Equilibrium Existence in Schelling games

Akash Kumar Singh, Siddhant Kar

Supervisor: Dr. Sunil Simon

1 Introduction

Segregation is a well known social phenomenon. If we take a community of people of different social backgrounds, they tend to segregate over time into homogeneous sub-communities. Hence, these local strategic choices on the micro-level lead to its emergence on a macro-level. For this phenomenon, Schelling proposed a simple and elegant model.

In Schelling's model, there are two types of agents, A and B. An agent remains at its current position if least a fraction τ of its neighbours are of its type. If this condition isn't met, the agent jumps to some other location or swaps its position with a second unhappy agent. This experiment resulted in almost homogeneous sub-communities.

In Schelling's experiment, agents chose their new locations at random. We will discard that assumption by analysing a game-theoretic version of Schelling's experiment.

2 Models

2.1 Basic

We here discuss the most basic model discussed in the papers. We have a graph $G = (V, E)$ and a set of agents $A = \{1 \dots n\}$. The agents are partitioned into k types $\{T_1 \dots T_k\}$. In each stage of our game, we have an injective mapping $\pi \colon A \to V$ which assigns all agents to nodes of the graph.

In any assignment π, an agent x is happy if the ratio $\frac{|N_\pi^+(x)|}{|N_\pi^+(x)|+|N_\pi^-(x)|}$ is greater than or equal to the tolerance parameter τ. Here, $N_\pi^+(x)$ and $N_\pi^-(x)$ are the neighbours of x of the same and not the same type respectively. The cost for each agent is given by $\max(0, \tau - \frac{|N_\pi^+(x)|}{|N_\pi^+(x)|+|N_\pi^-(x)|})$.

The strategy space for each agent is the set of all nodes. A feasible strategy is therefore an assignment of the agents to the nodes. The agents change their strategies using the following rules:

1. In a Swap Schelling Game (SSG), two agents swap their positions only if each of them strictly decreases their cost.
2. In a Jump Schelling Game (JSG), an agent jumps to an empty node such that their cost strictly decreases.

We may sometimes use stubborn agents as well, ones who do not want to change their position and thus are fixed at certain nodes in the graph. We will mention explicitly when we use such agents.

Also, we take $\tau = 1$ by default. If we use some other τ, we will mention it explicitly. For $\tau = 1$, instead of the cost function, we can use the utility function

$$\mathbf{u}_\pi(v) = \frac{|N_\pi^+(x)|}{|N_\pi^+(x)| + |N_\pi^-(x)|}.$$

2.2 Social

We can have a more generalized setting, where instead of types, each agent is part of a social network. Our model consists of the following:

- Social network $H = (A, F, N)$, a directed graph where the set of all nodes is the set all agents A, F are the edges between friends and N are the edges between enemies.
- Placement graph $P = (V, E)$.
- Assignment $\pi : A \to V$.
- $n_i(\pi)$, the set of all neighbours of π_i in P.
- $N^+_\pi(i)$, the set of all friends of i which are in the neighbourhood of $\pi(i)$ in P.
- $N^-_\pi(i)$, the set of all enemies of i which are in the neighbourhood of $\pi(i)$ in P.
- $N_\pi(i)$, the set of all neighbours of i in P.
- $\pi^{i \leftrightarrow j}$, the assignment after agents i, j are swapped.

So, the tuple $I = (G, P, \pi, \mathbf{u})$ is called the Social Schelling game. We will discuss the two following variants:

1. Elkind's Social Schelling game [1]: Here, there are 2 type of agents, stubborn and strategic. There are no enemies in this model. Utility of a strategic agent i is given by

$$\mathbf{u}_\pi(i) = \frac{\left|N^+_\pi(x)\right|}{\left|N_\pi(x)\right|}.$$

2. Embedded Stubbornness Social Schelling game: We propose this model. Here, the utility of a node $\mathbf{u}^{'}_i : V \to \mathbb{R}$ is defined as

$$\mathbf{u}_\pi(i) = \alpha_i \frac{\left|N^+_\pi(x)\right| - \left|N^-_\pi(x)\right|}{\left|N_\pi(x)\right|} + \beta_i \mathbf{u}^{'}_i(\pi)$$

   $\mathbf{u}^{'}_i(\pi)$ does not depend on the placement of other nodes. There are no stubborn agents, but for each agent i we have a set of favourite nodes $S_i \subseteq V$. We define

$$\mathbf{u}^{'}_i(v) = \begin{cases} 1 & \text{for } v \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

   We can keep $\alpha_i + \beta_i = 1$ to reduce the number of variables, since it does not change the dynamics of the game.

We will usually refer to a basic Schelling game as a Schelling game. When a Schelling game is social, we will specify it and also assume it is the embedded hardness social Schelling game unless mentioned otherwise.

3 Prior Work

In this section, we present some relevant results and their prooef in our own words.

