An Ordinal Approach to Approximation Algorithms

ELLIOT ANSHELEVICH, Rensselaer Polytechnic Institute
SHREYAS SEKAR, University of Washington

We study Matching, Densest subgraph, and other graph optimization problems in a partial information setting, where the true graph weights are hidden, and the algorithm only has access to ordinal preference information. Our model is motivated by the fact that in settings where the nodes of the graph represent self-interested agents, it may be preferable for the agents to rank their adjacent edges in the order of preferences as opposed to expressing the numerical value of each edge weight. Specifically, we study problems where the ground truth exists in the form of a weighted graph of agent utilities, but the algorithm receives as input only a preference ordering for each agent induced by the underlying weights. Against this backdrop, we design both truthful and non-truthful algorithms to approximate the true optimum solution with respect to the hidden weights. Perhaps surprisingly, such algorithms are possible for many important problems, as we show using our framework based on the techniques of greedy, random, and serial dictatorship. Our framework yields a 1.6-approximation algorithm for the maximum weighted matching problem, and a 4-approximation algorithm for Densest k-subgraph as the hidden weights constitute a metric. Under the additional constraint of strategyproofness, we obtain approximation factors of 1.77 and 8 respectively for the above problems. We also show that our framework yields constant factor approximations for a number of other graph maximization problems. Our results are the first non-trivial ordinal approximation algorithms for such problems, and indicate that in many situations, we can design robust algorithms even when we are agnostic to the precise agent utilities.

CCS Concepts: • **Theory of computation → Graph algorithms analysis**; **Algorithmic mechanism design**; *Approximation algorithms analysis*; *Algorithmic game theory*; • **Mathematics of computing →** *Matchings and factors*;

Additional Key Words and Phrases: Approximation Algorithms, Partial Information, Matching, Densest Subgraph, Mechanism Design

ACM Reference format:
Elliot Anshelevich and Shreyas Sekar. 2018. An Ordinal Approach to Approximation Algorithms. *ACM Trans. Algor.* 1, 1, Article 1 (March 2018), 43 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

In recent years, the field of algorithm design has been marked by a steady shift towards newer paradigms that take into the account the behavioral aspects and communication bottlenecks pertaining to self-interested agents. In contrast to traditional algorithms that are assumed to have

Author's addresses: E. Anshelevich, Computer Science Department, Rensselaer Polytechnic Institute; S.Sekar, Department of Electrical Engineering, University of Washington.

1549-6325/2018/3-ART1 $15.00
https://doi.org/0000001.0000001

complete information regarding the inputs, mechanisms that interact with autonomous individuals commonly assume that the input to the algorithm is controlled by the agents themselves. In this context, a natural constraint that governs the process by which the algorithm elicits inputs from these agents is *truthfulness* [37, 40]: agents cannot improve upon the resulting outcome by misreporting the inputs. Another constraint that has recently gained traction in network optimization problems is that of *ordinality* [11, 15, 36]: here, agents who correspond to the nodes of the network can only submit a preference list of their neighbors ranked in the order of the edge weights. The need for algorithms that are truthful or ordinal arises in a number of important settings; however, it is well known that it is impossible to obtain optimum solutions even when the algorithm is required to satisfy only one of these two constraints [11, 25, 40].

In this work, we study the design of approximation algorithms for popular graph optimization problems such as matching, clustering, and Densest subgraph with the goal of understanding the *price of ordinality*. To be more specific, we consider the above optimization problems on a weighted graph whose vertices represent the agents, and where the edge weights (that correspond to agent utilities) satisfy the triangle inequality, and pose the following question: "*How does a computationally efficient algorithm that only has access to each agent's edge weights in the form of preference rankings perform in comparison to an optimal algorithm that has full knowledge of the weighted graph?*". Additionally, we also consider the case when the edge weights constitute private information known only to the agents (nodes) on that edge and design truthful mechanisms that elicit preference orders from each agent and approximate the optimum solution using only the preference information.

A Case for an Ordinal World with Partial Information. A crucial question in algorithm and mechanism design is: "How much information about the agent utilities does the algorithm designer possess?". The starting point for the rest of our paper is the observation that in many natural settings, it is unreasonable to expect the mechanism to know the exact weights of the edges in the graph [11, 15]. For example, when clustering a set $\mathcal{N}$ of objects, it may be difficult to precisely quantify the true similarity level for every pair of objects; ordinal questions such as 'which is more similar to object x: y or z?' may be easier to answer. Such a situation would also arise when the graph represents a social network of agents, as the agents themselves may not be able to express 'exactly how much each friendship is worth', but would likely be able to form an ordering of their friends from best to worst. This phenomenon has also been observed in settings pertaining to voting or social choice, in which it is much easier to obtain ordinal preferences instead of true agent utilities [5, 39].

Motivated by this, we consider a model where for every agent $i \in \mathcal{N}$, we *only have access to a preference ordering* among the agents in $\mathcal{N} - \{i\}$ so that if $w(i, j) > w(i, k)$, then i prefers j to k—here, $w(i, j)$ is the weight on the edge $e \in \mathcal{N} \times \mathcal{N}$. The common approach in Learning Theory while dealing with such ordinal settings is to estimate the 'true ground state' based on some probabilistic assumptions on the underlying utilities [38, 43]. In this paper we take a different approach, and instead focus on the more demanding objective of designing *robust algorithms*, i.e., algorithms that provide good performance guarantees no matter what the underlying weights are.

Despite the large body of literature (e.g., see [3, 42]) on computing matchings, clusterings, and teams in settings with preference orderings, there has been much less work on quantifying the quality of these solutions. As is common in *social choice theory*, most papers (implicitly) assume that the underlying utilities cannot be measured or do not even exist, and hence there is no clear way to measure quality [1, 8, 29]. In such papers, the focus therefore is on computing solutions that satisfy normative properties such as stability or optimize a measure of efficiency that depends only on the preference orders, e.g., average rank. On the other hand, the literature on approximation

algorithms usually follows the *utilitarian* approach [30] of assigning a numerical quality to every solution; the presence of input weights is taken for granted. Our work combines the best of both worlds: we do *not* assume the availability of numerical information (only its latent existence), and yet our approximation algorithms must compete with algorithms that know the true input weights.

Model and Problem Statements. In this work, we consider a weighted, complete graph $G = (\mathcal{N}, E)$ where the vertex set $\mathcal{N}$ also denotes the set of agents. We design ordinal approximation algorithms for a class of optimization problems where the objective is to select a subset of edges $S \subseteq E$ satisfying some constraint in order to maximize the weight of the selected edges. For each $i \in \mathcal{N}$, we assume that the agent has a strict preference ordering P_i over the agents in $\mathcal{N} - \{i\}$, which are derived from the (hidden) underlying edge weights ($w(x, y)$ for edge $x, y \in E$), which satisfy the triangle inequality, i.e., for $x, y, z \in \mathcal{N}$, $w(x, y) \leq w(x, z) + w(y, z)$. These weights are considered to represent the *ground truth*, which is not known to the algorithm. We say that the preferences $\mathbf{P} = (P_i)_{i \in \mathcal{N}}$ are induced by weights w if $\forall x, y, z \in \mathcal{N}$, if x prefers y to z, then $w(x, y) \geq w(x, z)$.

For the specific kind of the problems that we study, the metric structure occurs in a number of well-motivated environments such as: (*i*) social networks, where the property captures a specific notion of friendship, (*ii*) Euclidean metrics: each agent is a point in a metric space which denotes her skills or beliefs, and (*iii*) edit distances: each agent could be represented by a string over a finite alphabet (e.g., a gene sequence) and the graph weights represent the edit or Levenshtein distances [44]. The reader is asked to refer to Appendix A for additional details on these specific applications and a mathematical treatment of friendship in social networks.

Our main goal is to form **ordinal approximation algorithms** for a variety of problems. An algorithm A is said to be ordinal if it only takes preference orderings $\mathbf{P}$ as input (and not the hidden numerical weights w). It is an α-approximation algorithm if for all possible weights w, and the corresponding induced preferences $\mathbf{P}$, we have that $\frac{OPT(w)}{A(\mathbf{P})} \leq \alpha$. Here $OPT(w)$ is the total value of the maximum weight solution with respect to w, and $A(\mathbf{P})$ is the value of the solution returned by the algorithm for input $\mathbf{P}$. In other words, such algorithms produce solutions which are always a factor α away from optimum, *without actually knowing* what the weights w are.

Truthful Mechanisms for an Ordinal World. We study the design of ordinal approximation algorithms for graph maximization problems under two informational paradigms: (*i*) the full information (but ordinal) case when the preference orders corresponding to each agent are readily available to the algorithm designer, and (*ii*) the private information model when the preference order P_i corresponding to agent i is known only to that agent. In the latter case, our objective is to design a truthful, ordinal mechanism that approximates the optimal solution, i.e., an algorithm that incentivizes agents to report their true preference rankings and uses these preference rankings as input to compute an approximately optimal solution for the problem at hand. The algorithm is truthful if no single agent can improve their utility by submitting a preference ordering different from the 'true ranking' induced by the graph weights. Here, the utility of each agent i is simply the total weight of the edges incident to i which are chosen. These utilities have a natural interpretation with respect to the problems considered in this work. For instance, for matching problems, an agent's utility corresponds to her affinity or weight to the agent to whom she is matched, and for densest subgraph as well as clustering, the utility is her aggregate weight to the agents in the same team or cluster.

Although we present approximation results for a variety of graph optimization problems, the majority of this paper will focus on the following two problems: (*i*) **Maximum Weighted Matching**(MWM): where the goal is to compute a matching to maximize the total (unknown) weight of the edges inside and (*ii*) **Densest k-Subgraph**: where the objective is to compute a set $S \subseteq \mathcal{N}$ of

Problem	Full Info (Metric Weights)	Our Results	
		Ordinal	Ordinal & Truthful
Max Weighted Matching	1 [21]	1.6	1.77
Densest k-Subgraph	2 [10, 31]	4	8

Table 1. A comparison of the approximation factors obtained by ordinal approximation algorithms presented in this work and previous results for the full information setting with and without the truthfulness requirement.

size k to maximize the weight of the edges inside S. For both of these problems, we design two types of algorithms, those that are simply ordinal and those that satisfy truthfulness in addition to ordinality. Finally, we remark that the approximation factor obtained by a mechanism that is both truthful as well as ordinal cannot be better than that obtained by an ordinal mechanism that has full information about agent preferences.

Challenges, Technique, and Contributions

We describe the challenges involved in designing ordinal algorithms through the lens of the Maximum Weighted Matching problem. First, different sets of edge weights may give rise to the same preference ordering and moreover, for each of these weights, the optimum matching can be different. Therefore, unlike for the full information setting, no algorithm (deterministic or randomized) can compute the optimum matching using only ordinal information. More generally, the restriction that only ordinal information is available precludes almost all of the well-known algorithms for computing a matching. This motivates the need for a new line of algorithmic thinking that specifically exploits preference rankings.

As a first step towards designing truthful ordinal mechanisms, we introduce three fundamental paradigms based on the popular algorithmic notions of *Greedy*, *Serial Dictatorship*, and *Uniformly Random*. These techniques have consistently featured in a number of works on matching and Densest subgraph, and indeed, it is not difficult to characterize the approximation ratio obtained via a direct application of these paradigms. However, the main contribution of the current work is designing algorithms that interleave these basic *greedy*, *random*, and *serial dictatorship* based solutions in order to beat the approximation guarantees obtained by a naive application of these techniques. For instance, the simple greedy and random techniques yield 2-approximations to the optimum matching for the Maximum Weighted Matching problem. However, by carefully studying the interplay between these two techniques, it is actually possible to do much better, and obtain a 1.6-approximation algorithm.

Our Contributions. The central contributions of this paper are truthful and non-truthful mechanisms that use ordinal information to obtain constant factor approximations for the maximum weighted matching and densest k-subgraph problems. The main approximation guarantees are summarized in Table 1. As seen in the table, our ordinal algorithms provide approximation factors that are close to the best known for the full information versions, where the algorithm designer has complete knowledge of the graph weights. In other words, we show that it is possible to find good solutions to such problems even if the graph weights are unknown and only ordinal information is presented to the algorithm, and even if the agents can lie about their preferences.

Our secondary contribution is a general approach towards designing ordinal approximation algorithms for a variety of optimization problems. More specifically, we show that the greedy, random, and serial dictatorships techniques can be leveraged to obtain constant factor approximation

Problem	Our Approximation Factor
Max k-Matching	2
k-sum Clustering	2
Max Traveling Salesman (TSP)	2
Max Spanning Tree	2
Max k-Vertex Cover	4

Table 2. Approximation factors for other graph maximization problems obtained via ordinal algorithms.

algorithms for a number of well-studied graph maximization problems. The approximation factors for these problems are given in Table 2; the problem definitions can be found in Section 5.

Related Work

Broadly speaking, the cornucopia of algorithms proposed in the matching literature belong to one of two classes: (*i*) Ordinal algorithms that ignore agent utilities, and focus on (unquantifiable) axiomatic properties such as *stability*, or *Pareto-optimality* and (*ii*) Optimization algorithms where the numerical utilities are fully specified. From our perspective, algorithms belonging to the former class, with the exception of *Greedy*, do not result in good approximations for the hidden optimum, whereas the techniques used in the latter (e.g., [18, 19]) depend heavily on *improving cycles* and thus, are unsuitable for ordinal settings. A notable exception to the above dichotomy is the class of optimization problems studying *ordinal measures of efficiency* [1, 8, 15, 35], for example, the average rank of an agent's partner in the matching. Such settings often involve the definition of 'new utility functions' based on given preferences, and thus are fundamentally different from our model where preexisting cardinal utilities give rise to ordinal preferences.

The truthful mechanisms in our work fall under the umbrella of 'mechanism design without money' [4, 12, 20, 25, 40], a recent line of work on designing strategyproof mechanisms for settings like ours, where monetary transfers are irrelevant. A majority of the papers in this domain deal with mechanisms that elicit agent utilities, specifically for one-sided matchings, assignments and facility location problems that are somewhat different from the graph problems we are interested in. The notable exceptions are the recent papers on truthful, ordinal mechanisms for one-sided matchings [12, 25] and general allocation problems [4]. While [25] looks at normalized agent utilities and shows that no ordinal algorithm can provide an approximation factor better than $\Theta(\sqrt{N})$, [12] considers *minimum* cost metric matching under a resource augmentation framework. The main differences between our work and these two papers are (1) we consider two-sided matching instead of one-sided, as well as other clustering problems, as well as non-truthful algorithms with better approximation factors, and (2) we consider maximization objectives in which users attempt to maximize their utility instead of minimize their cost. The latter may seem like a small difference, but it completely changes the nature of these problems, allowing us to create many different truthful mechanisms and achieve *constant-factor approximations*. Finally, [4] looks at the problem of allocating goods to buyers in a 'fair fashion'. In that paper, the focus is on maximizing a popular non-linear objective known as the *maximin share*, which is incompatible with our objective of social welfare maximization. That said, an interesting direction is to see if our techniques extend to other objectives.

After the publication of the preliminary versions of this work, a number of follow-up papers have focused on the design of ordinal algorithms for other graph optimization problems [2] or more general models of information [7]. Specifically, in [2], the authors present an ordinal greedy approach for a general class of graph optimization problems, where the approximation factors

depend on the degree and sparsity of the graph. Although our results in Section 5 are similar in spirit to those in [2], we also present constant factor approximation algorithms for problems that *do not* fall within the class of graph optimization problems studied in that work, particularly, namely k-sum clustering and max k-vertex cover. Secondly, [7] considers the problem of computing a matching on bipartite graphs and presents ordinal approximation algorithms for more general models of information where the designer only has access to partial ordinal information for each agent (e.g., top αN preferred nodes). Our results in Section 3 are not directly comparable to theirs as we study matchings on complete graphs where the weights satisfy the triangle inequality. That said, we believe that our algorithms

All of the problems studied in this paper have received considerable attention in the literature for the full information case with metric weights. In particular, metric Densest Subgraph (also known as *Maximum Dispersion* or *Remote Clique*) is quite popular owing to its innumerable applications [9, 10]. The close ties between the optimum solutions for Matching, k-sum Clustering, and Densest k-subgraph was first explored by Feo and Khellaf [24], and later by Hassin et al. [31].

Distortion in Social Choice Our work is similar in motivation to the growing body of research in computational social choice that study settings where the voter preferences are induced by a set of hidden utilities [5, 6, 11, 13, 14, 23]. The voting protocols in these papers are essentially ordinal approximation algorithms, albeit for the very specific problem of selecting the utility-maximizing candidate from a set of alternatives.

Finally, other models of incomplete information have been considered in the Matching literature, most notably online algorithms [32] and truthful algorithms (for strategic agents) [20]. Given the strong motivations for preference rankings in settings with agents, it would be interesting to see whether algorithms developed for other partial information models can be extended to our setting.

2 PRELIMINARIES AND FRAMEWORK FOR ORDINAL ALGORITHMS

In this section, we present a general framework for developing ordinal approximation algorithms using the simple paradigms mentioned previously. Our framework comprises of technical lemmas and properties corresponding to three algorithmic approaches: *greedy*, *uniform*, and *random serial dictatorship*. In the interest of extensibility, we present these techniques in reference to the Max k-matching problem, a strict generalization of the Maximum Weighted Matching problem, where the objective is to select a matching consisting only of $k \leq \frac{N}{2}$ edges. This has direct implications to both of the primary problems studied in this work.

We also discuss how these approaches can be leveraged to design truthful ordinal algorithms for matching and establish tight upper and lower bounds on the performance of algorithms that select matching edges either greedily or uniformly at random. In Sections 3 and 4, we develop more sophisticated mechanisms that build upon the simple paradigms presented here, leading to improved approximation factors. Finally, we remark that for the sake of convenience and brevity, we will often assume that N, the number of agents, is even, and sometimes that it is also divisible by 3. As we discuss in Appendix B, our results still hold if this is not the case, with only minor modifications. We begin with some pertinent definitions.

