Understanding SpringRank through Random Utility Models, Identifiability, and Online Updates

by

Aparajithan Venkateswaran

A thesis submitted to the

College of Engineering and Applied Science at the

University of Colorado in partial fulfillment

of the requirements for the degree of

Bachelor of Science

Department of Computer Science

2020

This thesis entitled:
Understanding SpringRank through Random Utility Models, Identifiability, and Online Updates
written by Aparajithan Venkateswaran
has been approved for the Department of Computer Science

_____
Prof. Daniel Larremore

_____
Prof. Stephen Becker

_____
Prof. Manuel Lladser

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Venkateswaran, Aparajithan (B.S., Computer Science)

Understanding SpringRank through Random Utility Models, Identifiability, and Online Updates

Thesis directed by Prof. Daniel Larremore

A special class of complex systems arises when the agents involved are in competition with one another. The outcomes of these interactions make it possible to reveal a latent ordinal hierarchy of the entities in the system. SpringRank is a ranking method that models the network as a physical system comprised of springs. It estimates ranks as locations of the springs that minimize the total energy of the system.

This thesis explores SpringRank in three ways, two of which are extensions to the model. Firstly, we make connections between SpringRank and the Boltzmann distribution, Random Utility Models, and linear regression. We characterize SpringRank as a Random Utility Model by using the Boltzmann distribution to notice that SpringRank makes a decision between the choices itself as opposed to the choice items. We reconcile an ordinary least squares interpretation of SpringRank with the results from the Boltzmann distribution. We conclude that various interpretations of SpringRank offer the same insights but in different ways. Secondly, we extend SpringRank to identify the effect of group characteristics on the outcomes of interactions. We develop three models – two where group memberships are fixed and one where group memberships can change. The two models when group memberships are fixed correspond to cases where the group effects stay constant or are allowed to change. Both these models are non-identifiable – we cannot identify the effect of group characteristics. We recover identifiability in the third model. Finally, we propose an online update algorithm to accurately and efficiently update ranks inferred by SpringRank. Our algorithm forces boundary conditions and only updates a small neighborhood. This algorithm fails the accuracy-efficiency trade-off as it is computationally expensive. Instead, we suggest other possible online update mechanisms.

Dedication

To my sister.

Acknowledgements

I would like to thank my thesis committee, Stephen Becker, Manuel Lladser, and Dan Larremore for their advice and guidance. I am deeply indebted to my advisor, Dan, for the continuous support and innumerable opportunities he has given me over the years. It has been extremely rewarding and invaquably delightful to work with him. I also want to thank the Larremore and Clauset Labs for constantly challenging me to perform better science. I am especially thankful to Sam Way, Allie Morgan, and Ian Van Buskirk for valuable discourses and mentoring. Special thanks to Johan Ugander for providing crucial insights regarding SpringRank.

I also want to thank the Applied Math and Computer Science departments, faculty, and my advisors for their valuable instruction, beneficial advice, and helpful guidance throughout my undergraduate study. I would like to thank the Engineering Honors Program for fostering a community that gave me the opportunity to grow as an individual, and for always reminding me to ask the more important questions. Many thanks to Aaron Clauset, Liz Bradley, Anne Dougherty, Scot Douglass, Mary Rader, Raf Frongillo, Lesley McDowell, Jonathan Kish, Rhonda Hoenigman, Chris Ketelsen, and Tony Wong.

I would also like to thank all of my friends. I am thankful to Avery Anderson and Suyog Soti for our discussions, and shenanigans together. I am especially grateful to Ellen Considine for her friendship, and many intellectual and inspiring conversations. And, lastly, I would like to thank my family, especially my sister, for their unwavering support through every endeavor, without which I would not be where I am today.

Contents

Chapter

Figures

Figure

Chapter 1

Introduction

Complex systems are systems with relatively simple components whose behavior is difficult to model due to dependencies, interactions, competitions and other types of relationships. Complex systems are ubiquitous. They are comprised of agents that interact with one another, either individually or as agglomerates. These interactions sustain and lend novelty to the entire system. For instance, ecosystems form a complex system with species interacting with each other. Even within an ecosystem, a single species could form a complex system. Social behavior of humans is one such example. Other examples of complex systems include the metabolic network, Earth's climate system, the human brain, and the entire universe. Complex systems can also be artificially constructed such as electrical power grids, transportation systems, and the world wide web.

A key factor driving such systems is the interaction between different components. A special class of complex systems arises when the agents involved are in competition with one another. The nature of those interactions reveals hierarchy – the outcomes are correlated with their positions in the hierarchy. The hierarchies can be naturally occurring, such as in animals like birds, primates and elephants that tend to organize themselves according to dominance hierarchies [Drews, 1993]. There is also evidence of such latent hierarchies in anthropological systems where hierarchies result from prestige, reputation, and social position [Power, 2017, Clauset et al., 2015]. In certain scenarios, such as sports tournaments, we construct systems by forcing interactions between entities for the sole purpose of identifying the rankings [Szymanski, 2003, Baumann et al., 2010]. Finally there are instances where the hierarchy arises as a byproduct of the construction of the system such as the

world wide web [Page et al., 1999].

From these outcomes, it is possible to reveal the latent rankings of the components, or entities, in the system. It is desirable to infer the rankings as they form the basis of strategies in complex systems. The outcome of interactions between entities give evidence to reveal these hierarchies. However, in some systems, even the existence of an interaction provides useful information to determine the rankings. For example, sports tournaments are designed to match players of similar skill levels [Szymanski, 2003, Baumann et al., 2010]. Therefore, we can infer rankings based on the existence and the outcomes of interactions. Further, inferring an *embedding* of hierarchies can be more useful than an ordinal hierarchy as it can reveal relative differences of the positions of entities in the hierarchy.

We are interested in identifying the latent hierarchies of entities in complex systems. This is the goal of ranking methods. Using observed data, often in the form of a directed network, they aim to discover the hierarchy that could have generated the data. Since complex systems appear throughout the world, there are many ranking methods that are unique to fields (or applications), each with their own advantages and disadvantages. There is a family of spectral methods that rank nodes in a network by performing random walks and calculating their stationary distribution [Bonacich, 1987, Page et al., 1999, Negahban et al., 2017]. There is another family of methods that generate ordinal rankings by minimizing penalty functions [Slater, 1961, Ali et al., 1986, Gupte et al., 2011]. There are ranking methods that compute ranks using proportions of wins and losses [Elo, 1978, Herbrich et al., 2007, Coulom, 2008]. There are also Random Utility models that infer real valued ranks from pairwise comparison data [Bradley and Terry, 1952, Train, 2009].

SpringRank is one such ranking method [De Bacco et al., 2018]. In SpringRank, the system is represented as a directed weighted network, where the existence of an edge between two nodes indicates the outcome of a single interaction between them. The direction of the edge $i \to j$ is related to the outcome of the interaction and suggests that i is ranked above j. This model treats the network as a system of oriented springs, where each spring represents an edge in the network,

and aims to minimize the total energy of the system. The optimal locations of the springs are mapped to the ranks of the nodes.

In this thesis, we will explore the SpringRank model in three distinct ways. Firstly, we will gain a deeper insight into SpringRank including why and how it works. In particular, we will make important connections between SpringRank and the Boltzmann distribution, Random Utility Models, and linear regression. This will also help us place SpringRank in the larger body of literature concerning these fields, and reconcile the various interpretations.

Secondly, we will extend the SpringRank model to identify the effect of group characteristics on the outcomes of the interactions. There are scenarios where some nodes may enjoy advantages over other nodes due to their group memberships. For instance, chess players may gain an advantage when they play White (or Black). In such cases, we are interested in distinguishing the "skill" of the player from the "color advantage" they get from playing White (or Black). We will develop three models to answer this question. In the process, we will address the problem of identifiability, the ability to distinguish the skill and group advantage, and state conditions under which the model is identifiable.

Thirdly, we will extend the SpringRank model to accurately and efficiently update ranks as new interactions occur over time. As we will see, SpringRank can be computationally expensive. We will explore a potential algorithm to address the problem and study its viability.

The rest of this thesis follows this natural organization: Chapter 2 formally introduces SpringRank and is devoted to the first exploration to gain a deeper understanding of SpringRank. Chapter 3 will deal with the second exploration (and the first extension) to identify the effect of group characteristics on outcomes of interactions. In Chapter 4, we will concern ourselves with the third exploration (and the second extension) to update ranks continuously as new information comes to light. Finally, Chapter 5 will conclude our findings and discuss possible directions for future research.

Chapter 2

SpringRank

SpringRank is a physical model for inferring a hierarchical ranking of nodes in a directed network [De Bacco et al., 2018]. Section 2.1 is dedicated to describing the model in detail following the original work [De Bacco et al., 2018]. The rest of this chapter is devoted to gaining a better understanding of SpringRank and how it works. Section 2.2 explores the connection between SpringRank and the Boltzmann distribution, that is briefly mentioned in [De Bacco et al., 2018]. Section 2.3 motivates the study of SpringRank as a Random Utility Model and helps understand the position of SpringRank in the larger body of literature concerning pairwise choice comparisons. Finally, Section 2.4 addresses the resemblance of SpringRank to ordinary least squares.

2.1 The original model

Consider a system with N nodes (players, individuals, universities, etc.) that interact with each other. Let A_{ij} describe the number of interactions between nodes i and j that suggest i is ranked above j. In a sports tournament, this might be the number of times i beat j when they faced off against each other. In faculty hiring, this might be the number of Ph.D. students university j hired from university i (therefore tending to suggest that university j endorses the quality of training university i gives its students). In online dating, this might be the number of times i and j send a first message to a person but only i gets a reply back. Using this data about the outcomes of many such interactions we can construct a weighted directed network A, where the weight is given by A_{ij}.

Under this setup, ranking algorithms attempt to infer the latent hierarchical ranking of nodes that generated the observed data. This is the goal of SpringRank.

2.1.1 SpringRank

SpringRank models the ranks as the optimal location of the nodes in a physical system. Each node i is embedded at location (rank) s_i, and each interaction $i \to j$ is modeled as a spring with displacement $s_i - s_j$. We are free to scale the embedding space and the energy space. So, without loss of generality, if we set the rest length and the spring constant to 1, the spring corresponding to the edge $i \to j$ has energy

$$H_{ij} = \frac{1}{2}(s_i - s_j - 1)^2, \tag{2.1}$$

which is minimized when $s_i - s_j = 1$. The original model explicitly refrains from allowing tunable parameters, and parameters that can be inferred (or chosen *a priori*) such as spring constants and rest lengths for different springs.

The total energy of the system is given by the Hamiltonian

$$H(\boldsymbol{s}) = \sum_{i,j=1}^{N} A_{ij} H_{ij} = \frac{1}{2} \sum_{i,j=1}^{N} A_{ij}(s_i - s_j - 1)^2. \tag{2.2}$$

The optimal rankings, $\boldsymbol{s}^* = (s_1^*, s_2^*, \ldots, s_N^*)$, correspond to ranks (locations) that minimize the energy of the system. We minimize the Hamiltonian by noticing that it is convex in $\boldsymbol{s}$. Therefore, we find the optimal ranking of the nodes, $\boldsymbol{s}^*$, by setting $\nabla H(\boldsymbol{s}) = 0$. This gives us the linear system

$$\Big[D^{\text{out}} + D^{\text{in}} - (A + A^T)\Big]\boldsymbol{s}^* = \Big[D^{\text{out}} - D^{\text{in}}\Big]\mathbf{1},$$

where $\mathbf{1}$ is vector of all-ones, and D^{out} and D^{in} are diagonal matrices whose entries are the weighted out- and in-degrees, $D^{\text{out}}_{ii} = \sum_j A_{ij}$ and $D^{\text{in}}_{ii} = \sum_j A_{ji}$. $D^{\text{out}} + D^{\text{in}} - (A + A^T)$ is the Laplacian of the network A. The derivation of this solution is omitted here and can be found in [De Bacco et al., 2018]. However, two other derivations that follow a similar flavor are shown in Sections 3.2.2 and 3.3.2.

For succinctness of notation, we will denote the Laplacian by L and $(D^{\text{out}} - D^{\text{in}})\mathbf{1} = \hat{\boldsymbol{d}}$. Further, we will refer to the optimal solution simply by $\boldsymbol{s}$ from this point. So, we can rewrite the

system as

$$L\boldsymbol{s} = \hat{\boldsymbol{d}}. \tag{2.3}$$

By construction, L is rank deficient. If there are k disconnected components in the undirected network of A, $\text{rank}(L) = N - k$. Therefore, $\text{rank}(L) \leq N - 1$. And, $\mathbf{1}$ resides in the kernel of L. This means that the family of solutions $\boldsymbol{s}$ is translation-invariant i.e., translating the ranks by a constant yields another valid solution. In practice, we solve this system by fixing the position of an arbitrary node $s_N = 0$, and compute the remaining ranks in an iterative fashion.

SpringRank assigns real-valued ranks to nodes in an embedding. This gives interpretability to ranks. Further, SpringRank assumes that interactions are more likely to occur between similarly ranked nodes. This assumption is discussed in detail in Section 2.3.1.

2.1.2 Other ranking methods

As previously noted, ranking is not a problem unique to one field and arises in various contexts. As a consequence, a plethora of ranking methods exists each with their own advantages and disadvantages. In this section, we will attempt to discuss some other ranking methods and explain how SpringRank differs.

There is a class of spectral ranking methods like Eigenvector Centrality [Bonacich, 1987], PageRank [Page et al., 1999], the method of Callaghan [Callaghan et al., 2003], and Rank Centrality [Negahban et al., 2017]. The ranks produced by these methods are given by stationary distributions of different types of random walks. However, by design, they tend to give high ranks to a small number of important nodes, and provide little information about the lower-ranked nodes. In contrast, SpringRank provides an embedding of the ranks and relative distances between the ranks of two nodes are interpretable.

A second class of methods such as Minimum Violation Rank [Slater, 1961, Ali et al., 1986, Gupte et al., 2011], SerialRank [Fogel et al., 2016], and SyncRank [Cucuringu, 2016] propose ordinal rankings that minimize various penalty functions. For instance, Minimum Violation Rank (MVR)

imposes a uniform penalty for every violation, defined as an edge that has a direction opposite to the one expected by the rank difference between the two nodes. Non-uniform penalties and other generalizations are often referred to as agony methods [Letizia et al., 2018]. The problem with ordinal rankings is, once again, interpretability.

Random Utility Models [Train, 2009] are designed to infer real-valued ranks from pairwise comparison data. Bradley-Terry-Luce (BTL) model [Bradley and Terry, 1952, Luce, 1959] is one such model. They assign utilities to various choices and define a probability, based on this utility, to the direction of an edge conditioned on its existence, but they do not assign a probability to the existence of an edge. They find applications in instances when an experimenter presents subjects with choices between pairs of items, and asks them which they prefer. They are commonly used in, but not limited to, economics. Section 2.3 is devoted to studying SpringRank as a Random Utility Model.

Another class of methods like David's Score [David, 1987] and the Colley matrix [Colley, 2002] compute rankings from proportions of wins and losses. Widely used win-loss methods such as Elo score [Elo, 1978], TrueSkill [Herbrich et al., 2007], and Go Rank [Coulom, 2008] update ranks after every match rather than taking all previous interactions into account. This specialization makes them useful when ranks evolve over sequential matches, but less useful otherwise. In Section 2.3, we briefly discuss how SpringRank is similar to one such method called Elo++ [Sismanis, 2010].

The methods highlighted in this section illustrate the range of techniques used to construct rankings from pairwise interaction data. Though many others exist, we now focus on some of the more popular approaches, comparing each one to SpringRank

2.2 SpringRank and the Boltzmann distribution

Consider a thermodynamic system consisting of multiple particles. The Boltzmann distribution gives us the probability of each microstate (a configuration of the particles) existing in the system as a function of its energy. In particular, at temperature T, if a microstate x has energy E_x, the

probability of this microstate existing is given by

$$P(x) \propto e^{-E_x/kT},$$

where k is the Boltzmann constant. To normalize the probability, we use the *canonical partition function*,

$$Q = \sum_{j \in \mathcal{X}} e^{-E_j/kT}$$
$$\implies P(x) = \frac{1}{Q} e^{-E_x/kT}$$

where $\mathcal{X}$ is the set of all possible microstates.

SpringRank uses the Boltzmann distribution to make predictions of outcomes. This intuitively makes sense because SpringRank is a physical system where each spring contributes to the total energy of the system, which we wish to minimize. In SpringRank, (oriented) springs constitute the microstates whose energy is given by the corresponding Hamiltonian contribution. For instance, take a spring oriented from $i \to j$. According to the Boltzmann distribution, the probability of this microstate is

$$P(i \to j) \propto e^{-\beta H_{ij}}$$

where $\beta = 1/kT$ is the inverse temperature and $H_{ij} = \frac{1}{2}(s_i - s_j - 1)^2$. Finally, since the Boltzmann distribution maximizes entropy of the system, SpringRank can be thought of as a maximum entropy model.

Consider an instance where i and j interact and there is only one winner. There are two possible outcomes: (1) i beats j, or (2) j beats i. These two outcomes correspond to two possible microstates: (1) a spring oriented $i \to j$, or (2) a spring oriented $j \to i$. Without loss of generality, we can compute the probability that i beats j, $P(i \to j)$, as

$$P(i \to j) = \frac{e^{-\beta H_{ij}}}{e^{-\beta H_{ij}} + e^{-\beta H_{ji}}} \tag{2.4}$$
$$= \frac{1}{1 + e^{-2\beta(s_i - s_j)}} \tag{2.5}$$

2.3 SpringRank and Random Utility Models

Now, let us take a step back and consider Random Utility Models (RUMs) [Train, 2009]. In this scenario, an agent is given multiple items to choose a single item from. Each item x has an observed utility u_x for the agent. Each item also has an unobserved utility, ϵ_x for the agent. The agent tries to maximize the utility. Random Utility Models model the decision of the agents.

According to the Bradley-Terry-Luce model (BTL) [Bradley and Terry, 1952, Luce, 1959], a commonly used RUM, the probability that the agent chooses an item x from a set of items $\mathcal{X}$ is given by the multinomial logit distribution,

$$P(x) = \frac{e^{u_x}}{\sum_{j \in \mathcal{X}} e^{u_j}}.$$

Multinomial logit model assumes that the unobserved utility is drawn from a Gumbel distribution [Train, 2009]. If the agent is presented only two choices, i and j, the probability that the agent chooses i over j is given by

$$\begin{aligned} P(i \to j) &= \frac{e^{u_i}}{e^{u_i} + e^{u_j}} \\ &= \frac{1}{1 + e^{-(u_i - u_j)}} \end{aligned} \tag{2.6}$$

Observe that the predictions from SpringRank (Equation 2.5) and RUM (Equation 2.6) both follow the logistic distribution. In fact, we can interpret SpringRank as a Random Utility Model.

In SpringRank, if we know that i and j interact with each other (and that exactly one of them will win), there are two possible microstates – two directed springs. So, in the language of RUMs, the choice is actually between the two directed springs, where the utility of each spring is given by $-\beta H_{ij}$ and $-\beta H_{ji}$ respectively. RUMs consider the utility of the *choice items*, u_i, u_j whereas SpringRank considers the utility of the *choice* itself $-\beta H_{ij}, -\beta H_{ji}$. In other words, SpringRank recasts the *springs* as the *choice items*. The spring $i \to j$ corresponds to the statement (decision) that i beats j.

Note that, we can add a constant to all of the utilities, and the results (probabilities and decisions) remain the same. So, to make the utilities positive in SpringRank, we can add the same

constant to all the utilities without changing the results. Although this is worth observing, we will not do that.

A missing piece of this interpretation is the unobserved utilities in SpringRank. This question is addressed in Section 2.5.

2.3.1 The quadratic utility

SpringRank models the utilities as a quadratic function. This has three interesting implications in the generative model. Firstly, SpringRank operates differently in how it places edges in the network. Assume that all ranks s_i are fixed. We wish to add a new edge (a *spring*) to the system. The probability of this new edge is given by the Boltzmann distribution,

$$P(i \to j) \ \propto \ e^{-\beta H_{ij}}.$$

According to this model, there is a high probability of adding an edge between i and j when $e^{-\beta H_{ij}}$ is large i.e., when H_{ij} is small. H_{ij} is small whenever i and j are close to each other. Therefore, SpringRank chooses to add edges between choices that are similarly positioned (ranked). And the model is forced to compromise when we need to make a choice between two items that are very far apart.

This is a well-motivated assumption. Sports tournaments and leagues are often designed to match players or teams based on similar skills [Szymanski, 2003, Baumann et al., 2010]. Further, [Hobson and DeDeo, 2015] find that this is also the case in parakeets' pecking order. Newly introduced birds have one bit of information on the other birds: are you below me or above me in the hierarchy? However, once they settle, after about a week, they start targeting just a little down the hierarchy. From the bird cognition point of view, this suggests that they can now estimate more than one bit about the other birds: how far above or below me are you? This is exactly what SpringRank is asking when placing new edges.

Secondly, the quadratic utility is that SpringRank functions, doubly, as regularization. In the absence of other information, SpringRank will place things near each other. RUMs like BTL

maximize the separation as much as possible. This prevents the ranks from diverging to $\pm\infty$. This regularization can also be thought of as a consequence of the maximum entropy framework provided by the Boltzmann distribution.

Finally, due to the coupled utility, SpringRank's requirement for inference is that the undirected graph needs to be connected. On the other hand, BTL requires the directed graph to be strongly connected.

2.3.2 SpringRank as a Context Dependent Random Utility Model

The Context Dependent Random Utility Model (CDM) [Seshadri et al., 2019] is an extension to the multinomial logit model that allowed for a general choice system in which the utility of an item is uniquely expanded into contextual utilities,

$$\begin{aligned} u(x \mid C) &= \sum_{B \subseteq C \setminus \{x\}} v(x \mid B) \\ &= \underbrace{v(x)}_{\text{1st order}} + \underbrace{\sum_{y \in C \setminus \{x\}} v(x \mid \{y\})}_{\text{2nd order}} + \cdots + \underbrace{v(x \mid C \setminus \{x\})}_{|C|\text{th order}} \end{aligned}$$

where C is a subset of $\mathcal{X}$ that has more than two elements. Here the pth order terms represent the contextual contributions to the utility that is not already modeled by lesser order terms. By truncating this expansion at order p, they develop a class of models called $\mathcal{M}_p$. This leads to $n-1$ classes of models: $\mathcal{M}_1 \subset \mathcal{M}_2 \subset \ldots \mathcal{M}_{n-1}$ where $\mathcal{M}_{n-1}$ is the universal logit model [McFadden et al., 1977]. Further, $\mathcal{M}_1$ is the multinomial logit model:

$$P(x \mid C) = \frac{e^{v(x)}}{\sum_{y \in C} e^{v(y)}}$$

They refer to $\mathcal{M}_2$, the minimal model that accounts for context effects, as the *context dependent random utility model*. In particular, they re-parametrize the utility as $u_{xz} = v(x \mid \{z\}) - v(x)$ as the "pairwise push and pull of z on x's utility" and rewrite the choice probability as

$$P(x \mid C) = \frac{\exp(\sum_{z \in C \setminus \{x\}} u_{xz})}{\sum_{y \in C} \exp(\sum_{z \in C \setminus \{x\}} u_{yz})}.$$

Notice that SpringRank is exactly a CDM when $C = \{x, y\}$. SpringRank cannot say anything about an arbitrarily sized C as the SpringRank choices are always between directed springs between two nodes. In the language of CDM, $v(x) = \beta s_x$ and $v(x \mid \{y\}) = -\frac{\beta}{2}(s_y^2 + s_x^2 - 2s_x s_y + 2s_y + 1)$ are the "pairwise push and pull".

The authors note that the pairwise contextual utilities u_{xy} can be modeled by a lower-dimensional parameterization, defined as the *low-rank CDM*. The utilities jointly admit a low-rank factorization $u_{xy} = \boldsymbol{c}_y^T \boldsymbol{t}_x$, where $\boldsymbol{t}_x, \boldsymbol{c}_x \in \mathbb{R}^r$ are the target and context vectors, and r is the rank of the CDM. Finally, they state that this featurization must by learned as we assume that it is not available in the notion of contextual utility.

2.3.3 SpringRank as a Blade-Chest model

The Blade-Chest model [Chen and Joachims, 2016a, Chen and Joachims, 2016b] is aimed at learning possibly intransitive relations from pairwise comparison data. In Blade-Chest, the probability that i beats j is also modeled as a logistic function, but differs in the argument:

$$P(i \text{ beats } j) = \frac{1}{1 + \exp(-M(i,j))}$$

where $M(i,j)$ is the *matchup* function of i and j. Notice that $M(i,j) = u_i - u_j$ recovers the multinomial logit, or the BTL model. Using this function, they define a $N \times N$ skew-symmetric matchup matrix M where $M_{ij} = M(i,j)$.

They define two d dimensional vectors $\boldsymbol{i}_{\text{blade}}$, $\boldsymbol{i}_{\text{chest}}$. These vectors can be viewed as the "attack" and "defense" of an item, and the interaction between these terms forms the basis of the matchup function. In particular, they describe two matchup functions – the Blade-Chest *dist* and Blade-Chest *inner* model – that represent any possible matchup matrix:

$$M(i,j) = \|\boldsymbol{j}_{\text{blade}} - \boldsymbol{i}_{\text{chest}}\|_2^2 - \|\boldsymbol{i}_{\text{blade}} - \boldsymbol{j}_{\text{chest}}\|_2^2 \qquad \text{Blade-Chest dist model}$$

$$M(i,j) = \boldsymbol{i}_{\text{blade}} \cdot \boldsymbol{j}_{\text{chest}} - \boldsymbol{j}_{\text{blade}} \cdot \boldsymbol{i}_{\text{chest}} \qquad \text{Blade-Chest inner model}$$

The authors term the ability of these two models to represent any possible matchup matrix as

expressiveness. They also remark, "although the two models are closely related, neither one generalizes the other."