Theorem 1 ([2])**.** *If $\tau \leq \frac{1}{2}$, then 2-typed SSG is a potential game. And it is guaranteed to converge in $O(|E|)$.*

Proof. We will show that the ordinal potential function is

$$\Phi(\pi_G) = \frac{1}{2}\sum_{x \in A} \left| N^-(\pi_G(x)) \right| .$$

Let us say agents x and y swap with each other. Then they must be of different types, else they are not willing to swap. Since $\tau \leq \frac{1}{2}$ and they were willing, we have

$$N^-(\pi_G(x)) > N^+(\pi_G(x))$$

$$N^-(\pi_G(y)) > N^+(\pi_G(y)).$$

Also, for other agents which are neighbours of x and y, N^- will decrease by 1 for $N^-(\pi_G(x))+N^-(\pi_G(y))$ agents and will increase by 1 for $N^+(\pi_G(x))+N^+(\pi_G(y))$ agents. The total change in Φ is thus

$$\Phi(\pi'_G) - \Phi(\pi_G) = \frac{1}{2}(2 \times (N^+(\pi_G(x)) + N^+(\pi_G(y)) - N^-(\pi_G(x)) - N^-(\pi_G(y)))) < 0.$$

As we can see, the ordinal potential function decreases by at least 1 and it is bounded between $|E|$ and 0. hence, it must converge in at most $|E|$ swaps. □

Theorem 2 ([3])**.** *The SSG on a regular graph is a potential game. And it is guaranteed to converge in $O(|E|)$.*

Proof. Here as well, the ordinal potential function is

$$\Phi(\pi_G) = \frac{1}{2}\sum_{x \in A} \left| N^-(\pi_G(x)) \right| .$$

Since the graph is regular, the utility function is a linear. The rest of the proof can be done in a similar way as in Theorem 1. □

Theorem 3 ([4])**.** *In a k-type SSG, an equilibrium may not exist even when there are no stubborn agents and the graph is a tree.*

Proof. For $k = 2$, one can verify that the following graph will not have an equilibrium.

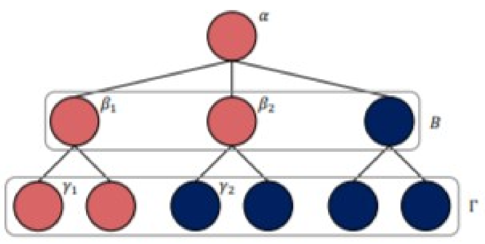

Figure 1: Construction for $k = 2$ [4].

For $k \geq 2$, there are $k(k^2 - 2)$ agents such that we have $k^2 - 2$ agents of each type. There will be $k(k-1) - 1$ nodes in B, and each node $b \in B$ will have k children, represented by the set Γ_b.

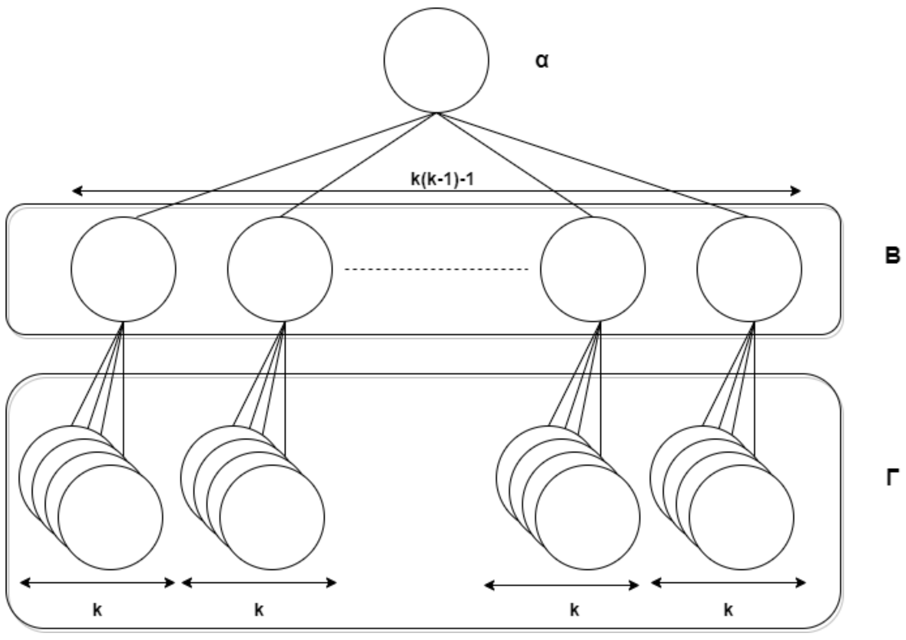

Figure 2: Construction for any $k \geq 2$.

Lemma 3.1. *For an assignment π, if 2 agents i and j in B are of same type, say T_x, and Γ_i and Γ_j have two nodes of same type T_y such that $T_x \neq T_y$, then it is not an equilibrium.*

Lemma 3.2. *If B is occupied by agents of every type, then it is not an equilibrium.*

Proof. Let the type of α be T_x. Then if for any node $\beta \in B$ of type T_x which has one child γ that is not of type T_x, γ and α can swap with each other.