2.1 Definitions

We study settings consisting of a set $\mathcal{N}$ of self-interested agents, where each agent $i \in \mathcal{N}$ has a preference ranking P_i over the other agents. We assume that the vector of preferences $\mathbf{P} := (P_i)_{i \in \mathcal{N}}$ is consistent with a set of true weights $(w_{ij})_{i,j \in \mathcal{N}}$ which the agents may or may not be aware of. Given this setup, we are interested in designing truthful, ordinal algorithms for some well-studied graph optimization problems where the objective, broadly speaking, is to compute a set of edges

S satisfying some constraint in order to maximize the weight of edges inside S. This includes Maximum Weighted Matching and Densest Subgraph which are the main problems studied in this work but also other popular problems such as k-sum Clustering, Max TSP, Max Spanning Tree, and Max (Weighted) k-Vertex Cover.

For all of these problems, we study two information models: a full information ordinal setting, where the weights are hidden but the preference orders corresponding to each agent are known to us, and a private information model where each agent's preference list P_i is known only to that agent. In the latter case, we are interested in designing mechanisms $\mathcal{M}$ that elicit this preference information from the agents and output a valid solution to the above problems. The input to this mechanism is a strategy profile $\mathbf{s} = (s_i)_{i \in \mathcal{N}}$ such that s_i denotes the preference ranking over the agents in $\mathcal{N} - \{i\}$ as reported by agent i, and its output is a set of edges $\mathcal{M}(\mathbf{s}) \subseteq E$ corresponding to the one of the problems defined above.

Agent Utilities: Suppose that $(w_{ij})_{i,j \in \mathcal{N}}$ denotes the true weights, then the utility of agent i under mechanism $\mathcal{M}$, and input profile $\mathbf{s}$ is the sum of weights of the edges in $\mathcal{M}(\mathbf{s})$ containing i, i.e., $u_i^{\mathcal{M}}(\mathbf{s}) = \sum_{j:(i,j) \in \mathcal{M}(\mathbf{s})} w(i,j)$. When the mechanism is clear from the context, we will omit the superscript $\mathcal{M}$ from the user's utility. These utilities have a natural interpretation with respect to the matching and team formation problems considered in this work. Our objective in this paper is to design truthful, ordinal mechanisms that maximize the overall social welfare, i.e., the sum of the utilities of all of the agents.

Truthful Ordinal Mechanisms. As mentioned previously, we are interested in designing incentive-compatible mechanisms that elicit ordinal preference information from the users, i.e., mechanisms where agents are incentivized to truthfully report their preferences in order to maximize their utility. We now formally define the notions of truthfulness pertinent to our setting. Throughout the rest of this paper, we will use P_i to represent the true preference ranking of agent i (i.e., one that is induced by the weights $w(i,j)$), and s_i to represent the preference ordering that agent i submits to the mechanisms (which will be equal to P_i if i is truthful).

Definition 2.1. (Truthful Mechanism) A deterministic mechanism $\mathcal{M}$ is said to be truthful if for every $i \in \mathcal{N}$, all $\mathbf{s}_{-i}, s_i'$, we have that $u_i(P_i, \mathbf{s}_{-i}) \geq u_i(s_i', \mathbf{s}_{-i})$, where u_i is the utility guaranteed to agent i by the mechanism.

Definition 2.2. (Universally Truthful Mechanisms) A randomized mechanism is said to be universally truthful if it is a probability distribution over truthful deterministic mechanisms.

Informally, in a universally truthful mechanism, a user is incentivized to be truthful even when she knows the exact realization of the random variables involved in determining the mechanism.

Definition 2.3. (Truthful in Expectation) A randomized mechanism is said to be truthful in expectation if an agent always maximizes her expected utility by truthfully reporting her preference ranking. The expectation is taken over the different outcomes of the mechanism.

All of our algorithms are *universally* truthful, not just in expectation. The reader is asked to refer to [17] for a useful discussion on the types of randomized mechanisms, and settings where universally truthful mechanisms are strongly preferred as opposed to the mechanisms that only guarantee truthfulness in expectation.

2.2 Approaches for Designing Truthful and Ordinal Mechanisms: Greedy

Our first algorithm is the ordinal analogue of the classic greedy matching algorithm, that has been extensively applied across the matching literature. In order to better understand this algorithm, we first define the notion of an undominated edge.

Definition 2.4. (Undominated Edges) Given a set E of edges, $(x, y) \in E$ is said to be an undominated edge if for all $(x, a), (y, b) \in E$, $w(x, y) \geq w(x, a)$ and $w(x, y) \geq w(y, b)$.

Given a set E, let us use the notation $E_\top$ to denote the set of undominated edges in E. Finally, we say that an edge set E is complete if there exists some $S \subseteq \mathcal{N}$ such that E is the complete graph on the nodes in S (minus the self-loops). We make the following two observations regarding undominated edges

(1) Every edge set E has at least one undominated edge. In particular, any maximum weight edge in E is obviously an undominated edge.
(2) Given an edge set E, one can efficiently find at least one edge in $E_\top$ *using only the ordinal preference information.* A naive algorithm for this is as follows. Consider starting with an arbitrary node x. Let (x, y) be its first choice out of all the edges in E (i.e., y is x's first choice of all the nodes it has an edge to in E). Now consider y's first choice. If it is x, then the edge (x, y) must be undominated, as desired. If instead it is some $z \neq x$, then continue this process with z. Eventually this process must cycle, giving us a cycle of nodes $x_0, x_1, \ldots, x_{\ell-1}$ such that x_i is the top preference of x_{i-1}, taken with respect to $\mod \ell$. This means that all edges in this cycle have equal weight, even though we do not know what this weight is, since x_i preferring x_{i+1} over x_{i-1} means that $w(x_i, x_{i+1}) \geq w(x_i, x_{i-1})$. Moreover, the edge weights of all edges in this cycle must be the highest ones incident on the nodes in this cycle, since they are all top preferences of the nodes. Therefore, all edges in this cycle are undominated, as desired.

In general, an edge set E may have multiple undominated edges that are not part of a cycle. Our first lemma shows that these different edges are comparable in weight.

Lemma 2.5. *Given a complete edge set E, the weight of any undominated edge is at least half as much as the weight of any other edge in E, i.e., if $e = (x, y) \in E_\top$, then for any $(a, b) \in E$, we have $w(x, y) \geq \frac{1}{2} w(a, b)$. This is true even if (a, b) is another undominated edge.*

Proof. Since (x, y) is an undominated edge, and since E is a complete edge set this means that $w(x, y) \geq w(x, a)$, and $w(x, y) \geq w(x, b)$. Now, from the triangle inequality, we get $w(a, b) \leq w(a, x) + w(b, x) \leq 2w(x, y)$. □

Now that we have obtained a better understanding of undominated edges, we are ready to present the ordinal greedy algorithm for the Max k-Matching problem, which we introduce in Algorithm 1. Recall that Max k-matching reduces to the MWM problem when $k = \frac{N}{2}$. For this case, it is not difficult to see that the output of Algorithm 1 coincides with that of the extremely popular greedy algorithm that picks the maximum weight edge at each iteration, and therefore, our algorithm yields an ordinal 2-approximation for the MWM problem. Our next result shows that the algorithm is truthful only for the $k = \frac{N}{2}$ case, i.e., the maximum weighted matching problem.

ALGORITHM 1: Ordinal Greedy Algorithm for Maximum k-Matching

Input: Parameter k;
$M := \emptyset$, T is the valid set of edges initialized to E, the complete graph on $\mathcal{N}$;
while T *is not empty* **do**
  pick an undominated edge[1] $e = (x, y)$ from T and add it to M;
  remove all edges containing x or y from T; if $|M| = k$, $T = \emptyset$.
end

Proposition 2.6. *Algorithm 1 is truthful for the Max k-Matching problem only when* $k = \frac{N}{2}$.

Proof. We need to prove that for any given strategy profile adopted by the other players $\mathbf{s}_{-i}$, player i maximizes her utility when she is truthful, i.e., if P_i is the true preference ordering of agent i and s_{-i} is any set of preference orderings for the other agents, then $u_i(P_i, \mathbf{s}_{-i}) \geq u_i(s'_i, \mathbf{s}_{-i})$ for any s'_i. Our proof will proceed via contradiction and will make use of the following fundamental property: *if Algorithm 1 (for some input) matches agent i to j during some iteration, then both i and j prefer each other to every other agent that is unmatched during the same round.*

We introduce some notation: suppose that M denotes the matching output by Algorithm 1 for input $(P_i, \mathbf{s}_{-i})$, and for every $x \in \mathcal{N}$, $m(x)$ is the agent to whom x is matched under M. Let e_j be the edge added to the matching M in round j of Algorithm 1, denote the round in which i is matched to $m(i)$ as round k. Assume to the contrary that for input $(s'_i, \mathbf{s}_{-i})$, i is matched to an agent she prefers more than $m(i)$. Let the altered matching be referred to as M', and let $m'(x)$ be the agent who x is matched with in M'.

We begin by proving the following claim: *For each $j < k$, we have that $e_j \in M'$*. In other words, all the edges which are included into M before i is matched by Algorithm 1 must appear in both matchings no matter what i does. Once we prove this claim, we are done, since e_k is the highest-weight edge from i to any node not in $e_1, \ldots, e_{k-1}$, so i maximizes its utility by telling the truth and receiving utility equal to the weight of e_k.

To prove the claim above, we proceed by induction. Note that if $k = 1$, then i is trivially truthful, since $m(i)$ is its top choice in the entire graph. Now suppose that we have shown the claim for edges $e_1, \ldots, e_{j-1}$. Let $e_j = (x, y)$, and without loss of generality suppose that x is matched in our algorithm constructing M' before y. At the time that x is matched with $m'(x)$, it must be that $m'(x)$ is the top choice of x from all available nodes. But, by the definition of our algorithm, y is the top choice of x that is not contained in $e_1, \ldots, e_{j-1}$. Since $m'(x)$ is not contained in $e_1, \ldots, e_{j-1}$ due to our inductive hypothesis, this means that x prefers y over $m'(x)$, and since y is not matched yet, this means that x and y will become matched together in M'. Thus, e_j is in M' as well. This completes the proof of our claim.

To see why this mechanism is not truthful for the max k-matching problem when $k < \frac{N}{2}$, notice that agents which would not be matched in the first k steps have incentive to lie and form undominated edges where none exist, in order to be matched earlier. Assume that the algorithm uses a deterministic tie-breaking rule to choose between multiple undominated edges in each round. While this does not really alter the final output for the perfect matching problem, the tie-breaking rule may lead to certain undominated edges not getting selected for the final matching.

Specifically, consider the max k-matching problem and suppose that when the input preferences are truthful, agents i, j are not present in the matching M returned by Algorithm 1. Moreover, suppose that (i) j's first preference is i, and (ii) the deterministic tie-breaking rule always prefers (i, j) over other edges (one can design preferences so that agents favoured by the tie-breaking are not selected for truthful inputs). Clearly i has incentive to alter its preferences to identify j as its most preferred node and receive a utility of $w(i, j)$, which can be strictly greater than its previous utility of zero for an appropriately defined instance. □

Greedy Algorithms for Other Problems Can we use a similar approach to design algorithms for the other problems that we are interested in? In Section 5, we present a general-purpose ordinal greedy algorithm that achieves constant factor approximations for a variety of problems such as k-sum Clustering, Max TSP, etc. For the Densest k-subgraph problem, the deceptively simple algorithm

[1]For the sake of completeness, we assume that when $T \neq \emptyset$ but there are no undominated edges left, e.g., when agents lie, the algorithm picks an edge uniformly at random

that picks an undominated edge greedily in each round and adds the two vertices constituting its end points to the solution yields a 4-approximation to the optimum omniscient solution. Unfortunately, the greedy approach **does not** lead to truthful algorithms for any of the problems studied in this work other than MWM. We now illustrate this via the Densest subgraph problem.

Consider the Densest k-subgraph problem with $k = 4$, and an instance with 6 nodes whose preferences we define partially: a's top 3 nodes are b, c, d; b's top two nodes are a and d; c's first two nodes are a, e; d's top two preferences are b and e and finally, e, f prefer each other as a first choice. Consider the greedy algorithm that first picks a matching M with $\frac{k}{2}$-edges and returns the same set of nodes as in the matching. Moreover, suppose that the algorithm's tie-breaking involves selecting edges containing a or b before edges containing e, f and then c, d. Now, under these preferences, we claim that node a stands to improve her utility by lying when all the other agents are being truthful. To see why, first observe that if node a truthfully reports her preferences, the algorithm returns $\{a, b, e, f\}$ as the solution set. On the other hand, if a lies and points to c as her first preference, then the algorithm picks (a, c) first followed by (b, d) resulting in the set $\{a, b, c, d\}$, which is strictly preferable from a's perspective. In a similar fashion, for other problems such as clustering, TSP, or vertex cover, using a greedy algorithm could result in agents underplaying their most preferred node if that node will be chosen in a later round regardless.

Random Algorithm for Matching and Other Problems

A much simpler approach that is completely oblivious to the input preferences involves selecting a solution uniformly at random. Such an algorithm (described in Algorithm 2 for max k-matching) naturally extends to all of the problems studied in this work and is obviously truthful. Somewhat surprisingly, this approach also leads to a good approximation for maximum weighted matching. We begin by describing the algorithm and then prove some independent technical lemmas that provide us with a lower bound on the quality of a random matching and upper bound on that of the optimum matching. For the purpose of generality, we define the algorithm with an arbitrary edge set as an input; for the MWM problem (for instance), the input edge set is simply the complete graph.

ALGORITHM 2: Random Algorithm for Max k-Matching

Input: Edge Set E' and Parameter k;
$M := \emptyset$, T is the valid set of edges initialized to E';
while T *is not empty and* $|M| < k$ **do**
  pick an edge $e = (x, y)$ from T uniformly at random and add it to M;
  remove all edges containing x or y from T
end

Lemma 2.7. *(Lower Bound)*

(1) *Suppose $G = (T, E')$ is a complete graph on the set of nodes $T \subseteq \mathcal{N}$ with $|T| = n$. Then, the expected weight of the random (perfect) matching returned by Algorithm 2 for the inputs E', $\frac{n}{2}$ is $E[w(M_R)] \geq \frac{1}{n} \sum_{(x,y)\in E'} w(x, y)$.*

(2) *Suppose $G = (T_1, T_2, E')$ is a complete bipartite graph on the set of nodes $T_1, T_2 \subseteq \mathcal{N}$ with $|T_1| = |T_2| = n$. Then, the weight of the random (perfect) matching returned by Algorithm 2 for the inputs E', n is $E[w(M_R)] = \frac{1}{n} \sum_{(x,y)\in E} w(x, y)$.*

Proof. We show both parts of the theorem using simple symmetry arguments. For the first part, i.e., the complete (non-bipartite) graph, let $\mathcal{M}$ be the set of all perfect matchings in E. Then, we

argue that every matching M in $\mathcal{M}$ is equally likely to occur. Therefore, the expected weight of M_R is

$$E[w(M_R)] = \frac{1}{|\mathcal{M}|} \sum_{M \in \mathcal{M}} w(M) = \sum_{e=(x,y) \in E} p_e w(x,y), \tag{1}$$

where p_e is the probability of edge e occurring in the matching. Since the edges are chosen uniformly at random, the probability that a given edge is present in M_R is the same for all edges in E. So $\forall e$, we have the following bound of p_e, which we can substitute in Equation 1 to get the first result.

$$p_e = \frac{|M_R|}{|E|} = \frac{n/2}{n(n-1)/2} = \frac{1}{n-1} \geq \frac{1}{n} \tag{2}$$

For the second case, where E is the set of edges in a complete bipartite graph, it is not hard to see that once again every edge e is present in the final matching with equal probability. Therefore, $p_e = \frac{|M_R|}{|E|} = \frac{n}{n^2} = \frac{1}{n}$. The final bound follows from substituting this value into Equation 2. □

Lemma 2.8. *(Upper Bound) Let $G = (T, E)$ be a complete subgraph on the set of nodes T with $|T| = n$. Let $\mathcal{S}$ be a superset of T such that $T \subseteq \mathcal{S} \subseteq \mathcal{N}$, and let M be any perfect matching on the larger set $\mathcal{S}$. Then, the following is an upper bound on the weight of M,*

$$w(M) \leq \frac{2}{n} \sum_{\substack{x \in T \\ y \in T}} w(x,y) + \frac{1}{n} \sum_{\substack{x \in T \\ y \in \mathcal{S} \setminus T}} w(x,y)$$

Proof. Fix an edge $e = (x, y) \in M$. Then, by the triangle inequality, the following must hold for every node $z \in T$: $w(x,z) + w(y,z) \geq w(x,y)$. Summing this up over all $z \in T$, we get

$$\sum_{z \in T} (w(x,z) + w(y,z)) \geq n w(x,y).$$

Once again, repeating the above process over all $e \in M$, and then all $z \in T$ we have

$$n w(M) \leq 2 \sum_{\substack{x \in T \\ y \in T}} w(x,y) + \sum_{\substack{x \in T \\ y \in \mathcal{S} \setminus T}} w(x,y)$$

Each $(x, y) \in E$ appears twice in the RHS: once when we consider the edge in M containing x, and once when we consider the edge with y. □

We conclude by proving that picking edges uniformly at random yields a 2-approximation for the MWM problem.

Claim 2.9. *Algorithm 2 is an ordinal 2-approximation algorithm for the Maximum Weighted Matching problem.*

Proof. From Lemma 2.7, we know that in expectation, the matching M_R output by the random algorithm when the input is $\mathcal{N}$ has a weight of at least $\frac{1}{N} \sum_{x \in \mathcal{N}, y \in \mathcal{N}} w(x,y)$. Substituing $T = \mathcal{S} = \mathcal{N}$ in Lemma 2.8 and $M = OPT$ (max-weight matching) gives us the following upper bound on the weight of OPT, $w(OPT) \leq \frac{2}{N} \sum_{x \in \mathcal{N}, y \in \mathcal{N}} w(x,y) \leq 2E[w(M_R)]$. □

Random Serial Dictatorship

Another popular approach to compute incentive compatible matchings that performs well in settings with metric weights (albeit usually for one-sided matchings [12, 25]) is random serial dictatorship. In some sense, this technique combines the best of greedy and random into a single algorithm. We formally define the algorithm below for our two-sided matching setting. It is not hard to see that the algorithm is trivially truthful.

