms once again a blocking pair. The same reasoning naturally holds for local arrangements of the form $(b_2,\ c,\ a)$.
- If the local arrangement around c consists of $(d_i,\ c,\ a)$, then at least one member of B, say b_1, is no neighbor of a and has utility 0. In that case, both d_i and b_1 can increase their utility by exchanging seats.
- At last, if it consists of $(d_i,\ c,\ b_1)$ or $(d_i,\ c,\ d_j)$, then agent a has utility 0, and d_i can always increase its utility by switching with a. Indeed, if a was his neighbor, he would exchange c for a member of $D \cup B$ while retaining a as a neighbor; otherwise he would exchange c for a for a second neighbor in $D \cup B$. Since moving close to c would always increase a's utility, (d_i, a) is a blocking pair. The same reasoning of course holds for local arrangements of the form $(d_i,\ c,\ b_2)$.

□

A.6 Proofs Omitted From Section 6.3.1

In this section, we prove Lemma 13, which we restate below for convenience.

Lemma 13. *Consider the four-agent preference profile $\mathcal{P}_4$ in Figure 2a. A path stable arrangement exists, yet swap dynamics started from certain arrangements cannot converge.*

Proof. Let π^* be the arrangement $(a,\ b,\ c,\ d)$. Since each agent approves of exactly one other agent, they can never get utility strictly greater than one. Since only d does not achieve utility one in π^*, it is stable. Moreover, consider arrangements $\pi_1 = (a,\ d,\ b,\ c)$ and $\pi_2 = (c,\ d,\ b,\ a)$. The only blocking pair in π_1 is $(a,\ c)$ as both b and d have utility one. Exchanging them leads to arrangement π_2. Similarly, in π_2, the only blocking pair is $(b,\ d)$. Exchanging them gives π_1 back, up to reversal of the seat numbers. Hence, the swap dynamics cannot converge. See Figure 2b for an illustration. □

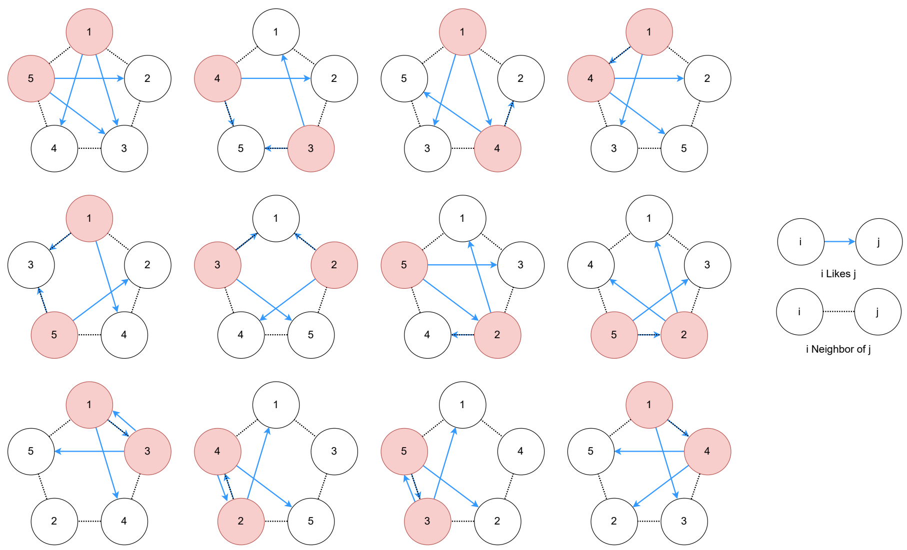


Figure 5: Graphical proof of the instability of $\mathcal{P}_5$.

B Stability Analysis of Profile $\mathcal{P}_5$

In this appendix, we briefly analyze the stability of profile $\mathcal{P}_5$ from Figure 1a, which is the only preference profile with $n = 5$ that is unstable on a cycle. Despite our best efforts, there does not seem to be an easy explanation for the emergence of instability in this case.