Observe that there are a couple of resemblances and allusions to how SpringRank works. Firstly, the probability that i beats j in SpringRank also reduces to the logistic function. However, it is not as straightforward as BTL which brings us to the second observation. The matchup function in Blade-Chest represents the interaction between the two choice items. The utilities in SpringRank perform a similar role.

This naturally raises the question, what is the Blade-Chest representation of SpringRank? As SpringRank involves Hamiltonians, it is easier to construct the Blade-Chest dist model with $i_{\text{blade}} = s_i,\ i_{\text{chest}} = s_i + 1$ and notice that

$$\begin{aligned} M(i,j) &= \|j_{\text{blade}} - i_{\text{chest}}\|_2^2 - \|i_{\text{blade}} - j_{\text{chest}}\|_2^2 \\ &= (s_j - s_i - 1)^2 - (s_i - s_j - 1)^2 \end{aligned}$$

The Blade-Chest inner formulation is less straightforward. Given that neither model generalizes the other, it is unclear if such a formulation even exists. However, in Section 2.3.2, we noted that CDMs admit a low-rank factorization. Since SpringRank is also a CDM, the contextual utilities must admit a factorization into target and context vectors. In the language of Blade-Chest, these correspond to the blade and chest vectors. So, indeed, SpringRank can be formulated as a Blade-Chest inner model, whose vectors need to be learned.

At this point, it is important to note that Blade-Chest abstracts away from the utilities associated with the choice items. In Section 2.3.4, we construct a family of utilities, using the SpringRank model as inspiration, that give rise to the multinomial logit model.

2.3.4 A family of utilities

Consider an arbitrary utility for the choice $x \to y$, $u(x,y)$, we want the difference in utilities $u(x,y) - u(y,x) = k(x-y)$ to give rise to a multinomial logit. Looking at the degree n Taylor

polynomial, we have

$$u(x,y) = \sum_{p=0}^{n} \sum_{q=0}^{n-p} k_{p,q} x^p y^q, \tag{2.7}$$

where $k_{p,q}$ are real constants. In order to generate the multinomial logit, the coefficients need to satisfy the following:

$$k_{p,p} = \text{ free constant} \tag{2.8}$$

$$k_{1,0} \neq k_{0,1} \tag{2.9}$$

$$k_{p,q} = k_{q,p}, \qquad (p,q) \neq (1,0),(0,1) \tag{2.10}$$

We can easily verify that the Hamiltonian in SpringRank satisfies this condition.

Now, instead, if we want the utility to be a function of a linear combination of x, y i.e., $u(x,y) = u(ax+by+c)$, we notice that this utility cannot have any $(ax+by+c)$ term with an exponent larger than 2. To see why, let

$$u(x,y) = (ax+by+c)^n, \quad n \geq 3$$

where the coefficient of $x^p y^q$ is given by

$$k_{p,q} = \frac{(n)!}{p!q!(n-p-q)!} a^p b^q c^{n-p-q}.$$

First, consider the case where n is odd. From Equation 2.10, $k_{n,0} = k_{0,n}$. Therefore $a = b$ as n is odd. But, from Equation 2.9, $k_{1,0} \neq k_{0,1}$ gives $a \neq b$. This is a contradiction.

Now, consider the case where n is even. Similarly Equation 2.9 gives $a \neq b$. From Equation 2.10, $k_{n,0} = k_{0,n}$. This gives $a = \pm b$. Therefore, $a = -b$. But, then $k_{1,2} \neq k_{2,1}$. This is a contradiction. Therefore, $u(x,y)$ have any $(ax+by+c)$ term with an exponent larger than 2.

Notice that SpringRank captures all possible quadratic solutions if we recall that the rest length of the spring was arbitrarily scaled to 1. In other words, SpringRank captures all possible utilities that build in regularization.

2.3.5 Other related models

Another pairwise ranking model that achieves a similar effect to SpringRank is Elo++ [Sismanis, 2010]. Sismanis attempts to rank chess players and predict the outcome of chess games by combining a logistic error term with a quadratic term that acts as regularization by penalizing ranks that deviate away from the average rank of their neighbors as

$$\hat{s}_i = \frac{\sum_k w_{ik} s_k}{\sum_k w_{ik}}$$
$$p_{ij} = \frac{1}{1 + \exp(s_j - s_i - \gamma)}$$
$$l = \sum_{i,j} w_{ij}(p_{ij} - o_{ij})^2 + \lambda \sum_i (s_i - \hat{s}_i)$$

where w_{ij} is a weight based on how recently i and j played (larger value indicating more recency), o_{ij} is the outcome of that game, γ, λ are global parameters and l is the total loss. Notice that $\hat{s}_i$ is the average rank of the neighbors of player i, weighted by how recently they played. The aim of Elo++ is to find ranks s_i that minimize the loss l.

As a consequence of the regularization, the rank of i is pulled closer to the average of the ranks of neighbors. This suggests that the rank of a player is similar to that of their opponents. Under the microstate interpretation of SpringRank, the probabilistic model suggests the same by favoring (assigning a higher likelihood) to edges whose nodes are similarly ranked. The energy of the spring automatically kicks in the same effect of the regularization of Elo++.

Two key differences between Elo++ and SpringRank are: (1) Elo++ attempts to minimize the prediction accuracy explicitly and simultaneously, while SpringRank does it implicitly through the probabilistic model, and (2) Elo++ weighs games based on how recently it occurred, while SpringRank is agnostic to time.

2.4 SpringRank as linear regression

Recall that the Hamiltonian is described as

$$H(\boldsymbol{s}) = \frac{1}{2} \sum_{i,j} A_{ij}(s_i - s_j - 1)^2$$

$$= \frac{1}{2}\sum_{i,j}(\sqrt{A_{ij}}(s_i - s_j) - \sqrt{A_{ij}})^2.$$

Our goal is to minimize $H(\boldsymbol{s})$. Notice that this resembles the method of least squares to solve a linear regression problem. And we can recover the linear system that generates this least squares formulation.

$$\begin{bmatrix} \sqrt{A_{11}} & 0 & \dots & 0 \\ 0 & \sqrt{A_{12}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{A_{nn}} \end{bmatrix} \begin{bmatrix} s_1 - s_1 \\ s_1 - s_2 \\ \vdots \\ s_n - s_n \end{bmatrix} = \begin{bmatrix} \sqrt{A_{11}} \\ \sqrt{A_{12}} \\ \vdots \\ \sqrt{A_{nn}} \end{bmatrix}$$

$$\implies A_{n^2}^{\frac{1}{2}} \begin{bmatrix} s_1 - s_1 \\ s_1 - s_2 \\ \vdots \\ s_n - s_n \end{bmatrix} = A_{n^2}^{\frac{1}{2}}\mathbf{1}. \tag{2.11}$$

We can rewrite the vector with difference in scores as

$$\begin{aligned} \begin{bmatrix} s_1 - s_1 \\ s_1 - s_2 \\ \vdots \\ s_1 - s_n \\ s_2 - s_1 \\ \vdots \\ s_n - s_n \end{bmatrix} &= \begin{bmatrix} s_1 \\ s_1 \\ \vdots \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} - \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \\ s_1 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} - \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} \\ &= I_{n^2}^{(+)}\boldsymbol{s} - I_{n^2}^{(-)}\boldsymbol{s} \\ &= \tilde{I}_{n^2}\boldsymbol{s}. \end{aligned} \tag{2.12}$$

Therefore, SpringRank can be re-imagined as solving the following regression problem via the method of least squares

$$A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2}\boldsymbol{s} = A_{n^2}^{\frac{1}{2}}\mathbf{1}. \tag{2.13}$$

Observe that $\tilde{I}_{n^2}$ does not have linearly independent columns. Therefore, $A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2}$ also does not have linearly independent columns. As a consequence, $\boldsymbol{s}$ cannot be determined uniquely. This is not surprising as we already know from SpringRank that $\boldsymbol{s}$ is translation-invariant.

Notice that if we attempt to directly solve this linear system using least squares, we have to pre-multiply by $(A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2})^T$. And we get

$$\begin{aligned}
(A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2})^T A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2}\boldsymbol{s} &= (A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2})^T A_{n^2}^{\frac{1}{2}}\mathbf{1} \\
\implies \tilde{I}_{n^2}^T A_{n^2}^{\frac{1}{2}^T} A_{n^2}^{\frac{1}{2}}\tilde{I}_{n^2}\boldsymbol{s} &= \tilde{I}_{n^2}^T A_{n^2}^{\frac{1}{2}^T} A_{n^2}^{\frac{1}{2}}\mathbf{1} \\
\implies \tilde{I}_{n^2}^T A_{n^2}\tilde{I}_{n^2}\boldsymbol{s} &= \tilde{I}_{n^2}^T A_{n^2}\mathbf{1}. \qquad (2.14)
\end{aligned}$$

The last equation is simply a decomposition of

$$L\boldsymbol{s} = \hat{\boldsymbol{d}}.$$

Therefore, we recover the SpringRank solution. This verifies the alternative least squares formulation of the problem.

Now, let us take a step back to the regression equations. We have

$$\sqrt{A_{ij}}(s_i - s_j) = \sqrt{A_{ij}} + \epsilon_{ij}, \quad \forall i \neq j \qquad (2.15)$$

where ϵ_{ij} is some unobserved quantity, which we will call residuals. We can obtain certain nice properties by assuming that ϵ_{ij} is drawn from some fixed probability distribution.

For instance, Gauss-Markov Theorem [Amemiya, 1985] asserts that under Gauss-Markov assumptions, which are

(i) $E(\epsilon_{ij}) = 0, \forall\ i, j,$

(ii) $Var(\epsilon_{ij}) = \sigma^2 < \infty, \forall\ i, j,$ and

(iii) $Cov(\epsilon_{ij}, \epsilon_{kl}) = 0,$ whenever $i \neq k$ and $j \neq l,$

the least squares estimate $\hat{\boldsymbol{s}}$ is the *best linear unbiased estimator*. It is *best* in the sense that is optimal to minimizing the mean squared error (the Hamiltonian). It is *linear* in the outcome

variables, $\sqrt{A_{ij}}$. And, it is *unbiased* as $E(\hat{\boldsymbol{s}}) = \boldsymbol{s}$. However, note that we may have to translate $\hat{\boldsymbol{s}}$ appropriately in order to satisfy the unbiased property, although it will not invalidate the other two properties.

Further, if we assume normality i.e., $\epsilon \sim Normal(0, \sigma^2 I)$, then the estimator $\hat{\boldsymbol{s}}$ is also normally distributed, $\hat{\boldsymbol{s}} \sim Normal(\boldsymbol{s}, \sigma^2 Q)$ [Amemiya, 1985]. Here $Q = ((A_{n^2}^{\frac{1}{2}} \tilde{I}_{n^2})^T A_{n^2}^{\frac{1}{2}} \tilde{I}_{n^2})^{-1}$ is the cofactor matrix. We can simplify this

$$\begin{aligned} Q &= ((A_{n^2}^{\frac{1}{2}} \tilde{I}_{n^2})^T A_{n^2}^{\frac{1}{2}} \tilde{I}_{n^2})^{-1} \\ &= (\tilde{I}_{n^2}^T A_{n^2}^{\frac{1}{2}^T} A_{n^2}^{\frac{1}{2}} \tilde{I}_{n^2})^{-1} \\ &= (\tilde{I}_{n^2}^T A_{n^2} \tilde{I}_{n^2})^{-1} \\ &= L^{-1} \end{aligned}$$

where L is the Laplacian of the network A. So, $\hat{\boldsymbol{s}} \sim Normal(\boldsymbol{s}, \sigma^2 L^{-1})$. Of course, this holds assuming that we shift $\hat{\boldsymbol{s}}$ and $\boldsymbol{s}$ appropriately and that we define L^{-1} as the Moore-Penrose pseudo-inverse because L is rank deficient.

This is interesting because [De Bacco et al., 2018] show that if we assume a multivariate Gaussian distribution for $\boldsymbol{s}$ with covariance matrix Σ, and use the formulation given by the Boltzmann distribution, we get that the covariance matrix $\Sigma = \frac{1}{\beta} L^{-1}$.

2.5 Reconciliation

The results in the preceding sections are intriguing. In Section 2.2, we started with the Boltzmann distribution to predict states in the SpringRank model. These states correspond to edge directions i.e., the Boltzmann distribution characterized the probability that node i beats node j. And this characterization is natural as SpringRank is a physical system of springs.

In Section 2.3, we saw that the multinomial logit model, a Random Utility Model, characterizes the same probability in a similar manner. We reconciled the two models saying that SpringRank is also a Random Utility Model. In particular, there is a generalized version of multinomial logit models, called the context dependent random utility model and SpringRank is just a special case

of the generalization. Through this lens, we decomposed the observed utility of choosing a winner into the utility determined solely by the winner and the contextual utility determined as a function of the opponent. However, we observed that there is a missing piece to the puzzle: what is the unobserved utility in the RUM formulation of SpringRank?

In Section 2.4, we discovered that SpringRank is a least squares solution to a linear regression problem with normally distributed residuals. We also noted that this is the same result obtained from the Boltzmann distribution. So, if the residuals, $\epsilon_{ij} \sim Normal(0, \sigma^2)$, then

$$\begin{aligned} \sqrt{A_{ij}}(s_i - s_j - 1) &\sim \sigma Normal(0,1) \\ \implies \frac{1}{2}(s_i - s_j - 1)^2 &\sim \frac{\sigma^2}{A_{ij}}\chi_1^2 \\ \implies -\beta H_{ij} &\sim \frac{-\beta\sigma^2}{A_{ij}}\chi_1^2 \end{aligned} \tag{2.16}$$

where we assume $A_{ij} \neq 0$ and χ_1^2 is the Chi-squared distribution with 1 degree of freedom. And this completes the Random Utility Model formulation. The unobserved utilities in SpringRank are drawn from a Gaussian distribution and takes the form of a Chi-squared distribution.

We have closed the problem of interpreting SpringRank through various perspectives by realizing that they offer the same insights. These interpretations are simply different sides to the same problem, each more natural in a particular context: When we want to infer the ranks from observed data, we use the least squares approach. When we wish to predict new edges i.e., winners in future interactions, we use the Boltzmann distribution. When we want to study the contextual effects of choosing the direction of the edge i.e., the winner in presence of different nodes, a Random Utility Model is best suited.

Chapter 3

SpringRank with Groups

There are scenarios where the nodes that are interacting may have an advantage (or disadvantage) that could potentially influence the outcome. For instance, in turn-based games such as chess we may believe that the player who goes first may gain an advantage. Home-field advantage in sports has been well studied [Courneya and Carron, 1992, Jones, 2007, Lidor et al., 2010]. [Clauset et al., 2015] report a hierarchical structure in faculty hiring that reflects a social inequality. When viewed as a competition for endorsements (e.g., by hiring doctoral students), this suggests that some institutions may enjoy an advantage over others. Further, there may be arbiters, having different preferences, who judge the outcome of the interaction. [Bruch and Newman, 2019] find that such a racial stratification occurs in online dating platforms.

These situations raise the question, how can we use SpringRank to identify these group preferences? SpringRank, by itself, does not account for such preferences. In this chapter, we develop methods to address this problem. In Section 3.1, we discuss some existing methods that tackle similar situations. In Section 3.2, we develop a model that takes into account group preferences when ranking nodes, assuming that group preferences are constant with respect to nodes. In Section 3.3, we extend the model to consider scenarios where arbiters making decisions have different group preferences, relaxing the constant group preferences assumption. Finally, in Section 3.4, we explore the model identifiability problem that arises throughout this chapter.

3.1 Related work

We want to infer the effect of group characteristics on the outcomes of interactions between nodes in the system. This is reminiscent of hierarchical models (or multilevel models) [Gelman and Hill, 2006]. Hierarchical models are often used when the data is nested. In such scenarios, each observation (node) is a member of a group and the group membership has significant effects on the outcome (ranks). Hierarchical models are used to estimate parameters that vary at more than one level i.e., at the group and individual level. [Slaughter and Koehly, 2016] discusses a hierarchical Bayesian approach to modeling exponential random graphs.

There has been work done in ranking individuals based on exclusively group comparisons, such as players in team-based games [Huang et al., 2008]. This does not help us because our system does not consist of just inter-group interactions; it also consists of intra-group interactions. There is also an existing body of literature that discusses network generation algorithms that produces high clustering of nodes [Ravasz and Barabási, 2003, Noh, 2003]. While we can estimate the parameters of such generative algorithms from data, this does not generalize well to systems where the group information is not correlated with the structure of the network.

3.2 Introducing groups for nodes

SpringRank is agnostic to characteristics that differentiate nodes thereby giving advantage (or disadvantage) with respect to the outcomes. By being able to identify these group *preferences* (or equivalently, penalties), we can get better insights into the behavior of the system. At this point, it is important to distinguish group memberships and community structure in networks. Community structure can be inferred from the network. Group membership information, which is the focus here, need not be evident from the structure of the network.

In this section, we extend the SpringRank model to infer these group preferences. Throughout this section, we will assume that the group preference for each node remains fixed. In other words, if a node i has a group preference θ_i, then it will have the same group preference in all of its

interactions recorded in the data. In Section 3.3, we will relax the assumption and deal with a more general situation.

3.2.1 Redefining the hamiltonian

Recall that in the original model, we set all springs to have the same rest positions. Here, we allow different springs to have different rest positions. In particular, we have *a priori* knowledge about the group memberships of the nodes. So, each node's final combined rank is a function of its individual rank (score) and the group membership. All nodes in a particular group have the same rest position. Consequentially, nodes belonging to one group may have an advantage over a different group, while nodes in the same group do not. These *group preferences* allow us to incorporate nested structure of the network when ranking its members. This nested structure can be given as such, or inferred from the network itself.

Now, assume there are k groups in the network, where $0 < k \leq N$. The group membership is given by the matrix $G \in \mathbb{R}^{N\times k}$, where $G_{it} = 1$ if node i belongs to group t and 0 otherwise. We can also define $G_{it} = \delta_{g_i,t}$. Here, $\delta_{a,b}$ is the Kronecker delta such that,

$$G_{it} = \delta_{g_i,t} = \begin{cases} 1, & g_i = t \\ 0, & g_i \neq t \end{cases}.$$

Let $\boldsymbol{\theta} \in \mathbb{R}^k$ denote the group penalties, where θ_t is the penalty for the t^{th} group. Finally, let s_i denote the rank (or total score) and a_i denote the individual score for node i. So, the spring corresponding to the edge $i \to j$ has energy,

$$\begin{aligned} H_{ij} &= \frac{1}{2}(s_i - s_j - 1)^2 \\ &= \frac{1}{2}([a_i + \theta_{g_i}] - [a_j + \theta_{g_j}] - 1)^2, \end{aligned} \tag{3.1}$$

where g_i is the group that node i belongs to. Therefore, the total energy of the system is given by the revised Hamiltonian

$$H(\boldsymbol{a}, \boldsymbol{\theta}) = \sum_{i,j=1}^{N} A_{ij}H_{ij} = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij}([a_i + \theta_{g_i}] - [a_j + \theta_{g_j}] - 1)^2. \tag{3.2}$$

3.2.2 Minimizing the hamiltonian

To minimize the Hamiltonian, observe that it is convex in $\boldsymbol{a}$ and $\boldsymbol{\theta}$. Therefore, we can find the optimal ranking, $\boldsymbol{a}$, and the optimal group penalties, $\boldsymbol{\theta}$, by solving $\nabla H(\boldsymbol{a}, \boldsymbol{\theta}) = 0$.

To solve $\nabla H(\boldsymbol{a}, \boldsymbol{\theta}) = 0$, where H is defined in Equation 3.2, first consider $\frac{\partial H}{\partial a_i}$.

$$\begin{aligned}\frac{\partial H}{\partial a_i} &= \sum_j A_{ij}([a_i + \theta_{g_i}] - [a_j + \theta_{g_j}] - 1) - \sum_j A_{ji}([a_j + \theta_{g_j}] - [a_i + \theta_{g_i}] - 1) \\ &= (a_i + \theta_{g_i}) \sum_j A_{ij} - \sum_j A_{ij}(a_j + \theta_{g_j}) - \sum_j A_{ij} \\ &\quad + (a_i + \theta_{g_i}) \sum_j A_{ji} - \sum_j A_{ji}(a_j + \theta_{g_j}) + \sum_j A_{ij}.\end{aligned}$$

Now, let $d_i^{\text{out}} = \sum_j A_{ij}$ and $d_i^{\text{in}} = \sum_j A_{ji}$. Further, for the sake of convenience, let us define a new vector $\boldsymbol{\theta}_g \in \mathbb{R}^N$ such that the i^{th} element of $\boldsymbol{\theta}_g$ denotes the group preference for node i. Thus,

$$\frac{\partial H}{\partial a_i} = (a_i + \theta_{g_i})(d_i^{\text{out}} + d_i^{\text{in}}) - \big((A + A^T)(\boldsymbol{a} + \boldsymbol{\theta}_g)\big)_i - (d_i^{\text{out}} - d_i^{\text{in}}).$$

We want $\frac{\partial H}{\partial a_i} = 0$. So,

$$(a_i + \theta_{g_i})(d_i^{\text{out}} + d_i^{\text{in}}) - \big((A + A^T)(\boldsymbol{a} + \boldsymbol{\theta}_g)\big)_i = (d_i^{\text{out}} - d_i^{\text{in}}).$$

Now, we can rewrite this as a system of N equations by recalling that D^{out} and D^{in} are diagonal matrices representing the out- and in-degrees. This gives us

$$[D^{\text{out}} + D^{\text{in}} - (A + A^T)](\boldsymbol{a} + \boldsymbol{\theta}_g) = [D^{\text{out}} - D^{\text{in}}]\mathbf{1}.$$

Here, $\mathbf{1}$ is the all-ones vector of appropriate dimension. Notice that $\boldsymbol{\theta}_g = G\boldsymbol{\theta}$. Therefore, we can write the linear system as

$$[D^{\text{out}} + D^{\text{in}} - (A + A^T)](\boldsymbol{a} + G\boldsymbol{\theta}) = [D^{\text{out}} - D^{\text{in}}]\mathbf{1}. \tag{3.3}$$

Now, consider $\frac{\partial H}{\partial \theta_t}$. This is more difficult to represent using θ_{g_i} because more than one node can belong to the same group. So, using the Kronecker delta, equivalently we have

$$\frac{\partial H}{\partial \theta_t} = \sum_{i,j,r} A_{ij}([a_i + \theta_t] - [a_j + \theta_r] - 1)\delta_{g_i,t}\delta_{g_j,r} - \sum_{i,j,r} A_{ij}([a_j + \theta_r] - [a_i + \theta_t] - 1)\delta_{g_i,t}\delta_{g_j,r}.$$

To simplify, it is worth investigating each term in this equation individually.