Proposition 2.10. *Algorithm 3 is universally truthful for the Maximum k-Matching problem.*

ALGORITHM 3: Random Serial Dictatorship for Max k-Matching

Input: Parameter k;
$M := \emptyset$, T is the set of available agents initialized to $\mathcal{N}$;
while T *is not empty and* $|M| < k$ **do**
  pick an available agent x uniformly at random from T;
  let y denote x's most preferred agent in $T - \{x\}$; add (x, y) to M;
  remove $\{x, y\}$ from T;
end

Serial dictatorship is among the most prominent of algorithms to feature in this work: our primary approximation algorithm for truthful Densest k-subgraph relies on serial dictatorship. Similar algorithms have received attention for other matching problems [12, 25] as well; ours is the first result showing that the algorithms can approximate the optimum matching up to a small constant factor for metric settings. Moreover, while serial dictatorship is usually easy to analyze, our algorithm greatly exploits the randomness to select good edges *in expectation*. We now show that this algorithm also leads to a 2-approximation for MWM. For the purpose of clear exposition, we defer the somewhat lengthy proof of the theorem to Appendix C.

Theorem 2.11. *Random serial dictatorship is a universally truthful mechanism that provides a 2-approximation for the Maximum k-Matching Problem.*

3 ORDINAL MATCHING ALGORITHMS

In the previous section, we presented three truthful algorithms for the MWM problem, all of which lead to 2-approximations. In this section, we improve upon these naive algorithms and present improved approximations for both the truthful and non-truthful versions of the problem.

3.1 1.6-Approximation Algorithm for Non-Truthful Matching

Here we present a better ordinal approximation than simply taking the random or greedy matching. The algorithm first performs the greedy subroutine until it matches two-thirds of the agents. Then it either creates a random matching on the unmatched agents, *or* it creates a random matching between the unmatched agents and a subset of agents which are already matched. We show that one of these matchings is guaranteed to be close to optimum in weight. Unfortunately since we have no access to the weights themselves, we cannot simply choose the best of these two matchings, and thus are forced to randomly select one, giving us good performance in expectation. More formally, the algorithm is:

Theorem 3.1. *For every input ranking, Algorithm 4 returns a* $\frac{8}{5} = 1.6$*-approximation to the maximum-weight matching.*

ALGORITHM 4: 1.6-Approximation Algorithm for Maximum Weight Matching

input : $\mathcal{N}, \mathbf{P}$
output: Perfect Matching M
E is the complete graph on $\mathcal{N}$, and $M_1 = M_2 = \emptyset$;
Let M_0 be the output returned by Algorithm 1 for E,$k = \frac{2}{3}\frac{N}{2}$;
Let B be the set of nodes in $\mathcal{N}$ not matched in M_0, and E_B is the complete graph on B.;
First Algorithm;
$M_1 = M_0 \cup$ (The perfect matching output by Algorithm 2 on E_B, $k = \frac{1}{3}\frac{N}{2}$) ;
Second Algorithm ;
Choose half the edges from M_0 uniformly at random and add them to M_2;
Let A be the set of nodes in $M_0 \setminus M_2$;
Let E_{ab} be the edges of the complete bipartite graph (A, B);
Run Algorithm 2 on the set of edges in E_{ab}, $k = \frac{2}{3}\frac{N}{2}$ to obtain a perfect bipartite matching and add the edges returned by the algorithm to M_2;
Final Output Return M_1 with probability 0.5 and M_2 with probability 0.5.

Proof. First, we provide some high-level intuition on why this algorithm results in a significant improvement over the standard half-optimal greedy and randomized approaches. We first remark that in order to obtain a half-approximation to OPT, it is sufficient to greedily select $\frac{2}{3}(N/2)$ edges (substitute $\alpha = \frac{2}{3}$, $\alpha^* = 1$ in Lemma 3.2). Choosing all $\frac{N}{2}$ edges greedily would be overkill, and so we choose the remaining edges randomly in the First Algorithm of Alg 4. Now, let us denote by Top, the set of $\frac{2}{3}N$ nodes that are matched greedily. The main idea behind the second Algorithm is that if the first one performs poorly (not that much better than half), then, all the 'good edges' must be going across the cut from Top to Bottom (B), where $B = \mathcal{N} \setminus Top$. In other words, $\sum_{(x,y)\in Top\times B} w(x, y)$ must be large, and therefore, the randomized algorithm for bipartite graphs should perform well. In summary, since we randomized between the first and second algorithms, we are guaranteed that at least one of them should have a good performance for any given instance.

We now prove the theorem formally. We begin with a general lemma that presents a lower bound on the quality of the greedy algorithm for Max k-matching for all values of k. This result may be of independent interest.

Lemma 3.2. *Given $k = \alpha\frac{N}{2}$, and $k^* = \alpha^*\frac{N}{2}$, the performance of the greedy k-matching with respect to the optimal k^*-matching (i.e., $\frac{OPT(k^*)}{Greedy(k)}$) is given by,*

(1) $\max(2, 2\frac{\alpha^*}{\alpha})$ *if* $\alpha^* + \alpha < 1$
(2) $\max(2, \frac{\alpha^*+1}{\alpha} - 1)$ *if* $\alpha^* + \alpha \geq 1$

Thus, for example, when $\alpha^* = 1$, and $\alpha = \frac{2}{3}$, we get the factor of 0.5, i.e., in order to obtain a half-approximation to the optimum perfect matching, it suffices to greedily choose two-thirds as many edges as in the perfect matching.

Proof. We show the claim via a charging argument where every edge in the optimum matching M^* is charged to one or more edges in the greedy matching M. Specifically, we can imagine that each edge $e \in M$ contains a certain (not necessarily integral) number of slots s_e, initialized to zero, that measure the number of edges in M^* charged to e. Our proof will proceed in the form of an algorithm: initially $U = M^*$ denotes the set of uncharged edges. In each iteration, we remove some edge from U, charge its weight to some edges in M and increase the value of s_e for the

corresponding edges so that the following invariant always holds: $\sum_{e \in M} s_e w_e \geq \sum_{e^* \in M^* \setminus U} w_{e^*}$. Finally, we can bound the performance ratio using the quantity $\max_{e \in M} s_e$.

We describe our charging algorithm in three phases. Before we describe the first phase, consider any edge $e^* = (a, b)$ in M^*. The edge must belong to one of the following two types.

(1) (Type I) Some edge(s) consisting of a or b (both a and b) are present in M.
(2) (Type II) No edge in M has a or b as an endpoint.

Suppose that M^* contains m_1 Type I edges, and m_2 Type II edges. We know that $m_1 + m_2 = k^*$. Also, let $T \subseteq M$ denote the top $\frac{m_1}{2}$ edges in M, i.e., the $\frac{m_1}{2}$ edges with the highest weight. In the first charging phase, we cover all the Type I edges using only the edges in T, and so that no more than two slots of each edge are required.

Claim 3.3. *(First Phase) There exists a mechanism by which we can charge all Type I edges in M^* to the edges in T so that $\sum_{e \in T} s_e w_e \geq \sum_{e \in TypeI(M^*)} w_e$ and for all $e \in T$, $s_e \leq 2$.*

Proof. We begin by charging the Type I edges to arbitrary edges in M, and then transfer the slots that are outside T to edges in T. Consider any Type I edge $e^* = (a, b)$: without loss of generality, suppose that $e = (a, c)$ is the first edge containing either a or b that was added to M by the greedy algorithm. Since the greedy algorithm only adds undominated edges, we can infer that $w_e \geq w_{e^*}$ (or else e would be dominated by e^*). Using this idea, we we charge the Type I edges as follows

> (Algorithm: Phase I (Charging)) Repeat until U contains no Type I edge: pick a type I edge $e^* = (a, b)$ from U. Suppose that $e = (a, c)$ is the first edge containing either a or b that was added to M. Since $w_e \geq w_{e^*}$, charge e^* to e, i.e., increase s_e by one and remove e^* from U.

At the end of the above algorithm, U contains no type I edge. Moreover, $\sum_{e \in M} s_e = m_1$ since every Type I edge requires only one slot. Finally, for every $e = (x, y) \in M$, $s_e \leq 2$. This is because any edge charged to (x, y) must contain at least one of x or y. Now, without altering the set of uncharged edges U, we provide a mechanism to transfer the slots to edges in T. The following procedure is based on the observation that for every $e, e' \in M$ such that $e' \in T$ and $e \notin T$, $w_{e'} \geq w_e$.

> (Algorithm: Phase I (Slot Transfer)) Repeat until $s_e = 0$ for every edge outside T: pick $e \notin T$ such that $s_e > 0$. Pick any edge $e' \in T$ such that $s_e < 2$. Transfer the edge originally charged to e to e', i.e., decrease s_e by one and increase $s_{e'}$ by one.

Notice that at the end of the above mechanism, $\sum_{e \in T} s_e = m_1$, $s_e \leq 2$ for all $e \in T$, and $s_e = 0$ for all $e \in M \setminus T$. □

Now, consider any type II edge e^*. We make a strong claim: for every $e \in M$, $w_e \geq \frac{1}{2} w_{e^*}$. This follows from Lemma 2.5 since at the instant when e was added to M, e was an undominated edge in the edge set E and e^* was also present in the edge set. Therefore, each type II edge can be charged using two (unit) slots from any of the edges in M (or any combination of them). We now describe the second phase of our charging algorithm that charges nodes only to edges in $M \setminus T$, recall that there $k - \frac{m_1}{2}$ such edges.

> (Second Phase) Repeat until $s_e = 2$ for all $e \in M \setminus T$ (or) until U is empty: pick any arbitrary edge e^* from U and $e \in M \setminus T$ such that $s_e = 0$. Since $w_{e^*} \leq 2w_e$, charge e^* using two slots of e, i.e., increase s_e by two and remove e^* from U.

During the second phase, every edge in M^* is charged to exactly (two slots of) one edge in $M \setminus T$. Therefore, the number of edges removed from U during this phase is $\min(m_2, k - \frac{m_1}{2})$. Since the number of uncharged edges at the beginning of Phase I was exactly m_2, we conclude that the number of uncharged edges at the end of the second phase, i.e., $|U|$ is $\min(0, m_2 - k + \frac{m_1}{2})$. If

$|U| = 0$, we are done, otherwise we can charge the remaining edges in U uniformly to all the edges in M using a fractional number of slots, i.e.,

> (Third Phase) Repeat until $U = \emptyset$: pick any arbitrary edge e^* from U. Since $w_{e^*} \leq 2w_e$ for all $e \in M$, charge e^* uniformly to all edges in M, i.e., increase s_e by $\frac{2}{k}$ for every $e \in M$ and remove e^* from M^*.

Now, in order to complete our analysis, we need to obtain an upper bound for s_e over all edges in e. Recall that at the end of phase II, $s_e \leq 2$ for all $e \in M$. In the third phase, s_e increased by $\frac{2}{k}$ for every edge in U, and the number of edges in U is $\min(0, m_2 - k + \frac{m_1}{2})$. Therefore, at the end of the third phase, we have that for every $e \in M$,

$$s_e \leq 2 + \frac{2}{k}[\min(0, m_2 - k + \frac{m_1}{2}).]$$

Since $m_1 + m_2 = k^*$, we can simplify the second term above and get

$$s_e \leq 2 + \frac{2}{k}[\min(0, \frac{m_2}{2} + \frac{k^*}{2} - k)] \tag{3}$$

$$= 2 + \min(0, \frac{m_2 + k^*}{k} - 2)] = \min(2, \frac{m_2 + k^*}{k}). \tag{4}$$

How large can m_2 be? Clearly, $m_2 \leq k^*$. But a more careful bound can be obtained using the fact that the m_2 Type II edges have no node in common with any of the k edges in M. But the total number of nodes is N, therefore, $2m_2 + 2k \leq N$ or $m_2 \leq \frac{N}{2} - k$. This gives us $m_2 \leq \min(k^*, \frac{N}{2} - k)$. Depending on what the minimum is, we get two cases:

(1) Case I: $k^* \leq \frac{N}{2} - k$ or equivalently, $k^* + k \leq \frac{N}{2}$. Substituting $m_2 \leq k^*$ in Equation 3, we get that for all e, $s_e \leq \min(2, \frac{2k^*}{k})$. Replacing k by $\alpha\frac{N}{2}$ and k^* by $\alpha^*\frac{N}{2}$, we get that when $\alpha + \alpha^* \leq 1$, $s_e \leq \min(2, 2\frac{\alpha^*}{\alpha})$.
(2) Case II: $k^* \geq \frac{N}{2} - k$ or equivalently $\alpha^* + \alpha \geq 1$. Substituting $m_2 \leq \frac{N}{2} - k$ in Equation 3, we get that $s_e \leq \min(2, \frac{\frac{N}{2} + k^* - k}{k})$ or equivalently $s_e \leq \min(2, \frac{\alpha^*+1}{\alpha} - 1)$.

□

Now, we proceed with the proof of the theorem. By linearity of expectation, $E[w(M)] = 0.5(E[w(M_1)] + E[w(M_2)])$. Now, look at the first algorithm, since M_0 has two-thirds as many edges as the optimum matching, we get from Lemma 3.2 that $w(M_0) \geq \frac{1}{2}w(OPT)$. As mentioned in the algorithm, B is the set of nodes that are not present in M_0; since we randomly match the nodes in B to other nodes in B, the expected weight of the random algorithm (from Lemma 2.7 with $n = \frac{N}{3}$) is $\frac{3}{N}\sum_{(x,y)\in B} w(x, y)$. Therefore, we get the following lower bound on the weight of M_1,

$$E[w(M_1)] \geq \frac{OPT}{2} + \frac{3}{N}\sum_{(x,y)\in B} w(x, y).$$

Next, look at the second algorithm: half the edges from M_0 are added to M_2. A constitutes the set of $\frac{N}{3}$ nodes from M_0 that are not present in M_2, these nodes are randomly matched to those in B. Let M_{AB} denote the matching going 'across the cut' from A to B. Since the set A is chosen

randomly from the nodes in Top, the expected weight of the matching from A to B is given by,

$$\begin{aligned} E[w(M_{AB})] &= \sum_{\substack{S \subset Top \\ |S|=\frac{N}{3}}} E\left[w(M_{AB}) \mid (A = S)\right] Pr(A = S) \\ &= \sum_{\substack{S \subset Top \\ |S|=\frac{N}{3}}} Pr(A = S) \sum_{(x,y) \in S \times B} \frac{3}{N} w(x,y) \\ &= \sum_{(x,y) \in Top \times B} \frac{3}{N} w(x,y) Pr(x \in A) \\ &= \sum_{(x,y) \in Top \times B} \frac{3}{N} w(x,y) \times \frac{1}{2} \end{aligned} .$$

The second equation above comes from Lemma 2.7 Part 2 for $n = \frac{N}{3}$, and the last step follows from the observation that $Pr(x \in A)$ is exactly equal to the probability that the edge containing x in M_0 is not added to M_2, which is one half (since the edge is chosen with probability 0.5). We can now bound the performance of M_2 as follows,

$$\begin{aligned} E[w(M_2)] &= \frac{1}{2} E[w(M_0)] + E[w(M_{AB})] \\ &= \frac{1}{2} w(M_0) + \frac{3}{2N} \sum_{(x,y) \in Top \times B} w(x,y). \end{aligned}$$

Now, let us apply Lemma 2.8 to the set $T = B$ ($n = \frac{N}{3}$), with OPT being the matching: we get that $w(OPT) \leq \frac{6}{N} \sum_{x \in B, y \in B} w(x,y) + \frac{3}{N} \sum_{x \in Top, y \in B} w(x,y)$ or equivalently $\frac{3}{2N} \sum_{(x,y) \in Top \times B} w(x,y) \geq \frac{w(OPT)}{2} - \frac{3}{N} \sum_{x \in B, y \in B} w(x,y)$. Substituting this in the above equation for M_2 along with the fact that $w(M_0) \geq \frac{w(OPT)}{2}$, we get the following lower bound for the performance of M_2 in terms of OPT,

$$E[w(M_2)] \geq \frac{w(OPT)}{4} + \frac{w(OPT)}{2} - \frac{3}{N} \sum_{x \in B, y \in B} w(x,y).$$

Recall that

$$E[w(M_1)] \geq \frac{w(OPT)}{2} + \frac{3}{N} \sum_{x \in B, y \in B} w(x,y).$$

The final bound comes from adding the two quantities above and multiplying by half. □

3.2 Truthful Mechanisms for Matching

In Section 2, we looked at three simple approaches for designing truthful mechanisms for the maximum weighted matching problem, all of which yield truthful 2-approximations to the optimum matching. Can we do any better without sacrificing truthfulness? Previously, we used a complex interleaving of greedy and random approaches to extract a *non-truthful* 1.6-approximation algorithm. Now, we present a simpler algorithm and rather surprising result: a simple random combination of Algorithms 1 and 2 results in a 1.77-approximation to the optimum matching. The main insight driving this result is the fact that the random and greedy approaches are in some senses complementary to each other, i.e., on instances where the approximation guarantee for the greedy algorithm is close to 2, the random algorithm performs much better.

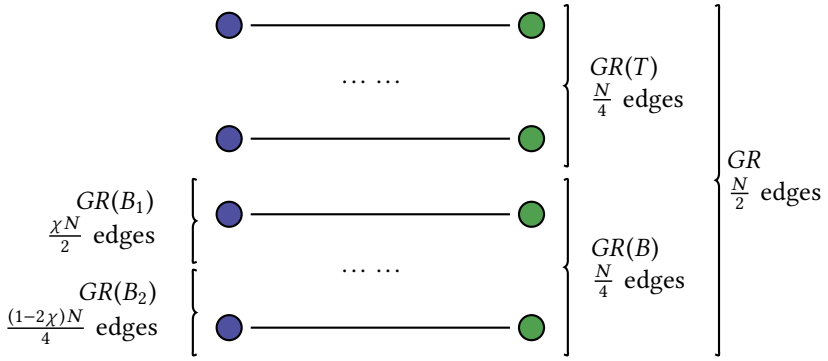

Fig. 1. An illustration of the greedy matching (*GR* specifies the edges in the greedy matching) partitioned into three subsets: $Gr(T), Gr(B_1), Gr(B_2)$ with $Gr(B) = Gr(B_1) \cup Gr(B_2)$ and $Gr = Gr(T) \cup Gr(B_1) \cup Gr(B_2)$. The edges depicted are arranged in the decreasing order of weight from top to bottom so that (*i*) every edge in $GR(T)$ has larger weight than any edge in $GR(B)$ and (*ii*) every edge in $GR(B_1)$ has larger weight than any edge in $GR(B_2)$.