If all the nodes in B which are assigned to agents of type T_x have all there children of type T_x, then there must be some agent i of type T_x in Γ such that its parent, p, is of the type $T_y, y \neq x$. If there is another agent j of type T_y in B, then i and j can swap with each other.

So for equilibrium, all the other agents occurring in Γ must be of type T_y. So, there can be at most $k - 1$ agents of type T_y in Γ_p. Hence, there are at least $k(k-1) - 2$ agents of type T_y which are placed in Γ whose parents are not of the type T_y, and also not T_x. Also, those agents of type T_y will have at least $k - 1$ parent in B. As we have $k - 2$ types left and $k - 1$ parents, at least two parents must have the same type. By lemma 3.1, we know that this is not an equilibrium. Hence, if B is occupied by an agent of every type, it is not an equilibrium. □

Lemma 3.3. *In an assignment, if there is a type T_i such that no agent of type T_i has been assigned a node in B, then it is not an equilibrium.*

Proof. Let agents of type T_x not occur in B. Then, there are at least $k^2 - 3$ agents of type T_x in Γ. Those $k^2 - 3$ will have at least k parents in B and we have $k - 1$ types to assign to those parents. So, at least 1 type must repeat. By lemma 3.1, this is not an equilibrium. □

Using lemma 3.2 and 3.3, we can say that in the given tree, no equilibrium exists. □

Theorem 4 ([4]). *It is NP-hard to decide whether a given k-type SSG has an equilibrium or not.*

Proof. We can check that the given assignment is an equilibrium in polynomial time, so it is member of the NP class. For NP-hardness, we will reduce it from the clique problem. The Clique problem is a decision problem in which we are given a graph $H = (X, Y)$ and an integer λ and have to decide whether it has a complete sub-graph of size λ or not. We will reduce Clique problem's instance to a 2-type (red and blue) SSG. d_H is the degree of graph H and d_v is degree of node $v \in X$.

Construction

- There will be λ strategic red agents and $|X| + 5$ strategic blue agents. All other agents are stubborn.
- The graph $G = (V, E)$ will be consist of 3 components, G_1, G_2, G_3:
  - $G_1 = (V_1, E_1)$. Every node v in X is first connected to a set of $2d_H - d_v + 2\lambda - 3$ nodes, say W_v. Then $V_1 = X \bigcup_{v \in V} W_v$ and $E_1 = Y \cup \big\{\{v, w\} : v \in X, w \in W_v\big\}$. Every node in G_1 has degree $d_1 = 2d_H + 2\lambda - 3$. In every W_v, d_H and $d_H - d_v + 2\lambda - 3$ nodes are occupied by stubborn red and blue agents respectively.
  - $G_2 = (A \cup B, E_2)$. It is a complete bipartite graph (A, B) in which $|A| = \lambda - 5$ and $|B| = 4d_1$. B is occupied by $2d_1 + 1$ stubborn red agents and $2d_1 - 1$ stubborn blue agents. So the strategic blue agents in A have utility $\frac{1}{2} - \frac{1}{4d_1}$ and strategic red agents have utility $\frac{1}{2} + \frac{1}{4d_1}$.
  - $G_3 = (V_3, E_3)$. We use the graph from figure 1, call it (V_3', E_3'). For every node v of degree 3, we connect $|Z_v|$ nodes to it, where Z_v is a set of $100d_1$ nodes, among which $50d_1$ are occupied by stubborn red agents and the other $50d_1$ by stubborn blue agents. And for every node of degree 1, we connect $|Z_v|$ nodes to it, where Z_v is a set of $10d_1$ nodes ($5d_1$ red and $5d_1$ blue). Then $V_3 = V_3' \cup_{v \in V_3'} Z_v$ and $E_3 = E_3' \cup \big\{\{v, w\} : v \in V_3', w \in Z_v\big\}$, so each strategic agent will get at least $\frac{5d_1 - 1}{10d_1 + 1} > \frac{1}{2} - \frac{1}{4d_1}$ and at most $\frac{5d_1 + 1}{10d_1 + 1} < \frac{1}{2} + \frac{1}{4d_1}$ utility.

Let there exist a clique of size λ, If we place all the strategic red agents in the clique and place all the blue agents in rest of the empty nodes, the utility of red strategic agents will be

$$u = \frac{\lambda - 1 + d_H}{d_1} = \frac{d_H + \lambda - 1.5 + 0.5}{2d_H + 2\lambda - 3} \geq \frac{1}{2} + \frac{1}{2d_1}$$

which is more then the maximum utility any red strategic agent can get in G_2 and G_3 components, and also more then what he gets by swapping with any other node in G_1. So, it is an equilibrium.