$$\begin{aligned}
\sum_{i,j,r} A_{ij} a_i \delta_{g_i,t} \delta_{g_j,r} &= \sum_i a_i \delta_{g_i,t} \sum_{j,r} A_{ij} \delta_{g_j,r} \\
&= \sum_i a_i \delta_{g_i,t} \sum_j A_{ij} && (\textstyle\sum_r \text{ becomes redundant.}) \\
&= \sum_i a_i d_i^{\text{out}} \delta_{g_i,t} = \sum_i G_{ti}^T D_{ii}^{\text{out}} s_i && (G_{ti}^T = G_{it} = \delta_{g_i,t}) \\
&= (G^T D^{\text{out}} \boldsymbol{a})_t. \\
\sum_{i,j,r} A_{ji} a_i \delta_{g_i,t} \delta_{g_j,r} &= (G^T D^{\text{in}} \boldsymbol{a})_t. && (\text{a similar argument}) \\
\sum_{i,j,r} A_{ij} \theta_t \delta_{g_i,t} \delta_{g_j,r} &= \theta_t \sum_i \delta_{g_i,t} \sum_{j,r} A_{ij} \delta_{g_j,r} \\
&= \theta_t \sum_i \delta_{g_i,t} \sum_j A_{ij} && (\textstyle\sum_r \text{ becomes redundant.}) \\
&= \theta_t \sum_i d_i^{\text{out}} \delta_{g_i,t} = \theta_t \sum_i G_{ti}^T D_{ii}^{\text{out}} G_{it} && (G_{ti}^T = G_{it} = \delta_{g_i,t}) \\
&= (G^T D^{\text{out}} G)_{tt} \theta_t = (G^T D^{\text{out}} G \boldsymbol{\theta})_t. \\
\sum_{i,j,r} A_{ji} \theta_t \delta_{g_i,t} \delta_{g_j,r} &= (G^T D^{\text{in}} G \boldsymbol{\theta})_t. && (\text{a similar argument}) \\
\sum_{i,j,r} A_{ij} a_j \delta_{g_i,t} \delta_{g_j,r} &= \sum_i \delta_{g_i,t} \sum_j A_{ij} a_j \sum_r \delta_{g_j,r} \\
&= \sum_i \delta_{g_i,t} \sum_j A_{ij} a_j && (\textstyle\sum_r \text{ becomes redundant.}) \\
&= \sum_i \delta_{g_i,t} (A\boldsymbol{a})_i = \sum_i G_{ti}^T (A\boldsymbol{a})_i && (G_{ti}^T = G_{it} = \delta_{g_i,t}) \\
&= (G^T A \boldsymbol{a})_t. \\
\sum_{i,j,r} A_{ji} a_j \delta_{g_i,t} \delta_{g_j,r} &= (G^T A^T \boldsymbol{a})_t. && (\text{a similar argument}) \\
\sum_{i,j,r} A_{ij} \theta_r \delta_{g_i,t} \delta_{g_j,r} &= \sum_i \delta_{g_i,t} \sum_j A_{ij} \sum_r \theta_r \delta_{g_j,r} \\
&= \sum_i \delta_{g_i,t} \sum_j A_{ij} \sum_r G_{jr} \theta_r && (G_{jr} = \delta_{g_j,r})
\end{aligned}$$

$$= \sum_i \delta_{g_i,t} \sum_j A_{ij}(G\boldsymbol{\theta})_j = \sum_i G^T_{ti}(AG\boldsymbol{\theta})_i \qquad (G^T_{ti} = G_{it} = \delta_{g_i,t})$$

$$= (G^T AG\boldsymbol{\theta})_t.$$

$$\sum_{i,j,r} A_{ji}\theta_r \delta_{g_i,t}\delta_{g_j,r} = (G^T A^T G\boldsymbol{\theta})_t. \qquad \text{(a similar argument)}$$

$$\sum_{i,j,r} A_{ij}\delta_{g_i,t}\delta_{g_j,r} = \sum_i \delta_{g_i,t} \sum_j A_{ij} \sum_r \delta_{g_j,r}$$

$$= \sum_i \delta_{g_i,t} \sum_j A_{ij} \qquad (\sum_r \text{ becomes redundant.})$$

$$= \sum_i d^{\text{out}}_i G_{it} = \sum_i G^T_{ti} D^{\text{out}}_{ii} G_{it} \qquad (G^T_{ti} = G_{it} = \delta_{g_i,t})$$

$$= (G^T D^{\text{out}} G)_t.$$

$$\sum_{i,j,r} A_{ji}\delta_{g_i,t}\delta_{g_j,r} = (G^T D^{\text{in}} G)_t. \qquad \text{(a similar argument)}$$

Substituting these simplifications into $\frac{\partial H}{\partial \theta_t} = 0$, we have

$$\left[G^T(D^{\text{out}} + D^{\text{in}} - (A + A^T))s\right]_t + \left[G^T(D^{\text{out}} + D^{\text{in}} - (A + A^T))G\theta\right]_t = \left[G^T(D^{\text{out}} - D^{\text{in}})G\right]_t.$$

Now, we can rewrite this as a system of k equations

$$\left[G^T(D^{\text{out}} + D^{\text{in}} - (A + A^T))\right](\boldsymbol{a} + G\boldsymbol{\theta}) = \left[G^T(D^{\text{out}} - D^{\text{in}})G\right]\mathbf{1},$$

where $\mathbf{1}$ is the vector of all-ones. If we impose the restriction that a node belongs to exactly one group, we can simplify $G\mathbf{1} = \mathbf{1}$ (assuming they are of appropriate dimensions). This gives us

$$\left[G^T(D^{\text{out}} + D^{\text{in}} - (A + A^T))\right](\boldsymbol{a} + G\boldsymbol{\theta}) = \left[G^T(D^{\text{out}} - D^{\text{in}})\right]\mathbf{1}. \tag{3.4}$$

Equations 3.3, 3.4 give us the following linear systems:

$$L(\boldsymbol{a} + G\boldsymbol{\theta}) = \hat{\boldsymbol{d}}, \tag{3.5}$$

$$G^T L(\boldsymbol{a} + G\boldsymbol{\theta}) = G^T\hat{\boldsymbol{d}}, \tag{3.6}$$

where $L = D^{\text{out}} + D^{\text{in}} - (A + A^T)$ and $\hat{\boldsymbol{d}}$ if the difference between out- and in-degrees. We can also visualize this as a single system of $N + k$ equations if we concatenate $\boldsymbol{a}$ and $\boldsymbol{\theta}$ vectors as

$$\begin{bmatrix} L & LG \\ G^T L & G^T LG \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}} \\ G^T\hat{\boldsymbol{d}} \end{bmatrix}. \tag{3.7}$$

3.2.3 Generalizing to arbitrary number of group characteristics

We can generalize the results in the previous section to m sets of group characteristics. For characteristic i, let k_i denote the number of groups, $G_i \in \mathbb{R}^{N \times k_i}$ denote the group membership matrix for characteristic, and $\boldsymbol{\theta}_i \in \mathbb{R}^{k_i}$ denote the preferences vector.

We get the following linear systems:

$$L(\boldsymbol{a} + \sum_{j=1}^{m} G_j \boldsymbol{\theta}_j) = \hat{\boldsymbol{d}}, \tag{3.8}$$

$$G_i^T L(\boldsymbol{a} + \sum_{j=1}^{m} G_j \boldsymbol{\theta}_j) = G_i^T \hat{\boldsymbol{d}}, \qquad 1 \leq i \leq m. \tag{3.9}$$

If we visualize this as a single system of $N + \sum_{i=1}^{m} k_i$ equations, we have:

$$\begin{bmatrix} L & LG_1 & LG_2 & \dots & LG_m \\ G_1^T L & G_1^T LG_1 & G_1^T LG_2 & \dots & G_1^T LG_m \\ G_2^T L & G_2^T LG_1 & G_2^T LG_2 & \dots & G_2^T LG_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ G_m^T L & G_m^T LG_1 & G_m^T LG_2 & \dots & G_m^T LG_m \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_m \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}} \\ G_1^T \hat{\boldsymbol{d}} \\ G_2^T \hat{\boldsymbol{d}} \\ \vdots \\ G_m^T \hat{\boldsymbol{d}} \end{bmatrix}. \tag{3.10}$$

3.2.4 Regularization

Observe that Equation 3.6 is identical to Equation 3.5, but left multiplied by G^T. This is in fact an underdetermined system. This system has at least $k + 1$ degrees of freedom as $\text{rank}(L) \leq N - 1$. This means that we cannot infer group penalties because we do not know if node i is inherently better than j or if it is because group g_i is superior to g_j. In other words, this is an identifiability problem. We will take a closer look at this in Section 3.4.

For now, if we introduce L_2 regularization for $\boldsymbol{a}$ and $\boldsymbol{\theta}$, with regularization coefficients λ_a and λ_θ respectively, we can get a unique solution to the system. For brevity, we will deal with the case with only one set of group characteristics, but we can directly extend this idea to m group characteristics. Specifically, in $m = 1$ case, we have the regularized Hamiltonian:

$$H_r(\boldsymbol{a}, \boldsymbol{\theta}) = H(\boldsymbol{a}, \boldsymbol{\theta}) + \tfrac{1}{2}\lambda_a ||\boldsymbol{a}||_2^2 + \tfrac{1}{2}\lambda_\theta ||\boldsymbol{\theta}||_2^2. \tag{3.11}$$

We want to solve $\nabla H_r(\boldsymbol{a}, \boldsymbol{\theta}) = 0$, which gives:

$$\nabla_a : \nabla_a H + \lambda_a \boldsymbol{a} = 0,$$

$$\nabla_\theta : \nabla_\theta H + \lambda_\theta \boldsymbol{\theta} = 0.$$

If we define, $\Lambda_a = \lambda_a I$ and $\Lambda_\theta = \lambda_\theta I$, where I is the identity matrix of appropriate dimensions, we get the following system:

$$\begin{bmatrix} L + \Lambda_a & LG \\ G^T L & G^T LG + \Lambda_\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}} \\ G^T \hat{\boldsymbol{d}} \end{bmatrix}. \tag{3.12}$$

The regularization coefficients control how much the group preferences and the individual scores contribute to the final ranks of the nodes. A relatively small λ_a (compared to λ_θ) places a greater importance on the individual scores. On the other hand, a relatively large λ_a places a greater importance on the group preferences. The choice of these regularization coefficients determine how we resolve the identifiability problem.

3.2.5 Results using synthetic data

Here, we present results from synthetic data. We generated data by first assigning groups to each node, G. Then, we drew individual scores a_{planted} and group preferences, θ_{planted} from Gaussian distributions. Thus, the total rank is $s_{\text{planted}} = a_{\text{planted}} + G\theta_{\text{planted}}$. Then an average degree $\langle k \rangle$ and inverse temperature β were chosen. Finally, edges were drawn from the generative model described in the original paper [De Bacco et al., 2018].

For Test 1, we drew individual scores for 500 nodes from $Normal(\frac{1}{2}, 10)$. We fixed the number of groups to be 4 and drew the corresponding group preferences from $Normal(2, \frac{1}{2})$. Here, $\beta = 0.1$, $\langle k \rangle = 10$. The results are shown in Figure 3.1. In each subfigure, the left panel plots the recovered individual scores $\hat{a}$ against the planted individual scores a. The middle panel plots the recovered group preferences $\hat{\theta}$ against the planted group preferences θ. And the right panel plots the recovered ranks $\hat{s}$ against the planted ranks s. Figure 3.1(a) corresponds to the solution when regularization was used. Figure 3.1(b) shows the solution when attempting to resolve identifiability by setting the

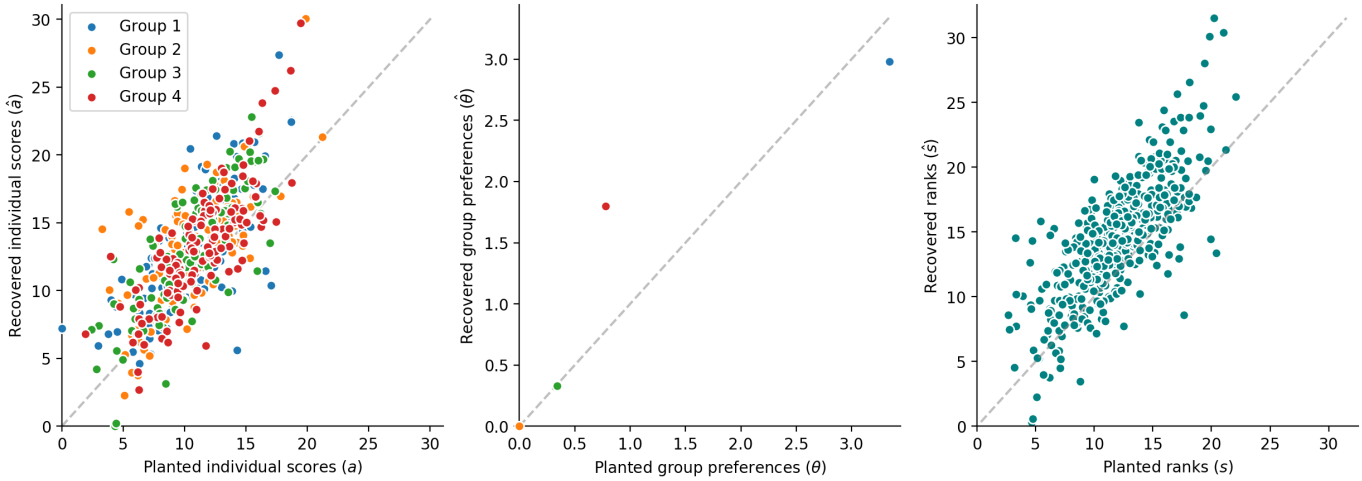
average individual scores across all groups to 0. Figure 3.1(c) shows the solution when we arbitrarily set the average individual scores for each group. Note that the final rank $\hat{s}$ remains unchanged. This illustrates the non-identifiability problem which is discussed in detail in Section 3.4 (where we will revisit this experiment).

To introduce variety, in Test 2 we drew individual scores for 600 node. 200 of them were drawn from $Normal(\frac{1}{2}, 10)$, 200 from $Normal(0, \frac{1}{2})$, 200 from $Normal(-1, 1)$. We fixed the number of groups to be 4 and drew the corresponding group preferences from $Normal(2, 10)$. The results are shown in Figure 3.2. The subfigures correspond to plots described above. We performed more tests under a similar setup but varied the inverse temperature β. These plots can be found in Appendix A.1.

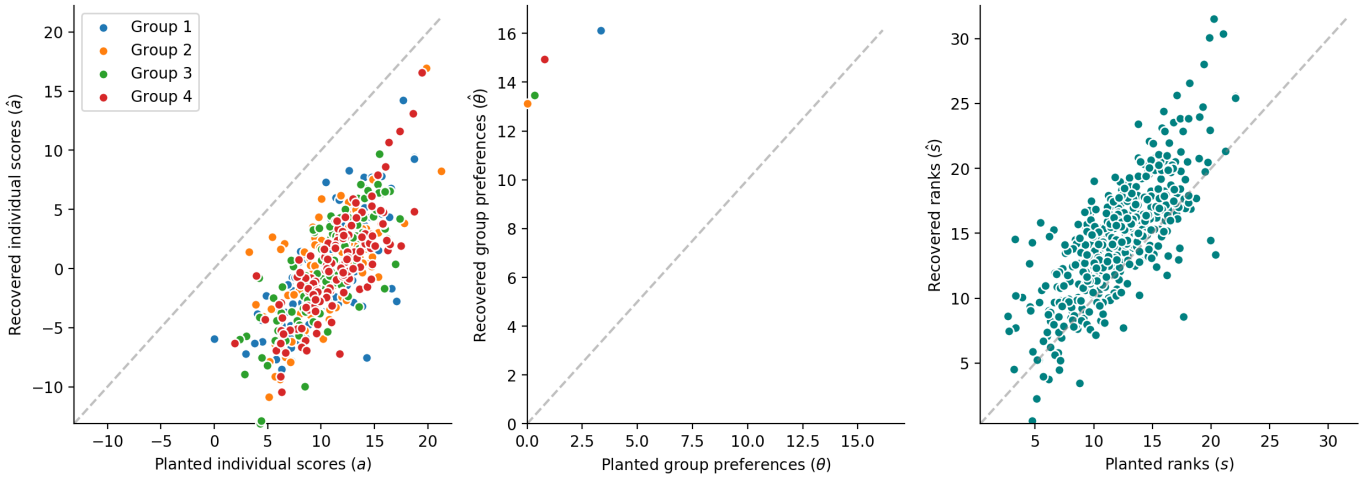
3.3 Introducing arbiters for interactions

In Section 3.2 we assumed that the group characteristics of nodes directly affect the outcome of the interactions. In this section, we introduce arbiters who determine the outcomes of interactions. So, the arbiters' preferences influence the outcomes. Further, we also assume that the arbiters have group memberships which determine their preferences. In essence, we are relaxing the assumption that group preferences stay constant across interactions. However, for now, we still assume that group memberships remain the same. We will briefly relax this assumption in in Section 3.4 when studying identifiability.

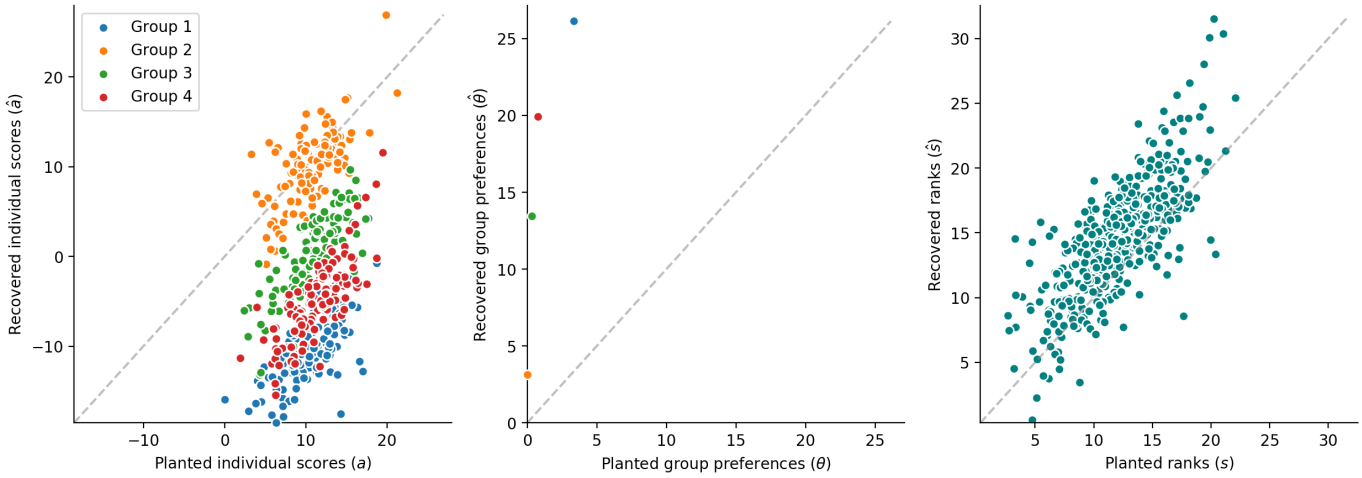
This is inspired by competition in online dating platforms [Bruch and Newman, 2018]. In such a system, suitors compete for attention. Concretely, suitor i beats suitor j when i gets the attention of an arbiter k, but j fails. This happens when i and j send a first message to k, who replies to i but not j. Suitors have characteristics such as ethnicity, age and education that may give an advantage. Arbiters also have the same characteristics which may determine their preferences for suitors. So, an arbiter belonging to group a will have preference $\theta_{a,s}$ for a suitor belonging to group s, while an arbiter in group b will have a, possibly, different preference $\theta_{b,s}$ for suitors in group s. The problem here is infer these group preferences.



(a) Solution using regularization.

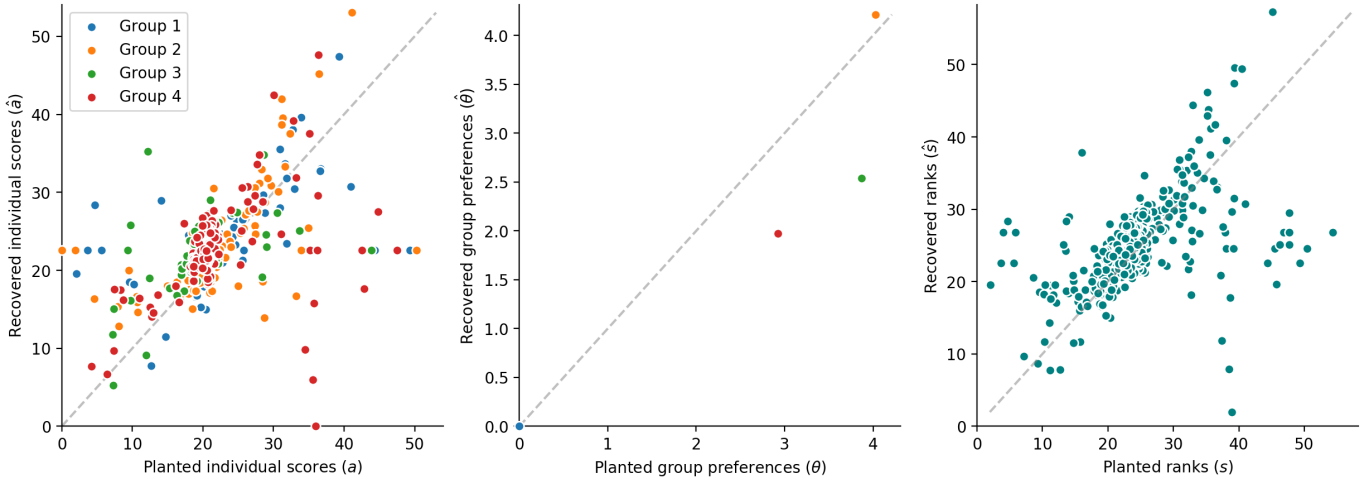


(b) Setting average individual scores across groups to 0.

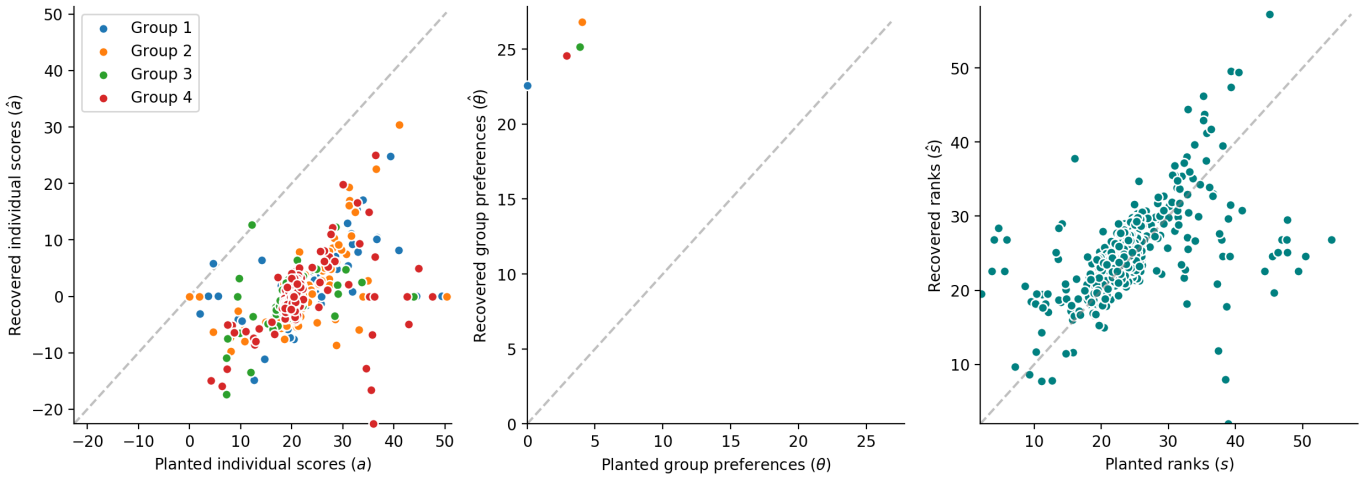


(c) Setting average individual scores across groups to $(10, -10, 0, 5)$.

Figure 3.1: Results from Test 1 with synthetic data (without arbiters). $N = 500$; $a_{\text{planted}} \sim Normal(\frac{1}{2}, 10)$; 4 groups; $\theta_{\text{planted}} \sim Normal(2, \frac{1}{2})$. The individual scores correspond to a_{planted} and group preferences refer to θ_{planted}. The ranks refer to the total score of each node (individual and group preference) i.e., s_{planted}. Here, $\beta = 0.1$, $\langle k \rangle = 10$.



(a) Solution using regularization.



(b) Setting average individual scores across groups to 0.

Figure 3.2: Results from Test 2 with synthetic data (without arbiters). $N = 600$: 200 from $Normal(\frac{1}{2}, 10)$; 200 from $Normal(0, \frac{1}{2})$; 200 from $Normal(-1, 1)$; 4 groups; $\theta_{\text{planted}} \sim Normal(2, 10)$. The individual scores correspond to a_{planted} and group preferences refer to θ_{planted}. The ranks refer to the total score of each node (individual and group preference) i.e., s_{planted}. Here, $\beta = 0.1$, $\langle k \rangle = 10$.

While this question is inspired online dating platforms, the answer is more broadly applicable to any pairwise comparison data. Consider a different scenario where a researcher presents pairs of items to an arbiter, who is to choose one item. In the language of Random Utility Models, the arbiter gains some utility from choosing each item and attempts to maximize their utility. Of course, each arbiter may gain different utilities from the same item (in the same situation). The problem, then, is to identify these preferences that are characteristic to arbiters.

To make this concrete, consider a situation where a publisher is choosing book covers for a book. The publisher knows that people are more likely to buy the book if the cover appeals to them. Further, different target audiences, based on age and region, may have different preferences for book covers. The question here is to identify these different preferences.

To motivate the solution, we will use the example of *suitors* and *arbiters* in Sections 3.3.1 and 3.3.2. After that, we will switch the terminology and refer to suitors as *items* (and continue using *arbiters*). This is to remind ourselves of the broader applicability.

3.3.1 Redefining the hamiltonian

Consider a situation with N suitors and M arbiters. The matrix $F \in \mathbb{R}^{N\times M}$ denotes the first message indicators. $F_{ik} = 1$ if suitor i sent a first message to arbiter k, and 0 otherwise. Let $R \in \mathbb{R}^{N\times M}$ denote the reply indicators. $R_{ik} = 1$ if arbiter k replied to a first message sent by suitor i, and 0 otherwise. Consequentially, let us define $\bar{R} = 1 - R$ for convenience. Hence, $\bar{R}_{ik} = 0$ if arbiter k replied to a first message sent by suitor i, and 1 otherwise. Using this notation, we can construct the adjacency matrix, $A \in \mathbb{R}^{N\times N}$, for the directed network as follows:

$$A_{ij} = \sum_{k=1}^{M} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}.$$

In other words, A_{ij} denotes the number of times suitor i beat j in the competition for attention. It is the number of times i and j sent a first message to the same arbiter say k ($F_{ik}F_{jk}$), and that arbiter k chose to reply to i, but not j ($R_{ik}\bar{R}_{jk}$).