Theorem 3.4. *The following algorithm is a universally truthful mechanism for the maximum weighted matching problem that obtains a* 1.7638*-approximation to the optimum matching.*
***Greedy-Random Mix Algorithm for Maximum Weighted Matching**: With probability $\frac{3}{7}$, return the output of Algorithm 1 for $k = \frac{N}{2}$ and with probability $\frac{4}{7}$, return the output of Algorithm 2 for $k = \frac{N}{2}$.*

Proof. Our proof mainly involves non-trivial lower bounds on the performance of the random matching which highlight its complementary nature to the greedy matching. We first define some important notation that allows us to partition the greedy matching into three components and quantify their weight separately. An illustration of this partition can be seen in Figure 1.

Notation. : Suppose that *GR* denotes the output of the greedy matching algorithm for the given instance, and *RD* is the random matching for the same instance.

(1) Let $GR(T)$ and $GR(B)$ denote the sub-matchings of *GR* of size $\frac{N}{4}$ containing the highest-weight and lowest-weight edges in *GR* respectively, i.e., $|GR(T)| = |GR(B)| = \frac{|GR|}{2}$.
(2) Suppose that χ is the weight of the edges in *B* relative to the optimum matching, i.e.,

$$\chi = \frac{w(GR(B))}{w(OPT)}.$$

(3) Let $GR(B_1)$ be the sub-matching of $GR(B)$ containing the $\frac{\chi N}{2}$ highest-weight edges in $GR(B)$ and let $GR(B_2) = GR(B) \setminus GR(B_1)$.
(4) Let T, B, B_1, B_2 denote the set of nodes that form the edges in $GR(T), GR(B), GR(B_1), GR(B_2)$ respectively.

Proof Outline. Although our algorithm is extremely simple, the proof is technically involved so we first outline the key ideas in the proof. As per definition, the weight of the greedy matching is given by $w(GR) = w(GR(T)) + w(GR(B))$. Our first step is to prove that $w(GR(T)) \geq \frac{w(OPT)}{2}$: this

is done via a charging argument similar to the one used in Lemma 3.2. It then follows that:

$$w(GR) = w(GR(T)) + w(GR(B)) \geq \frac{1}{2}w(OPT) + \chi w(OPT). \tag{5}$$

If χ is large, then the greedy algorithm already obtains a good approximation to the optimum matching. Our next step is to show that when χ is small, the random matching gives us an approximation factor better than half. This statement is formalized in the following lemma.

Lemma 3.5. *The weight of the random matching is always at least*

$$E[w(RD)] \geq \frac{5}{8}w(OPT) - \chi(1 - \frac{3}{2}\chi)w(OPT).$$

Moreover, when $\chi \leq \frac{1}{8}$, the following is a tighter lower bound for the random matching

$$E[w(RD)] \geq \frac{5}{8}w(OPT) - \chi(1 - 2\chi)w(OPT).$$

The main idea behind the above lemma is that when χ is small, the weight of the greedy matching on the nodes in B is small, and therefore, a large number of the heavy edges in the matching go 'across the cut' from T to B. The random matching is able to select these edges with high probability leading to an improved approximation factor when compared to the greedy matching.

The final approximation bound follows directly from Lemma 3.5. We show this in two cases depending on whether or not $\chi \leq \frac{1}{8}$.

Case I: $\chi \leq \frac{1}{8}$. Recall that we pick the random matching with probability $p = \frac{4}{7}$ and the greedy mathing with probability $1 - p = \frac{3}{7}$. Suppose we use $w(M)$ to denote the weight of the matching returned by our algorithm. Then applying Equation 5 and Lemma 3.5, we get,

$$\begin{aligned} E[w(M)] &= (1-p)\cdot w(GR) + p \cdot w(RD) \\ &\geq w(OPT)\{(1-p)(\frac{1}{2}+\chi) + (p)(\frac{5}{8} - \chi + 2\chi^2)\} \\ &= w(OPT)\{\frac{1}{2} + p\frac{1}{8} + \chi(1-2p) + 2p\chi^2\} \end{aligned}$$

Given $p = \frac{4}{7}$, the quantity $\chi(1-2p) + 2p\chi^2$ is minimized at $\chi = \frac{1}{2} - \frac{1}{4p} = \frac{1}{16}$. Substituting the values for p, χ (that result in the worst-case guarantee), we obtain $\frac{w(OPT)}{E[w(M)]} \leq 1.7638$.

Case II: $\chi \geq \frac{1}{8}$. In this case, we need to use a weaker lower bound for RD, also provided in Lemma 3.5.

$$\begin{aligned} E[w(M)] &= (1-p)\cdot w(GR) + p \cdot w(RD) \\ &\geq w(OPT)\{(1-p)(\frac{1}{2}+\chi) + (p)(\frac{5}{8} - \chi + \frac{3}{2}\chi^2)\} \\ &= w(OPT)\{\frac{1}{2} + p\frac{1}{8} + \chi(1-2p) + \frac{3}{2}p\chi^2\} \end{aligned}$$

The expression in the final line is a non-decreasing function of χ in the range $[\frac{1}{8}, \frac{1}{2}]$[2] and so, its minimum value is attained at $\chi = \frac{1}{8}$. Substituting this value above, we get $\frac{w(OPT)}{E[w(M)]} \leq 1.7638$. This completes the proof outline.

(Proof) Given the steps outlined above, it only remains for us to prove that (i) $w(GR(T)) \geq \frac{1}{2}w(OPT)$ and (ii) Lemma 3.5 holds. We do so below.

[2]Note that $\chi \leq \frac{1}{2}$ because $w(GR(T)) + \chi w(OPT) \leq w(OPT)$ and $w(GR(T)) \geq \frac{1}{2}w(OPT)$.

Lemma 3.6.

$$w(GR(T)) \geq \frac{w(OPT)}{2}.$$

Proof. We proceed via the standard charging argument applied to prove the half-optimality of the greedy algorithm. Pick any edge in OPT, say $e = (x, y)$, if $(x, y) \in GR$, we charge the edge to itself. Otherwise, at least one of x or y must be matched to an edge that yields it the same or better utility, i.e., w.l.o.g, $\exists (x, z) \in GR$ such that $w(x, z) \geq w(x, y)$. In this case, we charge (x, y) to (x, z). Clearly, every edge in GR has anywhere between 0 to 2 edges (from OPT) assigned to it.

For any $e \in GR$, suppose that s_e is the number of edges from OPT assigned to e. By our charging argument, the following inequality must be true,

$$OPT \leq \sum_{i=1}^{\frac{N}{2}} w_{e_i} s_{e_i},$$

where e_i is the i^{th} largest edge belonging to GR. Observe that $\sum_{i=1}^{\frac{N}{2}} s_{e_i} = \frac{N}{2}$. Consisder an alternative 'slot vector', $(\mathbf{q})_{e_i \in GR}$ such that $q_{e_i} = 2$ if $i \leq \frac{N}{4}$ and $q_{e_i} = 0$ otherwise. Since $w_{e_i} \geq w_{e_j}$ whenever $i < j$, it is not hard to deduce that:

$$\sum_{i=1}^{\frac{N}{2}} w_{e_i} s_{e_i} \leq \sum_{i=1}^{\frac{N}{2}} w_{e_i} q_{e_i} = 2w(GR(T)).$$

□

Before proving Lemma 3.5, we state two useful but technical propositions.

Proposition 3.7.

(1) *No edge in $GR(B_2)$ can have a weight larger than $w^* := \frac{2GR(B_1)}{\chi N}$.*
(2) *For any given instance, we have that $2\chi \leq \chi_1 := \frac{w(GR(B_1))}{w(GR(B))}$.*

The first part is true because this is the average of the edge weights in $GR(B_1)$, which must be larger than the weight of any edge in $GR(B_2)$. The second part comes from the fact that $GR(B_1)$ consists of $\frac{xN}{2}$ edges whereas $GR(B)$ consists of $\frac{N}{4}$ edges. From now on, we use χ_1 to denote the quantity $\frac{w(GR(B_1))}{w(GR(B))}$. Note that when χ_1 is smaller than $\frac{1}{2}$, the weights of the edges in $GR(B)$ are somewhat evenly distributed across $GR(B_1)$ and $GR(B_2)$.

Given a set $S \subseteq \mathcal{N}$ (e.g., $S = T, B, B_1, B_2$), we abuse notation and use $w(S)$ to denote the quantity $\sum_{x,y \in S} w(x, y)$. Similarly, given two distinct sets $S_1, S_2 \subseteq \mathcal{N}$, we use $w(S_1, S_2)$ to represent $\sum_{x \in S_1, y \in S_2} w(x, y)$. The following proposition relates the weight of quantities such as $w(T)$, $w(B)$, etc. to the weight of edges in the greedy matching: its proof uses similar ideas as the proofs of Lemmas 2.8 and 2.7 as well as new insights on the greedy matching. However, owing to its rather technical nature, the proof is deferred to Appendix D.

Proposition 3.8. *The following inequalities hold:*

$$w(B_1, B_2) \leq 2w(GR(B_1))|B_2| \tag{6}$$

$$w(B_2) \leq 2w(GR(B_2))\{|B_2| - \frac{w(GR(B_2))}{w^*}\} \tag{7}$$

$$w(B) \leq 2|B|w(GR(B))(1 - 2\chi) \quad \text{when } \chi_1 \in [0, \frac{1}{2}] \tag{8}$$

All that remains is for us to actually prove Lemma 3.5.

(Proof of Lemma 3.5) We prove this lemma in three parts depending on the values of χ, χ_1.

(1) (Part 1: $\chi \geq 0, \chi_1 \geq \frac{1}{2}$) $w(RD) \geq \frac{5}{8}w(OPT) - \chi(1 - \frac{3}{2}\chi)w(OPT)$.
(2) (Part 2: $\chi \leq \frac{1}{8}, \chi_1 \geq \frac{1}{2}$) $w(RD) \geq \frac{5}{8}w(OPT) - \chi(1 - 2\chi)w(OPT)$.
(3) (Part 3: $\chi \leq \frac{1}{4}, \chi_1 \leq \frac{1}{2}$) $w(RD) \geq \frac{5}{8}w(OPT) - \chi(1 - 2\chi)w(OPT)$.

Clearly, the above three parts are sufficient to prove the claim made in the statement of Lemma 3.5. Note that since $\chi_1 \geq 2\chi$, there is no need to consider the case where $\chi \geq \frac{1}{4}, \chi_1 \leq \frac{1}{2}$. We tackle the three parts sequentially.

(Part 1: $\chi \geq 0, \chi_1 \geq \frac{1}{2}$). The expected weight of the random matching can be expressed as

$$E[w(RD)] \geq \frac{1}{N}\left(w(T) + w(T, B) + w(B)\right). \tag{9}$$

However, applying Lemma 2.8 with $T = B, \mathcal{S} = \mathcal{N}$ and $M = OPT$, we get that $\frac{1}{N}(w(T, B) + w(B)) \geq w(OPT)/2 - w(B)/N$. Substituting this into Equation 9 gives us:

$$E[w(RD)] \geq \frac{1}{2}w(OPT) + \frac{1}{N}\{w(T) - w(B)\}. \tag{10}$$

Alternatively, we can also apply Lemma 2.8 with $T = B_2, \mathcal{S} = \mathcal{N}$ and $M = OPT$ to get that: $\frac{1}{|B_2|}(w(T \cup B_1, B_2) + w(B_2)) \geq w(OPT) - w(B_2)/|B_2|$. This gives us.

$$E[w(RD)] \geq \frac{1}{N}\{w(T) + w(T, B_1) + w(B_1) + \frac{|B_2|}{N}w(OPT) - w(B_2)\} \tag{11}$$

Adding Equations 10 and 11, dividing by two, and substituting $\frac{|B_2|}{N} = \frac{1}{2} - \chi$, we get:

$$E[w(RD)] \geq \frac{1}{2}w(OPT) - \frac{\chi}{2}w(OPT) + \frac{1}{N}\left(w(T) + \frac{1}{2}(w(T, B_1) - w(B_1, B_2)) - w(B_2)\right). \tag{12}$$

Applying Lemma 2.8 with $T = \mathcal{S} = T$, $M = GR(T)$ and $w(GR(T)) \geq \frac{w(OPT)}{2}$, we get that $\frac{w(T)}{N} \geq \frac{1}{8}w(OPT)$. Now for every edge $e = (x, y)$ in $GR(T)$, note that the triangle inequality implies that for any node $z \in B_1$, $w(x, z) + w(y, z) \geq w(x, y)$. Summing these up, we get that $w(T, B_1) \geq |B_1|w(GR(T)) \geq |B_1|w(OPT)/2$. Substituting these lower bounds for $w(T)$ and $w(T, B_1)$ back into Equation 12, we get a slightly simplified expression.

$$w(RD) \geq \frac{5}{8}w(OPT) - \frac{\chi}{4}w(OPT) - \frac{1}{N}\{\frac{1}{2}w(B_1, B_2) + w(B_2)\}. \tag{13}$$

From Proposition 3.8, Equation 6, we get that $w(B_1, B_2) \leq 2w(GR(B_1))|B_2| = 2\chi_1 \cdot \chi w(OPT)(1/2 - \chi)N$. To complete the proof, we have to provide an upper bound on $w(B_2)$. To do so, we can directly apply Equation 7 from Proposition 3.8 to obtain:

$$w(B_2) \leq 2w(GR(B_2))(\frac{N}{2} - \chi N - t),$$

where $t = \frac{w(GR(B_2))}{w^*} = \frac{N(1-\chi_1)\chi w(OPT)}{2\chi_1 w(OPT)} = \frac{N(1-\chi_1)\chi}{2\chi_1}$. In conclusion, we have that

$$\frac{1}{N}w(B_2) \leq (1 - \chi_1)\chi w(OPT)(1 - 2\chi - \frac{1 - \chi_1}{\chi_1}\chi) \leq (1 - \chi_1)\chi w(OPT)(1 - 2\chi). \tag{14}$$

Combining the above equation with our upper bound on $w(B_1, B_2)$, we get that:

$$\begin{aligned}\frac{1}{2N}w(B_1,B_2)+\frac{1}{N}w(B_2) &\leq \chi_1\cdot\chi w(OPT)(1/2-\chi)+(1-\chi_1)\chi w(OPT)(1-2\chi)\\ &= \chi w(OPT)(1/2-\chi)(\chi_1+2-2\chi_1) \qquad (15)\\ &\leq \chi w(OPT)(1/2-\chi)\frac{3}{2} \qquad (\text{Since } \chi_1\geq\frac{1}{2})\\ &= \frac{3}{4}\chi w(OPT)-\frac{3}{2}\chi^2 w(OPT).\end{aligned}$$

Plugging the final inequality into Equation 13 completes the proof of Part I. □

(Part 2: $\chi \leq \frac{1}{8}, \chi_1 \geq \frac{1}{2}$) The proof of the second part picks up from the previous part with only a few simple tweaks. Specifically, from Equation 15, we have that

$$\frac{1}{2N}w(B_1,B_2)+\frac{1}{N}w(B_2)\leq \chi w(OPT)\left[\chi_1(1/2-\chi)+(1-\chi_1)(1-\chi-\frac{\chi}{\chi_1})\right].$$

Using basic calculus, we infer that the expression inside the square parenthesis attains its maximum value for $\chi_1 = \frac{1}{2}$ in the given range of χ. Therefore, substituting $\chi_1 = \frac{1}{2}$, we get

$$\frac{1}{2N}w(B_1,B_2)+\frac{1}{N}w(B_2)\leq w(OPT)\{\frac{\chi}{4}-\frac{\chi^2}{2}+\frac{\chi}{2}-\frac{3}{2}\chi^2\}.$$

Directly plugging this upper bound into Equation 13 completes the proof of the second part. □

(Part 3: $\chi \leq \frac{1}{4}, \chi_1 \leq \frac{1}{2}$) Consider Equation 10,

$$w(RD)\geq\frac{1}{2}w(OPT)+\frac{1}{N}\{w(T)-w(B)\}.$$

Once again, applying Lemma 2.8 with $T = \mathcal{S} = T$, $M = GR(T)$ and $w(GR(T)) \geq \frac{w(OPT)}{2}$, we get that $\frac{w(T)}{N} \geq \frac{1}{8}w(OPT)$. Therefore, it suffices to prove an upper bound on $\frac{w(B)}{N}$. Recall that B consists of exactly $n = \frac{N}{2}$ nodes, $GR(B) = \chi w(OPT)$, and $w(GR(B_1)) = \chi_1\cdot\chi w(OPT) \leq \frac{1}{2}\chi w(OPT)$ since $\chi_1 \leq \frac{1}{2}$ by assumption in this part of the proof. So, directly applying Equation 8 within Proposition 3.8, we get that,

$$\frac{1}{n}w(B)=\frac{2}{N}w(B)\leq 2w(GR(B))(1-2\chi).$$

So, $\frac{1}{N}w(B) \leq \chi w(OPT)(1-2\chi)$. Putting this inside Equation 11 along with $w(T)/N \geq w(OPT)/8$, we complete the proof of this part of the lemma. □

Lemma 3.5 follows. □

3.3 Lower Bound Example for Ordinal Matchings

Before concluding this section, it is important to understand the limitations of ordinal information. As mentioned in the Introduction, different sets of weights can give rise to the same preference ordering, and therefore, we cannot suitably approximate the optimum solution for every possible weight. We now show that even for very simple instances, there can be no deterministic 1.5-approximation algorithm, and no randomized 1.25-approximation algorithm for maximum weighted matching.

Claim 3.9. *No deterministic ordinal approximation algorithm can provide an approximation factor better than* 1.5*, and no randomized ordinal approximation algorithm can provide an approximation factor better than* 1.25 *for Maximum Weighted Matching. No ordinal algorithm, deterministic or randomized can provide an approximation factor better than* 2 *for Max k-Matching.*

Proof. Consider an instance with 4 nodes having the following preferences: (i) $a : b > c > d$, (ii) $b : a > d > c$, (iii) $c : a > b > d$, (iv) $d : b > a > c$. Since the matching $\{(a,d),(b,c)\}$ is weakly dominated, it suffices to consider algorithms that randomize between $M_1 = \{(a,b),(c,d)\}$, and $M_2 = \{(a,c),(b,d)\}$, or deterministically chooses one of them.