Let us take the case when H does not have a clique of size λ. Suppose there are some strategic red agents in G_1. Then, not every strategic red agent will have $\lambda - 1$ strategic red agents in his neighbourhood, otherwise it will be a clique. The utility that a strategic red agent i will get is

$$u_i \leq \frac{d_H + \lambda - 2}{d_1} = \frac{d_H + \lambda - 1.5 - 0.5}{2d_H + 2\lambda - 3} = \frac{1}{2} - \frac{1}{2d_1}.$$

Also, the utility of a blue strategic agent when he jumps to $\pi(i)$ will be $1 - u_i$. Hence, the red strategic agent can swap with any strategic blue agent in G_2 and G_3. This is not an equilibrium. Thus, for an equilibrium, G_1 must be occupied by only strategic blue agents. Now, if there is a node in G_2 that is occupied by a strategic blue agent, and in G_3 by a strategic red agent, then they will swap. So, for equilibrium, G_2 must be occupied by strategic red agents. Thus, there will be 4 strategic blue agents and 5 strategic red agents left in G_3. One can verify that with this configuration of G_3 there is no equilibrium. Hence, there is no equilibrium in the constructed SSG. □

Theorem 5 ([1])**.** *Given a k-typed JSG I with n agents, where G is a tree, we can decide whether I admits an equilibrium (and compute one if it exists) in time poly$\big(n^k\big)$, i.e., this problem lies in the complexity class* XP *with respect to the number of types k.*

Proof. In [1]. We skip this proof since we prove a similar result for the swap case (theorem 6). □

4 Our Contribution

In this section, we present our own results.

Theorem 6. *Given a t-typed Swap-Schelling game I with n agents, where G is a tree of degree d, we can decide whether I admits an equilibrium (and compute one if it exists) in time poly($n^{t^2d^2}$), i.e., this problem lies in the complexity class* XP *with respect to the number of types t and degree d of the graph.*

Proof. Let T_i denote the set of all agents of i^{th} type. Throughout the proof, we use the convention that a fraction of the form $\frac{a}{b}$ evaluates to 0 whenever $a = 0$.

Consider an instance I with n agents and a tree $G = (V, E)$. Pick an arbitrary node r to be the root of G. Let *tree*(v) denote the set of descendants of v (including v), and let *child*(v) be the set of children of v. Observe that the utility of a strategic agent takes values in the set $\mathcal{U} = \{i/j : i \in [d], j \in [d], i \leq j\} \cup \{0\}$; note that $|\mathcal{U}| \leq d^2$.

We use the following dynamic programming approach. For each node $v \in V$, we fill out a table τ_v, which contains an entry $\tau_v(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u})$ for each tuple $(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u})$, where

- $C \in [t]$,
- $C_p \in [t] \cup e$,
- $\mathbf{n} = (n_1, n_2, \ldots, n_t) \in [n]^t$,
- $\mathbf{k} = (\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_t) \in [n]^t$,
- $\mathbf{u} = (\mathbf{u}_{iju} : i \in [t], j \in [t], u \in \mathcal{U}, \mathbf{u}_{iju} \in \mathcal{U}) \in \mathcal{U}^{t^2|\mathcal{U}|}$.

The value of each entry is either *true* of *false*. Specifically, $\tau_v(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u}) =$ *true* if and only if there exists an assignment of a subset of agents to the nodes in *tree*(v) that satisfies the following conditions:

1. v is assigned to an agent of color C.

2. Parent of v is assigned to an agent of color C_p, if parent doesn't exist(in case of root node) C_p is assigned e.

3. Exactly n_i nodes of *tree*(v) are assigned to agents of i type.

4. Exactly $\mathbf{k}_i$ nodes of *child*(v) are assigned to agents of i type.

5. If agent of type j swaps with an agent of type i having utility u, then he will get a maximum utility of $\mathbf{u}_{iju}$ (where agent of type i is in $tree(v)\backslash\{v\}$ and agent of type j is not in the tree).

6. All agents in nodes of *tree*(v) do not have an incentive to deviate to nodes of *tree*(v).

Condition 6 directly relates to stability of *tree*$(v)\backslash\{v\}$, whereas conditions 1–5 are auxiliary, providing the necessary information that we need in order to determine the stability of node v, and fill out the dynamic programming table for the parent of v.

Consider the table τ_r at the root node r. The game admits an equilibrium if and only if there exists $(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u})$ such that $n_i = |T_i|$, $\forall i \in [t]$, $\tau_r(C, e, \mathbf{n}, \mathbf{k}, \mathbf{u}) =$ *true* for the root node r of G. The existence of a tuple $(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u})$ with these properties can be decided in polynomial time by going through all entries of τ_r. It remains to show that τ_r can be filled in in polynomial time.

Given C, we write $\mathbb{1}_i(C) = 1$ if $C = i$ and 0 otherwise, and $|\mathbf{k}| = \sum_{i=1}^{t} \mathbf{k}_i$. We fill the tables for all nodes starting from the leaf nodes of G.