Let the suitors be grouped into n groups and the arbiters into m. Let g_x denote the group

membership of individual x. Let $G_S \in \mathbb{R}^{N\times n}$ denote the group membership matrix of suitors. Further, let $A^{(t)}$ denote the directed network formed when we only consider the competitions for arbiters in group t i.e., competitions decided by arbiters belonging to group t. Clearly, $A = \sum_{t=1}^{m} A^{(t)}$. With this group membership information, we can introduce group preferences for arbiters. Let us define this group preference matrix by $\Theta \in \mathbb{R}^{m\times n}$ where Θ_{rs} is the group preference of arbiters in group r for suitors in group s. Further, let $\boldsymbol{\theta}_r \in \mathbb{R}^n$ be the vector of group preferences of arbiters in group r. It is easy to see that this corresponds to row r in the matrix Θ.

Now, we are ready to define the new Hamiltonian, $H(\boldsymbol{a}, \Theta)$:

$$H(\boldsymbol{a}, \Theta) = \frac{1}{2}\sum_{i,j}\sum_{k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}(a_i + \Theta_{g_k,g_i} - a_j - \Theta_{g_k,g_j} - 1)^2, \tag{3.13}$$

where a_i is the individual rank (or score) for suitor i. If Θ is identically 0, we recover the original SpringRank model defined in Equation 2.2 as $A_{ij} = \sum_k F_{ik}F_{jk}R_{ik}\bar{R}_{jk}$. Further, if there is only one group for the arbiters, we recover SpringRank with groups (and without arbiters) defined in Equation 3.2.

3.3.2 Minimizing the hamiltonian

We minimize the Hamiltonian by noticing that it is once again convex in $\boldsymbol{a}$ and Θ. So, we can minimize the energy, by solving for $\nabla H(\boldsymbol{a}, \Theta) = 0$.

To solve for $\nabla H(\boldsymbol{a}, \Theta) = 0$, first, consider $\frac{\partial H}{\partial a_i}$:

$$\begin{aligned}\frac{\partial H}{\partial a_i} &= \sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}(a_i + \Theta_{g_k,g_i} - a_j - \Theta_{g_k,g_j} - 1) \\ &\quad - \sum_{j,k} F_{jk}F_{ik}R_{jk}\bar{R}_{ik}(a_j + \Theta_{g_k,g_j} - a_i - \Theta_{g_k,g_i} - 1).\end{aligned}$$

Let us look at just the first term,

$$\begin{aligned}&\sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}(a_i + \Theta_{g_k,g_i} - a_j - \Theta_{g_k,g_j} - 1) \\ &\qquad = a_i\sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk} + \sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}\Theta_{g_k,g_i} - \sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}a_j\end{aligned}$$

$$
\begin{aligned}
&- \sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}\Theta_{g_k,g_j} + \sum_{j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk} \\
&= a_i d_i^{\text{out}} + \sum_{j,t} A_{ij}^{(t)}\Theta_{t,g_i} - \sum_j A_{ij}a_j - \sum_{j,t} A_{ij}^{(t)}\Theta_{t,g_j} - d_i^{\text{out}} \\
&= a_i d_i^{\text{out}} + \Big(\sum_t D^{(t),\text{out}} G_S \boldsymbol{\theta}_t\Big)_i - (A\boldsymbol{a})_i - \Big(\sum_t A^{(t)} G_S \boldsymbol{\theta}_t\Big)_i - d_i^{\text{out}}.
\end{aligned}
$$

Here, $(\cdot)^{(t)}$ corresponds to the quantity $(\cdot)$ formed by arbiters belonging to group t. The second term is symmetrical,

$$
\begin{aligned}
&\sum_{j,k} F_{jk}F_{ik}R_{jk}\bar{R}_{ik}(a_j + \Theta_{g_k,g_j} - a_i - \Theta_{g_k,g_i} - 1) \\
&\qquad = -\left[a_i d_i^{\text{in}} + \Big(\sum_t D^{(t),\text{in}} G_S \boldsymbol{\theta}_t\Big)_i - (A^T\boldsymbol{a})_i - \Big(\sum_t A^{(t),T} G_S \boldsymbol{\theta}_t\Big)_i + d_i^{\text{in}}\right].
\end{aligned}
$$

Combining both, setting the sum to 0, and vectorizing, we get:

$$
L\boldsymbol{a} + \sum_{t=1}^{m} L^{(t)} G_S \boldsymbol{\theta}_t = \hat{\boldsymbol{d}},
$$

where L is the full graph Laplacian and $L^{(t)}$ is the Laplacian of the graph formed by arbiters belonging to group t.

Now, consider $\frac{\partial H}{\partial \Theta_{rs}}$.

$$
\begin{aligned}
\frac{\partial H}{\partial \Theta_{rs}} &= \sum_{i,j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}(a_i + \Theta_{rs} - a_j - \Theta_{r,g_j} - 1)\delta_{g_k,r}\delta_{g_i,s} \\
&\quad - \sum_{i,j,k} F_{jk}F_{ik}R_{jk}\bar{R}_{ik}(a_j + \Theta_{r,g_j} - a_i - \Theta_{rs} - 1)\delta_{g_k,r}\delta_{g_i,s}.
\end{aligned}
$$

This is slightly more complicated. So, let us analyze this term-by-term.

$$
\begin{aligned}
\sum_{i,j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}a_i\delta_{g_k,r}\delta_{g_i,s} &= \sum_i a_i\delta_{g_i,s}\sum_j\sum_k F_{ik}F_{jk}R_{ik}\bar{R}_{jk}\delta_{g_k,r} \\
&= \sum_i a_i\delta_{g_i,s}\sum_j A_{ij}^{(r)} \\
&= \Big(G_S^T D^{(r),\text{out}}\boldsymbol{a}\Big)_s. \\
\sum_{i,j,k} F_{jk}F_{ik}R_{jk}\bar{R}_{ik}a_i\delta_{g_k,r}\delta_{g_i,s} &= \Big(G_S^T D^{(r),\text{in}}\boldsymbol{a}\Big)_s. \\
\sum_{i,j,k} F_{ik}F_{jk}R_{ik}\bar{R}_{jk}a_j\delta_{g_k,r}\delta_{g_i,s} &= \sum_j a_j\sum_i \delta_{g_i,s}\sum_k F_{ik}F_{jk}R_{ik}\bar{R}_{jk}\delta_{g_k,r}
\end{aligned}
$$

$$= \sum_j a_j \sum_i A^{(r)}_{ij} \delta_{g_i,s}$$

$$= \sum_j (G_S^T A^{(r)})_{sj} a_j$$

$$= \left(G_S^T A^{(r)} \boldsymbol{a} \right)_s .$$

$$\sum_{i,j,k} F_{jk} F_{ik} R_{jk} \bar{R}_{ik} a_j \delta_{g_k,r} \delta_{g_i,s} = \left(G_S^T A^{(r),T} \boldsymbol{a} \right)_s .$$

$$\sum_{i,j,k} F_{ik} F_{jk} R_{ik} \bar{R}_{jk} \Theta_{rs} \delta_{g_k,r} \delta_{g_i,s} = \sum_j \sum_i \delta_{g_i,s} \sum_k F_{ik} F_{jk} R_{ik} \bar{R}_{jk} \Theta_{rs} \delta_{g_k,r}$$

$$= \sum_j \sum_i \delta_{g_i,s} A^{(r)}_{ij} \Theta_{rs}$$

$$= \sum_j (G_S^T A^{(r)})_{sj} \Theta_{rs}$$

$$= \left(G_S^T D^{(r),\text{out}} G_S \boldsymbol{\theta}_r \right)_s .$$

$$\sum_{i,j,k} F_{jk} F_{ik} R_{jk} \bar{R}_{ik} \Theta_{rs} \delta_{g_k,r} \delta_{g_i,s} = \left(G_S^T D^{(r),\text{in}} G_S \boldsymbol{\theta}_r \right)_s .$$

$$\sum_{i,j,k} F_{ik} F_{jk} R_{ik} \bar{R}_{jk} \Theta_{r,g_j} \delta_{g_k,r} \delta_{g_i,s} = \sum_i \delta_{g_i,s} \sum_j \sum_k \sum_k F_{ik} F_{jk} R_{ik} \bar{R}_{jk} \Theta_{r,g_j} \delta_{g_k,r}$$

$$= \sum_i \delta_{g_i,s} \sum_j A^{(r)}_{ij} \Theta_{r,g_j}$$

$$= \sum_i \delta_{g_i,s} \sum_{j,v} A^{(r)}_{ij} \Theta_{r,g_j} \delta_{g_j,v}$$

$$= A^{(r)}_{ij} \Theta_{r,g_j} (A^r G_S \boldsymbol{\theta}_r)_s$$

$$= \left(G_S^T A^{(r)} G_S \boldsymbol{\theta}_r \right)_s .$$

$$\sum_{i,j,k} F_{jk} F_{ik} R_{jk} \bar{R}_{ik} \Theta_{r,g_j} \delta_{g_k,r} \delta_{g_i,s} = \left(G_S^T A^{(r),T} G_S \boldsymbol{\theta}_r \right)_s .$$

$$\sum_{i,j,k} F_{ik} F_{jk} R_{ik} \bar{R}_{jk} \delta_{g_k,r} \delta_{g_i,s} = \sum_{i,j} \delta_{g_i,s} A^{(r)}_{ij}$$

$$= \left(G_S^T \boldsymbol{d}^{(r),\text{out}} \right)_s .$$

$$\sum_{i,j,k} F_{jk} F_{ik} R_{jk} \bar{R}_{ik} \delta_{g_k,r} \delta_{g_i,s} = \left(G_S^T \boldsymbol{d}^{(r),\text{in}} \right)_s .$$

This gives us the following linear system:

$$G_S^T L^{(r)} \boldsymbol{a} + G_S^T L^{(r)} G_S \boldsymbol{\theta}_r = G_S^T \hat{\boldsymbol{d}}^{(r)}$$

Together, this gives us the following linear system:

$$L\boldsymbol{a} + \sum_{t=1}^{m} L^{(t)} G_S \boldsymbol{\theta}_t = \hat{\boldsymbol{d}}^{(t)}, \tag{3.14}$$

$$G_S^T L^{(t)} \boldsymbol{a} + G_S^T L^{(t)} G_S \boldsymbol{\theta}_t = G_S^T \hat{\boldsymbol{d}}^{(t)}, \qquad 1 \leq t \leq m, \tag{3.15}$$

We can visualize Equations 3.14, 3.15 as a single system of equations:

$$\begin{bmatrix} L & L^{(1)}G_S & L^{(2)}G_S & \dots & L^{(m)}G_S \\ G_S^T L^{(1)} & G_S^T L^{(1)} G_S & 0 & \dots & 0 \\ G_S^T L^{(2)} & 0 & G_S^T L^{(2)} G_S & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ G_S^T L^{(m)} & 0 & 0 & \dots & G_S^T L^{(m)} G_S \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_m \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}} \\ G_S^T \hat{\boldsymbol{d}}^{(1)} \\ G_S^T \hat{\boldsymbol{d}}^{(2)} \\ \vdots \\ G_S^T \hat{\boldsymbol{d}}^{(m)} \end{bmatrix}. \tag{3.16}$$

3.3.3 Regularization

Once again, the system is non-identifiable. To see why, notice the following:

$$\sum_{r=1}^{m} L^{(r)} = L$$

$$\implies \sum_{t=1}^{m} \left(G_S^T L^{(t)} \boldsymbol{a} + G_S^T L^{(t)} G_S \boldsymbol{\theta}_t \right) = G_S^T L \boldsymbol{a} + \sum_{t=1}^{m} L^{(t)} G_S \boldsymbol{\theta}_t. \tag{3.17}$$

In other words, Equation 3.15 is a rewritten version of Equation 3.14. This means that the system is low-rank and the model is non-identifiable i.e., we cannot distinguish the group preferences from the individual scores. We will take a closer look at identifiability in Section 3.4.

For now, we can make the system full-rank by introducing regularization. With regularization coefficients $\lambda_a, \lambda_{\theta_1}, \lambda_{\theta_2}, \dots, \lambda_{\theta_m}$, and the corresponding diagonal matrices $\Lambda_a, \Lambda_{\theta_1}, \Lambda_{\theta_2}, \dots, \Lambda_{\theta_m}$ of appropriate dimensions, we can solve the regularized system described in Equation 3.18.

$$\begin{bmatrix} L+\Lambda_a & L^{(1)}G_S & L^{(2)}G_S & \dots & L^{(m)}G_S \\ G_S^TL^{(1)} & G_S^TL^{(1)}G_S+\Lambda_{\theta_1} & 0 & \dots & 0 \\ G_S^TL^{(2)} & 0 & G_S^TL^{(2)}G_S+\Lambda_{\theta_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ G_S^TL^{(m)} & 0 & 0 & \dots & G_S^TL^{(m)}G_S+\Lambda_{\theta_m} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_m \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}} \\ G_S^T\hat{\boldsymbol{d}}^{(1)} \\ G_S^T\hat{\boldsymbol{d}}^{(2)} \\ \vdots \\ G_S^T\hat{\boldsymbol{d}}^{(m)} \end{bmatrix}. \tag{3.18}$$

The regularization coefficients determine how the scores and group preferences are controlled. A large regularization coefficient makes the individual score vector (or corresponding group preference vector) small giving more importance to other vectors when finding a solution.

3.3.4 Results from synthetic data

Here, we present results from synthetic data. We generated data by first assigning groups to each item. Then, we fixed the number of groups the arbiters can belong to. Then, we drew the individual scores for the items and the group preferences for each class of arbiters. To generate the full directed network, we generated a directed network for each group of arbiters like in Section 3.2.

For Test 1, we drew individual scores for 500 items from $Uniform(1, 10)$. We fixed $n = m = 4$ groups for the items and arbiters. We introduced a systematic bias against the fourth group of items i.e., a negative group preference drawn from a $Normal(0, 1)$. For others, arbiters from group i had a positive preference towards items from group i, and no preference for other items. Here, $\beta = 0.5$. The results are shown in Figure 3.3. Figure 3.3(a) shows the solution obtained using regularization. Figure 3.3(b) shows the solution after correcting for identifiability. The left panel in each subfigure plots the recovered individual scores against the planted individual scores. Each of the other four panels plot the recovered group preferences against the planted group preferences, for each group of arbiters.

For Test 2, we drew individual scores for 600 items from $Uniform(1, 10)$. We fixed $n = m = 4$ groups for the items and arbiters. This time, the group preferences were completely randomized

and drawn from $Normal(2, 1)$. Here, $\beta = 0.1$. The results are shown in Figure 3.4. The plots in each subfigure is described above. We performed more tests whose results are shown in Appendix A.2 varying β and the distributions from which the ranks were drawn from.

3.4 Model identifiability

In our development of models so far, we have constantly been confronted by model identifiability. In this section, we will delve deeper into understanding why the models are non-identifiable and attempt to provide conditions under which we can possibly recover identifiability. Here, we will concern ourselves only with the case where there are no arbiters i.e., group preferences remain fixed, for each node, across all interactions. However, the arguments made will generalize to the scenario with aribters having varied preferences.

Recall that the Hamiltonian is given by Equations 3.1 and 3.2:

$$\begin{aligned} H_{ij} &= \frac{1}{2}(s_i - s_j - 1)^2 \\ &= \frac{1}{2}([a_i + \theta_{g_i}] - [a_j + \theta_{g_j}] - 1)^2, && \text{(3.1 revisited)} \\ H(\boldsymbol{a}, \boldsymbol{\theta}) &= \frac{1}{2}\sum_{i,j=1}^{N} A_{ij}([a_i + \theta_{g_i}] - [a_j + \theta_{g_j}] - 1)^2. && \text{(3.2 revisited)} \end{aligned}$$

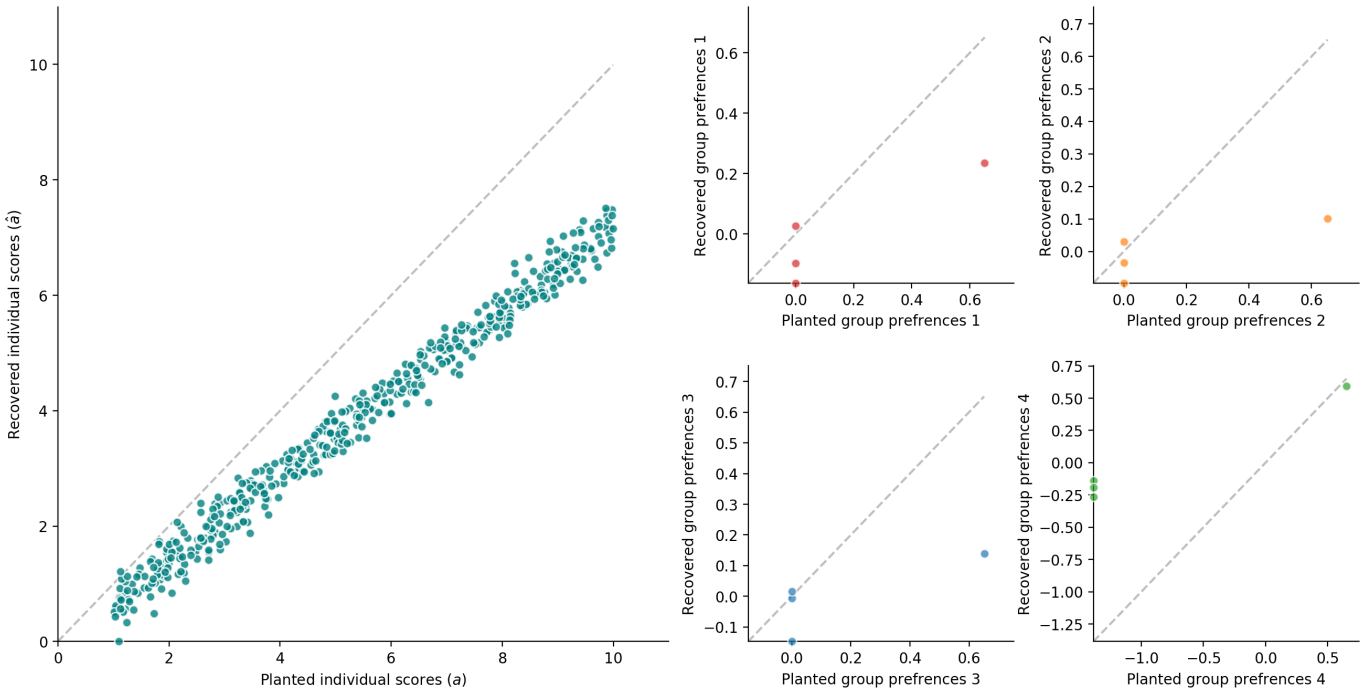
And the solution is given by Equations 3.5, 3.6, and 3.7:

$$L(\boldsymbol{a} + G\boldsymbol{\theta}) = \hat{\boldsymbol{d}}, \qquad \text{(3.5 revisited)}$$

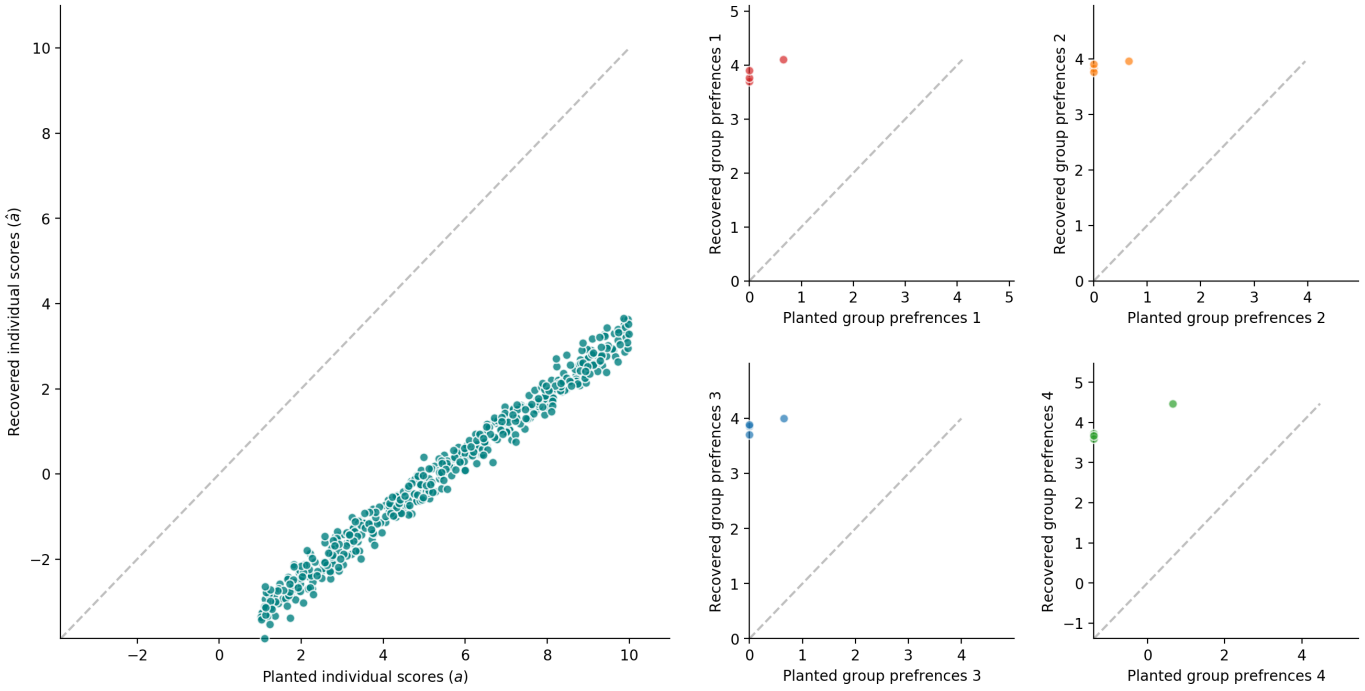
$$G^T L(\boldsymbol{a} + G\boldsymbol{\theta}) = G^T \hat{\boldsymbol{d}}, \qquad \text{(3.6 revisited)}$$

$$\begin{bmatrix} L & LG \\ G^T L & G^T LG \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}} \\ G^T \hat{\boldsymbol{d}} \end{bmatrix}. \qquad \text{(3.7 revisited)}$$

Here, model identifiability corresponds to finding a unique linear decomposition of s_i as $s_i = a_i + \theta_{g_i}$.

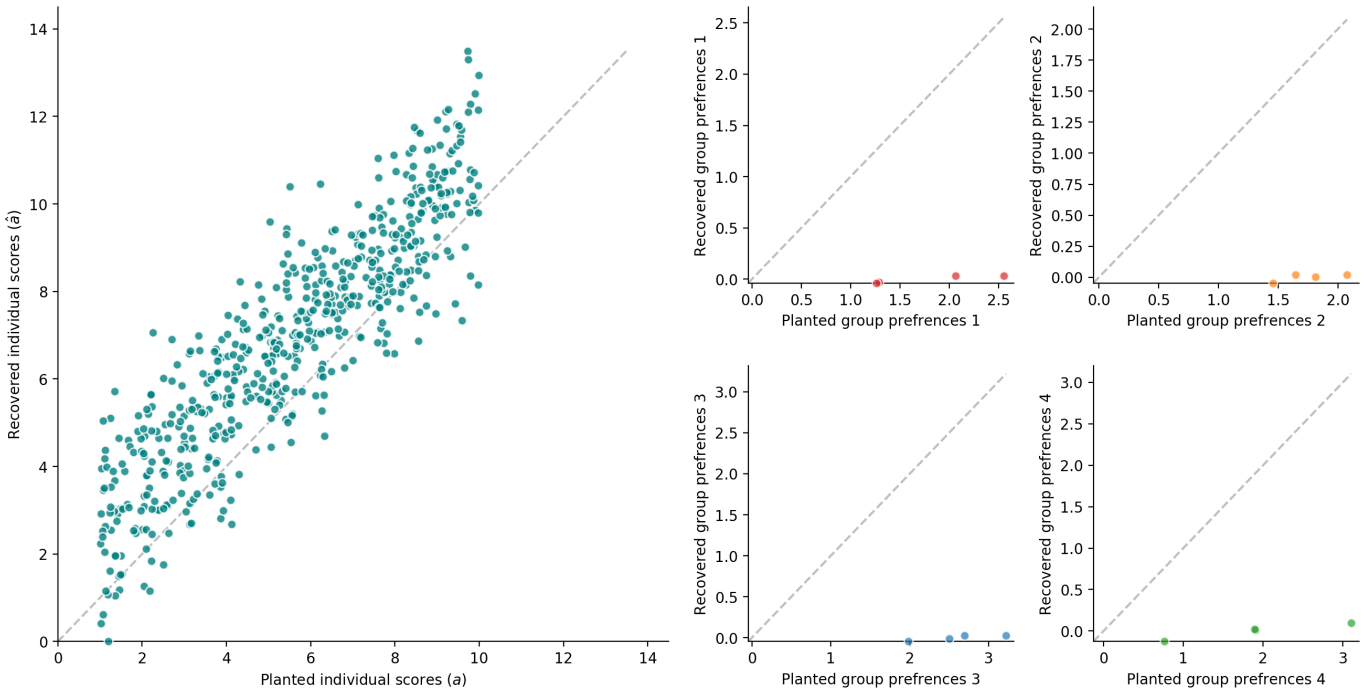


(a) Solution using regularization.