Now, consider the following two sets of weights, both of which induce the above preferences but whose optima are M_2 and M_1 respectively: $W_1 := w(a,b) = w(a,c) = w(b,d) = w(a,d) = w(b,c) = 1$, $w(c,d) = \epsilon$, and $W_2 := w(a,b) = 2, w(a,c) = w(b,d) = w(c,d) = w(a,d) = w(b,c) = 1$. The best deterministic algorithm always chooses the matching M_2, but for the weights W_2, this is only a $\frac{3}{2}$-approximation to OPT.

Consider any randomized algorithm that chooses M_1 with probability x, and M_2 with probability $(1-x)$. With a little algebra, we can verify that just for W_1, and W_2, the optimum randomized algorithm has $x = \frac{2}{5}$, yielding an approximation factor of 1.25.

For the Max k-Matching problem, our results are tight. For small values of k, it is impossible for any ordinal algorithm to provide an approximation factor better than two. To see why, consider an instance with $2N$ nodes $\{a_1, b_1, a_2, b_2, \ldots, a_N, b_N\}$. Every a_i's first choice is b_i and vice-versa, the other preferences can be arbitrary. Pick some i uniformly at random and set $w(a_i, b_i) = 2$, and all the other weights are equal to 1. For $k = 1$, it is easy to see that no randomized algorithm can always pick the max-weight edge and therefore, as $N \to \infty$, we get a lower bound of 2. □

4 DENSEST K-SUBGRAPH

Previously, we considered the Maximum Weighted Matching problem and presented both truthful and non-truthful algorithms that improve upon the naive guarantee. Here, we move on to the somewhat harder Densest k-subgraph problem and present approximation algorithms for both the information models. It is important to note that the metric Densest k-subgraph problem is NP-Hard even when the weights are fully known [24, 41]. Finally, given any set $S \subseteq \mathcal{N}$, node x, $w(S)$ will denote the total weight of the edges inside S, and $w(x,S) := \sum_{j \in S} w(x,j)$.

4.1 Ordinal Densest k-Subgraph without Truthfulness

Our 4-approximation for Densest k-subgraph comes via a black box reduction to the Max k-matching problem. Specifically, we show how to convert a matching into a feasible solution for densest subgraph with only a factor two loss in approximation. In combination with the greedy algorithm, this gives a factor 4 for densest subgraph.

Theorem 4.1. *We can compute in polynomial time a 4-approximation to the optimum for Densest k-subgraph.*

The algorithm is as follows: *Use algorithm 1 to compute a matching M of size $\frac{k}{2}$. Return the nodes that form the endpoints of the edges in M as a solution for Densest k-subgraph.*

Proof. Once again, we use M^* to denote the optimum $\frac{k}{2}$-matching and M to denote the solution returned by Algorithm 1. Let O be the optimum solution to the Densest k-subgraph problem for the given value of k. Then our algorithm simply returns the solution S comprising of the endpoints of all the edges in M.

First, we establish a lower bound on the quality of our solution S in terms of the weight of M.

$$\begin{aligned} w(S) &= \sum_{(x,y)\in S} w(x,y) \\ &\geq \frac{k}{2} w(M) && \text{(Lemma 2.8).} \end{aligned}$$

Next, we present an upper bound for the optimum solution in terms of the weight of M^*. Suppose that $M^*(O)$ is the optimum matching that can be formed using the nodes in O.

$$\begin{aligned} w(O) &= \sum_{(x,y)\in O} w(x,y) \\ &\leq k w(M^*(O)) && \text{(Lemma 2.7)} \\ &\leq k w(M^*). \end{aligned}$$

To conclude, it suffices to show that the solution returned by the greedy matching algorithm M is a 2-approximation to the optimum k-matching M^*. This follows from Lemma 3.2. □

4.2 Truthful Algorithm for Densest k-Subgraph

In this section we present our truthful, ordinal algorithm for Densest k-subgraph, which requires techniques somewhat different from the ones outlined in Section 2. While "conventional" approaches such as Greedy and Serial Dictatorship *do* lead to good approximations for this problem, they are not truthful, whereas random approaches are truthful but result in poor worst-case approximation factors. We combat this problem with a somewhat novel approach that combines the best of both worlds by designing a semi-oblivious algorithm that has the following property: if agent i is included in the solution, then changing her preference ordering s_i does not affect the mechanism's output.

ALGORITHM 5: Hybrid Algorithm for Densest k-Subgraph

Input: $\mathcal{N}, k$;
$S := \emptyset$, T is the set of available agents initialized to $\mathcal{N}$;
while $|S| < k$ **do**
  pick an anchor agent a and another node x, both uniformly at random from T;
  let b denote a's most preferred agent in $T - \{a, x\}$;
  with probability $\frac{1}{2}$, add a, x to S, and set $T = T - \{a, x\}$;
  with probability $\frac{1}{2}$, add b, x to S and set $T = T - \{a, b, x\}$;
end

Theorem 4.2. *Algorithm 5 is a universally truthful mechanism that yields a* 8*-approximation for the Densest* k*-Subgraph problem.*

To see why this is truthful, note that for any particular choice of the anchor agent a, the only case in which a's preference ordering makes a difference is when a is definitely not added to the final set of chosen nodes. Therefore, by lying, a cannot influence her utility in the event that she is actually chosen.

Remark on size of k Without loss of generality, we assume that $k \leq \frac{N}{2}$ so that T does not become empty before $|S| = k$. When $k \geq \frac{N}{2}$, there is a trivial algorithm that yields a 8-approximation to the optimum densest subgraph (see Appendix E). Since we are interested in asymptotic performance bounds, we also assume that k is even.

Proof. We begin by defining some notation pertinent to the analysis. For any given set $H \subseteq \mathcal{N}$, let $Alg_r(H)$ denote the (random) solution output by Algorithm 5 for the inputs H, r—i.e., the Densest r-subgraph returned by our algorithm on the set of nodes in H. Similarly, let $OPT_r(H)$ denote the optimum densest subgraph for this specific instance. Next, let us examine the inner workings of the algorithm. In any given round, the algorithm select an ordered triplet $\Delta := \{a, x, b\} \subset T$, where a is referred to as the anchor node, x is a node selected uniformly at random, and b is a's most preferred agent in $T - \{a, x\}$. Given Δ, the algorithm adds either a, x or b, x to S with equal probability.

(Outline of Proof) We prove by induction that for any given set[3] $T \subseteq \mathcal{N}$, $w(OPT_r(T)) \leq 8E[w(Alg_r(T))]$—the induction proceeds on even values of r. Suppose that $\Delta = \{a, x, b\}$ denotes the random triplet selected by the algorithm in the first round. Our proof is based on charging the weight of the edges in both $OPT_r(T)$ and $Alg_r(T)$ to the edge (a, b). More specifically, we identify special nodes $p, q \in OPT_r(T)$ and using these as intermediaries, show that,

$$\text{(Upper Bound)} \qquad w(OPT_{r+2}(T)) - w(OPT_r(T_2)) \leq 4(r+1)w(a,b)$$

$$\text{(Lower Bound)} \; E[w(Alg_{r+2}(T))] - E[w(Alg_r(T_2))] \geq \frac{r+1}{2} w(a,b),$$

where T_2 denotes the remaining nodes in T after the algorithm discards either a, x or a, b, x with equal probability. The rest of the proof is based on applying the induction hypothesis for the set T_2 and parameter r. Finally, we also identify a corner-case in which the upper bound for $OPT_{r+2}(T)$ does not follow from standard techniques—this corresponds to the scenario where $\Delta \subset OPT_{r+2}(T)$ and our algorithm removes three nodes that belong to the optimum solution (with probability one-half). For this case alone, we redo the analysis and derive stronger upper bounds using the simple fact that for any $i \in OPT_{r+2}(T)$, $w(a, i) \leq \max(w(a, x), w(a, b))$.

Before showing our induction hypothesis, we state an independent claim that will be useful later.

Proposition 4.3. *The following statements hold:*

1. *Consider any set of nodes T, and suppose that for some $x \in T$, y denotes x's most preferred node in T. Then for every $i, j \in T$, we have that $w(i, j) \leq 2w(x, y)$.*
2. *Consider any two sets of nodes O, T such that $O \subseteq T$ and let $p \in T$. Then, $w(p, O) \leq |O| w^*_{max}$, where w^*_{max} is the highest weight edge that can be formed using the nodes in T.*
3. *Consider a set of nodes O and any two nodes i, j. Then, $w(i, O) + w(j, O) \geq |O| w(i, j)$.*

Proof. The proof of the first statement is similar to that of Lemma 2.5. Indeed, using the triangle inequality, we see that $w(i, j) \leq w(i, x) + w(x, j) \leq 2w(x, y)$. The second statement is fairly straightforward: $w(p, O) = \sum_{i \in O} w(i, p) \leq \sum_{i \in O} w^*_{max} = |O| w^*_{max}$. The final statement comes from the triangle inequality; $w(i, O) + w(j, O) = \sum_{q \in O} (w(i, q) + w(j, q)) \geq \sum_{q \in O} w(i, j) = |O| w(i, j)$. □

Induction Hypothesis: $w(OPT_r(T)) \leq 8E[w(Alg_r(T))]$ **for even** $r \leq k$ **and all** $T \subseteq \mathcal{N}$

We implicitly assume that $|T| \geq 2r$ as mentioned previously.

(**Base Case:** $r = 2$) $w(OPT_2(T)) \leq 8E[w(Alg_2(T))]$: The base case is quite straightforward. Suppose that $w^*_{max} = \max_{i,j \in T} w(i, j)$. Clearly, $w(OPT_2(T)) = w^*_{max}$. Suppose that $\Delta = \{a, x, b\}$ is the triplet selected by the algorithm such that b denotes a's most preferred node in $T - \{x\}$. Then, it is not hard to deduce that a's most preferred node in T is one of b or x. So, the highest weight edge containing the node a must be either (a, x) or (a, b).

Therefore, we can apply Proposition 4.3 (statement 1) and get that $w^*_{max} \leq 2 \max(w(a, x), w(a, b))$. Next, consider the set $Alg_2(T)$ for this fixed choice of Δ. Since $Alg_2(T) = \{a, x\}$ with probability

[3]Recall that for a given set $T \subseteq \mathcal{N}$ and node $i \in \mathcal{N}$, $w(T) := \sum_{x,y \in T} w(x, y)$ and $w(i, T) = \sum_{x \in T} w(i, x)$.

one-half and $\{b, x\}$ with probability one-half, we have the following easy inequalities:

$$w(Alg_2(T)) = \frac{1}{2}(w(a,x) + w(b,x)) \geq \frac{1}{2}w(a,b) \quad \text{(Triangle Inequality)},$$
$$w(Alg_2(T)) \geq \frac{1}{2}w(a,x).$$

From the above two inequalities, we have that $w(Alg_2(T)) \geq \frac{1}{2}\max(w(a,x), w(a,b)) \geq \frac{1}{4}w^*_{max} = \frac{OPT_2(T)}{4}$. Taking the expectation over every such triplet Δ, we get that $w(OPT_2(T)) \leq 8E[w(Alg_2(T))]$.

Inductive Step: To Prove $w(OPT_{r+2}(T)) \leq 8E[w(Alg_{r+2}(T))]$

We assume that $w(OPT_r(T')) \leq 8E[w(Alg_r(T'))]$ for all $T' \subseteq \mathcal{N}$. Consider a fixed instantiation of $\Delta = \{a, x, b\}$, the ordered triplet of nodes selected by the algorithm in its first iteration. Let $w^*_a = \max(w(a,b), w(a,x))$, $T_1 = T \setminus \{a, x\}$, and $T_2 = T \setminus \{a, x, b\}$. We show an upper bound on $w(OPT_{r+2}(T))$ and a lower bound on $E[w(Alg_{r+2}(T))]$ in terms of w^*_a. As mentioned previously, the proof of the upper bound proceeds in two cases—when $\Delta \subseteq OPT_{r+2}(T)$ and when $|\Delta \cap OPT_{r+2}(T)| < 3$.

Upper Bound Case I: $|\Delta \cap OPT_{r+2}(T)| < 3$: We begin by defining two 'special nodes' p, q that allow us to relate $w(OPT_{r+2}(T))$ to $w(a,b)$ and $w(a,x)$.

- If $|\Delta \cap OPT_{r+2}(T)| = 2$, then $\{p, q\} \coloneqq \Delta \cap OPT_{r+2}(T)$.
- If $|\Delta \cap OPT_{r+2}(T)| = 1$, then $\{p\} \coloneqq OPT_{r+2}(T) \cap \Delta$ and q is any arbitrary node in $OPT_{r+2}(T)$ that is not p.
- If $|OPT_{r+2}(T) \cap \Delta| = 0$, then p, q are two arbitrary but different nodes belonging to $OPT_{r+2}(T)$

Informally, we first assign the nodes common to both Δ and $OPT_{r+2}(T)$ to p and(/or) q. If this does not suffice, then p and q are assigned arbitrary nodes belonging to $OPT_{r+2}(T)$. As a consequence of this careful definition, we have that the set $O \coloneqq OPT_{r+2}(T) \setminus \{p, q\}$ does not contain any node in Δ. Now, the weight of the edges inside the set $OPT_{r+2}(T)$ can be expanded as:

$$w(OPT_{r+2}(T)) = w(O) + w(p, O) + w(q, O) + w(p, q).$$

We can rewrite the term $w(p, O) + w(q, O)$ by applying Proposition 4.3(second statement) to both p and q and adding up the resulting inequalities. This gives us,

$$w(p, O) + w(q, O) \leq 2|O| \max_{i,j \in T} w(i, j).$$

Further, we know from Proposition 4.3 (statement 1) that $\max_{i,j \in T} w(i, j) \leq 2\max(w(a,b), w(a,x)) = 2w^*_a$. So, we have that $w(p, O) + w(q, O) \leq 4|O|w^*_a$, where $|O| = r$. In summary, we can bound $OPT_{r+2}(T)$ in terms of w^*_a as follows:

$$\begin{aligned} w(OPT_{r+2}(T)) &\leq w(O) + 4rw^*_a + w(p, q) \\ &\leq w(O) + (4r + 2)w^*_a \\ &\leq \frac{1}{2}\left(w(OPT_r(T_1)) + w(OPT_r(T_2))\right) + (4r + 2)w^*_a. \end{aligned} \tag{16}$$

The penultimate inequality stems from the fact that $w(p, q) \leq \max_{i,j \in T} w(i, j) \leq 2w^*_a$. Since $O \subseteq T_1$ and $O \subseteq T_2$ by definition, it follows that the weight of the edges inside O is smaller than or equal to the weight of the densest subgraph of size r that can be obtained from either T_1 or T_2. Combining the two, we get that $w(O) \leq \frac{1}{2}\left(w(OPT_r(T_1)) + w(OPT_r(T_2))\right)$, which leads to the final inequality.

Upper Bound Case II: $|\Delta \cap OPT_{r+2}(T)| = 3$: Note that in this case, $a, x, b \in OPT_{r+2}(T)$. This is a tricky case because our algorithm (with probability one-half) removes three nodes belonging to

the optimum set from consideration. Let $O_2 = OPT_{r+2}(T) \setminus \Delta$. Using the same kind of ideas as in the previous case, we can expand on the weight of the optimum solution as follows:

$$w(OPT_{r+2}(T)) \leq w(O_2) + w(a, O_2) + w(x, O_2) + w(b, O_2) + w(a, b) + w(a, x) + w(b, x).$$

Since $O_2 \subseteq T$ and w^*_a denotes the maximum weight of any edge connected to a inside of T, we have that: $w(a, O_2) = \sum_{i \in O_2} w(a, i) \leq \sum_{i \in O_2} w^*_a = (r-1)w^*_a$. Moreover, we can now apply Proposition 4.3 (statement 2) to (upper) bound $w(x, O_2) + w(b, O_2)$ by $2(r-1)\max_{i,j \in T} w(i, j)$, which we know to be smaller than or equal to $4(r-1)w^*_a$. Adding up the upper bounds for these three quantities, we get that:

$$\begin{aligned} w(OPT_{r+2}(T)) &\leq w(O_2) + 5(r-1)w^*_a + w(a, b) + w(a, x) + w(b, x) \\ &\leq w(O_2) + 5(r-1)w^*_a + w^*_a + w^*_a + 2w^*_a \\ &\leq w(OPT_{r-1}(T_2)) + 5rw^*_a \\ &\leq w(OPT_r(T_2)) + 5rw^*_a. \end{aligned} \tag{17}$$

Moreover, using similar steps as in the proof of the first case, we can alternatively write out $w(OPT_{r+2}(T))$ in terms of the set $O_1 := OPT_{r+2}(T) \setminus \{a, x\}$—this corresponds to the scenario where our algorithm removes the nodes $\{a, x\}$ from T but retains b.

$$\begin{aligned} w(OPT_{r+2}(T)) &\leq w(O_1) + w(a, O_1) + w(x, O_1) + w(a, x) \\ &\leq w(O_1) + rw^*_a + 2rw^*_a + w^*_a \\ &\leq w(OPT_r(T_1)) + (3r+1)w^*_a \end{aligned} \tag{18}$$

Recall that $T_1 = T \setminus \{a, x\}$. The second inequality comes from the fact that $w(a, O_1) = \sum_{i \in O_1} w(a, i) \leq \sum_{i \in O_1} w^*_a = rw^*_a$. By adding the inequalities in (17) and (18) and then dividing by two, we can show our final upper bound.

$$\begin{aligned} w(OPT_{r+2}(T)) &\leq \frac{1}{2}\left(w(OPT_r(T_1)) + w(OPT_r(T_2))\right) + \frac{3r+1}{2}w^*_a + \frac{5r}{2}w^*_a \\ &\leq \frac{1}{2}\left(w(OPT_r(T_1)) + w(OPT_r(T_2))\right) + (4r+2)w^*_a \end{aligned} \tag{19}$$

This completes the upper bounds for both cases. Despite the difference in the underlying techniques, we obtain the same upper bound irrespective of whether $\Delta \subseteq OPT_{r+2}(T)$ is true or not. For the rest of this proof, we can simply use Equation (19) (which happens to be the same as Equation (16)) as a universal upper bound on the optimum solution.