For every leaf node v, we have

$$\tau_v(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u}) = \begin{cases} \textit{true}, & \text{if } \mathbf{n} = \left(\mathbb{1}_1(C), \mathbb{1}_2(C), \ldots, \mathbb{1}_t(C)\right), \\ & \mathbf{k} = (0, 0, \ldots, 0) \text{ and} \\ & \mathbf{u}_{iju} = 0 \forall i \in [t], j \in [t], u \in \mathcal{U} \\ \\ \textit{false}, & \text{otherwise.} \end{cases} \tag{1}$$

Suppose now that for a node w we have constructed the table τ_v for each $v \in$ *child*(w). We will construct τ_w using these tables as follows. Let *child*$(w) = \{v_1, \ldots, v_L\}$. We create an intermediate table θ_w^ℓ for each $\ell \in [L]$. This table has an entry $\theta_w^\ell(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u}, \mathbf{u}_\dagger, \hat{\mathbf{u}})$ for every tuple $(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u}, \mathbf{u}_\dagger, \hat{\mathbf{u}})$. $\mathbf{u}, \mathbf{u}_\dagger, \hat{\mathbf{u}}$ are explained below:

1. $\mathbf{u} = (\mathbf{u}_{iju} : i \in [t], j \in [t], u \in \mathcal{U}, \mathbf{u}_{iju} \in \mathcal{U}) \in \mathcal{U}^{t^2|\mathcal{U}|}$
   If agent of type j swaps with an agent of type i having utility u, then he will get a maximum utility of $\mathbf{u}_{iju\dagger}$ [agent of type $i \in$ *tree*$(w^\ell)\backslash$*child*(v)]

2. $\mathbf{u}\dagger = (\mathbf{u}_{iju\dagger} : i \in [t], j \in [t], u \in \mathcal{U}, \mathbf{u}_{iju\dagger} \in \mathcal{U}) \in \mathcal{U}^{t^2|\mathcal{U}|}$
   If agent of type j swaps with an agent of type i having utility u, then he will get a maximum utility of $\mathbf{u}_{iju\dagger}$ [agent of type $i \in$ *child*(w^ℓ)] $\mathbf{u}_\dagger$ is similar to $\mathbf{u}$, but it holds information for *child*(v) nodes only.

3. $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_{iu} : i \in [t], u \in \mathcal{U}) \in \mathcal{U}^{t|\mathcal{U}|}$
   It is the maximum utility w (the root node) will get if he swaps with one of his child node of type i having utility u.

Rest of the symbols has the same meaning as earlier.
If $\theta_w^\ell(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u}, \mathbf{u}_\dagger, \hat{\mathbf{u}})$ is true then there is an equilibrium state in w^ℓ, if root doesn't swaps.

We construct θ_w^ℓ sequentially for $\ell = 0, \ldots, L$. We can fill out θ_w^0 using Equation (1). Next, suppose that we have filled out the first ℓ tables, i.e., $\theta_w^0, \ldots, \theta_w^{\ell-1}$. We combine $\theta_w^{\ell-1}$ and τ_{v_ℓ} in order to build θ_w^ℓ as follows: $\theta_w^\ell(C, C_p, \mathbf{n}, \mathbf{k}, \mathbf{u}, \mathbf{u}_\dagger, \hat{\mathbf{u}}) =$ *true* if and only if there exist a pair of tuples $(C', C'_p, \mathbf{n}', \mathbf{k}', \mathbf{u}', \mathbf{u}'_\dagger, \hat{\mathbf{u}}')$ and $(C'', C''_p, \mathbf{n}'', \mathbf{k}'', \mathbf{u}'')$ such that $\theta_w^{\ell-1}(C', C'_p, \mathbf{n}', \mathbf{k}', \mathbf{u}', \mathbf{u}'_\dagger, \hat{\mathbf{u}}') = \tau_{v_\ell}(C'', C''_p, \mathbf{n}'', \mathbf{k}'', \mathbf{u}'') =$ *true* and the following conditions hold:

1. $C''_p = C' = C$.
2. $C'_p = C_p$.
3. $\mathbf{n}'' + \mathbf{n}' = \mathbf{n}$.
4. $\mathbb{1}_i(C'') + \mathbf{k}'_i = \mathbf{k}_i \; \forall i \in [t]$.
5. For each $i, j \in [t], u \in \mathcal{U}$,
$$\mathbf{u}''_{jil} \leq u \forall l < \mathbf{u}'_{iju},$$
$$\mathbf{u}''_{jil} \leq u \forall l < \mathbf{u}'_{iju\dagger}.$$
These two conditions will ensure that there is no swap possible between *tree*$(w^{\ell-1} \backslash$*child*$(w))$ and *child*(w).
6. If $i \neq C''$ or $u < \frac{\mathbf{k}''_{C''} + \mathbb{1}_{C''}(C')}{|\mathbf{k}''|+1}$,
$$\mathbf{u}_{iju\dagger} = \mathbf{u}'_{iju\dagger}.$$
Else,
$$\mathbf{u}_{iju\dagger} = max\left(\mathbf{u}'_{iju\dagger}, \frac{\mathbf{k}''_j + \mathbb{1}_j(C')}{|\mathbf{k}''| + 1}\right).$$
7. If $i \neq C''$ or $u < \frac{\mathbf{k}''_{C''} + \mathbb{1}_{C''}(C')}{|\mathbf{k}''|+1}$,
$$\hat{\mathbf{u}}_{iu} = \hat{\mathbf{u}}'_{iju}.$$
Else,
$$\hat{\mathbf{u}}_{iu} = max\left(\hat{\mathbf{u}}'_{iu}, \frac{\mathbf{k}''_C}{|\mathbf{k}''| + 1}\right).$$
8. For each $i \in [t]$,
$$\mathbf{u}_{iju} = max(\mathbf{u}'_{iju}, \mathbf{u}''_{iju}).$$