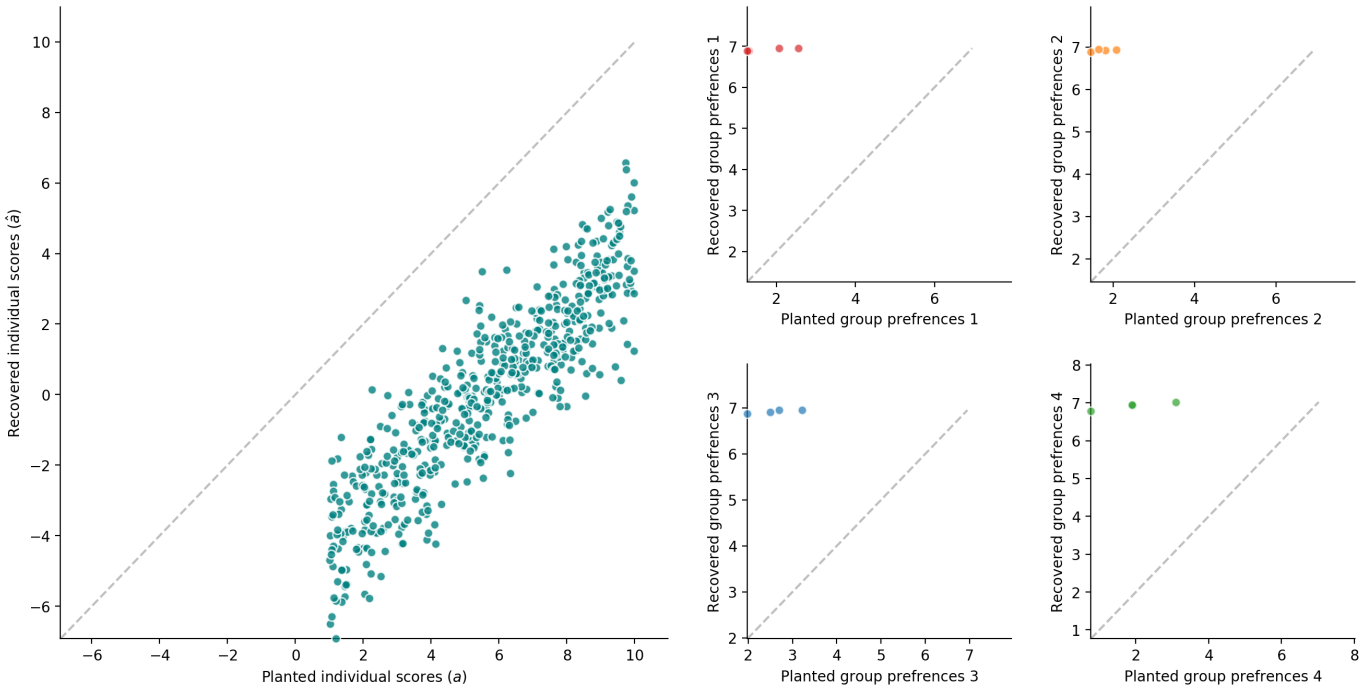


(b) Setting average individual scores across groups to 0.

Figure 3.3: Results from Test 1 with synthetic data (with arbiters). 500 items from $Uniform(1, 10)$. There are 4 groups for items and arbiters. There is a systematic bias against the fourth group of items i.e., a negative group preference drawn from a $Normal(0, 1)$. For others, arbiters from group i had a positive preference towards items from group i, and no preference for other items. Here, $\beta = 0.5$.



(a) Solution using regularization.



(b) Setting average individual scores across groups to 0.

Figure 3.4: Results from Test 2 with synthetic data (with arbiters). 600 items from $Uniform(1, 10)$. There are 4 groups for items and arbiters. Group preferences were drawn from $Normal(2, 1)$. Here, $\beta = 0.1$.

3.4.1 The model is non-identifiable

The operator in Equation 3.7 is low-rank. To see this observe that Equation 3.6 is a compressed version of Equation 3.5. This is a consequence of attempting to decompose the rank (full score) of a node, s_i, into individual score, a_i, and group preference (group score), θ_{g_i}. This is a model identifiability problem.

The full score of node i is invariant in terms of an arbitrary constant c if we consider the following shift:

$$\begin{aligned} a'_i &= a_i + c \\ \theta'_{g_i} &= \theta_{g_i} - c \\ \implies s'_i &= a'_i + \theta'_{g_i} \\ &= (a_i + c) + (\theta_{g_i} - c) \\ &= s_i. \end{aligned}$$

Therefore, the energy of this spring does not change. Now, if we increase the individual scores of all nodes belonging to group g_i by c, the Hamiltonian does not change. Hence, the Hamiltonian itself is invariant in terms of the constant c. In other words, it is no longer meaningful to compare the individual scores of nodes (and the groups scores) between two different groups. It is only meaningful to compare the full scores of nodes.

This *confounding* of the model is illustrated in Figure 3.1. Figure 3.1(a) shows the solution when we use regularization. In Figure 3.1(b), we shift the individual scores (and group preferences) by arbitrary constants such that the average individual score for each group is 0. Firstly, notice that the final ranks, s_i remain unchanged. This verifies that the Hamiltonian is still minimized. Then, notice that the distribution of individual ranks also remains relatively unchanged. This is because we drew all individual scores from the same Gaussian distribution. In an attempt to confound the model, in Figure 3.1(b), we shift the individual scores (and group preferences) by arbitrary constants such that the average individual score for each group is very different. Observe that the

final ranks still remain unchanged. However, the individual scores follow different distributions. If we did not have prior information about the individual scores, it will be impossible to distinguish results in Figures 3.1(b) and 3.1(c).

Consider an example of chess games. We have a set of players who always play Black and a different set of players who always play White[1]. We wish to score (rank) players and understand how much of their score can be attributed to their inherent skill (individual score) versus the color of the pieces they play with (group preference). Using this, we wish to compare players based on their inherent skill. Since the players can only play with the same color, it is impossible for us to discern their skill from the color they choose. Mathematically, we can arbitrarily increase the individual scores of all Black players by c units and decrease the group score for Black by c units. And suddenly, when comparing the inherent skills (individual scores) of Black and White players, Black players appear to be more highly skilled. But, we could repeat this arbitrary shift procedure for just the White players and White players will appear to be more skilled than Black players. Clearly, we cannot decompose the full score into individual and group score preserving comparability.

3.4.2 The Equality assumption

So, when can we meaningfully compare the individual scores, if at all possible? For now, we will continue assuming that once a chess player chooses a color, they can never change it.

The decomposition fails because the individual scores for each group becomes coupled with that particular group preference, but is not directly coupled with the individual scores of different groups. In other words, we can shift individual scores for nodes within a certain group without altering the individual scores of other nodes. To make the individual scores comparable across groups, we can remove the dependence on the group score using the following *Equality assumption*:

$$E\big[\boldsymbol{a} \mid g_i = r\big] = E\big[\boldsymbol{a}\big] \tag{3.19}$$

[1] In chess, White always goes first.

Equation 3.19 means that the average individual score of nodes belonging to a group r, is the same as the average individual score of all nodes. Note that this is weaker than assuming that nodes across all groups have identically distributed individual scores. We call this the *Equality assumption* because it implicitly says, "All nodes are created equally, on average." Once this assumption is satisfied, we can compare the group preferences directly without worrying about confounding SpringRank.

We will argue that to confront the identifiability problem in the absence of external information, this assumption is necessary. The word *external* refers to information about the system that cannot be inferred from the win-loss and group matrices. For the sake of contradiction, assume that there is some grouping of nodes $\mathcal{F}$ such that each group has a different expected value of individual scores. There are two cases:

(i) We want to infer group preferences in the same group space i.e., the group preferences are to be inferred using a grouping $\mathcal{G}$ that is a partition of $\mathcal{F}$.[2] Any group preference that $\mathcal{G}$ provides will become indistinguishable from $\mathcal{F}$ because we will only see the effects due to both preferences. We have a model identifiability problem once again.

(ii) We want to infer group preferences in a different group space, $\mathcal{G}$. Note that this includes the cases where $\mathcal{G}$ is "orthogonal" to $\mathcal{F}$, and where $\mathcal{F}$ is a partition of $\mathcal{G}$. This case is not meaningful to consider. If we are in a different group space, then the group preferences that came with the original grouping simply manifest themselves in the form of individual scores.

Therefore, in the absence of external information, this assumption is a necessary evil. As a remark, if we do have external information about the relative differences in expected values across groups, or group preferences, then we can easily impose those restrictions by directly modifying Equation 3.19.

Now, we will see how to impose this assumption. We want to find the proper constant c_r

[2]Partition means that if nodes x, y belong to different groups under $\mathcal{F}$, then they cannot belong to the same group under $\mathcal{G}$.

for each group r such that the resulting translated individual scores satisfy Equation 3.19. First, we find the average individual scores of nodes in group r, $\langle \boldsymbol{a} \rangle_r$. Then, we translate the individual scores of nodes in group r by this value. To preserve the Hamiltonian of the system, we also need to translate the group score θ_r accordingly. Therefore, for each group r, we perform the following translation:

$$\hat{a}_i = a_i - \langle \boldsymbol{a} \rangle_r, \qquad \forall\ i \text{ where } g_i = r \tag{3.20}$$

$$\hat{\theta}_r = \theta_r + \langle \boldsymbol{a} \rangle_r. \tag{3.21}$$

The translation ensures that the average individual score in each group is 0. Therefore, the average individual score across all groups is also 0. Therefore, Equation 3.19 is satisfied. Thus, in order to resolve the model identifiability problem, and be able to compare individual and group scores, we need to fix the group scores to be the average individual score of nodes in that group. But, from the example shown in Figure 3.1, we know that this assumption can be wrongly imposed to generate false results and interpretations. So, we must tread with caution when imposing this.

To illustrate this on a real dataset, we ranked universities in North America based on Computer Science faculty hiring networks [Clauset et al., 2015]. The data consists of 4388 tenure-track faculty hired by 205 Computer Science departments. If department v hires a doctoral student from department u for a tenure-track position, then the edge goes from $u \to v$, because endorsed v department u. We ranked the 205 departments using SpringRank, and grouped them into 5 regions – Canada, Midwest, Northeast, South, and West.

The results are shown in Figure 3.5. The left panel shows the distribution of ranks of the departments sorted by region. In the middle panel, we shift the individual scores in order to conform to the Equality assumption. The right panel shows the "region preferences". Just by looking at this, it is tempting to say that departments located in the Northeast get a boost to their rank simply by being there. However, given our discussion about identifiability, we should be careful not to jump to such conclusions without knowing more about the underlying process that generated these ranks.

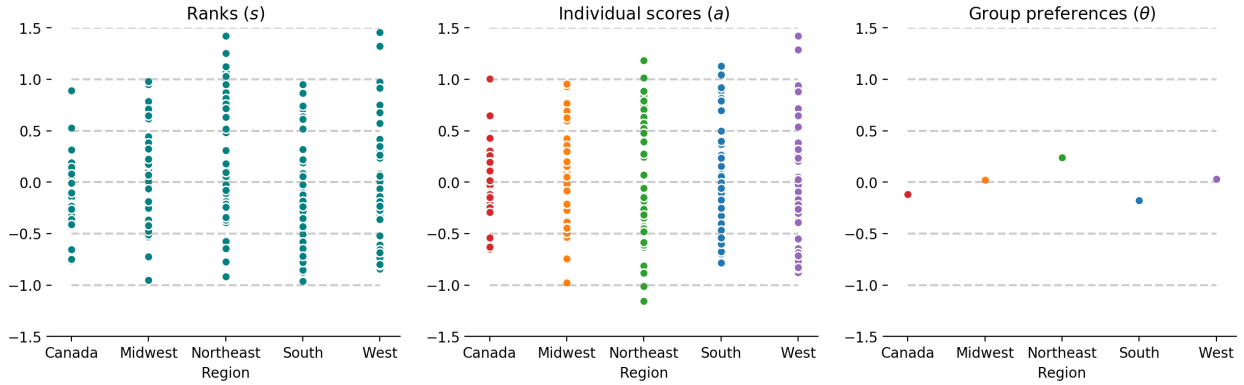


Figure 3.5: 205 Computer Science departments in North America ranked based on tenure-track faculty hiring network using SpringRank grouped into 5 regions – Canada, Midwest, Northeast, South, and West. Given our arguments about non-identifiability, we must be careful when interpreting these results about group preferences.

3.4.3 Resolving identifiability

So far we have been spuriously assuming that chess players are fixed to playing with one color throughout their career. Although it has served us well in developing our arguments, we now turn to a realistic situation (at least in some scenarios, including the game of chess): players are not restricted to a single color. In other words, let us assume that the groups are not fixed (nodes can change their groups in their interactions).

For the sake of simplicity, assume there are two different groupings given by the group matrices $G^{(1)}, G^{(2)}$. Assume that the individual scores $\boldsymbol{a}$, and the group preferences $\boldsymbol{\theta}$ are fixed under both groupings. When the nodes interact under $G^{(1)}$, call the resulting directed network $A^{(1)}$, the difference between out- and in-degrees $\hat{\boldsymbol{d}}^{(1)}$, and the Laplacian $L^{(1)}$. Similarly, define $A^{(2)}, \hat{\boldsymbol{d}}^{(2)}, L^{(2)}$ when the nodes interact under $G^{(2)}$.

If we were agnostic to groups, the two corresponding SpringRank solutions will take the form:

$$L^{(1)}\boldsymbol{s}^{(1)} = \hat{\boldsymbol{d}}^{(1)}$$

$$L^{(2)}\boldsymbol{s}^{(2)} = \hat{\boldsymbol{d}}^{(2)}.$$

But, we want $\boldsymbol{s}^{(1)}$ and $\boldsymbol{s}^{(2)}$ to decompose as:

$$\boldsymbol{s}^{(1)} = \boldsymbol{a} + G^{(1)}\boldsymbol{\theta}$$

$$\boldsymbol{s}^{(2)} = \boldsymbol{a} + G^{(2)}\boldsymbol{\theta}$$

Therefore,

$$L^{(1)}\boldsymbol{a} + L^{(1)}G^{(1)}\boldsymbol{\theta} = \hat{\boldsymbol{d}}^{(1)} \tag{3.22}$$

$$L^{(2)}\boldsymbol{a} + L^{(2)}G^{(2)}\boldsymbol{\theta} = \hat{\boldsymbol{d}}^{(2)} \tag{3.23}$$

$$\implies \begin{bmatrix} L^{(1)} & L^{(1)}G^{(1)} \\ L^{(2)} & L^{(2)}G^{(2)} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}}^{(1)} \\ \hat{\boldsymbol{d}}^{(2)} \end{bmatrix}$$

This is a system with $2N$ equations and $N + k$ variables. To make this a system of $N + k$ equations, we can pre-multiply Equation 3.23 by $G^{(2),T}$ and we get

$$\begin{bmatrix} L^{(1)} & L^{(1)}G^{(1)} \\ G^{(2),T}L^{(2)} & G^{(2),T}L^{(2)}G^{(2)} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{d}}^{(1)} \\ G^{(2),T}\hat{\boldsymbol{d}}^{(2)} \end{bmatrix} \tag{3.24}$$

Unlike in previous cases, Equation 3.24 generally has rank $N + k - 2$. This is because Equations 3.22 and 3.23 are generally independent from each other. So, the resulting rank of the system is $N + k - 2$, where the -2 comes from the fact that each Laplacian is rank deficient. This gives us two free variables, one for each of $\boldsymbol{a}$ and $\boldsymbol{\theta}$, and we can uniquely solve for the individual scores and the group preferences (up to one free variable). Therefore, we recover identifiability.

Note that this system can still be low-rank in specific scenarios where we lose identifiability. One such instance is when the top row of the matrix in Equation 3.24 is a linear combination of the bottom row. This corresponds to a case where every group interacts with every node the same number of times under both groupings. To see why observe that $G^T L = G^T[D^{\text{out}} + D^{\text{in}} - A - A^T]$ has dimensions $k \times N$ and in particular reflects the interactions between every group and node. Therefore, when every group interacts with every node the same number of times under both groupings, $G^{(1),T}L^{(1)} = G^{(2),T}L^{(2)}$. This also implies that $G^{(1),T}L^{(1)}G^{(1)} = G^{(2),T}L^{(2)}G^{(2)}$ and $G^{(1),T}\hat{\boldsymbol{d}}^{(1)} = G^{(2),T}\hat{\boldsymbol{d}}^{(2)}$. In other words, Equations 3.22 and 3.23 become linearly dependent when

grouped together, and Equation 3.24 has rank $N-1$ resulting in non-identifiability. Of course, now we can relax that assumption and conclude that when the ratio of number of interactions between every group and every node across both groupings is constant, we lose identifiability.

In order to demonstrate this, we performed several tests with synthetic data. For Test 1, we generated 500 nodes whose individual scores were drawn from $Normal(0, 10)$. We fixed the number of groups to be 2 and the group preferences was $[0.5, 0]$. The 500 nodes were randomized into two groups, in two different ways. Under the first grouping, a directed network was generated with $\langle k^{(1)} \rangle = 20$. And under the second grouping, a directed network was generated with $\langle k^{(2)} \rangle = 5$. Here, $\beta = 0.1$. The results are shown in Figure 3.6. The left panel plots the recovered individual scores against the planted individual scores. And the right panel plots the recovered group preferences against the planted group preferences.

For Test 2, we generated data in a fashion similar to chess i.e., only opposite groups can interact with each other. We drew 500 individual scores from $Normal(1, \frac{5}{2})$. We fixed the number of groups to be 2 and the group preferences was $[0.5, 0]$. The 500 nodes were randomized into a grouping. In the second grouping, the groups were flipped. Under the first grouping, a directed network was generated with $\langle k^{(1)} \rangle = 20$. And under the second grouping, a directed network was generated with $\langle k^{(2)} \rangle = 5$. Here, $\beta = 0.4$. We performed more tests, whose results are shown in Appendix A.3.

3.5 Conclusion

In this chapter, we developed a series of three models in order to identify the effect of node characteristics on the outcome of interactions. In order to account for node characteristics, we sorted nodes into different groups, each of which provide an additional boost. To model this boost, which we called group preference, we attempted to decompose the rank, s, into individual scores, a, and the group preference θ.

In the first model, we assumed that the groups of the nodes were fixed and that the group preference remained constant in all the interactions. In the second model, we introduced arbiters

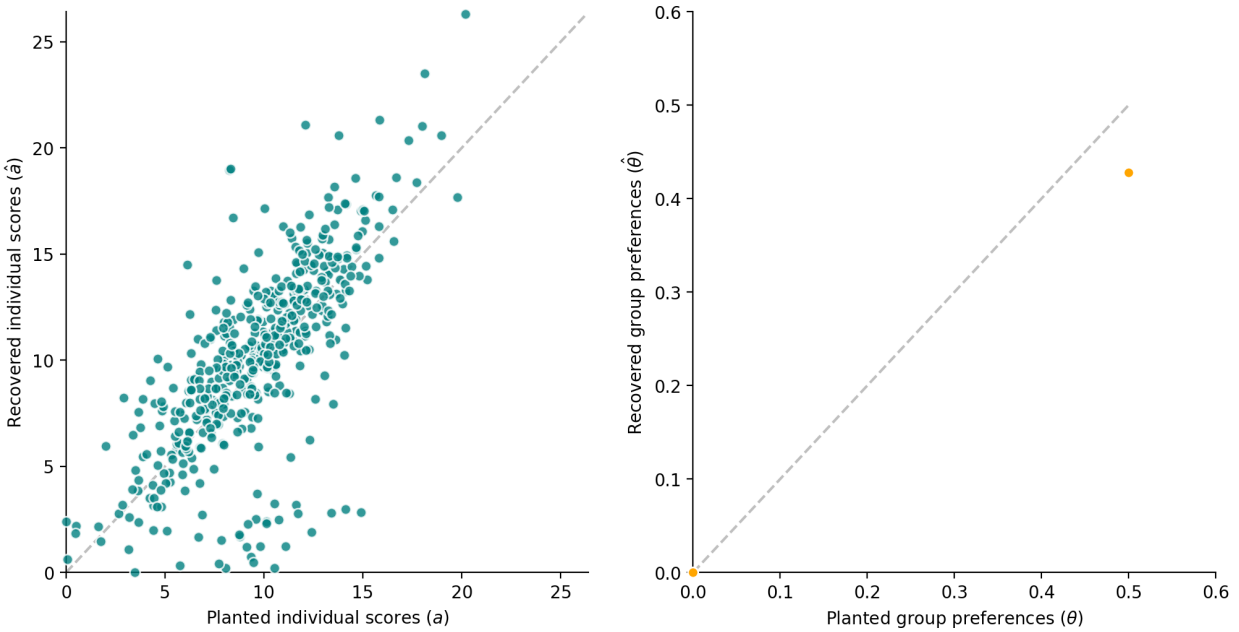


Figure 3.6: Results on identifiability with synthetic data - Test 1. 500 nodes drawn from $Normal(0, 10)$. There were two groups and the group preferences was $[0.5, 0]$. Under the two groupings, network was generated with $\langle k^{(1)} \rangle = 20$ and $\langle k^{(2)} \rangle = 5$. Here, $\beta = 0.1$

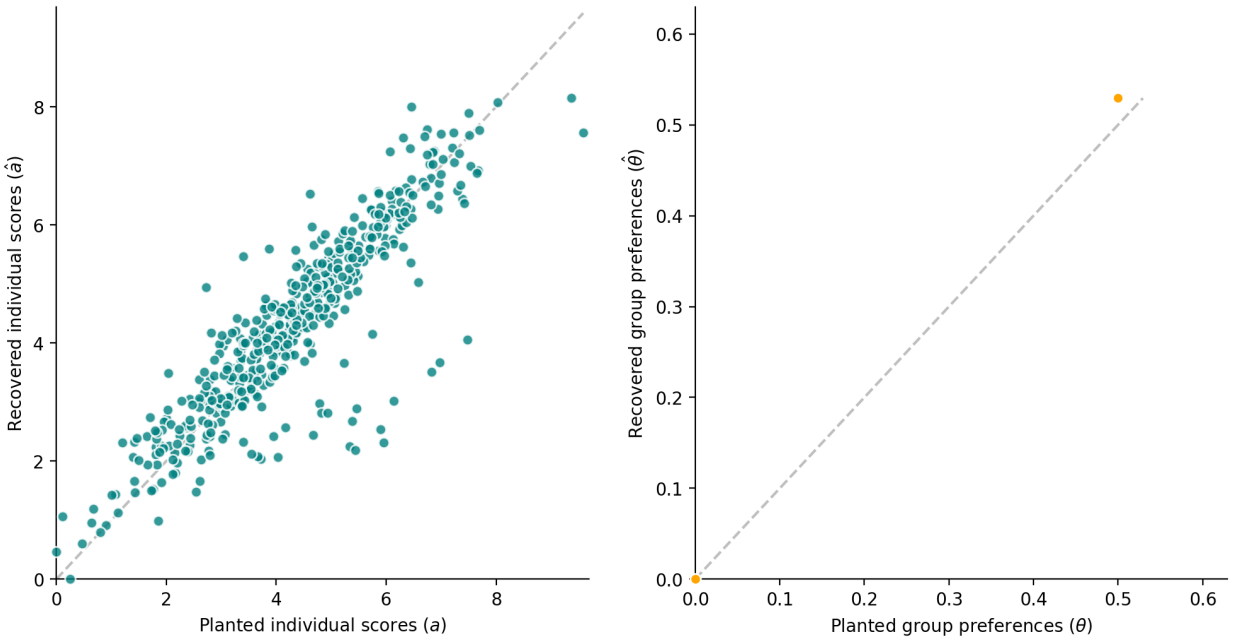


Figure 3.7: Results on identifiability with synthetic data - Test 2. 500 nodes drawn from $Normal(1, \frac{5}{2})$. There were two groups and the group preferences was $[0.5, 0]$. We allowed only opposite groups to interact with each other. Further, in the second grouping, nodes were assigned groups opposite from their first assignment. Under the two groupings, network was generated with $\langle k^{(1)} \rangle = 20$ and $\langle k^{(2)} \rangle = 5$. Here, $\beta = 0.4$

who determined the outcome of the interactions, and thus the group preferences could change over the interactions. We saw that both of these models are non-identifiable i.e., we cannot discern the individual score from the group preference. We studied identifiability in detail and saw that this corresponds to underdetermined systems. Without external information, we cannot recover identifiability. If we are willing to make an assumption that the average individual score for each each group is the same, we can partly recover identifiability. However, we noted that this assumption is flawed and can lead to misinterpretation of results. To recover identifiability, we developed a third model where the nodes can change group memberships in different interactions. This model is, generally, identifiable although there might be specific instances where we lose identifiability again.

Chapter 4

Online SpringRanking

So far, we have considered inferring rankings where all pairwise interactions are observed up front. However, in many applications, it's desirable to be able to compute rankings while interactions are being observed (e.g., inferring sports rankings throughout the season, as games are won and lost). In this chapter, we propose an extension to SpringRank to accurately and efficiently update rankings as new interactions occur over time. We term this process *online ranking* or an *online update*.

SpringRank computes the ranks by solving a linear system. The system grows quadratically with the number of nodes in the network. Recomputing the ranks every time a new interaction occurs can, therefore, be an expensive process. By being able to accurately and efficiently update existing ranks, we can study the temporal behavior, and the life cycle of the system. This has interesting applications in short time-scale networks (such as career trajectories of sports players and faculty), and long time-scale networks (such as behavior and evolution of biological species).

The rest of this chapter is organized as follows: In Section 4.1, we will discuss some related work that has already been done. In Section 4.2, we describe some properties that we may desire for such an online ranking algorithm. In Section 4.3, we propose an iterative update algorithm and discuss the results.