Coming back to our induction hypothesis, now that we are done with our upper bound, we prove a lower bound on the weight of the edges inside the set $Alg_{r+2}(T)$ for a fixed choice of $\Delta = \{a, x, b\}$. We abuse notation and use $E[w(Alg_{r+2}(T))]$ to refer to the expected weight of the solution returned by our algorithm given that Δ is the triplet selected in the first round—the expectation is over the nodes actually added to the solution (either a, x or b, x) as well as the random choices made by the algorithm in subsequent rounds.

Recall that $T_1 = T \setminus \{a, x\}$ and $T_2 = T \setminus \{a, b, x\}$. With probability one-half, our algorithm adds the nodes a, x to the final solution. In this case, the weight of the solution returned by our algorithm equals

$$E[w(Alg_r(T_1))] + w(a, Alg_r(T_1)) + w(x, Alg_r(T_1)) + w(a, x).$$

If the algorithm adds the nodes b, x to the final solution, then all three nodes in Δ are discarded from future consideration and therefore, the weight of the solution returned by the algorithm equals

$$E[w(Alg_r(T_2))] + w(b, Alg_r(T_2)) + w(x, Alg_r(T_2)) + w(b, x).$$

Since both of these events occur with equal probability, the actual solution returned by our algorithm for a fixed choice of Δ can be characterized as:

$$E[w(Alg_{r+2}(T))] = \frac{1}{2}\Big(E[w(Alg_r(T_1))] + E[w(Alg_r(T_2))] + w(a, Alg_r(T_1)) + w(x, Alg_r(T_1)) + w(b, Alg_r(T_2)) + w(x, Alg_r(T_2)) + w(a, x) + w(b, x)\Big). \tag{20}$$

We now carefully lower bound each of these terms using the third statement in Proposition 4.3. Specifically, by applying the proposition, we get that:

$$w(a, Alg_r(T_1)) + w(x, Alg_r(T_1)) \geq rw(a, x)$$
$$w(b, Alg_r(T_2)) + w(x, Alg_r(T_2)) \geq rw(b, x)$$

Moreover, we note that $rw(a, x) + rw(b, x) \geq rw(a, b)$ by means of the triangle inequality. So, finally, adding the two equations above leads to the following lower bound:

$$w(a, Alg_r(T_1)) + w(x, Alg_r(T_1)) + w(b, Alg_r(T_2)) + w(x, Alg_r(T_2)) \geq r \max(w(a, b) + w(a, x)) = rw_a^*.$$

Using this and the fact that $w(a, x) + w(b, x) \geq w_a^*$ we can simplify Equation (20):

$$\begin{aligned} E[w(Alg_{r+2}(T))] &\geq \frac{1}{2}\Big(E[w(Alg_r(T_1))] + E[w(Alg_r(T_2))] + rw_a^* + w_a^*\Big) \\ &\geq \frac{1}{2}\Big(E[w(Alg_r(T_1))] + E[w(Alg_r(T_2))] + (r+1)w_a^*.\Big) \end{aligned} \tag{21}$$

Armed with the upper bound in Equation (19) as well as the lower bound in Equation (21), we are now ready to complete the proof. Consider the upper bound in Equation 19 and recall that by the induction hypothesis $w(OPT_r(T_1)) \leq 8E[w(Alg_r(T_1))]$ and $w(OPT_r(T_2)) \leq 8E[w(Alg_r(T_2))]$. Therefore, we have that

$$\begin{aligned} w(OPT_{r+2}(T)) &\leq \frac{1}{2}\left(w(OPT_r(T_1)) + w(OPT_r(T_2))\right) + (4r+2)w_a^* \\ &\leq \frac{1}{2}\left(8E[w(Alg_r(T_1))] + 8E[w(Alg_r(T_2))]\right) + (4r+2)w_a^* \\ &\leq 8\left(\frac{1}{2}E[w(Alg_r(T_1))] + \frac{1}{2}E[w(Alg_r(T_2))] + \frac{(r+1)}{2}w_a^*\right) \\ &= 8E[w(Alg_{r+2}(T))] \quad \text{From Equation (21))} \end{aligned}$$

Taking the expectation over Δ, we prove the induction hypothesis. Consequently, the actual theorem follows from substituting $T = \mathcal{N}$ and $r = k$ in the induction hypothesis. □

4.3 Lower Bounds

Claim 4.4. *No ordinal approximation algorithm, deterministic or randomized, can provide an approximation factor better than 2 for Densest k-subgraph.*

Proof. Since randomized algorithms are more general than deterministic algorithms, it suffices to show the claim just for randomized algorithms.

Given a parameter k, consider an instance of Densest k-Subgraph with Mk nodes for a large enough value of M (say M is much larger than k). The set of nodes in the graph can be divided into M clusters $N_1, N_2, \ldots, N_M$, each containing k nodes. The preference ordering is given as follows: for a given i, every node in N_i prefers all the nodes in N_i over every node outside of N_i. The exact preference ordering within N_i and outside of N_i can be arbitrary.

Now, randomly choose one of M clusters and assign a weight of 2 to all the edges strictly inside that cluster. Assign a weight of 1 to every other edge in the graph. It is easy to see that these weights induce the given preference orderings. Now, without loss of generality, it suffices to consider only algorithms that choose k nodes within a fixed cluster. Moreover, since the clusters are identical from an ordinal point of view, the optimum ordinal algorithm for this instance just picks one of the M clusters uniformly at random, and therefore, its approximation ratio is $\frac{2}{1+\frac{1}{M}}$ which approaches 2 as $M \to \infty$ □

5 ORDINAL APPROXIMATION ALGORITHMS FOR GENERAL GRAPH MAXIMIZATION PROBLEMS

In the previous sections, we provided ordinal approximation algorithms for matching and densest subgraph. All of our algorithms were based on the simple techniques of greedy and random as well as serial dictatorship, which essentially combines the two approaches. Now, we further highlight the power of these paradigms for settings with ordinal information by showing that they lead to approximation algorithms with small constant factors for a number of graph maximization problems such as *k-sum Clustering, Max Spanning Tree, Max Traveling Salesman, and Max (Weighted) k-Vertex Cover.*

The common thread that connects all of these problems is that they involve selecting a subset of edges $\bar{E} \subseteq \mathcal{N} \times \mathcal{N}$ that satisfies some constraint in order to maximize the weight of the edges inside $\bar{E}$, i.e., $\sum_{(x,y)\in\bar{E}} w(x,y)$. Note that both maximum weighted matching and densest subgraph can be cast in this framework. However, while we present direct applications of greedy, random, and serial dictatorship algorithms in this section, our results in Section 3 and 4 are much stronger as we combined these approaches in a non-trivial fashion to obtain better approximation guarantees. We now formally define the problems studied in this section. We also remark that all of these problems have been studied previously in the optimization literature with respect to specific applications [22, 24, 26, 27, 33, 34].

- **k-sum Clustering**: Given an integer k, partition the nodes into k disjoint sets $(S_1, \ldots, S_k)$ of equal size in order to maximize $\sum_{i=1}^{k} \sum_{x,y\in S_i} w(x,y)$. (It is assumed that N is divisible by k). When $k = N/2$, k-sum clustering reduces to maximum weighted matching.

  This problem is NP-hard even when we have full access to the hidden graph weights and when the weights satisfy the triangle inequality [24].
- **Maximum Spanning Tree:** Select a tree (a set of edges containing no cycles) $\bar{E} \subseteq E$ such that for any two $x, y \in \mathcal{N}$, there is a path between x and y in $\bar{E}$. Our objective is to maximize the weight of the edges in $\bar{E}$, i.e., $\sum_{(x,y)\in\bar{E}} w(x,y)$.

  The problem can be solved efficiently when we have access to the edge weights as it is equivalent to the minimum spanning tree problem.
- **Maximum Traveling Salesman:** (Max TSP) In the maximum traveling salesman problem, the objective is to compute a tour T (cycle that visits each node in $\mathcal{N}$ exactly once) to maximize $\sum_{(x,y)\in T} w(x,y)$.

  This problem is known to be NP-hard in the full information case even when the weights satisfy the triangle inequality [34].
- **Maximum (Weighted) k-Vertex Cover:** Select a subset of k vertices S in order to maximize the total weight of edges incident to S in E, i.e., maximize $\sum_{\substack{(x,y)\in E \\ \{x,y\}\cap S\neq\emptyset}} w(x,y)$.

  This problem was first introduced in [22] and has been extensively studied amidst other coverage problems.

Our main theorem follows.

THEOREM 5.1. *We can compute in polynomial time ordinal approximation algorithms with the following guarantees:*

(1) *A universally truthful 2-approximation algorithm for k-sum Clustering based on the random algorithm.*
(2) *A universally truthful 2-approximation algorithm for Maximum Traveling Salesman based on random serial dictatorship.*
(3) *A greedy 2-approximation algorithm for Maximum Spanning Tree.*
(4) *A greedy 4-approximation algorithm for Maximum (Weighted) k-Vertex Cover.*

We now describe the algorithms for the four problems.

(1) **k-sum Clustering:** Initialize $T = \mathcal{N}$. For $i = 1$ to k, pick a set of $\frac{N}{k}$ nodes S_i uniformly at random from T and update $T = T \setminus S_i$. Output $(S_1, S_2, \ldots, S_k)$.
(2) **Maximum Traveling Salesman:** Initialize $\bar{E}$ to be a random edge $e_0 = (x_0, y_0)$ from E, $S = \mathcal{N} \setminus \{x_0, y_0\}$. While $S \neq \emptyset$, pick one of the end-points of $\bar{E}$, say x. Let y denote x's most preferred agent in S. Add (x, y) to $\bar{E}$ and remove y from S. Complete $\bar{E}$ to form a Hamiltonian cycle, and return $\bar{E}$.
(3) **Maximum Spanning Tree:** Initialize $\bar{E} = \emptyset$, $T = E$. While $T \neq \emptyset$: pick an undominated edge from T, add it to $\bar{E}$. Remove all edges from T whose addition to $\bar{E}$ would induce a cycle. Return $\bar{E}$.
(4) **Maximum (Weighted) k-Vertex Cover:** Initialize $S = \emptyset$, $T = E$. While $|S| < k$: pick an undominated edge (x^*, y^*) from T, set $S = S \cup \{x^*, y^*\}$. Remove all edges containing x^* or y^* from T. Return S.

PROOF. **Proof of 2-approximation for k-sum Clustering**

The algorithm is clearly truthful since it is completely oblivious to the input. The analysis of the algorithm involves non-trivial lower bounds on the performance of our random solution, and upper bounds on the optimum solution analogous to Lemmas 2.7 and 2.8 respectively. Suppose that $\gamma = \frac{N}{k}$.

LEMMA 5.2. *(Lower Bound) The expected value of the objective function for the clustering returned by our algorithm $(S_1, \ldots, S_k)$ is given by*

$$E[\sum_{i=1}^{k} \sum_{x,y \in S_i} w(x,y)] = \frac{\gamma - 1}{N - 1} \sum_{(x,y) \in \mathcal{N} \times \mathcal{N}} w(x,y).$$

PROOF. As with Lemma 2.7, we proceed via a symmetry argument although it is not hard to verify that the same bound can be obtained via a more exhaustive counting argument. First, by linearity of expectation, we have that the value of the objective (in expectation) is $\sum_{(x,y) \in \mathcal{N} \times \mathcal{N}} w(x,y) Pr(\exists i \textbf{ s.t. } x, y \in S_i)$ where the second term is the probability that x and y belong to the same cluster in S. Using a symmetry argument (since our process chooses nodes for each cluster uniformly at random), we argue that the probability $Pr(x, y \in S_i)$ is the same for every $x, y \in \mathcal{N}$.

Now, fix any arbitrary node $x \in \mathcal{N}$: since there $\gamma - 1$ other nodes in the same cluster as x, this means that $\sum_{y \neq x} Pr(x, y \in S_i) = \gamma - 1$. Therefore, for every (x, y), $Pr(x, y \in S_i) = \frac{\gamma - 1}{N - 1}$. Substituting this in the expected value of the objective function gives us the desired result. □

LEMMA 5.3. *(Upper Bound) Suppose that $O = (O_1, \ldots, O_k)$ is the optimum solution for a given instance of the k-sum Clustering problem. Then, we have the following upper bound on the value of*

the optimum solution

$$OPT = \sum_{i=1}^{k} \sum_{x,y \in O_i} w(x,y) \leq \frac{2(\gamma-1)}{N-1} \sum_{(x,y) \in \mathcal{N} \times \mathcal{N}} w(x,y).$$

Proof. Suppose that x and y are two nodes belonging to the same cluster in O (say cluster O_i). Then, by the triangle inequality, we have that for every $z \in \mathcal{N}$ (including x and y), $w(x,z)+w(y,z) \geq w(x,y)$. Summing this up over all $z \in \mathcal{N}$, we have $\sum_{z \in \mathcal{N}}(w(x,z)+w(y,z)) \geq Nw(x,y)$. Repeating this process over all $(x,y) \in O_i$ and $z \in \mathcal{N}$, we get

$$\sum_{i=1}^{k} \sum_{x,y \in O_i} \sum_{z \in \mathcal{N}} (w(x,z)+w(y,z)) \geq N \sum_{i=1}^{k} \sum_{x,y \in O_i} w(x,y)$$
$$= NOPT.$$

Now, given some edge (x,z), how many times does $w(x,z)$ appear in the LHS? Without loss of generality, suppose that $x \in O_i$ and $z \in O_j$. Then, x has $\gamma-1$ edges inside O_i and $w(x,z)$ appears once in the LHS for each of these neighboring edges. Similarly, z has $\gamma-1$ edges inside O_j and $w(x,z)$ appears once in the LHS for each edge. Therefore, for every $x,z \in \mathcal{N}$, $w(x,z)$ appears $2(\gamma-1)$ times in the LHS of the above equation. Substituting this, we prove the lemma,

$$\sum_{x,y \in \mathcal{N}} 2(\gamma-1)w(x,y) \geq NOPT.$$

□

The rest of the theorem follows immediately from the two lemmas.

Proof of 2-approximation for Max TSP

It is easy to see that this algorithm is truthful: when an agent i is asked for its preferences, the first edge of T incident to agent i has already been decided, so i cannot affect it. Thus, to form the second edge of T incident to i, it may as well specify its most-preferred edge. Suppose that T^* denotes the optimal tour and $T = \bar{E}$ denotes the tour returned by our algorithm.

The proof proceeds via a straightforward argument where we charge edges in T^*, the welfare maximizing tour, to those in T, the solution returned by our algorithm. We first introduce some notation beginning with a simple tie-breaking rule that allows for convenient analysis. Specifically, suppose that (a,b) denotes the first (random) edge added to T. Then, pick one of a or b (say a) uniformly at random, and term this node as the 'dead node'. For the rest of algorithm, a does not get to select another edge and remains as an end-point of T. The second edge containing a is added only when the tour is completed to form a cycle. We remark that the randomization in the first step is *essential:* if we had selected the first edge based on the input preferences, then the first node could improve its utility by lying, and the algorithm would no longer be strategy-proof.

Next, for any $i \in \mathcal{N}$, we will use $t_1^*(i)$ and $t_2^*(i)$ to denote the two nodes that i is connected to in T^*, and $t_1(i)$, $t_2(i)$ to the nodes connected to i in T. Finally, suppose that e_r denotes the random first edge selected by the algorithm and i_d, the (random) dead node. In this proof, we show that for any realization of e_r, i_d, the optimum tour is at most twice the tour returned by our algorithm. Therefore, the same approximation bound also holds in expectation.

Fix some instantiation of e_r, i_d, call it $\tilde{e}_r, \tilde{i}_d$. Our charging argument comprises of two phases: in the first phase, we charge to the edges in T all of the edges in T^* except the ones containing the dead node $\tilde{i}_d$. While doing so, we ensure that for each edge in T, at most two edges in T^* are charged to this edge. In the final phase, we carefully charge the edges in T^* containing $\tilde{i}_d$ to certain edges in T that were charged at most once in the first phase.

First Phase Charging. Suppose that we use S_i to denote the set of available nodes at the instant in our algorithm (for this particular instantiation of e_r, i_d) when an edge containing i is added to T. The algorithm then proceeds to pick i's most preferred agent in S_i and adds the corresponding edge to T. Suppose that for every $i \in \mathcal{N}$, $t_2(i)$ denotes its most preferred node in S_i.

Now consider any edge (x^*, y^*) in T^* such that $x^*, y^* \neq \tilde{i}_d$. Suppose that x^* was removed from the set of available nodes before y^* during the course of the algorithm. Then, $y^* \in S_{x^*}$ and so, $w(x^*, t_2(x^*)) \geq w(x^*, y^*)$ and we can charge the edge $(x^*, y^*) \in T^*$ to $(x^*, t_2(x^*)) \in T$.

After repeating this charging argument for every edge in T^* except $(\tilde{i}_d, t_1^*(\tilde{i}_d))$, $(\tilde{i}_d, t_2^*(\tilde{i}_d))$, we end up with the following proposition.