Combining states

These steps will be performed when $l = L$, to get $\tau_w(C, C_p\mathbf{n}, \mathbf{k}, \mathbf{u})$ from $\theta_w^L(C', C'_p, \mathbf{n}', \mathbf{k}', \mathbf{u}', \mathbf{u}'_\dagger, \hat{\mathbf{u}}')$. If at least one entry in $\theta_w^L(C', C'_p, \mathbf{n}', \mathbf{k}', \mathbf{u}', \mathbf{u}'_\dagger, \hat{\mathbf{u}}')$ that corresponds to $\tau_w(C, C_p\mathbf{n}, \mathbf{k}, \mathbf{u})$ is *true* then $\tau_w(C, C_p\mathbf{n}, \mathbf{k}, \mathbf{u})$ is true

1. $C = C'$.
2. $C_p = C'_p$.

3. $\mathbf{n} = \mathbf{n}'$.

4. $\mathbf{k} = \mathbf{k}'$.

5. First we will check if any agent can swap with the present root node. Let $u_i = \frac{\mathbf{k}_i - 1 + \mathbb{1}_i(C_p)}{|\mathbf{k}|+1}$, it is the utility if any agent of type i in *child*(v) will get if he swaps with v. For no swap to occur between v and *child*(v) following condition must be followed,

$$\hat{\mathbf{u}}_{it} \leq \frac{\mathbf{k}_C + \mathbb{1}_C(C_p)}{|\mathbf{k}| + 1} \forall t < u_i.$$

   And for no swap between v and *tree*$(v)\backslash$*child*(v), we must have

$$\mathbf{u}_{iCt} \leq \frac{\mathbf{k}_C + \mathbb{1}_C(C_p)}{|\mathbf{k}| + 1} \forall t < \frac{\mathbf{k}_i + \mathbb{1}_i(C_p)}{|\mathbf{k}| + 1}.$$

6. Merging $\mathbf{u}$ and $\mathbf{u}_\dagger$,

$$\mathbf{u}_{iju} = max(\mathbf{u}_{iju}, \mathbf{u}_{iju\dagger}).$$

□

Theorem 7. *Given Elkind's Social JSG I with n agents, where the placement graph is a tree, it is NP-hard to decide whether I has an equilibrium or not.*

Proof. We make a reduction from the Hamiltonian Path Problem. Given a graph $G' = (V', E')$, it is NP-complete to find the Hamiltonian path. We will take this graph and create our social network using it. There will be no enemy edges.

Agents and edges in social network

1. For each vertex in V' there is a strategic agent in A. Denote the set of all these agents as A'.

2. For each edge in E', there is an corresponding edge in the social network.

3. There are $4|V'|$ stubborn nodes. These are not friends with anyone. Denote the set of these agents as B.

4. There are $|V'| + 2$ stubborn nodes. These are friends with every agent in A. Denote the set of these agents as C.

5. For each $a \in A'$ there are:

   (a) 41 stubborn and 1 strategic agent who are friends with a and also with each other. Denote this set as D_a and the strategic agent as d_a, call them red agents. We will represent the set $\{d_a : a \in A'\}$ as d_A.

   (b) 61 stubborn agents who are not friends with a. Denote this set as E_a, call them blue.

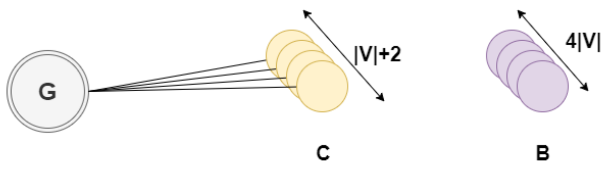

Figure 3: All nodes of G are connected to all nodes of C.

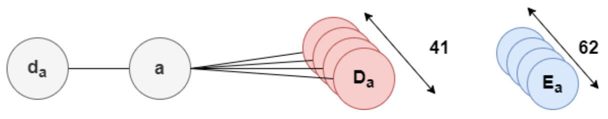

Figure 4: Each agent $a \in A'$ is then also connected to D_a and d_a.

Placement graph G

1. The first component G_1 consists of a path of $|V'| + 2$ length, whose start and end nodes are occupied by agents in C. And each node in the path is connected to 5 other nodes, among which 4 are from B and 1 from C. So, any agent $a \in A$ has minimum utility 1/7 and maximum 3/7.