Lemma 16. *Binary preferences $\mathcal{P}_5$ induce no stable arrangements on a cycle.*

Proof. Figure 5 provides a graphical proof of the instability of $\mathcal{P}_5$. Each of the twelve possible cyclic arrangements is displayed and a blocking pair is shown in red. The outgoing edges of the nodes participating in the blocking pairs are shown in blue for effortless verification. □

C Evidence for Exponential Convergence

In this section, we revisit the potential function used to prove convergence of the swap dynamics in Section 6.3.1. In essence, we show that each increase in potential might be very small, leading to an exponential number of increases. Note, however, that we do not exhibit preference instances where such exponential behavior can be observed. Moreover, we also do not exclude the possibility that the function can be shown to increase enough with each swap on average when the dynamics are carried out. What we show, instead, is that there is an exponentially long chain of potential values such that between any two consecutive values in the chain there are preferences for which the transition could occur by exchanging a blocking pair.

Lemma 17. *The potential argument alone cannot guarantee polynomial time convergence of swap dynamics.*

Proof. We first consider sequences of the auxiliary potential $S(\pi)$ with values in $\{1, 2\}$ and study the effect of exchanges at distances one and two that keep the social welfare constant. To simplify notations, we map $S(\pi) \in \{1, 2\}^{n-1}$ to $S_b(\pi) \in \{0, 1\}^{n-1}$ by subtracting one.

Performing an exchange at distance one while keeping the social welfare constant corresponds to the following modification of a subsequence of $S_b(\pi)$: $0x1 \to 1\overline{x}0$, where $x \in \{0,1\}$ (see Figure 6). Indeed, an exchange at distance one modifies the utility of at most four people, hence modifies a subsequence of length at most three of the auxiliary potential, the latter being defined not on vertices but edges; we call this operation f_3.

$$f_3 : 0x1 \to 1\overline{x}0$$

Similarly, we define the operation corresponding to exchanging at distance two that keeps the social welfare constant: it maps the subsequence of $S_b(\pi)$ of length four $0xy1$ to $1\overline{y}\overline{x}0$. For the same reason as above, we call this operator f_4.

$$f_4 : 0xy1 \to 1\overline{y}\overline{x}0$$

In the following, we denote by "Apply f_i at position j" the application of f_i on the subsequence of indices $[j, j+i]$.

Based on those two operators, we further define the operator f_8 mapping the sequence of length 8 $(0,0,0,0,0,0,0,1)$ to $(1,0,0,0,0,0,0,0)$. It consists of the following operations:

1. Apply f_3 at position 6: $(0,0,0,0,0,0,0,1) \to (0,0,0,0,0,1,1,0)$.
2. Apply f_4 at position 3: $(0,0,0,0,0,1,1,0) \to (0,0,1,1,1,0,1,0)$.
3. Apply f_4 at position 2: $(0,0,1,1,1,0,1,0) \to (0,1,0,0,0,0,1,0)$.
4. Apply f_4 at position 4: $(0,1,0,0,0,0,1,0) \to (0,1,0,1,1,1,0,0)$.
5. Apply f_4 at position 3: $(0,1,0,1,1,1,0,0) \to (0,1,1,0,0,0,0,0)$.
6. Apply f_3 at position 1: $(0,1,1,0,0,0,0,0) \to (1,0,0,0,0,0,0,0)$.

For $k \geq 3$, we then recursively define the operators f_{3k-1} mapping the sequence of length $3k-1$ $(0,\ldots,0,1)$ to $(1,0,\ldots,0)$ through the following operations:

1. Apply f_3 at position $3(k-1)$: $(0,\ldots,0,1) \to (0,\ldots,0,1,1,0)$.
2. Apply $f_{3(k-1)-1}$ at position 2: $(0,\ldots,0,1,1,0) \to (0,1,0\ldots,0,1,0)$.
3. Apply $f_{3(k-1)-1}$ at position 3: $(0,1,0\ldots,0,1,0) \to (0,1,1,0,\ldots,0)$.
4. Apply f_3 at position 1: $(0,1,1,0,\ldots,0) \to (1,0,\ldots,0)$.