Proposition 5.4. *The following are true at the end of the first phase of charging.*

(1) *At most two edges in T^* are charged to any one edge in T.*
(2) *No edges are charged to $(\tilde{i}_d, t_1(\tilde{i}_d))$, $(\tilde{i}_d, t_2(\tilde{i}_d)) \in T$.*
(3) *At most one edge in T^* is charged to any of the edges in T containing $t_1^*(\tilde{i}_d)$, $t_2^*(\tilde{i}_d)$.*

Proof. **(Statement 1)** Consider any edge of the form $(i, t_2(i))$, as per our definitions, i became unavailable before $t_2(i)$. Thus, the only edges charged to $(i, t_2(i))$ are those in T^* containing i, and there can only be two such edges.
Statement 2 Further, suppose that $(\tilde{i}_d, t_1(\tilde{i}_d))$ denotes the random edge in T. Clearly, edges in T^* containing $t_1(\tilde{i}_d)$ are not charged to the random edge. Finally, no edge in T^* is charged to $(\tilde{i}_d, t_2(\tilde{i}_d))$, since the latter node is the absolute last node to become unavailable.
Statement 3 This is a direct consequence of the fact that we have not charged $(\tilde{i}_d, t_1^*(\tilde{i}_d))$, $(\tilde{i}_d, t_2^*(\tilde{i}_d))$ to any edge in T. □

Second Phase Charging. We use the triangle inequality to charge the edge $(\tilde{i}_d, t_1^*(\tilde{i}_d))$:

$$w(\tilde{i}_d, t_1^*(\tilde{i}_d)) \leq w(\tilde{i}_d, t_2(\tilde{i}_d)) + w(t_2(\tilde{i}_d), t_1^*(\tilde{i}_d)) \leq w(\tilde{i}_d, t_2(\tilde{i}_d)) + w(t_1^*(\tilde{i}_d), t_2(t_1^*(\tilde{i}_d))).$$

The final inequality is due to the fact that $t_2(\tilde{i}_d) \in S_{t_1^*(\tilde{i}_d)}$ (since $t_2(\tilde{i}_d)$ is the absolute last node added to the tour, and so it is available during the entire runtime of the algorithm), and so $w(t_2(\tilde{i}_d), t_1^*(\tilde{i}_d)) \leq w(t_1^*(\tilde{i}_d), t_2(t_1^*(\tilde{i}_d)))$. Therefore, the edge $(\tilde{i}_d, t_1^*(\tilde{i}_d))$ can be charged to two edges in T, namely $(\tilde{i}_d, t_2(\tilde{i}_d))$ and $(t_1^*(\tilde{i}_d), t_2(t_1^*(\tilde{i}_d))$

Using exactly the same kind of argument, we can also charge the second edge containing $\tilde{i}_d$ in T^* to two edges in T, namely $(\tilde{i}_d, t_2(\tilde{i}_d))$ and $w(t_2^*(\tilde{i}_d), t_2(t_2^*(\tilde{i}_d))$. This concludes the second phase of charging.

In conjunction with Proposition 5.4, we have successfully charged every edge in OPT by using at most two edges in T. This completes our two approximation.

Proof of 2-approximation for Maximum Spanning Tree

We suppose that the algorithm proceeds in rounds $\{1, 2, \ldots, N-1\}$: the spanning tree contains N nodes and exactly $N-1$ edges. Suppose that $\bar{E}_k$ and T_k denote the set of edges constituting the partial solution and candidate pool at the beginning of round k. Let (x_k^*, y_k^*) be the undominated edge from T_k selected in round k. Finally, we use OPT to denote the optimum spanning tree and OPT_k to refer to the optimum spanning tree that contains the edges in S_k. Note that $OPT_1 = OPT$ and $OPT_N = \bar{E}$.

Our proof will proceed as follows: for every k, we prove that $w(OPT_k) - w(OPT_{k+1}) \leq w(\bar{E}_{k+1}) - w(\bar{E}_k)$.This simple claim suffices to prove the theorem as illustrated below.

$$\begin{aligned}
&\sum_{i=1}^{N-1} w(OPT_i) - w(OPT_{i+1}) \leq \sum_{i=1}^{N-1} w(\bar{E}_{k+1}) - w(\bar{E}_k) \\
\implies\ & w(OPT_1) - w(OPT_N) \leq w(\bar{E}_N) - w(\bar{E}_1) \\
\implies\ & w(OPT) - w(\bar{E}) \leq w(\bar{E}) \\
\implies\ & w(OPT) \leq 2w(\bar{E}).
\end{aligned}$$

Note that by definition $\bar{E}_1 = \emptyset$ and so its weight equals zero. It remains for us to prove the central claim. Fix some iteration k, and consider the spanning tree OPT_k. Clearly, $\bar{E}_k$ and $\bar{E}_{k+1}$ differ only in the edge (x_k^*, y_k^*). Since OPT_k is a spanning tree, there must be a path between every two nodes in $\mathcal{N}$. Let (x, y) denote an edge that lies on the simple path between x_k^* and y_k^* in OPT_k and also belongs to T_k. Such an edge must necessarily exist because:

(1) No edge in $E \setminus (T_k \cup \bar{E}_k)$ can be present in OPT_k as the addition of any such edge would induce a cycle, by definition of T_k.
(2) $OPT_k \setminus \bar{E}_k$ must therefore only contain edges from T_k.
(3) The nodes x_k^* and y_k^* are not connected in $\bar{E}_k$ and so, the path between these nodes in OPT_k must contain at least one edge from T_k.

Next, define $O'_k = (OPT_k \setminus \{(x, y)\}) \cup \{(x_k^*, y_k^*)\}$. It is not hard to deduce that O'_k is also a spanning tree, but perhaps most importantly, it is a spanning tree that contains all of the edges in $\bar{E}_{k+1}$. By definition, we know that OPT_{k+1} is the maximum weight spanning tree that contains $\bar{E}_{k+1}$. Therefore, we infer that $w(O'_k) \leq w(OPT_{k+1})$. Expanding upon this, we get:

$$w(OPT_{k+1}) \geq w(O'_k) = w(OPT_k) + w(x_k^*, y_k^*) - w(x, y). \tag{22}$$

Therefore, we have that

$$\begin{aligned}
w(OPT_k) - w(OPT_{k+1}) &\leq w(x, y) - w(x_k^*, y_k^*) \\
&\leq 2w(x_k^*, y_k^*) - w(x_k^*, y_k^*) \qquad \textbf{(From Lemma 2.5)} \\
&= w(\bar{E}_{k+1}) - w(\bar{E}_k).
\end{aligned}$$

This completes the proof that our algorithm produces a 2-approximation to the optimum maximum spanning tree.

Proof of 4-Approximation for Maximum k-Vertex Cover

This proof uses a similar approach to that of the 2-approximation algorithm for max spanning tree, wherein we bound the difference in 'successive optimal solutions' in terms of the solution returned by our algorithm. Formally, we define S_ℓ to be the set containing the first ℓ nodes chosen by our algorithm, N_ℓ denotes the nodes in $\mathcal{N} \setminus S_\ell$, T_ℓ is the edge set representing the complete graph on N_ℓ, and finally, OPT_ℓ denotes the optimum solution to the k-vertex cover problem that contains S_ℓ. Clearly, OPT_0 is the optimum solution OPT and $OPT_k = S_k$. For any given set $B \subseteq \mathcal{N}$, we define the value of the objective function for this solution to be:

$$w^{VC}(B) = \sum_{x,y \in B} w(x, y) + \sum_{x \in B, y \in \mathcal{N} \setminus B} w(x, y).$$

Consider the sets S_ℓ and $S_{\ell+2}$ for some $0 \leq \ell \leq k-2$. Suppose that $S_{\ell+2} = S_\ell \cup \{x_\ell, y_\ell\}$, where (x_ℓ, y_ℓ) is an undominated edge in T_ℓ. Recall that $N_{\ell+2} = N_\ell \setminus \{x_\ell, y_\ell\}$. Therefore, we have that

$$\begin{aligned} w^{VC}(S_{\ell+2}) &= w^{VC}(S_\ell) + w(x_\ell, y_\ell) + \sum_{z \in N_{\ell+2}} (w(x_\ell, z) + w(y_\ell, z)) \\ &\geq w^{VC}(S_\ell) + w(x_\ell, y_\ell) + \sum_{z \in N_{\ell+2}} w(x_\ell, y_\ell) \\ &= w^{VC}(S_\ell) + (N - \ell - 1) w(x_\ell, y_\ell). \end{aligned}$$

Next, we need to provide an upper bound for the difference in the quality of the solutions $OPT_{\ell+2}$ and OPT_ℓ. Let us demarcate two nodes $a, b \in OPT_\ell \setminus S_\ell$ as follows:

(1) If $x_\ell \in OPT_\ell$, then set $a = x_\ell$, else set a to be some arbitrary node in $OPT_\ell \setminus S_\ell$ that is not y_ℓ.
(2) If $y_\ell \in OPT_\ell$, then set $b = y_\ell$, else set b to be some arbitrary node in $OPT_\ell \setminus S_\ell$ that is not a.

Define $N' = \mathcal{N} \setminus (OPT_\ell \cup \{x_\ell, y_\ell\})$ and $O' = OPT_\ell \setminus \{a, b\} \cup \{x_\ell, y_\ell\}$ and $a, b \in N_\ell$. Observe that by definition, $N' \subseteq N_{\ell+2}$. Clearly, O' is a candidate for the maximum (weighted) k-vertex cover that also contains the set $S_{\ell+2}$. So, we have that $w^{VC}(O') \leq w^{VC}(OPT_{\ell+2})$. Expanding this, we get that:

$$\begin{aligned} w^{VC}(OPT_{\ell+2}) &\geq w^{VC}(O') \\ &\geq w^{VC}(OPT_\ell) - w(a, b) - \sum_{z \in N'} (w(a, z) + w(b, z)) + w(x_\ell, y_\ell) + \sum_{z \in N'} (w(x_\ell, z) + w(y_\ell, z)) \end{aligned} \tag{23}$$

Since $a, b \in N_\ell$ and (x_ℓ, y_ℓ) is an undominated edge in T_ℓ, we can apply Lemma 2.5 to get that $w(a, b) \leq 2w(x_\ell, y_\ell)$. In fact for all $z \in N'$, we can apply this lemma to get $w(a, z) \leq 2w(x_\ell, y_\ell)$ and $w(b, z) \leq 2w(x_\ell, y_\ell)$. Continuing from Equation 23,

$$\begin{aligned} w^{VC}(OPT_{\ell+2}) &\geq w^{VC}(OPT_\ell) - w(a, b) - \sum_{z \in N'} (w(a, z) + w(b, z)) + w(x_\ell, y_\ell) + \sum_{z \in N'} (w(x_\ell, z) + w(y_\ell, z)) \\ &\geq w^{VC}(OPT_\ell) - 2w(x_\ell, y_\ell) - \sum_{z \in N'} (2w(x_\ell, y_\ell) + 2w(x_\ell, y_\ell)) + w(x_\ell, y_\ell)) - \sum_{z \in N'} w(x_\ell, y_\ell) \\ &\geq w^{VC}(OPT_\ell) - (4|N'| + 2) w(x_\ell, y_\ell) + (|N'| + 1) w(x_\ell, y_\ell) \\ &= w^{VC}(OPT_\ell) - (3|N'| + 1) w(x_\ell, y_\ell) \\ &\geq w^{VC}(OPT_\ell) - (3(N - k) + 1) w(x_\ell, y_\ell) \\ &\geq w^{VC}(OPT_\ell) - (3(N - \ell - 2) + 1) w(x_\ell, y_\ell) \qquad (24) \\ &\geq w^{VC}(OPT_\ell) - 3(N - \ell - 1) w(x_\ell, y_\ell) \\ &\geq w^{VC}(OPT_\ell) - 3(w^{VC}(S_{\ell+2}) - w^{VC}(S_\ell)). \end{aligned}$$

Equation 24 comes from that (by definition) $\ell \leq k-2$ and $|N'| \leq N - k$. Therefore $|N'| \leq N - k \leq N - \ell - 2$. In conclusion, we have that

$$w^{VC}(OPT_\ell) - w^{VC}(OPT_{\ell+2}) \leq 3(w^{VC}(S_{\ell+2}) - w^{VC}(S_\ell)).$$

The approximation bound of four follows from a direct telescopic summation of the above inequality from $\ell = 0$ to $\ell = k-2$ and using the fact that $OPT_0 = OPT$ and $OPT_k = S_k = S$. □

6 CONCLUSION

In this paper we study ordinal algorithms, i.e., algorithms which are aware only of preference orderings instead of the hidden weights or utilities which generate such orderings. Perhaps surprisingly, our results indicate that for many problems including Matching, Densest Subgraph, and Traveling Salesman, ordinal algorithms perform almost as well as the best algorithms which know the underlying metric weights. This indicates that for settings involving agents where it is expensive, or impossible, to obtain the true numerical weights or utilities, one can use ordinal mechanisms without much loss in welfare.

While many of our algorithms are randomized, and the quality guarantees are "in expectation", similar techniques can be used to obtain weaker bounds for deterministic algorithms (bounds of 2 for Matching and TSP, and of 4 for the other problems considered). It may also be possible to improve the deterministic approximation factor for matching to be better than 2: although this seems to be a difficult problem which would require novel techniques, such an algorithm would immediately provide new deterministic algorithms for the other problems using black-box reductions similar to one used in the proof of Theorem 4.1. Finally, our bound of 2 for TSP holds even in the absence of the triangle inequality; it would be very interesting to see how well ordinal algorithms perform if the weights obeyed some structure other than the metric one.

REFERENCES

[1] David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn. 2007. Popular Matchings. *SIAM J. Comput.* 37, 4 (2007), 1030–1045.

[2] Ben Abramowitz and Elliot Anshelevich. 2018. Utilitarians Without Utilities: Maximizing Social Welfare for Graph Problems using only Ordinal Preferences. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)* (2018).

[3] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)* 55, 5 (2008), 23.

[4] Georgios Amanatidis, Georgios Birmpas, and Evangelos Markakis. On Truthful Mechanisms for Maximin Share Allocations. In *Proceedings of IJCAI 2016.*

[5] Elliot Anshelevich, Onkar Bhardwaj, and John Postl. 2015. Approximating Optimal Social Choice under Metric Preferences. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.* 777–783.

[6] Elliot Anshelevich and John Postl. Randomized Social Choice Functions Under Metric Preferences. In *Proceedings of IJCAI 2016.*

[7] Elliot Anshelevich and Wennan Zhu. 2017. Tradeoffs Between Information and Ordinal Approximation for Bipartite Matching. In *Proceedings of the 10th International Symposium on Algorithmic Game Theory SAGT 2017.* 267–279.

[8] Anand Bhalgat, Deeparnab Chakrabarty, and Sanjeev Khanna. 2011. Social Welfare in One-Sided Matching Markets without Money. In *Proceedings of the Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2011).* 87–98.

[9] Sayan Bhattacharya, Sreenivas Gollapudi, and Kamesh Munagala. 2011. Consideration set generation in commerce search. In *Proceedings of the 20th International Conference on World Wide Web, (WWW '11).* 317–326.

[10] Benjamin Birnbaum and Kenneth J Goldman. 2009. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica* 55, 1 (2009), 42–59.

[11] Craig Boutilier, Ioannis Caragiannis, Simi Haber, Tyler Lu, Ariel D. Procaccia, and Or Sheffet. 2015. Optimal social choice functions: A utilitarian view. *Artif. Intell.* 227 (2015), 190–213.

[12] Ioannis Caragiannis, Aris Filos-Ratsikas, Soren Kristoffer Stiil Frederiksen, Kristoffer Arnsfelt Hansen, and Zihan Tan. Truthful Facility Assignment with Resource Augmentation: An Exact Analysis of Serial Dictatorship. In *Proceedings of WINE 2016.*

[13] Ioannis Caragiannis, Swaprava Nath, Ariel D. Procaccia, and Nisarg Shah. Subset Selection Via Implicit Utilitarian Voting. In *Proceedings of IJCAI 2016.*

[14] Ioannis Caragiannis and Ariel D. Procaccia. 2011. Voting almost maximizes social welfare despite limited communication. *Artif. Intell.* 175, 9-10 (2011), 1655–1671.

[15] Deeparnab Chakrabarty and Chaitanya Swamy. 2014. Welfare maximization and truthfulness in mechanism design with ordinal preferences. In *Proceedings of the 5th Innovations in Theoretical Computer Science conference (ITCS 2014).* 105–120.

[16] James A Davis. 1977. Clustering and structural balance in graphs. *Social networks. A developing paradigm* (1977), 27–34.

[17] Shahar Dobzinski, Noam Nisan, and Michael Schapira. 2012. Truthful randomized mechanisms for combinatorial auctions. *J. Comput. Syst. Sci.* 78, 1 (2012), 15–25. https://doi.org/10.1016/j.jcss.2011.02.010

[18] Doratha E. Drake and Stefan Hougardy. 2003. Improved Linear Time Approximation Algorithms for Weighted Matchings. In *Proceedings of APPROX 2003*. 14–23.

[19] Ran Duan and Seth Pettie. 2010. Approximating Maximum Weight Matching in Near-Linear Time. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, (FOCS 2010)*. 673–682.

[20] Shaddin Dughmi and Arpita Ghosh. 2010. Truthful assignment without money. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC 2010),*. 325–334.

[21] Jack Edmonds. 1965. Paths, trees, and flowers. *Canadian Journal of mathematics* 17, 3 (1965), 449–467.

[22] Uriel Feige and Michael Langberg. 2001. Approximation Algorithms for Maximization Problems Arising in Graph Partitioning. *J. Algorithms* 41, 2 (2001), 174–211.

[23] Michal Feldman, Amos Fiat, and Iddan Golomb. On Voting and Facility Location. In *Proceedings of ACM EC 2016*.

[24] Thomas A Feo and Mallek Khellaf. 1990. A class of bounded approximation algorithms for graph partitioning. *Networks* 20, 2 (1990), 181–195.

[25] Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, and Jie Zhang. 2014. Social Welfare in One-Sided Matchings: Random Priority and Beyond. In *Proceedings of the 7th International Symposium on Algorithmic Game Theory (SAGT 2014)*. 1–12.

[26] Giulia Galbiati, Angelo Morzenti, and Francesco Maffioli. 1997. On the Approximability of Some Maximum Spanning Tree Problems. *Theor. Comput. Sci.* 181, 1 (1997), 107–118.

[27] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. 2012. Inferring Networks of Diffusion and Influence. *TKDD* 5, 4 (2012), 21:1–21:37.

[28] Steven M Goodreau, James A Kitts, and Martina Morris. 2009. Birds of a feather, or friend of a friend? Using exponential random graph models to investigate adolescent social networks. *Demography* 46, 1 (2009), 103–125.

[29] Dan Gusfield and Robert W Irving. 1989. *The stable marriage problem: structure and algorithms*. MIT press.

[30] John C Harsanyi. 1976. *Cardinal welfare, individualistic ethics, and interpersonal comparisons of utility*. Springer.

[31] Refael Hassin, Shlomi Rubinstein, and Arie Tamir. 1997. Approximation algorithms for maximum dispersion. *Operations research letters* 21, 3 (1997), 133–137.

[32] Bala Kalyanasundaram and Kirk Pruhs. 1993. Online Weighted Matching. *J. Algorithms* 14, 3 (1993), 478–488.

[33] S. Rao Kosaraju, James K. Park, and Clifford Stein. 1994. Long Tours and Short Superstrings (Preliminary Version). In *Proceedings of the35th Annual Symposium on Foundations of Computer Science (FOCS 1994)*. 166–177.