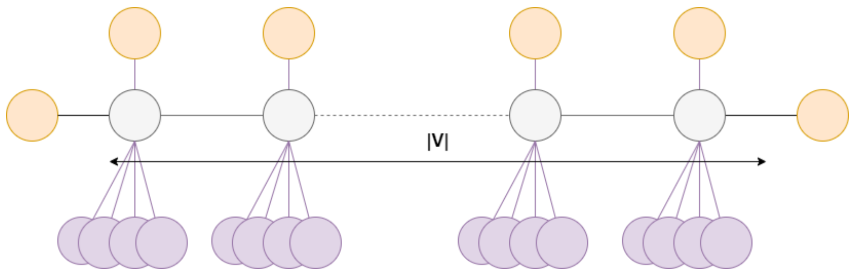

Figure 5: Component G_1: orange nodes have C, purple nodes have B.

2. Corresponding to each agent $a \in A$, there is a graph, say G_a, which has three empty nodes, denoted x, y, and z, and 102 nodes for stubborn agents ($D_a \cup E_a$). There is an edge between nodes x and y; also, x is connected to 4 red agents and 7 blue agents; y is connected to 8 red agents and 13 blue agents; and z is connected to 29 red agents and 41 blue agents. This graph has a similar structure to the one used in Theorem 4.2 of [1]. And a will get a utility between 1/3 $(2/7 < 1/3)$ and 5/12 $(5/12 < 3/7)$ in this graph.

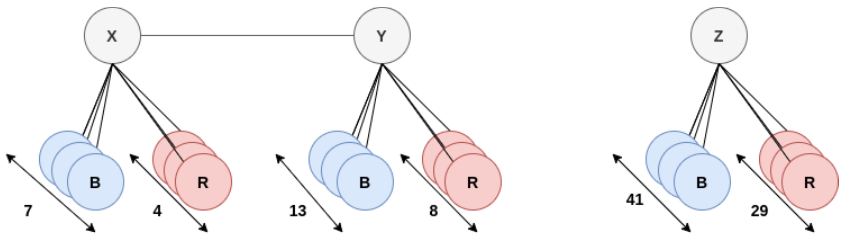

Figure 6: Component G_a.

Proof

1. If any agent $a \in A'$ is in $G_{a'}$ such that $a' \neq a$ and there is an empty node in either G_a or in G_1, then a will deviate there.

2. If strategic agent d_a is not in G_a and G_a has an empty node, then d_a agent will deviate there.

3. At any instant, there exist at least $|V'|$ empty nodes in all the G_a combined, i.e., at least $|V'|/3$ G_a will have at least 1 empty node.

4. If none of the agents in d_A were initially in their respective G_a, then at least $|V'|/3$ can jump to their corresponding G_a.

5. Let there be y agents of d_A in G_1. Then, G_a corresponding to those y agents must be completely occupied. In those y G_a components, there can be at most y agents from A'. Else, there will be at least one agent $a \in A'$ such that it is not in G_a or G_1 and there will be at least one empty node in G_1. This is not an equilibrium due to point 1.

   Let there be z $(z \leq y)$ G_a components that are completely occupied. This will require at least $2z$ agents from d_A. There will be at least z agents of d_A which are not in their corresponding G_a components, and those components have at least one node empty. Those agents can jump to their corresponding component. Hence, this is not an equilibrium.

6. If agent $a \in A'$ is in $G'_a : a' \neq a$, then there could be an empty node in G_1, so it is not an equilibrium due to point 1, or it will be occupied by some agent in d_A which is not an equilibrium due to point 5.

7. Let some agent in d_A be not in its corresponding G_a component. Then, its corresponding component will be completely occupied. Then, also by point 5, we can say that it will not be an equilibrium.

8. If a and d_a are in G_a and the remaining node is empty in G_a, then it is not an equilibrium state.

9. So, the only possibility for equilibrium is when a is in G_1 for all $a \in A'$, and each d_a is in its corresponding G_a at node z.

10. If both of the neighbours in G_1 of a are friends of a for all $a \in A'$. Then only all the agents $a \in A'$ will remain in G_1, which will be an Hamiltonian path.

11. The above construction was a forest. But, we can make it a tree by creating edges between stubborn agents of different components.

So, an equilibrium only exists when the placement in G_1 is the Hamiltonian path of G'. Also note that the maximum degree of the tree is also constant. So, an algorithm in $poly(n^d)$ is also not possible. Also, for a bipartite placement graph, we can make a reduction from 3-SAT. In that case, the diameter of the graph is constant. □

Theorem 8. *Given an Embedded Stubbornness Social JSG I with n agents, it is NP-hard to decide whether I admits equilibrium or not, even when the placement graph is a tree.*

Proof. This proof is similar to the proof of theorem 7. We again make a reduction from the Hamiltonian Path Problem. We will again take an instance of this problem and create our social network using it. Every agent i has $\alpha_i = 0.4$ and $\beta_i = 0.6$.

Agents and edges in social network

1. For each vertex in V' there is an agent in N, denote the set of all those agents as A'.
2. For each edge in E' there is an corresponding edge in the social network.
3. There are $4|V'|$ agents. These are not friends with anyone. Denote this set of these agents as B.
4. There are $|V'| + 2$ nodes. These are friends with every agent in A'. Denote the set of these agents as C.
5. For each $a \in A'$ there are:
   (a) 42 agents who are friends with a. Denote this set as D_a, call them red agents.
   (b) 61 agents who are not friends with a. Denote this set as E_a, call them blue agents.