Note that this definition of $f_{3(k+1)-1}$ indeed ensures it acts on a subsequence of length $3k-1+3$. Moreover, it is correctly initialized since f_8 is defined independently.

Noting that operator f_{3k-1} contains more than 2^{k-1} uses of f_3 and f_4 for all $k \geq 3$ concludes the proof. □

D Non-Monotonicity of Stability

In this section, we show that stability is non-monotonic, i.e., adding agents to a given instance can both introduce or destroy stability. For instance, consider preferences $\mathcal{P}_5$ in Figure 1a. On a cycle, all binary preferences with either four or six agents possess a stable arrangement, implying that adding a fifth agent could destroy stability while adding a sixth agent would restore it. We now show that this phenomenon can occur for all values $n \geq 7$:

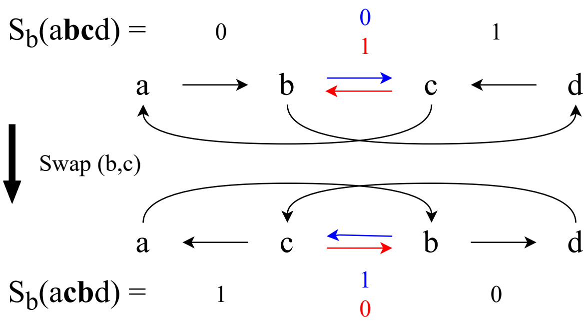


Figure 6: Change of $S_b(\pi)$ after an exchange at distance one preserving the social welfare.

Theorem 18. *For all $n \geq 7$, adding an agent can both destroy stability or restore it.*

Proof. We first prove that for $n \geq 7$ adding an agent can break stability. Consider the preferences of size $n+1$ in Theorem 10: we show that removing one agent creates a stable arrangement. Indeed, remove agent a and consider sitting c between b_1 and b_2: everyone but c has maximum utility (since b_1 and b_2 love only one person, they cannot get utilities higher than 1), hence c cannot exchange seats with anyone and the arrangement is stable.

We now prove that for $n \geq 7$ adding an agent can create stability. Consider the preferences of size n in Theorem 10, and add one extra agent $b_3 \in B$; i.e., which only likes a and c and is only liked by the members of D. Consider placing b_1, c, b_2, a, b_3 consecutively on the cycle in this order: agent b_2 and all members of D have utility 2, hence only b_1, b_3 or c can be part of a blocking pair. However, b_1 and b_3 both have utility one and would therefore only exchange for a seat with utility 2: there exists only one such seat, currently held by b_2, who has maximum utility and would hence not agree to swap places. As a result, neither b_1 nor b_3 are part of a blocking pair, so the arrangement is stable. □

E Blockwise-Diagonal Preferences

In this section, we show that, even when the preferences of the agents are, in a certain sense, highly decomposable, knowledge about the stability of the subparts is unlikely to help us to find stable arrangements for the instance as a whole. In essence, we show why non-monotonicity can make reasoning about stability rather challenging, even in reasonably simple cases. To make the previous statements precise, we introduce the concept of agent *components*, as follows.

Definition 19. *We say that a set of agents $\mathcal{C} \subseteq \mathcal{A}$ is isolated if for all $a \in \mathcal{C}$ and $b \in \mathcal{A} \setminus \mathcal{C}$, it holds that $p_a(b) = p_a(b) = 0$. Set $\mathcal{C}$ is called a component if none of its proper subsets are isolated.*

Note also that components and classes are two different notions: agents in the same component may not have the exact same preferences, but instead are limited to only caring about agents in their component. Assume that the set of agents $\mathcal{A}$ is partitioned into components $\mathcal{A} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_k$ (this partition always exists and is unique).[1] The preference matrix can then be represented, after a potential reordering of the agents, as a blockwise-diagonal matrix. When the partition into components is non-trivial; i.e., $k > 1$; intuitively, finding an arrangement that is stable for such preferences should be easier than for general