4.1 Related work

The problem of updating ranks continuously as new interactions occur is well studied. These include online ranking adaptations of commonly used methods such as PageRank [Page et al., 1999], and the Bradley-Terry-Luce model [Bradley and Terry, 1952, Luce, 1959]. Methods such as Elo score [Elo, 1978] and TrueSkill [Herbrich et al., 2007] are designed to be online in nature – updates occur after every game. There are also Bayesian ranking systems such as Glicko [Glickman, 1995] and Whole History Ranking [Coulom, 2008]. These methods approach ranking from a different perspective with contrasting assumptions, and are therefore ill-suited for adaptation to online SpringRank. These methods do not allow for conservation of ranks i.e., the order in which interactions are used to update the ranks can change the final ranking. Further, ranking methods that solely focus on pairwise interactions fail to propagate the changes in ranks throughout the entire system. The drawbacks suggest that a novel approach is needed to solve the problem of online ranking in SpringRank.

4.2 Desired properties of the system

To make this scenario concrete, consider an example. We have results from the original set of games in the form of a directed network A_0, and the corresponding ranking $\boldsymbol{s}_0$. Then, the players play additional games. The results from the original and additional games are described in the new directed network A. The corresponding final ranking is given by $\boldsymbol{s}$. According to the original model, we have to repeat the computation to calculate $\boldsymbol{s}$. Here, we consider the possibility of updating the old rankings $\boldsymbol{s}_0$ with new information from the additional games to get an approximation of $\boldsymbol{s}$. A successful online update will account for the following properties:

(i) *Coupled system*: Due to the coupled nature of the system, games between a pair of players, has the potential to not only affect the players' ranks, but also the ranking of other players in the system.

(ii) *Conserved system*: The rankings are computed from the Hamiltonian of a physical represen-

tation of the system. This means that the resulting rankings are conserved. In other words, the order in which the games are played should not affect the final rankings.

(iii) *Non-conserved system*: On the other hand, sometimes, we may desire the rankings to be dependent on the order of edge formation. An example of such a situation is where the players learn from the games and gain experience. Here, the ability to learn creates a friction that changes the underlying rank distribution. Another instance where the system may not be conserved is when a new node enters the system.

4.3 Updating the neighborhood

In this method, we propose to update the ranks of nodes in the neighborhood of the change, by fixing the ranks of all other nodes. Here, we define the k-th neighborhood of a node to be the set of all nodes in the network that are k edges away i.e., can be reached in k steps.

4.3.1 The update step

Let $\boldsymbol{s}_0$ be the original ranks, L_0 be the original Laplacian, and $\hat{\boldsymbol{d}}_0$ be the original difference of out- and in-degrees. Consider adding new edges between a subset of nodes, $P = \{x_1, x_2, \dots x_m\}$. Let L be the new Laplacian, $\hat{\boldsymbol{d}}$ be the new difference of out- and in-degrees. Let $\boldsymbol{\delta}$ denote the change in ranks.

As new notation, let $\mathcal{X}_Q$ denote the sampled version of $\mathcal{X}$ where we keep only elements (rows and columns) corresponding to the set Q and remove all others. So, L_P is the sampled Laplacian for nodes in P and $\boldsymbol{s}_P$ is the corresponding sampled ranks. Note that L_P is, still, a square matrix of dimension $|P| \times |P|$ and $\boldsymbol{s}_P$ is a vector of length $|P|$.

First, we will consider the case where we add new edges between nodes x and y. Then, we can generalize the results. Now, consider the new Hamiltonian

$$H(\boldsymbol{s}) = \frac{1}{2} \sum_{i,j} A_{ij}(s_i - s_j - 1)^2.$$

Recall that we are fixing the ranks of all nodes except for x and y. Therefore, when we want to minimize the new Hamiltonian, we only need to consider terms involving s_x and s_y. Let us look at the two partials,

$$\begin{aligned}
\frac{\partial H}{\partial s_x} &= \sum_i A_{xi}(s_x - s_i - 1) - \sum_i A_{ix}(s_i - s_x - 1) \\
&= s_x(d_x^{\text{out}} + d_x^{\text{in}}) - \sum_i (A_{xi} + A_{ix})s_i - d_x^{\text{out}} + d_x^{\text{in}} \\
&= (s_x - s_{0,x} + s_{0,x})d_x - (A_{xy} + A_{yx})(s_y - s_{0,y}) - ((A + A^T)\boldsymbol{s}_0)_x - \hat{d}_x \\
&= d_x\delta_x - (A_{xy} + A_{yx})\delta_y + (L\boldsymbol{s}_0)_x - \hat{d}_x. \\
\text{Similarly, } \frac{\partial H}{\partial s_y} &= d_y\delta_y - (A_{yx} + A_{xy})\delta_x + (L\boldsymbol{s}_0)_y - \hat{d}_y.
\end{aligned}$$

After setting the partials to 0 (to minimize H), we can visualize the equations as the following linear system:

$$\begin{bmatrix} d_x & -(A_{xy} + A_{yx}) \\ -(A_{yx} + A_{xy}) & d_y \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} (\hat{\boldsymbol{d}} - L\boldsymbol{s}_0)_x \\ (\hat{\boldsymbol{d}} - L\boldsymbol{s}_0)_y \end{bmatrix}. \tag{4.1}$$

Now, observe that the 2×2 matrix in Equation 4.1 is exactly the the Laplacian L but with only the rows and columns corresponding to nodes x and y. The same is true for the vector on the right hand side – it is the vector $\hat{\boldsymbol{d}} - L\boldsymbol{s}_0$ but with only elements corresponding to nodes x and y. Essentially, this is the sampled version of the original SpringRank model, concerning only the nodes between which we added new edges. So, in the case where we add edges between nodes in the set P, Equation 4.1 generalizes to

$$L_P\boldsymbol{\delta}_P = (\hat{\boldsymbol{d}} - L\boldsymbol{s}_0)_P,$$

where $\mathcal{X}_Q$ is the sampled version of $\mathcal{X}$. Note that, without loss of generality, we can set P to be any arbitrary subset of the nodes. So, we can set P to be the k-th neighborhood of the nodes x, y.

As a final observation, note that if P is the entire set of nodes in the network, we have

$$L\boldsymbol{\delta} = \hat{\boldsymbol{d}} - L\boldsymbol{s}_0$$

$$\implies L(\boldsymbol{s}_0 + \boldsymbol{\delta}) = \hat{\boldsymbol{d}}$$

$$\implies L\boldsymbol{s} = \hat{\boldsymbol{d}},$$

and we recover the full SpringRank problem.

We can encompass these results into the following update steps:

$$L_P \boldsymbol{\delta}_P = (\hat{\boldsymbol{d}} - L\boldsymbol{s}_0)_P \qquad \text{(Step (i))} \tag{4.2}$$

$$\boldsymbol{s}_P = \boldsymbol{s}_{0,P} + \boldsymbol{\delta}_P \qquad \text{(Step (ii))} \tag{4.3}$$

$$\boldsymbol{s}_{\bar{P}} = \boldsymbol{s}_{0,\bar{P}}, \qquad \text{(Step (iii))} \tag{4.4}$$

where $\bar{P}$ is the complement of P.

4.3.2 One-pass update algorithm

Using Equations 4.2 – 4.4, we have a one-pass update algorithm described in Algorithm 1.

Algorithm 1: One-pass update algorithm for online ranking

Input : Original network A_0
Original ranks $\boldsymbol{s}_0$
New edges ΔA
Subset of nodes between which new edges were added P
Neighborhood k

$A = A_0 + \Delta A$
$L =$ Laplacian(A)
$\boldsymbol{s} = \boldsymbol{s}_0$
$\hat{\boldsymbol{d}} =$ Difference out- and in-degrees of A
$N = k$-th neighborhood of nodes in P
Solve for $\boldsymbol{\delta}_N$, $L_N \boldsymbol{\delta}_N = (\hat{\boldsymbol{d}} - L\boldsymbol{s}_0)_N$
$\boldsymbol{s}_N = \boldsymbol{s}_N + \boldsymbol{\delta}_N$

Output : Updated ranks $\boldsymbol{s}$

This algorithm does not perform as expected. It seems that when we fix the boundary conditions (fix ranks of nodes outside the neighborhood), the ranks of the nodes inside the neighborhood change drastically in order to minimize the Hamiltonian. This is also interesting because in our experiments, we observed that the ranks of the nodes outside the neighborhood do not change at all! We conjecture that this could be due to the one-pass update algorithm creating a subsystem

whose Hamiltonian we are trying to minimize. And since this subsystem is closed (to nodes from outside the neighborhood), minimizing the Hamiltonian results in a different solution to the overall problem.

In the following simulation, we generated a network with 10,000 nodes. We computed the original ranks of the nodes using SpringRank. Then, we randomly chose two nodes and added a directed edge between them. We computed the new ranks using SpringRank. Then, we used the one-pass update algorithm to compute the online ranks by changing the neighborhood degree from $k = 0$ (just the two nodes) to $k = 7$. $k = 7$ was chosen as an upper bound because it generally captured the entire network. The results from this simulation are shown in Figures 4.1, 4.2

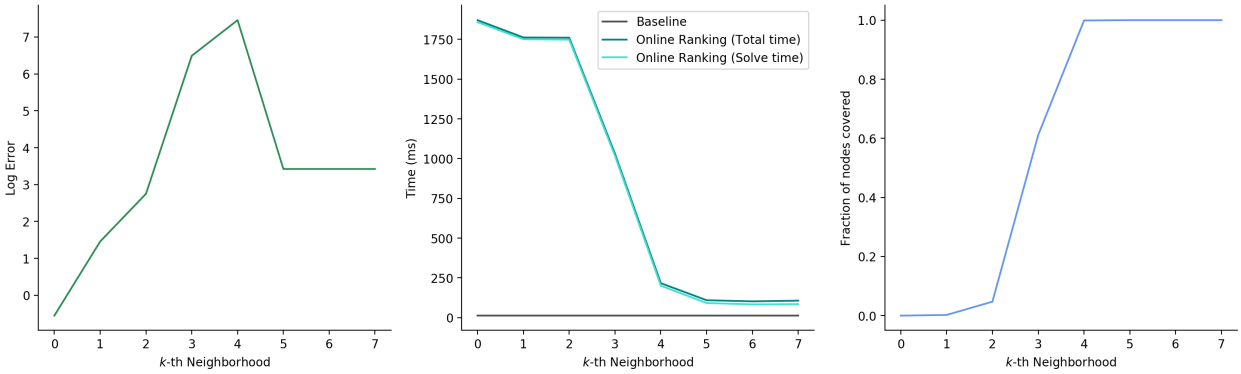


Figure 4.1: Accuracy and runtime analysis for Algorithm 1.

In Figure 4.1, the left panel shows the log error between estimates of ranks using Algorithm 1 and SpringRank for each neighborhood. The error was computed as the L_2 norm of the difference between the SpringRank estimate and the online estimate. Notice that the error first increases before decreasing. The increase can be attributed to the fixed boundary conditions. The error decreases when the entire network is captured in the neighborhood. This situation is exactly the same as plain SpringRank. The middle panel shows the runtime for the algorithm. The gray line denotes the baseline from using SpringRank to compute the ranks and the green-blue line shows the runtime of Algorithm 1. A further decomposition shows the solve time decreases with a larger neighborhood, but the neighbor collection process becomes more expensive. This decrease is due to

the usage of sparse solvers which perform faster than traditional solvers when the network is very large with few non-zero entries. The right panel shows the fraction of the network covered with each neighborhood.

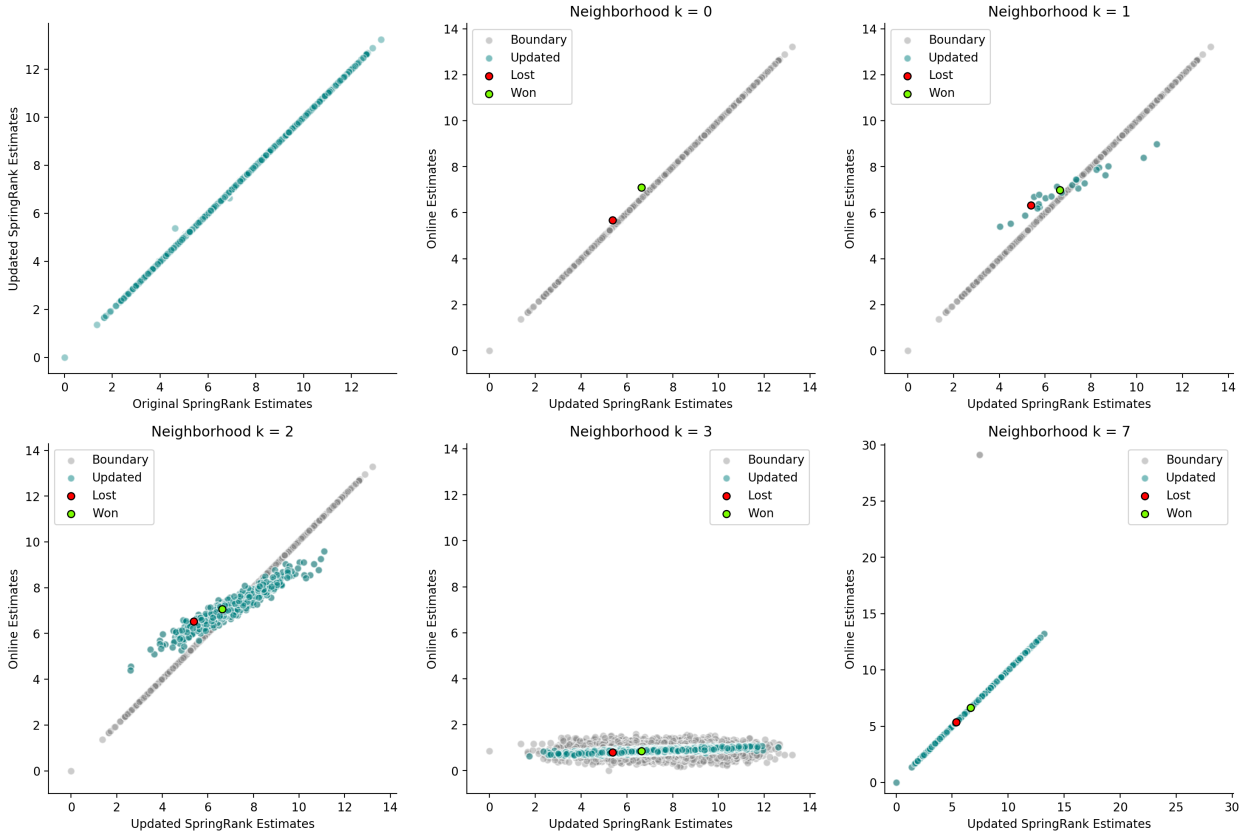


Figure 4.2: Change in ranks as we vary k in Algorithm 1.

In Figure 4.2, the top-left plot shows the ranks of the nodes after and before the new node was introduced. Both these estimates are calculated using SpringRank. In each subsequent plot, the estimates from Algorithm 1 (for each k) are plotted against the SpringRank estimates. The green and red nodes are the winning and losing nodes respectively. The grey nodes are the boundary nodes i.e., their ranks are fixed. The green-blue nodes are nodes that belong to that corresponding neighborhood, and are therefore updated. As we can see, the fixed boundary conditions change the ranks of nodes inside the neighborhood drastically in order to minimize the Hamiltonian. Once the neighborhood covers the entire network, however, our results completely agree with the estimates

from SpringRank.

4.3.3 Recursive update algorithm

Using Equations 4.2 – 4.4 we can also design an algorithm to recursively update the ranks. This algorithm is defined in Algorithm 2. Here the convergence can be defined by some tolerance ϵ and we declare convergence if $\frac{||\boldsymbol{s}^{(i)}-\boldsymbol{s}^{(i-1)}||_2}{||\boldsymbol{s}^{(i-1)}||_2} < \epsilon$, where $\boldsymbol{s}^{(i)}$ is the updated ranks after iteration i.

Algorithm 2: Recursive update algorithm for online ranking

Input : Original network A_0
Original ranks $\boldsymbol{s}_0$
New edges ΔA
Subset of nodes between which new edges were added P

$A = A_0 + \Delta A$
$L =$ Laplacian(A)
$\boldsymbol{s} = \boldsymbol{s}_0$
$\hat{\boldsymbol{d}} =$ Difference out- and in-degrees of A
Solve for $\boldsymbol{\delta}_P$, $L_P\boldsymbol{\delta}_P = (\hat{\boldsymbol{d}} - L\boldsymbol{s}_0)_P$
$\boldsymbol{s}_P = \boldsymbol{s}_P + \boldsymbol{\delta}_P$
$N =$ neighbors of P in A
while $\boldsymbol{s}$ **not coverged do**
  Solve for $\boldsymbol{\delta}_N$, $L_N\boldsymbol{\delta}_N = (\hat{\boldsymbol{d}} - L\boldsymbol{s})_N$
  $\boldsymbol{s}_N = \boldsymbol{s}_N + \delta_N$
  $N =$ neighbors of N in A

Output : Updated ranks $\boldsymbol{s}$

The error and speed will depend on the number of nodes in P, $|P|$, as it determines the number of iterations until convergence. Generally, we expect the effect of change in s_x on a second-degree neighbor of x to be smaller than the effect on a first-degree neighbor, which will result in convergence. In fact, as $\epsilon \to 0$, this algorithm is asymptotically the same as SpringRank. However, this algorithm becomes computationally more expensive due to the recursive nature.

In the following simulation, we generated a network with 10000 nodes. We computed the original ranks of the nodes using SpringRank. Then, we randomly chose two nodes and added a directed edge between them. We computed the new ranks using SpringRank. Then, we used the recursive update algorithm to compute the online updates. The results are shown in Figure 4.3.

In Figure 4.3, the left panel shows the error as a function of the number of iterations. The

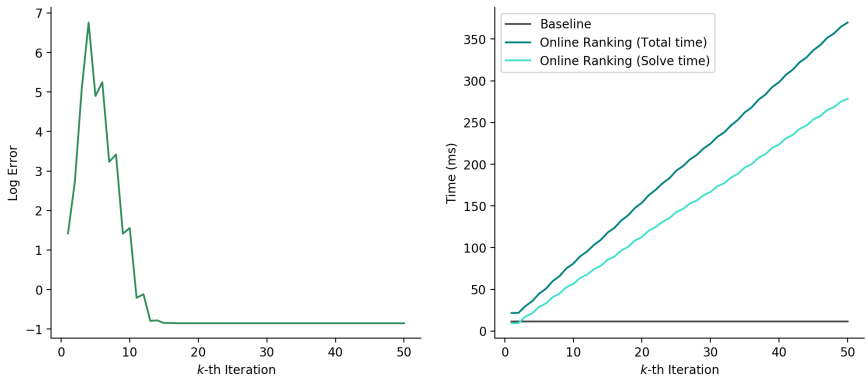


Figure 4.3: Accuracy and runtime analysis for Algorithm 2.

error is computed as the norm of the difference between the online and SpringRank estimates. As we can see, the error first increases, but then the online estimates converge to the SpringRank estimates asymptotically. The right panel shows the runtime as a function of the number of iterations. The gray line shows the time it takes SpringRank to solve the system i.e., the time we need to beat. Comparing both the plots, notice that while Algorithm 2 is accurate, it compromises on the speed. This shows that Algorithm 2 is actually worse than SpringRank.

4.4 Conclusion

We developed two algorithms that focused on updating the ranks in a particular neighborhood of the network. When the neighborhood spans the network, we saw that we recovered SpringRank. However, when updating the ranks for just a small neighborhood, the boundary conditions created a smaller system whose energy, the Hamiltonian, we minimize. And the solution we obtain by minimizing the Hamiltonian for the smaller system corresponds to a different solution to the problem. Therefore, when the neighborhood does not span the network, the updated ranks diverge from the expected ranks. Although the updated ranks converge to the expected ranks when the neighborhood spans the network, the algorithm becomes computationally more expensive than SpringRank. This suggests that we need to find alternative methods to accurately and efficiently update ranks in SpringRank.

Chapter 5

Conclusions and Future Directions

5.1 Conclusions

In the preceding chapters, we developed several insights into, and extensions of, SpringRank, a model for inferring a hierarchical ranking of nodes in a network. In Chapter 2, we made important connections between SpringRank, the Boltzmann distribution, Random Utility Models, and linear regression. This helped us gain a better understanding of how the model fits into the larger body of literature concerning these fields. In Chapter 3, we extended SpringRank in order to identify the effect of node covariates that may influence the outcome of the interactions. We saw that such a model is generally non-identifiable and found conditions under which the model becomes identifiable. Finally, in Chapter 4, we designed an online update algorithm to update the ranks, efficiently and accurately, as new information comes to light.

Chapter 2 was devoted to understanding SpringRank. SpringRank is a ranking algorithm to rank nodes in a network. The network can be thought of as a win-loss matrix representing the outcomes of interactions between nodes. SpringRank models the network as a physical system with edges representing springs and aims to minimize the energy of the system, given by the Hamiltonian. From statistical physics, this gives rise to the Boltzmann distribution for predicting new edges. The Boltzmann distribution simplifies to a multinomial logit model that is a commonly used Random Utility Model. We reconciled the two different models by observing that SpringRank makes a choice between the springs (the choices itself) as opposed to making a choice between the nodes (like in the multinomial logit model). In order to concretely justify SpringRank as a Random Utility Model,

we formulated SpringRank as a specific instance of the Context Dependent Random Utility Model, which is a generalization of Random Utility Models. Finally, we noted that SpringRank can be viewed as an ordinary least squares solution to a linear regression problem. By assuming that the noise is drawn from a multivariate Gaussian distribution, we recover results obtained from using the Boltzmann distribution to predict new edges. We concluded our discussion by noting that there are multiple perspectives to SpringRank, and that all of them offer the same insights.

In Chapter 3, we developed a series of three models in order to identify the effect of node characteristics on the outcome of interactions. In order to account for node characteristics, we sorted nodes into different groups, each of which provide an additional boost. To model this boost, which we called group preference, we attempted to decompose the rank s, into individual scores a, and the group preference θ. In the first model, we assumed that the groups of the nodes were fixed and that the group preference remained constant in all the interactions. In the second model, we introduced arbiters who determined the outcome of the interactions, and thus the group preferences could change with respect to the interactions. Both of these models are non-identifiable i.e., we cannot discern the individual score from the group preference. To recover identifiability, we developed a third model where the nodes can change group memberships in different interactions.

In Chapter 4, we designed an algorithm to continuously update ranks efficiently and accurately. Our algorithm updated a small neighborhood of the network centered around the change, keeping all other ranks constant. However, imposing these boundary conditions changed the behavior of the system, thereby causing the updated ranks to diverge from the expected ranks. In order for convergence, we noted that neighborhood should span the entire network. But then, it becomes computationally cheaper to simply perform SpringRank again instead of using our algorithm. Therefore, we concluded that we should not force boundary conditions on the physical system. While this formulation proved to be less efficient than SpringRank, our analyses reveal promising directions for future investigations.

5.2 Future directions

This thesis explored three important ideas to broaden the applicability of SpringRank to diverse problems. While the research answers many questions, it asks new questions in the process. Here, we summarize these questions for possible future work.

Chapter 2 closed by conclusively tying several approaches to SpringRank together. This does not imply that there is no future work left here. This actually questions the existence of other perspectives to SpringRank that can offer additional insights to our understanding. For instance, we discovered a family of utilities that simplify to the multinomial logit model, without fully exploring them. What models do each of these utilities correspond to? Another idea that was briefly mentioned was learning the feature vectors that the Context Dependent Random Utility Model proposes. It will be interesting to find closed form expressions for these feature vectors in order to better understand the model.

In Chapter 3, we developed models to identify the effect of group characteristics and concluded that the model is non-identifiable, unless in special cases. We introduced the *Equality assumption* in order to compare group preferences and noted that, in the absence of external information, this assumption can be flawed. A natural question to ask is, what other information is needed to recover identifiability? Following this, how do we encode this information into the model? Our inability to do so restrained us from applying our methods to real world applications to prevent incorrect interpretation of the results. Explicitly encoding the external information into our model will help us successfully apply our methods to real data and interpret the results accurately. This may help us quantify social inequalities, in the language of Random Utility Models, mentioned by [Clauset et al., 2015] in faculty hiring networks. This may also be applicable while studying social support networks [Power, 2017] and online social interactions [Bruch and Newman, 2019].

In Chapter 4, we designed an online update algorithm to update ranks by forcing boundary conditions and updating only a neighborhood in the network. We saw that this boundary condition changes the behavior of the system. Our attempts to update the ranks by only considering certain

neighborhoods of the network fails to be accurate *and* efficient. And this was a key requirement in order to beat SpringRank. Here, we propose four other avenues worth investigating in the space of online ranking.

Firstly, during our discussion, we noted that a change in rank of node x has a larger effect on its first-degree neighbors than second- or third-degree neighbors. We can leverage this to weigh the perturbations in ranks depending on how far they are from the local disturbance. Secondly, we assumed that the system is closed. However, adding an edge is an external interaction. Therefore, the system is most likely not closed. So, it makes sense to take a simulated annealing approach by increasing the energy of all springs and then allowing them to settle down to final ranks. As a third approach, let us assume that we have already computed and stored the inverse of the graph Laplacian L, L^{-1}. If the new edges added to the network is a low-rank perturbation to L, then we can exactly update L^{-1} using the matrix inversion lemma [Hager, 1989]. Using this updated inverse, we can compute the new ranks. Finally, we can take a stochastic approach. Whole History Ranking leverages the Bradley-Terry-Luce model to model the variations (changes) in rankings over time as a Wiener process [Coulom, 2008]. Then, a maximum likelihood estimation is performed to find the *best* change in rankings. As another approach, we can also model the ranks as an Ornstein-Uhlenbeck process, which is essentially a random walk with a drift to the long-term mean of the ranks.