The total number of agents is $|A|$.

4.1 Placement graph

1. The first component G_1 consists of a path of $|V'|$ length. Each node G_{1i} in this path is connected to $|A|$ nodes (denote it as $G_{1i}.s$).

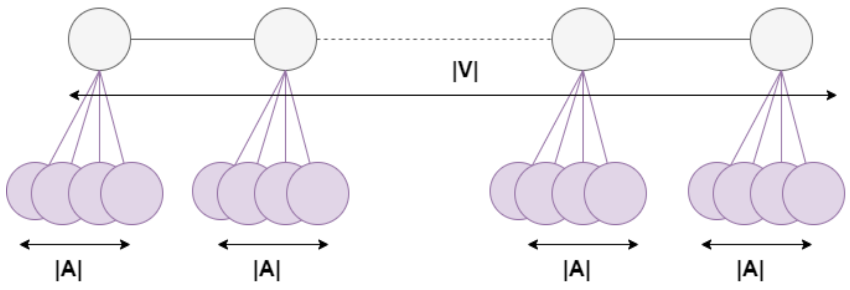

Figure 7: Component G_1.

2. Corresponding to each agent $a \in A'$, there is a graph G_a, which has three nodes x_a, y_a and z_a, and these 3 nodes are connected to $|A|$ other nodes (call them $x_a.s$, $y_a.s$ and $z_a.s$ respectively). There is an edge between the nodes x_a and y_a.

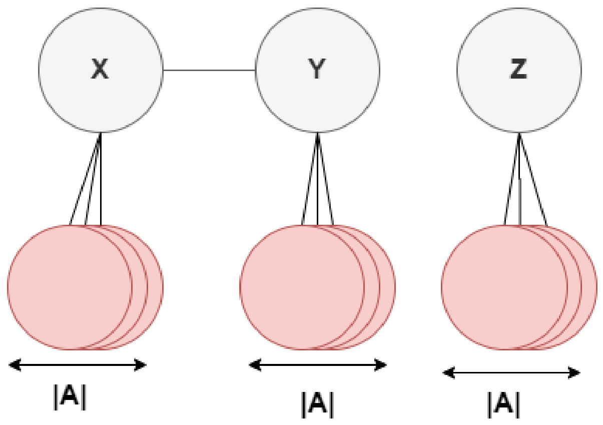

Figure 8: Component G_a.

Favourite nodes

1. Every agent a in A' has the whole path of G_1 and $\{x, y, z\}$ of G_a as favourite.
2. 1 agent of type C and 4 agents of type B have all the nodes in the set $\{G_{1i}.s\}$ as favourite, except when i is the start or end node of the path. $G_{1i}.s$ corresponding to these two i is a favourite node for 2 agents of type C and 4 agents of type B.
3. For each G_a component,
    (a) one red agent has x, y and z as favourite.
    (b) 4 red agents and 7 blue agents have $x_a.s$ as favourite.
    (c) 8 red agents and 13 blue agents have $y_a.s$ as favourite.
    (d) 29 red agents and 41 blue agents have $z_a.s$ as favourite.

Claim. If there is any agent that is not placed at one of its favourite node, then it is not an equilibrium.
Since $\alpha_i < \beta_i \; \forall i \in A'$, utility due to favourite nodes will dominate the strategic nature of agents.

- **Case 1:** Suppose any agent which has $x_a.s, y_a.s, z_a.s, G_{1i}.s$ as its favourite node is not placed at its favourite node. As all those agents have $|A|$ distinct nodes as their favourite nodes, at least 1 of the nodes must be empty. So, they can jump to that position.
- **Case 2:** Now, only those agents are left that were strategic in Theorem 7. Using the same proof, we can say that if any such agent is not at its favourite node, then it is not an equilibrium.

Also, using the same proof of theorem 7, we can say that an equilibrium can only exist when the corresponding input graph has a Hamiltonian path. □

5 Open Problems

- For a social SSG (both models), is it NP-hard to decide whether an equilibrium exists or not, even in case both the social network and placement graph are trees?
- For a basic Schelling game, is it NP-hard to decide whether an equilibrium exists or not, when there are no stubborn agents?
- Does an equilibrium always exist in case of Embedded Stubbornness Social SSGs, when the placement graph is regular?

References

[1] E. Elkind, J. Gan, A. Igarashi, W. Suksompong, and A. A. Voudouris, "Schelling games on graphs," 2019.

[2] A. Chauhan, P. Lenzner, and L. Molitor, "Schelling segregation with strategic agents," 2018.

[3] H. Echzell, T. Friedrich, P. Lenzner, L. Molitor, M. Pappik, F. Schöne, F. Sommer, and D. Stangl, "Convergence and hardness of strategic schelling segregation," 2019.

[4] A. Agarwal, E. Elkind, J. Gan, and A. A. Voudouris, "Swap stability in schelling games on graphs," 2019.