[1]This is because the components correspond to the connected components of the undirected graph with vertex set $\mathcal{A}$ and edges (a, b) for any two distinct agents such that either $u_a(b) \neq 0$ or $u_b(a) \neq 0$.

ones: first, find a stable arrangement on a path for each component, and then join all those paths to obtain a stable arrangement on a cycle (or on a path). While this method is indeed guaranteed to produce a stable arrangement whenever each component admits a path stable arrangement (at least for non-negative preference values), we will actually show that there are many instances where a stable arrangement exists but can not be produced by this approach. Before showing this, we need a technical lemma for cycles, stated next.

Lemma 20. *Let π be an arrangement where each agent sits between two agents from different components. If for any two distinct components $\mathcal{C}_i$ and $\mathcal{C}_j$ there is at most one pair of agents $(a, b) \in \mathcal{C}_i \times \mathcal{C}_j$ such that a and b are neighbors in π, then π is stable on a cycle.*

Proof. First, notice that in such an arrangement all agents get utility zero. Let $a \neq b$ be two agents. We want to show that they do not form a blocking pair. Assume $i, j \in [k]$ are such that $a \in \mathcal{C}_i$ and $b \in \mathcal{C}_j$. If $i = j$, then by assumption a would also have utility zero when seating in b's seat, so (a, b) is not a blocking pair. Now, assume $i \neq j$. Let a_ℓ and a_r denote the two neighbors of a on π; we define b_ℓ and b_r similarly. Suppose a and b wanted to switch place: this means a would have strictly positive utility at b's seat, and b would have strictly positive utility at a's seat. Assuming a and b are not neighbors, this means that $\{a_\ell, a_r\} \cap \mathcal{C}_j \neq \varnothing$ and $\{b_\ell, b_r\} \cap \mathcal{C}_i \neq \varnothing$. Let a' and b' be such that $a' \in \{a_\ell, a_r\} \cap \mathcal{C}_j$ and $b' \in \{b_\ell, b_r\} \cap \mathcal{C}_i$. Since (a, a') and (b', b) are both in $\mathcal{C}_i \times \mathcal{C}_j$ and are distinct pairs of adjacent agents in π, this contradicts the hypothesis. Furthermore, if a and b are neighbors, then b envying a implies that a's second neighbor is also in $\mathcal{C}_j$, from which the pairs of agents formed by a and its neighbors similarly contradict the hypothesis. Therefore, the arrangement is stable. □

Armed as such, we now show that any preference profile inducing no stable arrangements on a path can be used to construct preferences whose components all resemble this profile and yet the joint profile admits an arrangement that is stable on a cycle. In other words, even when a profile decomposes non-trivially into components, knowledge about the stability of the components does not necessarily help in resolving the stability of the instance as a whole.

Theorem 21. *Let $\mathcal{P}_{path} \in \{0, 1\}^{\ell \times \ell}$ be a one-component preference profile such that no stable arrangement on a path exists. Then, there exists a preference profile $\mathcal{P}$ whose components all resemble $\mathcal{P}_{path}$ admitting a stable arrangement on a cycle.*

Proof. We construct a larger blockwise-diagonal preference matrix $\mathcal{P} \in \{0, 1\}^{k \times k}$ by copying $\mathcal{P}_{path}$ a number $k = 2\ell + 1$ of times:

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{path}^{(1)} & & & 0 \\ & \mathcal{P}_{path}^{(2)} & & \\ & & \cdots & \\ 0 & & & \mathcal{P}_{path}^{(k)} \end{bmatrix}$$

Note that $\mathcal{P}_{path}^{(1)}, \ldots, \mathcal{P}_{path}^{(k)}$ naturally gives a partition into components $\mathcal{C}_1, \ldots, \mathcal{C}_k$; moreover, the choice of $\mathcal{P}_{path}$ immediately gives that all $\mathcal{C}_1, \ldots, \mathcal{C}_k$ are unstable on paths.