In the end, our goal is developing tools to analyze complex systems. And we wish to use apply these tools to understand such systems that surround us. Answering these questions will give us further insights into uncovering the hierarchical structure of complex systems.

Bibliography

[Ali et al., 1986] Ali, I., Cook, W. D., and Kress, M. (1986). On the minimum violations ranking of a tournament. Management Science, 32(6):660–672.

[Amemiya, 1985] Amemiya, T. (1985). Advanced econometrics. Harvard university press.

[Baumann et al., 2010] Baumann, R., Matheson, V. A., and Howe, C. A. (2010). Anomalies in tournament design: The madness of march madness. Journal of Quantitative Analysis in Sports, 6(2).

[Bonacich, 1987] Bonacich, P. (1987). Power and centrality: A family of measures. American journal of sociology, 92(5):1170–1182.

[Bradley and Terry, 1952] Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika, 39(3/4):324–345.

[Bruch and Newman, 2018] Bruch, E. E. and Newman, M. (2018). Aspirational pursuit of mates in online dating markets. Science Advances, 4(8):eaap9815.

[Bruch and Newman, 2019] Bruch, E. E. and Newman, M. (2019). Structure of online dating markets in us cities. Sociological science, 6:219–234.

[Callaghan et al., 2003] Callaghan, T., Mucha, P. J., and Porter, M. A. (2003). Random walker ranking for ncaa division ia football. arXiv preprint physics/0310148.

[Chen and Joachims, 2016a] Chen, S. and Joachims, T. (2016a). Modeling intransitivity in matchup and comparison data. In Proceedings of the ninth acm international conference on web search and data mining, pages 227–236.

[Chen and Joachims, 2016b] Chen, S. and Joachims, T. (2016b). Predicting matchups and preferences in context. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 775–784.

[Clauset et al., 2015] Clauset, A., Arbesman, S., and Larremore, D. B. (2015). Systematic inequality and hierarchy in faculty hiring networks. Science advances, 1(1):e1400005.

[Colley, 2002] Colley, W. N. (2002). Colleys bias free college football ranking method: The colley matrix explained. Princeton University, Princeton.

[Coulom, 2008] Coulom, R. (2008). Whole-history rating: A bayesian rating system for players of time-varying strength. In International Conference on Computers and Games, pages 113–124. Springer.

[Courneya and Carron, 1992] Courneya, K. S. and Carron, A. V. (1992). The home advantage in sport competitions: a literature review. Journal of Sport & Exercise Psychology, 14(1).

[Cucuringu, 2016] Cucuringu, M. (2016). Sync-rank: Robust ranking, constrained ranking and rank aggregation via eigenvector and sdp synchronization. IEEE Transactions on Network Science and Engineering, 3(1):58–79.

[David, 1987] David, H. A. (1987). Ranking from unbalanced paired-comparison data. Biometrika, 74(2):432–436.

[De Bacco et al., 2018] De Bacco, C., Larremore, D. B., and Moore, C. (2018). A physical model for efficient ranking in networks. Science advances, 4(7):eaar8260.

[Drews, 1993] Drews, C. (1993). The concept and definition of dominance in animal behaviour. Behaviour, 125(3-4):283–313.

[Elo, 1978] Elo, A. E. (1978). The rating of chessplayers, past and present. Arco Pub.

[Fogel et al., 2016] Fogel, F., d'Aspremont, A., and Vojnovic, M. (2016). Spectral ranking using seriation. The Journal of Machine Learning Research, 17(1):3013–3057.

[Gelman and Hill, 2006] Gelman, A. and Hill, J. (2006). Data analysis using regression and multilevel/hierarchical models. Cambridge university press.

[Glickman, 1995] Glickman, M. E. (1995). The glicko system. Boston University, 16.

[Gupte et al., 2011] Gupte, M., Shankar, P., Li, J., Muthukrishnan, S., and Iftode, L. (2011). Finding hierarchy in directed online social networks. In Proceedings of the 20th international conference on World wide web, pages 557–566.

[Hager, 1989] Hager, W. W. (1989). Updating the inverse of a matrix. SIAM review, 31(2):221–239.

[Herbrich et al., 2007] Herbrich, R., Minka, T., and Graepel, T. (2007). Trueskill: a bayesian skill rating system. In Advances in neural information processing systems, pages 569–576.

[Hobson and DeDeo, 2015] Hobson, E. A. and DeDeo, S. (2015). Social feedback and the emergence of rank in animal society. PLoS computational biology, 11(9).

[Huang et al., 2008] Huang, T.-K., Lin, C.-J., and Weng, R. C. (2008). Ranking individuals by group comparisons. Journal of Machine Learning Research, 9(Oct):2187–2216.

[Jones, 2007] Jones, M. B. (2007). Home advantage in the nba as a game-long process. Journal of Quantitative Analysis in Sports, 3(4).

[Letizia et al., 2018] Letizia, E., Barucca, P., and Lillo, F. (2018). Resolution of ranking hierarchies in directed networks. PloS one, 13(2).

[Lidor et al., 2010] Lidor, R., Bar-Eli, M., Arnon, M., and Bar-Eli, A. A. (2010). On the advantage of playing the second game at home in the knock out stages of european soccer cup competitions. International Journal of Sport and Exercise Psychology, 8(3):312–325.

[Luce, 1959] Luce, R. D. (1959). On the possible psychophysical laws. Psychological review, 66(2):81.

[McFadden et al., 1977] McFadden, D., Tye, W. B., and Train, K. (1977). An application of diagnostic tests for the independence from irrelevant alternatives property of the multinomial logit model. Institute of Transportation Studies, University of California Berkeley.

[Negahban et al., 2017] Negahban, S., Oh, S., and Shah, D. (2017). Rank centrality: Ranking from pairwise comparisons. Operations Research, 65(1):266–287.

[Noh, 2003] Noh, J. D. (2003). Exact scaling properties of a hierarchical network model. Physical Review E, 67(4):045103.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[Power, 2017] Power, E. A. (2017). Social support networks and religiosity in rural south india. Nature Human Behaviour, 1(3):0057.

[Ravasz and Barabási, 2003] Ravasz, E. and Barabási, A.-L. (2003). Hierarchical organization in complex networks. Physical review E, 67(2):026112.

[Seshadri et al., 2019] Seshadri, A., Peysakhovich, A., and Ugander, J. (2019). Discovering context effects from raw choice data. arXiv preprint arXiv:1902.03266.

[Sismanis, 2010] Sismanis, Y. (2010). How i won the" chess ratings-elo vs the rest of the world" competition. arXiv preprint arXiv:1012.4571.

[Slater, 1961] Slater, P. (1961). Inconsistencies in a schedule of paired comparisons. Biometrika, 48(3/4):303–312.

[Slaughter and Koehly, 2016] Slaughter, A. J. and Koehly, L. M. (2016). Multilevel models for social networks: hierarchical bayesian approaches to exponential random graph modeling. Social Networks, 44:334–345.

[Szymanski, 2003] Szymanski, S. (2003). The economic design of sporting contests. Journal of economic literature, 41(4):1137–1187.

[Train, 2009] Train, K. E. (2009). Discrete choice methods with simulation. Cambridge university press.

[Weng and Lin, 2011] Weng, R. C. and Lin, C.-J. (2011). A bayesian approximation method for online ranking. Journal of Machine Learning Research, 12(Jan):267–300.

Appendix A

Supplementary Results to Chapter 3

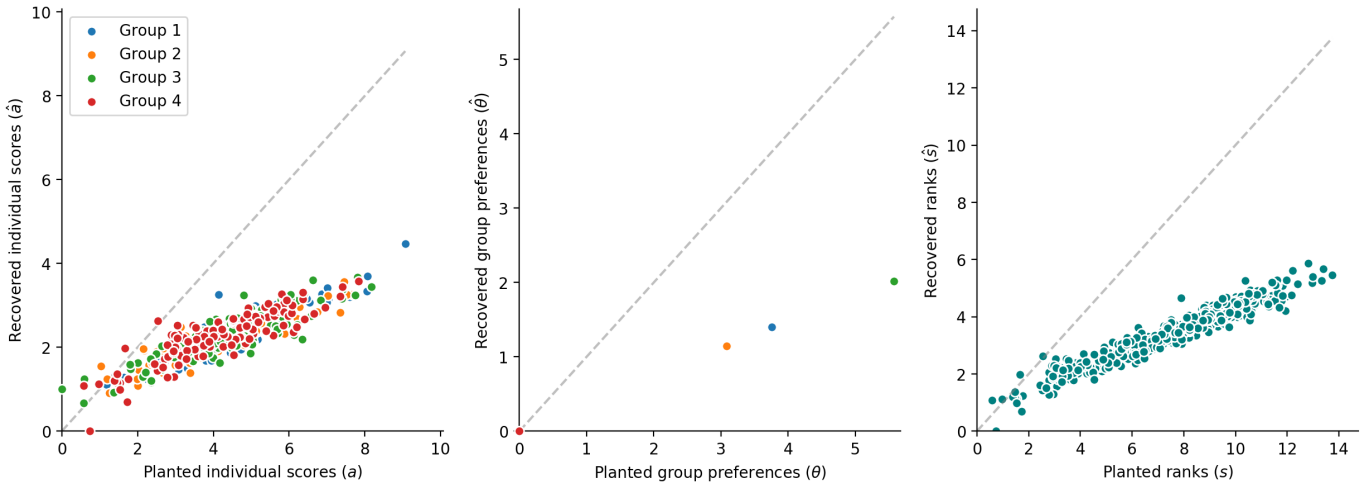
A.1 Recovering group preferences (without arbiters)

For each subfigure in the following plots, the left panel plots the recovered individual scores $\hat{a}$ against the planted individual scores a. The middle panel plots the recovered group preferences $\hat{\theta}$ against the planted group preferences θ. And the right panel plots the recovered ranks $\hat{s}$ against the planted ranks s. The top subfigure corresponds to the solution when regularization was used. The bottom subfigure shows the solution when setting the average individual scores across all groups to 0. Note that the final rank $\hat{s}$ remains unchanged. This illustrates the non-identifiability problem which is discussed in detail in Section 3.4.

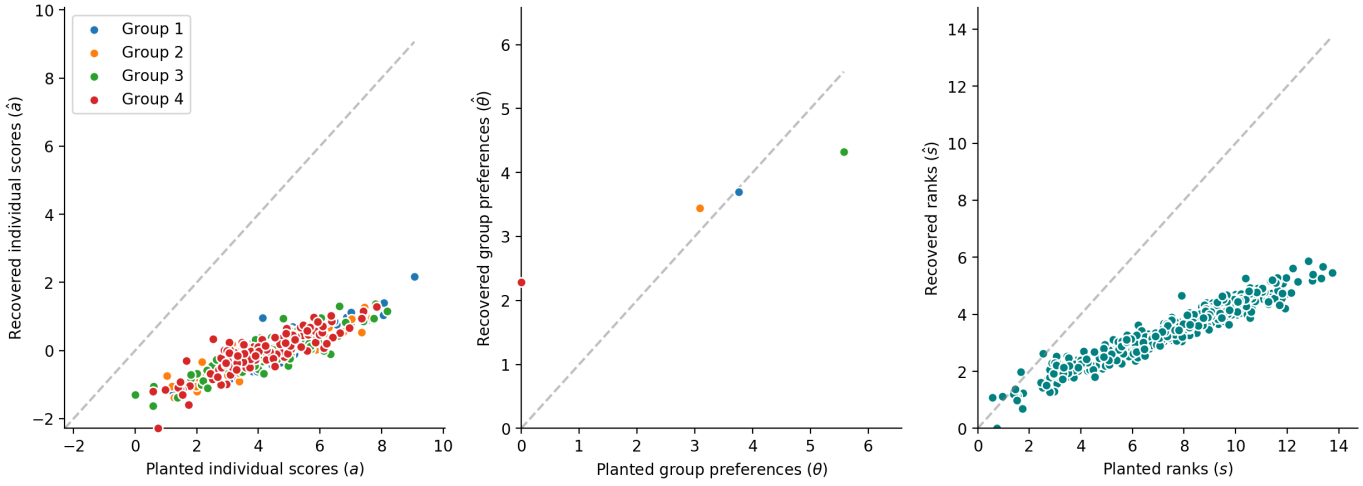
For Test 3, we generated 500 nodes with individual scores drawn from $Normal(1, \frac{5}{2})$. There were 4 groups with group preferences drawn from $Normal(2, \frac{5}{2})$. Here, $\beta = 0.4$ and $\langle k \rangle = 10$. The results are shown in Figure A.1.

For Test 4, we generated 500 nodes with individual scores drawn from $Normal(0, 0.2)$. There were 4 groups with group preferences drawn from $Normal(0, 0.2)$. Here, $\beta = 5$ and $\langle k \rangle = 10$. The results are shown in Figure A.2.

For Test 5, we generated 600 nodes. 200 individual scores were drawn from $Normal(0, 1)$, 200 from $Normal(1, \frac{5}{2})$, and 200 from $Normal(4, 1)$. There were 4 groups with group preferences drawn from $Normal(1, 1)$. Here, $\beta = 0.4$ and $\langle k \rangle = 10$. The results are shown in Figure A.3.

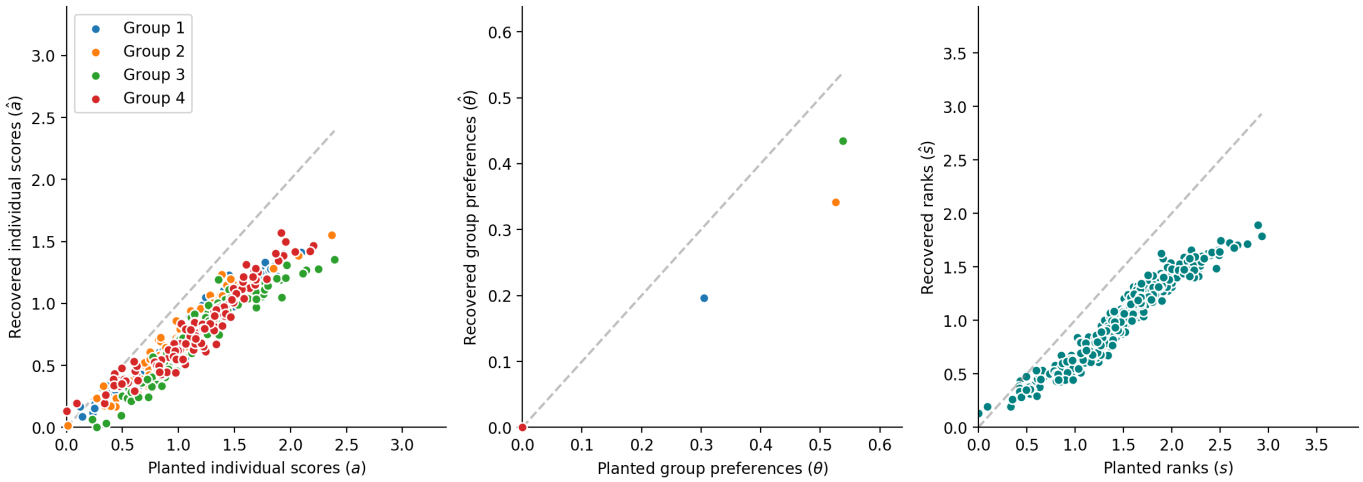


(a) Solution using regularization.

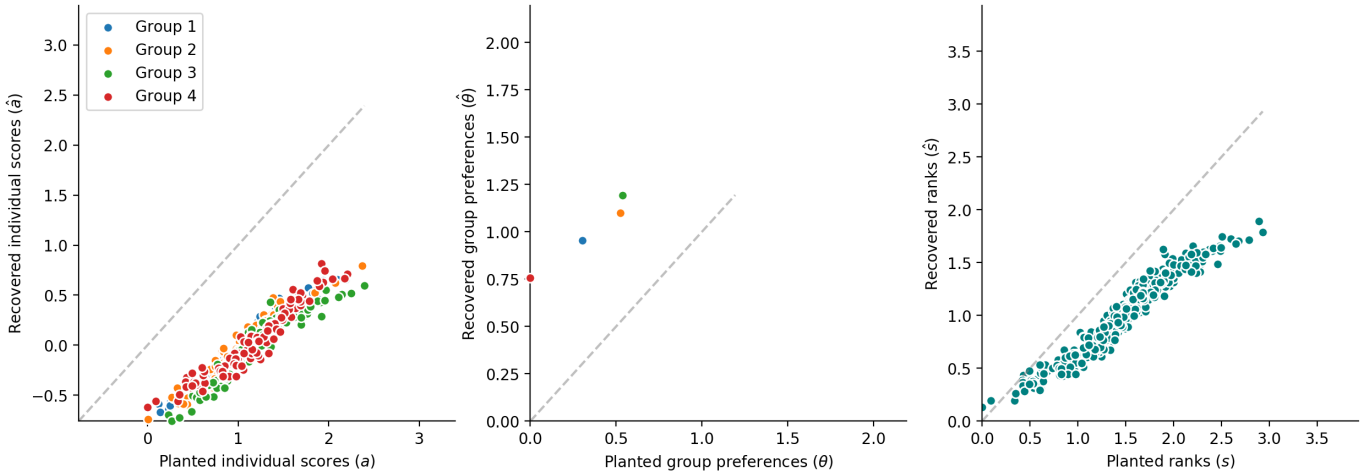


(b) Setting average individual scores across groups to 0.

Figure A.1: Results from Test 3 with synthetic data (without arbiters). $N = 500$; $a_{\text{planted}} \sim Normal(1, \frac{5}{2})$; 4 groups; $\theta_{\text{planted}} \sim Normal(2, \frac{5}{2})$. The individual scores correspond to a_{planted} and group preferences refer to θ_{planted}. The ranks refer to the total score of each node (individual and group preference) i.e., s_{planted}. Here, $\beta = 0.4$, $\langle k \rangle = 10$.

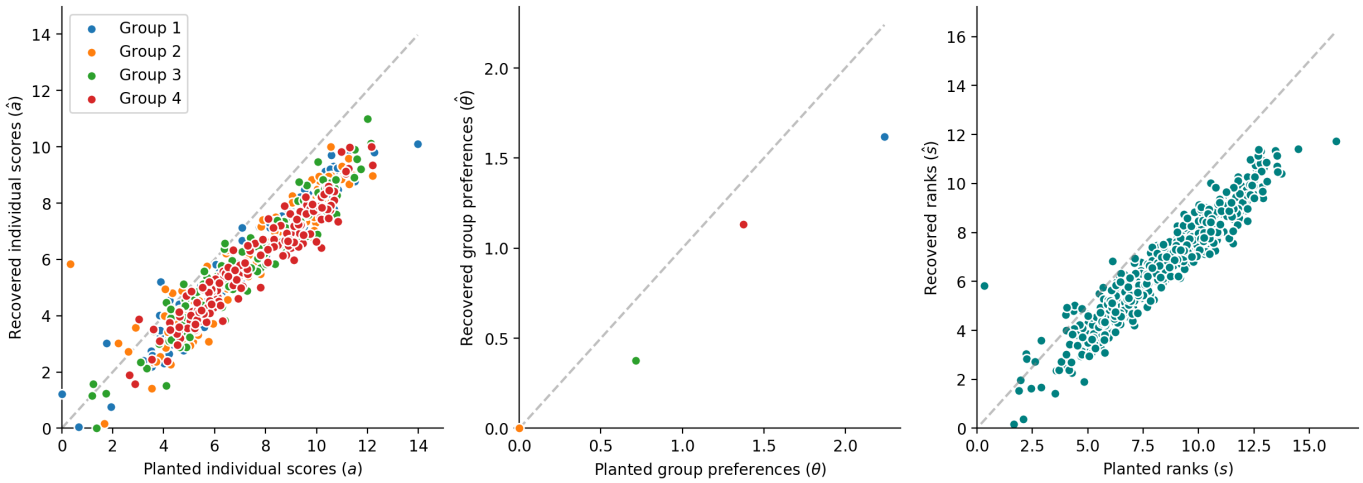


(a) Solution using regularization.

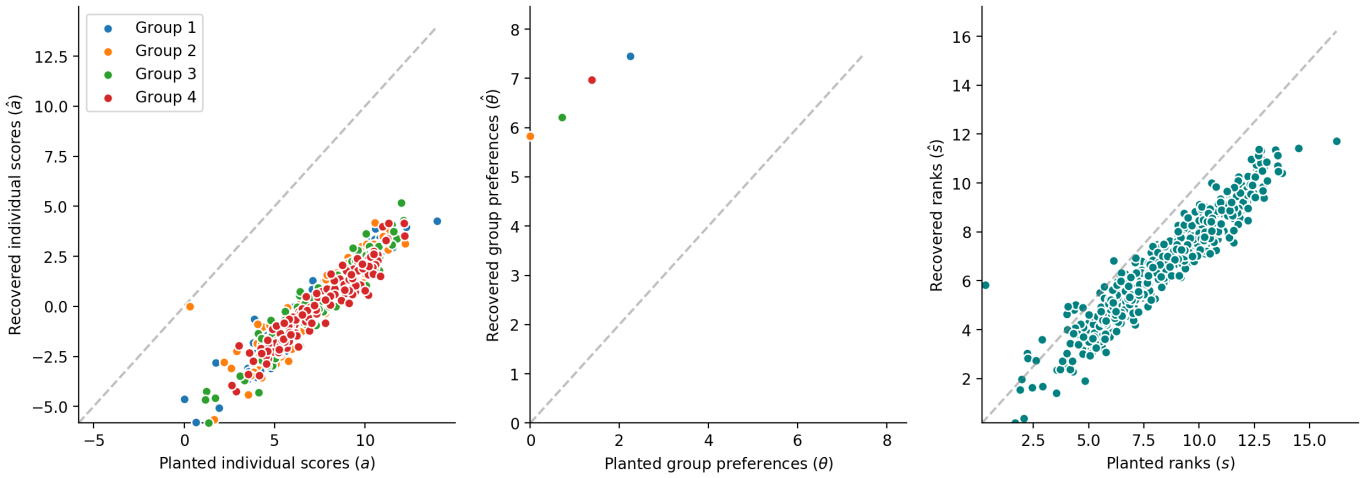


(b) Setting average individual scores across groups to 0.

Figure A.2: Results from Test 4 with synthetic data (without arbiters). $N = 500$; $a_{\text{planted}} \sim Normal(0, 0.2)$; 4 groups; $\theta_{\text{planted}} \sim Normal(0, 0.2)$. The individual scores correspond to a_{planted} and group preferences refer to θ_{planted}. The ranks refer to the total score of each node (individual and group preference) i.e., s_{planted}. Here, $\beta = 5$, $\langle k \rangle = 10$.



(a) Solution using regularization.



(b) Setting average individual scores across groups to 0.

Figure A.3: Results from Test 5 with synthetic data (without arbiters). $N = 600$: 200 from $Normal(0, 1)$; 200 from $Normal(1, \frac{5}{2})$; 200 from $Normal(4, 1)$; 4 groups; $\theta_{\text{planted}} \sim Normal(1, 1)$. The individual scores correspond to a_{planted} and group preferences refer to θ_{planted}. The ranks refer to the total score of each node (individual and group preference) i.e., s_{planted}. Here, $\beta = 0.4$, $\langle k \rangle = 10$.

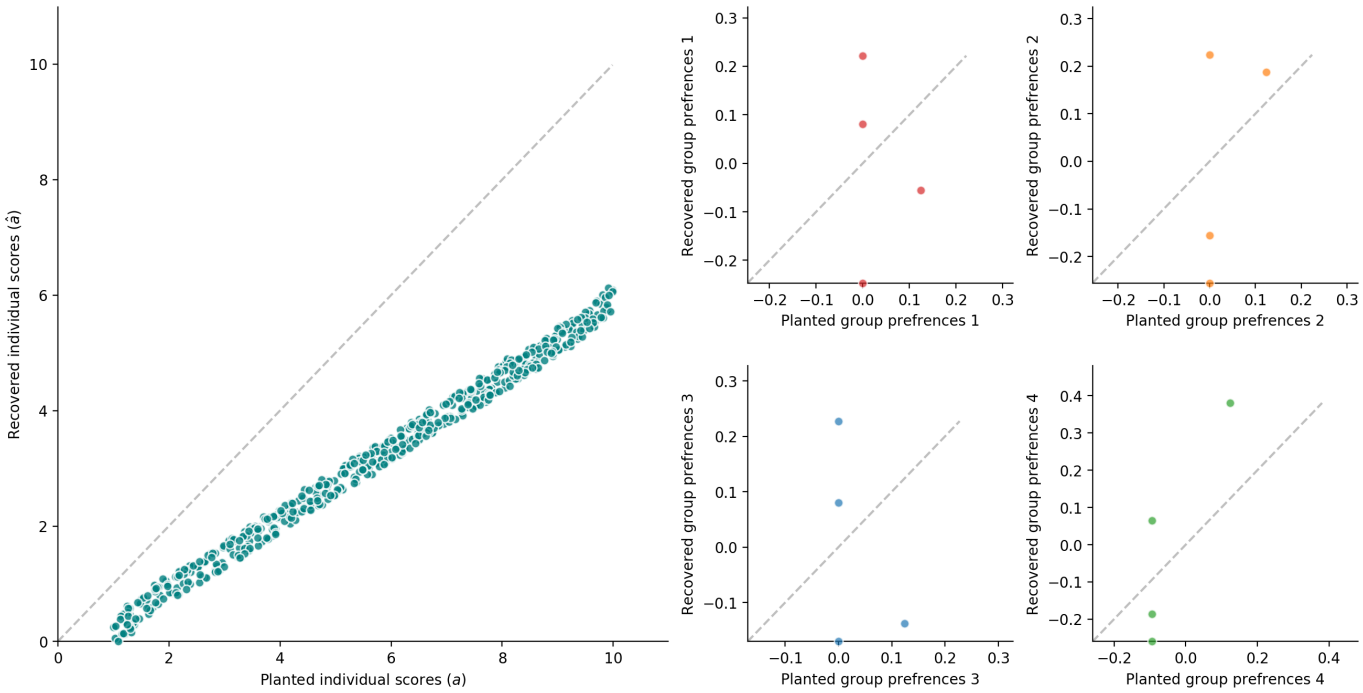
A.2 Recovering group preferences (with arbiters)

The left panel in each subfigure plots the recovered individual scores against the planted individual scores. Each of the other four panels plot the recovered group preferences against the planted group preferences, for each group of arbiters. The top subfigure shows the solution obtained using regularization. The bottom subfigure shows the solution when setting the average individual scores across all groups to 0.