We now construct a cycle stable arrangement for $\mathcal{P}$. Since k is odd, graph K_k, which is the undirected clique graph of size k, possesses an Euler tour T where each vertex is visited ℓ times. We construct the arrangement π (more precisely π^{-1}) on the cycle by replacing every occurrence of i in T by an agent $a_i \in \mathcal{C}_i$, without repetition, starting for example from seat one. For the π we have just constructed, it then holds that the conditions to apply Lemma 20 are satisfied since the Euler Tour traverses each edge in K_k precisely once. By the lemma, π is a cycle stable arrangement of $\mathcal{P}$. □

F Stability of Random Binary Preferences

In this chapter, we conduct a study of stability using probabilistic tools. In particular, we employ the Lovász Local Lemma:

Lemma 22 (Lovász Local Lemma)**.** *Let $A_1, A_2, \ldots, A_k$ be a sequence of events such that each event occurs with probability at most p and is independent of all but at most d other events. If $epd < 1$, then the probability that none of the events occurs $P\left(\cap_{i=1}^{k} \overline{A_i}\right)$ is greater than or equal to $\left(1 - \frac{1}{d+1}\right)^k$.*

We now give a lower bound on the expected number of arrangements stable on a cycle when the preference graph is sampled from the Erdős-Rényi model $G(n,p)$ with average node degree either $O(\sqrt{n})$ or $n - O(\sqrt{n})$.

Theorem 23. *Suppose a binary preference graph $\mathcal{P}$ is drawn at random using the Erdős-Rényi model $G(n,p)$ with $p \leq Cn^{-1/2}$, where $C = (96e)^{-1/2}$.*
Then, the expected number of stable arrangements on a cycle is at least:

$$\frac{1}{2}(n-1)! \exp\left(-\frac{n(n-1)}{2n-3}\right).$$

The same results holds for $p \geq 1 - Cn^{-1/2}$.

Proof. Let $S = \sum_{\pi} S_{\pi}$ be the random variable counting the stable arrangements on a cycle for preference $\mathcal{P}$, where $S_{\pi} = 1$ if arrangement π is stable and 0 otherwise. Since all permutations of $\mathcal{P}$ follow the same distribution, all S_{π} have the same expectation and we only need to consider the identity arrangement $\pi = id$.
For $i, j \in [n]$, event L_{ij} corresponds to the i^{th} agent liking the j^{th} one, event E_{ij} to the i^{th} agent envying the j^{th} one, and event B_{ij} to agents (i,j) forming a blocking pair. Finally, let the random variable U_i denote the utility of the i^{th} agent. First, note that events E_{ij} and E_{kl} are independent for all $i \neq k$; from which $Pr(E_{ij}) = Pr(E_{ji})$. We therefore get:

$$\begin{aligned} Pr(B_{ij}) &= Pr(E_{ij} \cap E_{ji}) \\ &= Pr(E_{ij}) Pr(E_{ji}) \\ &= Pr(E_{ij})^2 \end{aligned} \tag{1}$$

By symmetry, we only have to calculate $(Pr(E_{1j}))_{2 \leq j \leq n}$.
If $j \in \{2, 3\}$:

$$Pr(E_{1j}) = Pr\left(E_{1j} \mid \overline{L_{1n}}\right) Pr\left(\overline{L_{1n}}\right) = p(1-p)$$

If $j \in \{n-1, n\}$:

$$Pr(E_{1j}) = Pr\left(E_{1j} \mid \overline{L_{12}}\right) Pr\left(\overline{L_{12}}\right) = p(1-p)$$

If $3 < j < n-1$:

$$\begin{aligned} Pr(E_{1j}) &= Pr\left(E_{1j} \mid U_1 = 0\right) Pr(U_1 = 0) + Pr\left(E_{1j} \mid U_1 = 1\right) Pr(U_1 = 1) \\ &= (2p(1-p) + p^2)(1-p)^2 + p^2 2p(1-p) \end{aligned}$$

Together with Equation (1), we subsequently get that:

$$Pr(B_{ij}) = \begin{cases} p^2(1-p)^2 & \text{if } |i-j| \leq 2 \\ \left(2p^3(1-p) + 2p(1-p)^3 + p^2(1-p)^2\right)^2 & \text{otherwise.} \end{cases} \tag{2}$$

Now, consider the family of events $(B_{ij})_{1\leq i<j\leq n}$. Note that each even B_{ij} is independent of events B_{kl} where $\{i,j\}\cap\{k,l\}=\varnothing$, but dependent of events with which it shares an index, so B_{ij} depends on at most $d=2(n-2)$ other events. Therefore, when $Pr(B_{ij})<\frac{1}{2e(n-2)}$, Lemma 22 gives:

$$\begin{aligned}\mathbb{E}[S_\pi]=Pr(S_\pi=1)&\geq\left(1-\frac{1}{2n-3}\right)^{\frac{n(n-1)}{2}}\\&\geq\exp\left(-\frac{n(n-1)}{2n-3}\right)\end{aligned}\tag{3}$$

By linearity of expectation, we finally get:

$$\mathbb{E}[S]\geq\frac{1}{2}(n-1)!\exp\left(-\frac{n(n-1)}{2n-3}\right)\tag{4}$$

It is only left to verify that $Pr(B_{ij})\leq\frac{1}{2e(n-2)}$. Note that B_{ij} has at most six terms all strictly smaller than $8p^2$. Hence, for $p\leq\frac{1}{\sqrt{96en}}\leq\frac{1}{\sqrt{96e(n-2)}}$, we have:

$$\begin{aligned}Pr(B_{ij})&<6\times 8p^2\\&\leq\frac{48}{96e(n-2)}=\frac{1}{2e(n-2)}\end{aligned}$$

Comparison with $6\times 8(1-p)^2$ instead gives the result for $p\geq 1-\frac{1}{\sqrt{96en}}$. $\square$

In practice, our result implies that, for random preferences of average out-degree at most $O(\sqrt{n})$, a naive approach sampling arrangements uniformly at random on average determines a stable arrangement using exponentially fewer samples than the theoretically required $(n-1)!/2$.

G Z3 Solver for Binary Preferences

In this section, we describe the Z3 Solver[2] employed in Section 6.3 to check whether all binary preferences for $n\leq 7$ are stable on paths and cycles. We only describe the case of cycles, as for paths it is enough to add one additional agent with null preferences from and toward all other agents.

Listing 1 shows the main body of the solver. In line 8, we introduce a function associating the boolean "i likes j" to each pair (i,j). In line 14, we define an array of n integers encoding the index of the agent placed in each seat; lines 17 to 24 constrain this array to be a permutation representing one of the $(n-1)!/2$ cycles. In lines 15, 16, and 26 we implement the main constraint: all arrangements on a cycle must induce a blocking pair.

Note that all simultaneous permutations of rows and columns of a solution are themselves solutions, as this corresponds to relabeling the agents. Therefore, it is desirable for efficiency to implement some kind of symmetry breaking. We do this in Listing 2 by requiring that the agents are sorted by the number of agents they approve of, breaking ties by the number of agents that approve them.

Finally, Listing 3 shows how to check whether agent-pair (i,j) is a blocking. Note that adjacent and non-adjacent seats require different treatments, and so lead to different logical expressions.