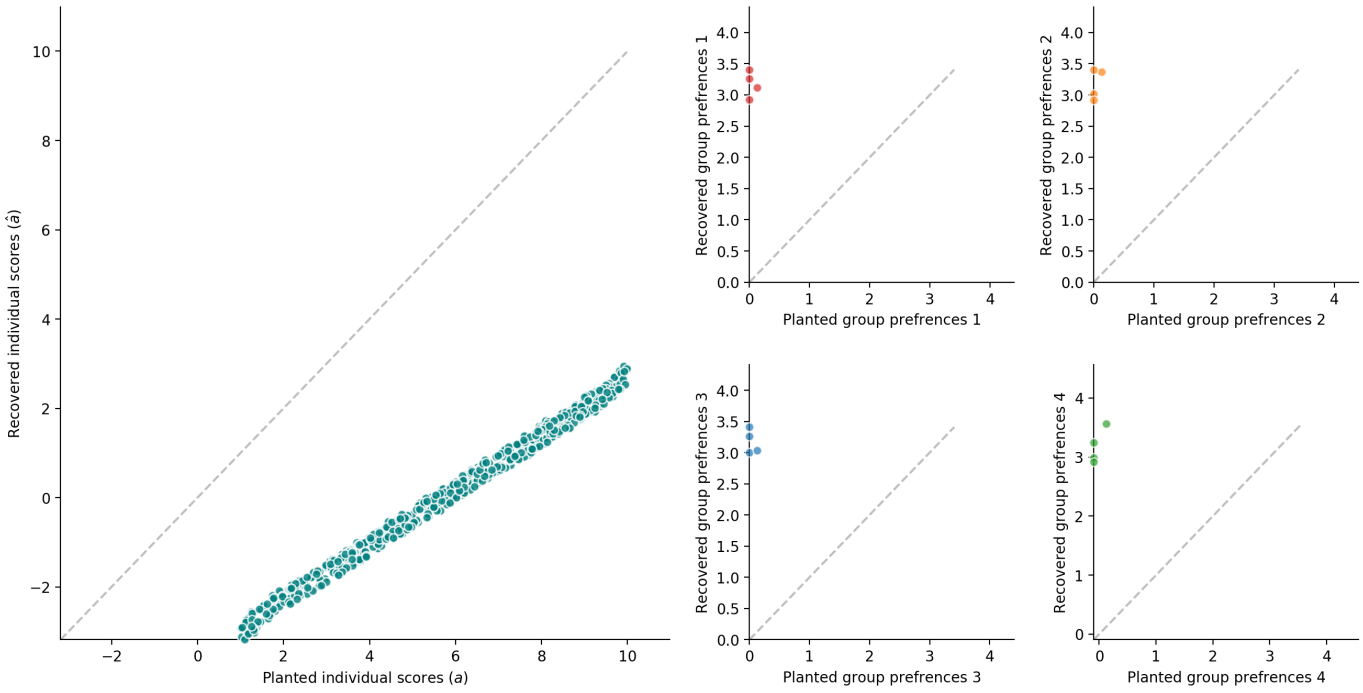
For Test 3, we drew 500 individual scores for items from $Uniform(1, 10)$. We fixed 4 groups for the items and the arbiters. There is a systematic bias against the fourth group of items. There is a systematic bias against the fourth group of items i.e., a negative group preference drawn from a $Normal(0, 1)$. For others, arbiters from group i had a positive preference, also drawn from $Normal(0, 1)$ towards items from group i, and no preference for other items. Here, $\beta = 5$. The results are shown in Figure A.4.

For Test 4, we drew 600 individual scores for items from $Normal(0, 1)$. We fixed 4 groups for the items and the arbiters. The group preference matrix was drawn from $Uniform(-5, 5)$. Here, $\beta = 0.5$. The results are shown in Figure A.5.

For Test 5, we drew 600 individual scores for items from $Normal(2, 10)$. We fixed 4 groups for the items and the arbiters. The group preference matrix was drawn from $Uniform(-2, 2)$. Here, $\beta = 1$. The results are shown in Figure A.6.

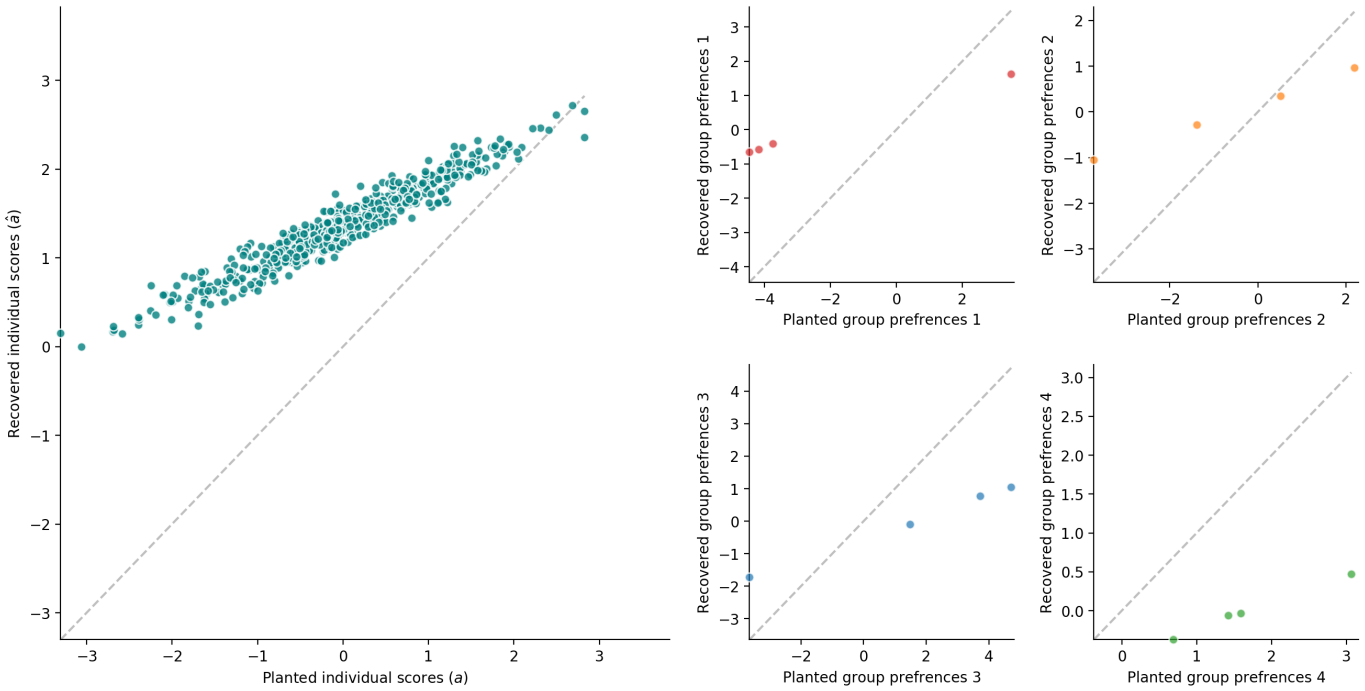


(a) Solution using regularization.

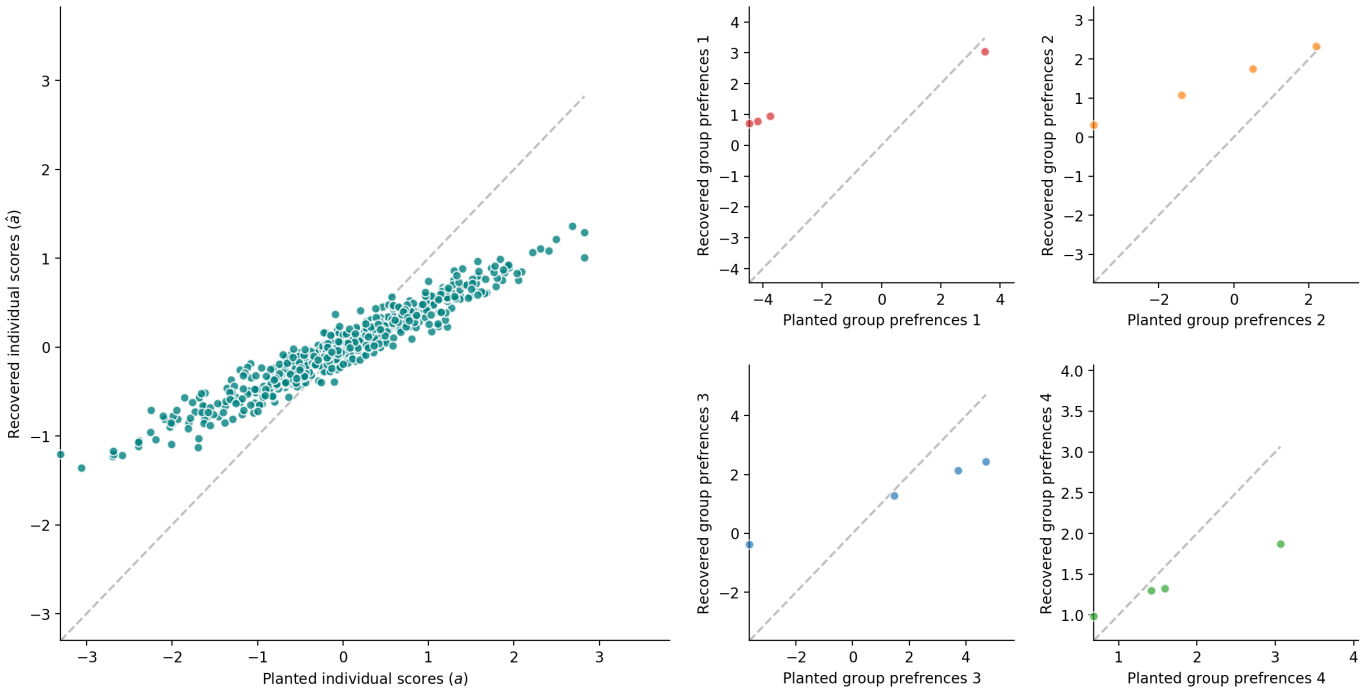


(b) Setting average individual scores across groups to 0.

Figure A.4: Results from Test 3 with synthetic data (with arbiters). 500 items were drawn from $Uniform(1, 10)$. There are 4 groups for items and arbiters. There is a systematic bias against the fourth group of items i.e., a negative group preference drawn from a $Normal(0, 1)$. For others, arbiters from group i had a positive preference towards items from group i, and no preference for other items. Here, $\beta = 5$.

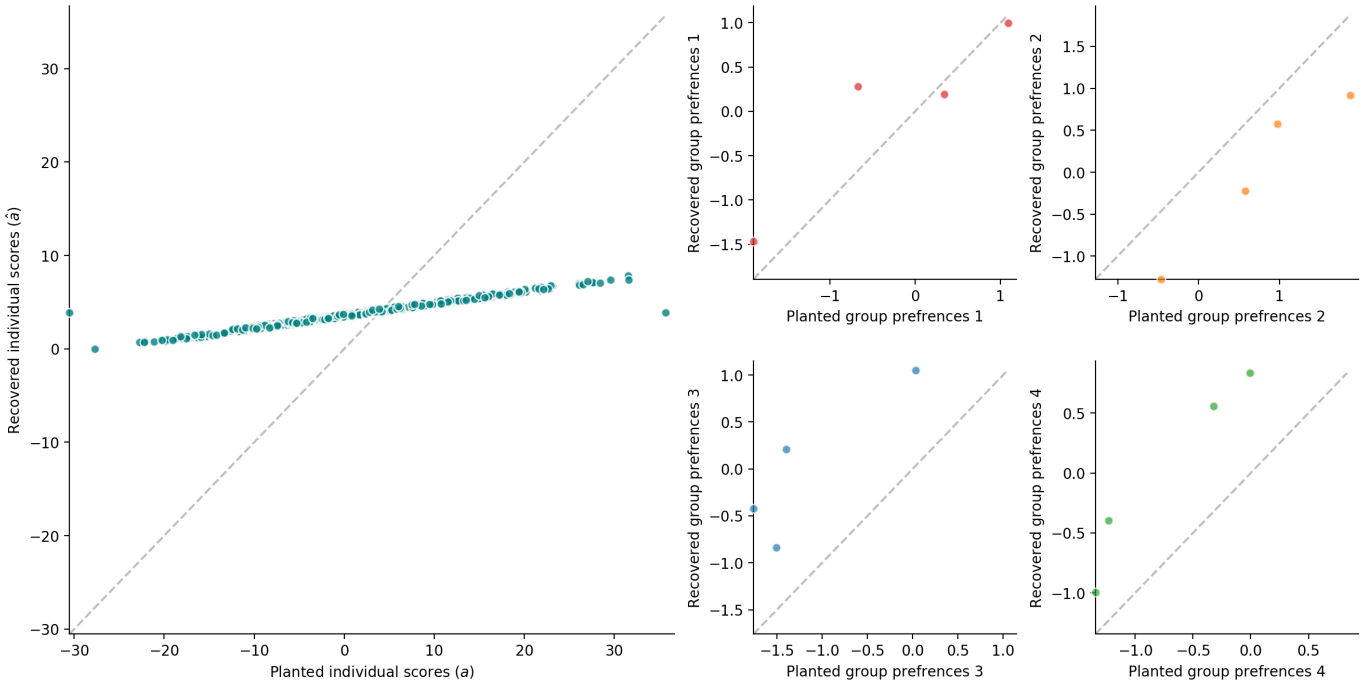


(a) Solution using regularization.

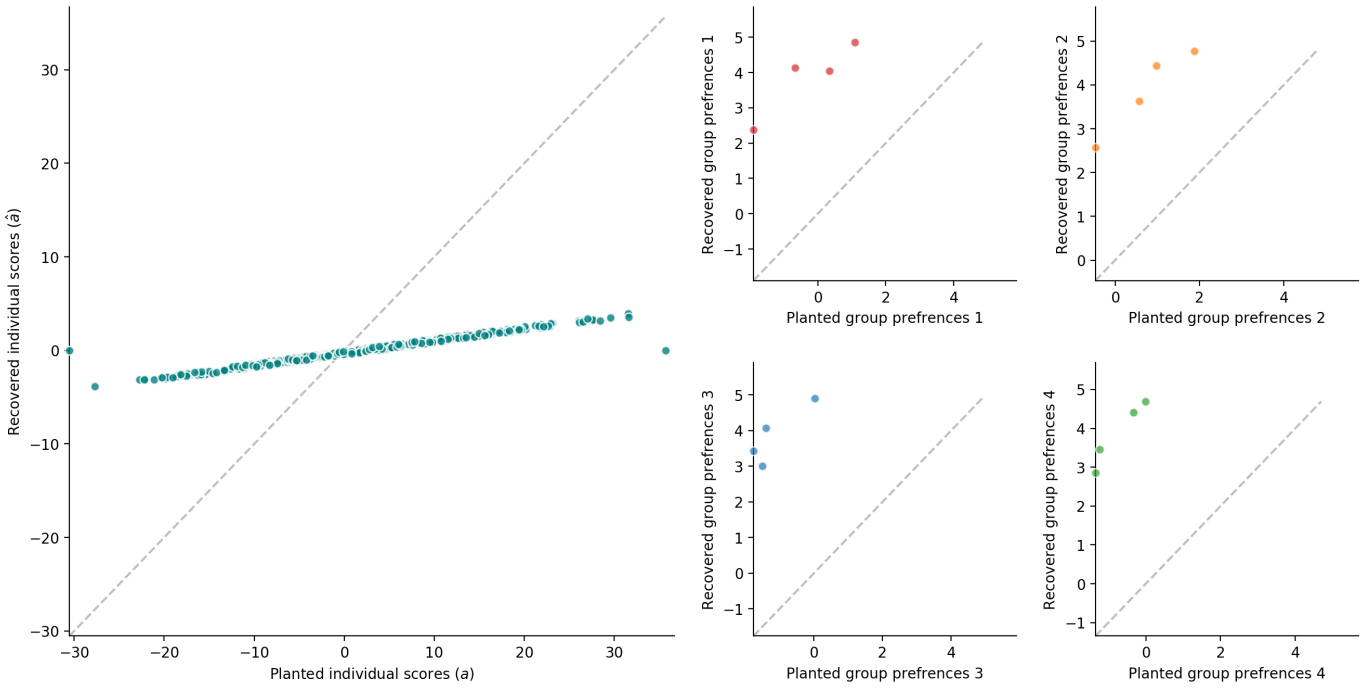


(b) Setting average individual scores across groups to 0.

Figure A.5: Results from Test 4 with synthetic data (with arbiters). 600 items from $Normal(0, 1)$. There are 4 groups for items and arbiters. Group preferences were drawn from $Uniform(-5, 5)$. Here, $\beta = 0.5$.



(a) Solution using regularization.



(b) Setting average individual scores across groups to 0.

Figure A.6: Results from Test 5 with synthetic data (with arbiters). 600 items from $Normal(2, 10)$. There are 4 groups for items and arbiters. Group preferences were drawn from $Uniform(-2, -2)$. Here, $\beta = 1$.

A.3 Recovering group preferences (identifiable models)

In each of the following figures, the left panel plots the recovered individual scores against the planted individual scores. And the right panel plots the recovered group preferences against the planted group preferences.

For Test 3, we drew 100 nodes from $Normal(1, \frac{5}{2})$, $Normal(0, 1)$, and $Normal(-2, 2)$ each for a total of 300 nodes. There were two groups and the group preferences was $[0.5, 0]$. Under the two groupings, network was generated with $\langle k^{(1)} \rangle = 20$ and $\langle k^{(2)} \rangle = 5$. Here, $\beta = 0.4$. The results are shown in Figure A.7.

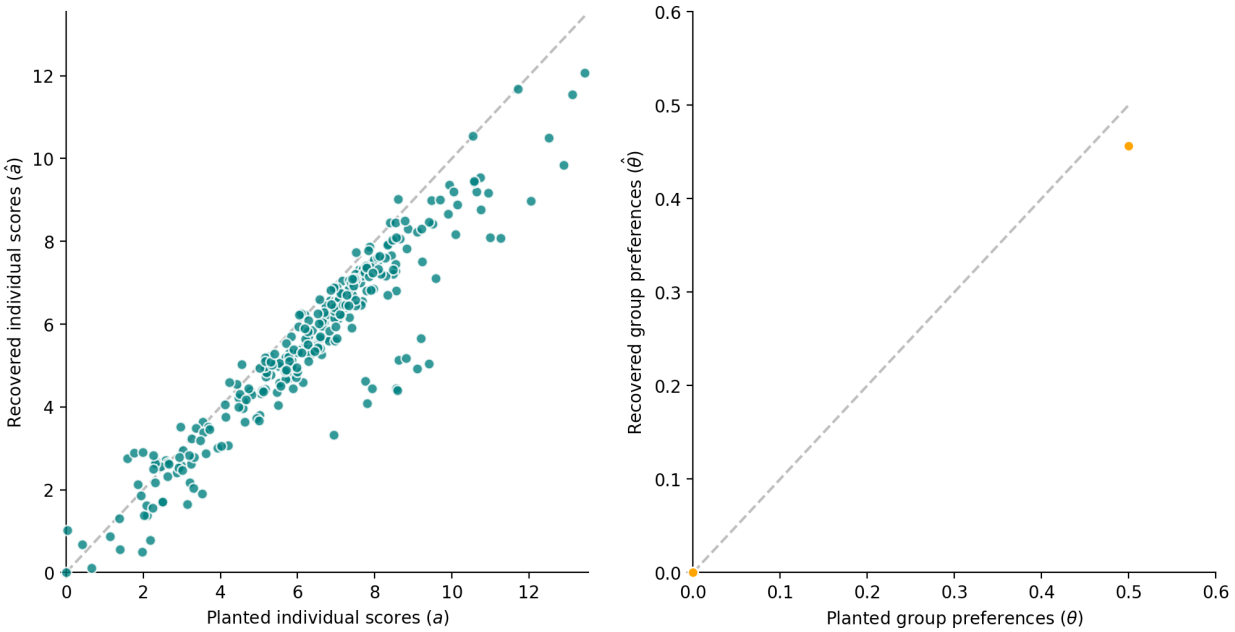


Figure A.7: Results on identifiability with synthetic data - Test 3. 100 nodes drawn from $Normal(1, \frac{5}{2})$, $Normal(0, 1)$, and $Normal(-2, 2)$ each for a total of 300 nodes. There were two groups and the group preferences was $[0.5, 0]$. Under the two groupings, network was generated with $\langle k^{(1)} \rangle = 20$ and $\langle k^{(2)} \rangle = 5$. Here, $\beta = 0.4$.

For Test 4, we drew 200 nodes from $Normal(3, \frac{5}{2})$, $Normal(0, 1)$, and $Normal(-3, 2)$ each for a total of 600 nodes. There were two groups and the group preferences was drawn from $Uniform(-3, 3)$. We allowed only opposite groups to interact with each other. Further, in the second grouping, nodes were assigned groups opposite from their first assignment. Under the two

groupings, network was generated with $\langle k^{(1)} \rangle = 10$ and $\langle k^{(2)} \rangle = 10$. Here, $\beta = 0.4$. The results are shown in Figure A.8.

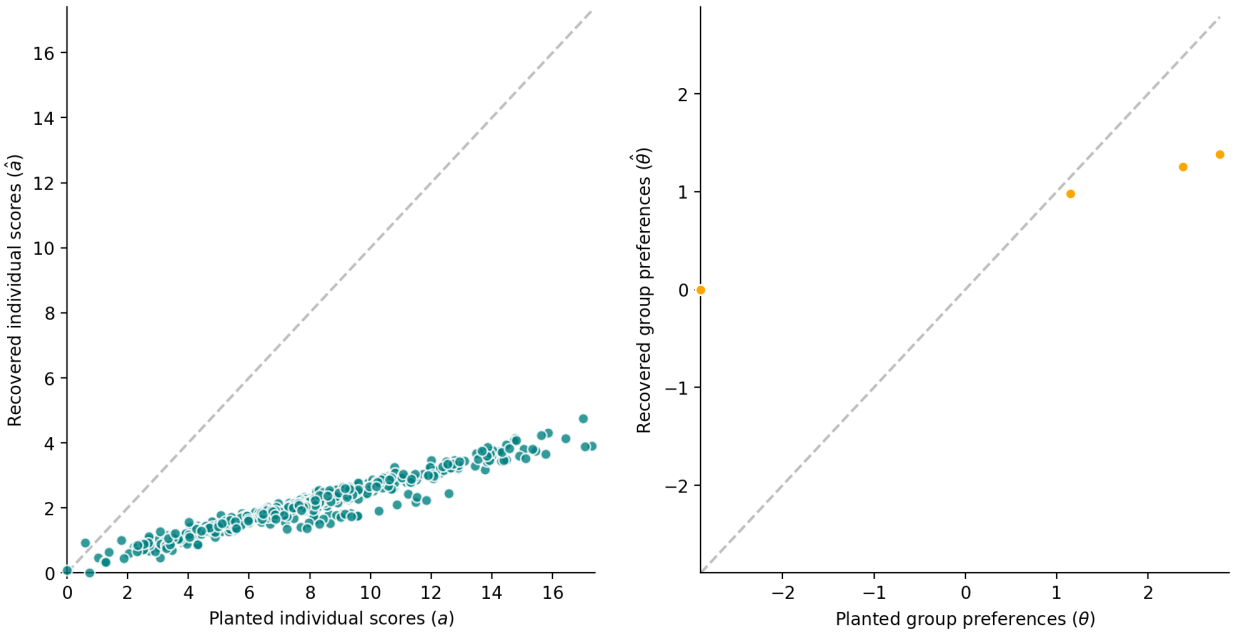


Figure A.8: Results on identifiability with synthetic data - Test 4. 200 nodes drawn from $Normal(3, \frac{5}{2})$, $Normal(0, 1)$, and $Normal(-3, 2)$ each for a total of 600 nodes. There were two groups and the group preferences was drawn from $Uniform(-3, 3)$. We allowed only opposite groups to interact with each other. Further, in the second grouping, nodes were assigned groups opposite from their first assignment. Under the two groupings, network was generated with $\langle k^{(1)} \rangle = 10$ and $\langle k^{(2)} \rangle = 10$. Here, $\beta = 0.4$.