[2] `https://github.com/Z3Prover/z3`

Finally, note that Z3 returns either “Unsatisfiable” when no solutions exist, or “Satisfiable” and one solution otherwise. Finding all possible solutions is therefore rather tedious: after finding a solution, to get another one, we need to add a constraint that “the preferences are **not** these ones.” This detail is omitted for brevity.

```python
from z3 import *

N = 7

def UnstableCounterExample():
    s = Solver()
    # Pref(i, j): bool("Agent i likes agent j").
    Pref = Function('Pref', IntSort(), IntSort(), BoolSort())
    # Constrains Pref(i, i) = False.
    s.add([Not(Pref(i, i)) for i in range(N)])
    # Symmetry breaking on Pref.
    s.add(SymBreakSumRowCol(Pref))
    # Seat i is taken by agent Arr[i].
    Arr = [Int(f"Arr_{i}") for i in range(N)]
    s.add(ForAll(
        Arr, Implies(
            # Seat 0 is taken by agent 0.
            And(Arr[0] == 0,
                # Arr is a permutation.
                And([Arr[i] > 0 for i in range(1, N)]),
                And([Arr[i] < N for i in range(1, N)]),
                Distinct([Arr[i] for i in range(N)]),
                # Symmetry breaking on Arr.
                Arr[1] < Arr[N - 1]),
            # Agents at seats (i, j) form a blocking pair.
            Or([isBlockingPair(Pref, Arr, i, j) for i in range(1, N) for j in range(i)]))))
    SolveAndPrint(s)

def SolveAndPrint(s):
    print("Solving", s)
    val = s.check()
    if val == sat:
        print("Satisfiable", s.model())
    elif val == unsat:
        print("Unsatisfiable")
    else:
        print("Unknown")

UnstableCounterExample()
```

Listing 1: Main body of the solver.

```
def SymBreakSumRowCol(pref):
    """Returns True iff rows of pref have increasing sums, and in case of
    equality, the sums on the corresponding columns are increasing."""
    sums_row = [Sum([pref(i, j) for j in range(N)]) for i in range(N)]
    sums_col = [Sum([pref(i, j) for i in range(N)]) for j in range(N)]
    # First order on the sums of the rows.
    sum_row_ord = And([sums_row[i] <= sums_row[i + 1] for i in range(N - 1)])
    # Then order on the sums of the columns.
    sum_col_ord = And([Implies(sums_row[i] == sums_row[i + 1], sums_col[i] <= sums_col[i])
                       for i in range(N - 1)])
    return And(sum_row_ord, sum_col_ord)
```

Listing 2: Symmetry breaking of preferences.

```
def isBlockingPair(pref, arr, i, j):
    """Returns True if agents at seats i and j form a blocking pair.
    We suppose 0 <= j < i <= N - 1."""
    if j == i - 1:                  # Seats i and j adjacent.
        blockpair = And(Not(pref(arr[i], arr[(i + 1) % N])), pref(arr[i], arr[(j - 1 + N) % N]),
                                    # Agent at seat i envies agent at seat j.
                        Not(pref(arr[j], arr[(j - 1 + N) % N])), pref(arr[j], arr[(i + 1) % N]))
                                    # Agent at seat j envies agent at seat i.
        return blockpair
    if j == 0 and i == N - 1:       # Seats i and j adjacent.
        blockpair = And(Not(pref(arr[i], arr[(i - 1 + N) % N])), pref(arr[i], arr[(j + 1) % N]),
                                    # Agent at seat i envies agent at seat j.
                        Not(pref(arr[j], arr[(j + 1) % N])), pref(arr[j], arr[(i - 1 + N) % N]))
                                    # Agent at seat j envies agent at seat i.
        return blockpair
    else:                           # Seats i and j non adjacent.
        blockpair = And(Sum(pref(arr[i], arr[(i - 1 + N) % N]), pref(arr[i], arr[(i + 1) % N]))
                        < Sum(pref(arr[i], arr[(j - 1 + N) % N]), pref(arr[i], arr[(j + 1) % N])),
                                    # Agent at seat i envies agent at seat j.
                        Sum(pref(arr[j], arr[(j - 1 + N) % N]), pref(arr[j], arr[(j + 1) % N]))
                        < Sum(pref(arr[j], arr[(i - 1 + N) % N]), pref(arr[j], arr[(i + 1) % N])))
                                    # Agent at seat j envies agent at seat i.
        return blockpair
```

Listing 3: Testing for a blocking pair.