[34] Łukasz Kowalik and Marcin Mucha. 2009. Deterministic 7/8-approximation for the metric maximum TSP. *Theoretical Computer Science* 410, 47 (2009), 5000–5009.

[35] Piotr Krysta, David Manlove, Baharak Rastegari, and Jinshan Zhang. 2014. Size versus truthfulness in the House Allocation problem. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC)*. 453ÂŰ–470.

[36] Wei-Ping Liu, Jeffrey B Sidney, and André Van Vliet. 1996. Ordinal algorithms for parallel machine scheduling. *Operations Research Letters* 18, 5 (1996), 223–232.

[37] Noam Nisan and Amir Ronen. 2001. Algorithmic Mechanism Design. *Games and Economic Behavior* 35, 1-2 (2001), 166–196. https://doi.org/10.1006/game.1999.0790

[38] Sewoong Oh and Devavrat Shah. 2014. Learning Mixed Multinomial Logit Model from Ordinal Data. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS 2014)*. 595–603.

[39] Ariel D. Procaccia and Jeffrey S. Rosenschein. 2006. The Distortion of Cardinal Preferences in Voting. In *Proceedings of 10th Intl. Workshop on Cooperative Information Agents CIA 2006*. 317–331.

[40] Ariel D. Procaccia and Moshe Tennenholtz. 2013. Approximate Mechanism Design without Money. *ACM Trans. Economics and Comput.* 1, 4 (2013), 18. https://doi.org/10.1145/2542174.2542175

[41] Sekharipuram S Ravi, Daniel J Rosenkrantz, and Giri Kumar Tayi. 1994. Heuristic and special case algorithms for dispersion problems. *Operations Research* 42, 2 (1994), 299–310.

[42] Alvin E Roth and Marilda Sotomayor. 1992. Two-sided matching. *Handbook of game theory with economic applications* 1 (1992), 485–541.

[43] Hossein Azari Soufiani, David C. Parkes, and Lirong Xia. 2012. Random Utility Theory for Social Choice. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS 2012)*. 126–134.

[44] Juan Ramón Troncoso-Pastoriza, Stefan Katzenbeisser, and Mehmet Utku Celik. Privacy preserving error resilient dna searching through oblivious automata. In *Proceedings of ACM CCS 2007*.

A FRIENDSHIP NETWORKS

The classic theory of Structural Balance [16] argues that agents embedded in a social network must exhibit the property that 'a friend of a friend is a friend'. This phenomenon has also been observed in many real-life networks (see for example [28]). Mathematically, if we have an unweighted social network $G = (V, E)$ of (say) friendships, it is easy to check if this property holds. If for any $(i, j) \in E$, (j, k) is also an edge, then (i, k) must also belong to E. For this reason, this property has also been referred to as transitive or triadic closure.

What about weighted networks that capture the 'intensity of friendships'? There is no obvious way as to how this property can be extended to weighted graphs without placing heavy constraints on the weights. For example, if we impose that for every edge (i, j), and every node k, $w(i, k) \geq \min(w(i, j), w(j, k))$, we immediately condemn all triangles to be isosceles (w.r.t the weights). Instead, we argue that a more reasonable mathematical property that extends triadic closure to Weighted graphs is the following

Definition A.1. (α-Weighted Friendship Property) Given a social network $G = (V, E, W)$, and a fixed parameter $\alpha \in [0, \frac{1}{2}]$, for every (i, j, k): $w(i, k) \geq \alpha[w(i, j) + w(j, k)]$.

In a nutshell, this property captures the idea that if (i, j) is a 'good edge', and (j, k) is a good edge, then so is (i, k). The parameter α gives us some flexibility on how stringently we can impose the property. Notice that in a sense, this property appears to be the opposite of the metric inequality, here we require that $w(i, k)$ is not too small compared to $w(i, j) + w(j, k)$. However, we show that this is not the case; in fact for every $\alpha \geq \frac{1}{3}$, any weighted graph that satisfies the α-Weighted Friendship Property must also satisfy the metric inequality.

Claim A.2. *Suppose that $G = (V, E, W)$ is a weighted complete graph that satisfies the α-Weighted Friendship Property for some $\alpha \in [\frac{1}{3}, \frac{1}{2}]$. Then, for every (i, j, k), we have $w(i, j) \leq w(i, k) + w(j, k)$.*

Proof. Without loss of generality, it suffices to show the proof for the case where $w(i, j)$ is the heaviest edge in the triangle (i, j, k). Now consider $w(i, k)$,$w(j, k)$ and without loss of generality, suppose that $w(i, k) \geq w(j, k)$. Then, since the α-friendship property is obeyed, we have

$$w(j, k) \geq \alpha[w(i, j) + w(i, k)] \geq \alpha w(i, j) + \alpha w(j, k).$$

This gives us that $w(i, j) \leq \frac{1-\alpha}{\alpha} w(j, k)$. It is easy to verify that for $\alpha \in [\frac{1}{3}, \frac{1}{2}]$, the quantity $\frac{1-\alpha}{\alpha} \leq 2$. Therefore, we get

$$w(i, j) \leq 2w(j, k) \leq w(i, k) + w(j, k).$$

□

B ODD NUMBER OF AGENTS: EXTENSIONS

In many of our algorithms, we assumed that N (the number of agents) is even for the sake of convenience and in order to capture our main ideas concisely without worrying about the boundary cases. Here, we show that all of our algorithms can be extended to the case where N is odd or not divisible by 3 with only minor modifications to the proofs and bounds obtained.

B.0.1 Matching. We begin by arguing that of all our algorithms and proofs for matching hold even when N is odd. First of all, it is not hard to see that our framework does not really depend on the parity of N and the lemmas on the greedy and random techniques carry over to the case when N is not even. In particular, note that in Lemma 2.7, we had that $E[w(M_R)] \geq \frac{|M_R|}{|E|} \sum_{x,y \in N} w(x, y)$, where E is the total number of edges in the complete graph. When N is odd, $|M_R| = \frac{N-1}{2}$, and $|E|$ is still $\frac{N(N-1)}{2}$. Therefore, we still get that $E(w(M_R)] \geq \frac{1}{N} \sum_{x,y \in N} w(x, y)$.

Next, we argue that our main 1.6-algorithm still holds for arbitrary N (not divisible by two and/or three) with an ϵ multiplicative error term that vanishes as $N \to \infty$. In Algorithm 4, when N is not divisible by three, we may need to choose a matching with $\lceil \frac{N}{3} \rceil$ edges for M_0. Let B be the largest matching outside of M_0, then $|B| = \lfloor \frac{N}{6} \rfloor$. Then, the second sub-routine in Algorithm 4 proceeds by selecting $|B|$ edges from M_0, and matching those nodes arbitrarily to the nodes in B. Ideally, we would like $\lceil \frac{N}{3} \rceil = |M_0| = 2|B| = 2\lfloor \frac{N}{6} \rfloor$. The worst case multiplicative error happens when $\lceil \frac{N}{3} \rceil$ is much larger than $2\lfloor \frac{N}{6} \rfloor$; this happens when N is odd, and has the form $3p + 2$ for some positive integer p. With some basic algebra, we can show that the multiplicative error is at most $\frac{7}{8(2N-3)}$, which approaches zero as N increases.

B.0.2 Max k-Sum. In the case of the black-box theorem, the 2α reduction still holds if we modify the algorithm as follows when $\frac{N}{k}$ is odd. Instead of selecting the optimum perfect matching, we need to select a matching of size $\frac{1}{2}(\frac{N}{k} - 1) * k = \frac{1}{2}(N - k)$, assign every pair of matched nodes to the same cluster, and the unmatched nodes arbitrarily. The rest of the proof is the same. Now when we apply this black-box result, we can no longer invoke the 1.6-approximation algorithm for perfect matchings and only use the 2-approximation greedy algorithm for a matching that selects $\frac{N-k}{2}$ edges, and therefore, the black-box result only yields a 4-approximation when $\frac{N}{k}$ is odd, but our main algorithm gives a 2-approximation algorithm irrespective of its parity.

Densest k-Subgraph. All of the proofs for the Densest k-subgraph hold. When k is odd, we simply select a matching with $\lfloor \frac{k}{2} \rfloor$ edges instead of $\frac{k}{2}$. The rest of the proof is exactly the same.

Max TSP. During the proofs for Max TSP, we extensively make use of the fact that $w(M^*) \geq \frac{w(T^*)}{2}$, where M^* is the optimum matching, and T^* is the optimal tour. This may no longer be true when N is odd. However, suppose that M_f^* is the optimum fractional matching, it is not hard to verify that $w(M_f^*) \geq \frac{w(T^*)}{2}$; this follows from taking T^* and choosing each edge with probability $\frac{1}{2}$. Moreover, observe that all of our proofs in this paper for the greedy algorithm (namely Lemma 3.2) are true when we compare the solution to the optimum fractional matching of a given size. Therefore, the black-box result itself carries over.

C PROOF OF THEOREM 2.11

Proof. **Notation:** Given any set of nodes S, we use $\bar{G}(S)$ to denote the directed first preference graph on S defined as follows: for every $i \in S$, there is a directed edge from i to its most preferred agent in $S - \{i\}$. Our algorithm could be viewed as selecting one edge from $\bar{G}(T)$ uniformly at random in each iteration, where T denotes the set of available agents.

For any set S, define $S^{-1} \subset S$ to be the random set of nodes remaining in S after removing one edge uniformly at random from $\bar{G}(S)$, i.e., $S^{-1} := S - \{i, j\}$ with probability $\frac{1}{|S|}$ for every $(i, j) \in \bar{G}(S)$. Finally, we define $OPT(S, r)$ to denote the (weight of the) maximum weight r-matching in S (containing r edges). When it is clear from the context, we will abuse notation and use $OPT(S, r)$ to denote the optimum r-matching itself (as opposed to its value).

Our proof depends on the following crucial structural claim: we show that for any set S, $OPT(S, r) - E[OPT(S^{-1}, r - 1)]$ is at most twice the weight of an edge chosen uniformly at random from $\bar{G}(S)$. This recursive claim implies that if at all we end up selecting a sub-optimal edge, then this does not hurt our solution by much since $E[OPT(S^{-1}, r - 1)]$ is bound to be large, and we apply the algorithm recursively on S^{-1}.

Claim C.1. *(Structural Claim) For any given set $S \subseteq \mathcal{N}$ and $r \leq \frac{|S|}{2}$, we have that*

$$OPT(S, r) \leq E[OPT(S^{-1}, r-1)] + \frac{2}{|S|} \sum_{e \in \bar{G}(S)} w(e)$$

In the above claim, the expectation is taken over the different realizations of S^{-1}. In words, the claim bounds the change in the optima using the 'increase in profit' of our algorithm. We first show how this claim can be used to complete the proof of Theorem 2.11, and then detail the proof of Claim C.1.

Proposition C.2. *As long as Claim C.1 is obeyed for every S, r, our algorithm provides a 2-approximation to the Max k-matching.*

Proof. Suppose that the algorithm proceeds in rounds (1 to k) where in each round exactly one edge is selected from the first preference graph. Define S_i to denote the random set of available nodes at the beginning of round i ($S_1 = \mathcal{N}$, and is deterministic). Then taking expectation over Claim C.1, we get that for every $i \leq k$,

$$E_{S_i}[OPT(S_i, k-i+1)] - E_{S_{i+1}}[OPT(S_{i+1}, k-i)] \leq E_{S_i}[\frac{2}{|S_i|} \sum_{e \in \bar{G}(S_i)} w(e)].$$

Moreover, if we define Alg_i to denote the expected weight of chosen edge in round i, the term in the RHS is simply twice Alg_i. Therefore, we can simplify the above inequality as follows.

$$E_{S_i}[OPT(S_i, k-i+1)] - E_{S_{i+1}}[OPT(S_{i+1}, k-i)] \leq 2Alg_i. \tag{25}$$

We also know that $OPT(\mathcal{N}, k)$ can be written as a telescoping summation, $OPT(\mathcal{N}, k) = \sum_{i=1}^{k} E[OPT(S_i, k-i+1)] - E[OPT(S_{i+1}, k-i)]$. After bounding the terms in the right hand side of the summation using Equation 25, we complete the proof,

$$OPT(\mathcal{N}, k) - E[OPT(S_{k+1}, 0)] \leq \sum_{i=1}^{k} 2Alg_i.$$

Since $E[OPT(S_{k+1}, 0)] = 0$, and the RHS of the above algorithm is exactly the expected weight of the solution returned by our algorithm, the proposition follows. □

□

It only remains to prove Claim C.1, which we complete now.

Proof of Claim C.1. We need to prove that $OPT(S, r) \leq E[OPT(S^{-1}, r-1)] + \frac{2}{|S|} \sum_{e \in \bar{G}(S)} w(e)$. Now, for any $i \in S$, we use o_i to denote the agent i is matched to in $OPT(S, r)$. If the agent is unmatched in $OPT(S, r)$, we let o_i be a null element. We also extend the notion of edge weights so that $w(i, \emptyset) = 0$ for all i. Finally, given any $i \in S$, let s_i denote i's most preferred node in S, i.e., the node to which i has an outgoing edge in $\bar{G}(S)$.

Suppose that the (random) serial dictatorship removes the edge (a, s_a) from $\bar{G}(S)$. We proceed in two cases based on whether or not a is matched to a non-null agent in $OPT(S, r)$. Let E_1 denote the subset of edges in $\bar{G}(S)$ such that a is matched to an actual agent in OPT, i.e., $o_a \neq \emptyset$. Note that s_a may or may not be matched in OPT. Then, for any $(a, s_a) \in E_1$, we have that

$$OPT(S - \{a, s_a\}, r-1)) \geq OPT(S, r) - w(a, o_a) - w(s_a, o_{s_a}) + w(o_a, o_{s_a}).$$

That is, $OPT(S - \{a, s_a\}, r-1)$ is at least as good as the matching obtained by pairing up o_a, o_{s_a} and leaving the other edges of $OPT(S, r)$. The above inequality is robust to o_{s_a} being empty.

Observe that by definition $w(a, o_a) \leq w(a, s_a)$ and via the triangle inequality, $w(s_a, o_{s_a}) \leq w(o_a, s_a) + w(o_a, o_{s_a}) \leq w(o_a, s_{o_a}) + w(o_a, o_{s_a})$. So, we get a simplified charging argument for edges in E_1,

$$OPT(S - \{a, s_a\}, r - 1)) \geq OPT(S, r) - w(a, s_a) - w(o_a, s_{o_a}). \tag{26}$$

Finally, let us denote the remaining edges in $\bar{G}(S)$ as E_2, for every edge in E_2; we know that $o_a = \emptyset$, o_{s_a} may or may not be empty. We claim that for both of these cases

$$OPT(S - \{a, s_a\}, r - 1)) \geq OPT(S, r) - 2w(a, s_a). \tag{27}$$

The main idea in this case is provided by the second statement in Proposition 4.3, from which we infer that the weight of any edge in $OPT(S, r)$ is at most twice $w(a, s_a)$. So, if $o_{s_a} = \emptyset$, then $OPT(S - \{a, s_a\}, r - 1))$ is at least $OPT(S, r) - w(a, o_a)$. In the case that both of them are null, one can simply remove any one edge from $OPT(S, r)$ to get a lower bound for $OPT(S - \{a, s_a\}, r - 1))$.

We are now ready to complete the proof.

$$\begin{aligned} E[OPT(S^{-1}, r-1)] =& \frac{1}{|S|} \sum_{(a,s_a)\in\bar{G}(S)} OPT(S - \{a, s_a\}, r - 1) \\ &\geq OPT(S, r) - \frac{1}{|S|}\{ \sum_{(a,s_a)\in E_1} w(a, s_a) + w(o_a, s_{o_a}) + \sum_{(a,s_a)\in E_2} 2w(a, s_a)\} \\ &\geq OPT(S, r) - \frac{1}{|S|} \sum_{(a,s_a)\in\bar{G}(S)} 2w(a, s_a) \end{aligned}$$

The crucial observation that leads us from line 2 to line 3 is that for any $(a, s_a) \in E_1$, the edge (o_a, s_{o_a}) must also belong to E_1. Therefore, in both of these cases, we are only counting $w(o_a, s_{o_a})$ twice. □ □

D PROOF OF PROPOSITION 3.8

Before proving the proposition, we state two important claims which are required to show the main proposition. Recall that GR denotes the greedy matching, $GR(T)$ is the set of $\frac{N}{4}$ highest-weight edges in GR and $GR(B) = GR \setminus GR(T)$. Moreover, given that $GR(B) = \chi w(OPT)$, we use $GR(B_1)$ to denote the highest-weight $\frac{\chi N}{2}$ edges in $GR(B)$ and $GR(B_2) = GR(B) \setminus GR(B_1)$. A graphical illustration can be found in Figure 1. For any given sets $S, S' \subseteq \mathcal{N}$ we use $w(S)$ to denote $\sum_{x,y\in S} w(x, y)$ and $w(S, S') = \sum_{x\in S, y\in S'} w(x, y)$. Finally, for the purposes of this proof, we will use w_i^{GR} to denote the weight of the i^{th} largest (heaviest-weight) edge in GR.

Claim D.1. ***(Local Stability)*** *Suppose that Algorithm 1 is run on the sub-instance consisting only of the nodes in T (alternatively B, B_1, B_2) with the same set of preferences. Then its output is $GR(T)$ (respectively $GR(B), GR(B_1), GR(B_2)$).*

Local stability has powerful consequences. For a given instance, we can take a subset of the greedy matching and show that all of the properties that apply to greedy matchings in general also apply to the subset, as it can be treated as an independent greedy matching. For the rest of this proof, we will treat greedy sub-matchings as independent greedy matchings on sub-instances, when it suits our needs. Our next claim follows from basic algebraic arguments so we simply state it without proof.

Claim D.2. *Consider two vectors $(w_i^1)_{i=1}^n$ and $(w_i^2)_{i=1}^n$ that satisfy the following conditions*

(1) $\sum_{i=1}^{n} w_i^1 = \sum_{i=1}^{n} w_i^2$
(2) $\exists$ *some index* k *such that for every* $r \le k$, $w_r^1 \ge w_r^2$ *and for every* $r > k$, $w_r^1 \le w_r^2$.

Let $\mathbf{a}$ *be any fixed vector of the same length satisfying* $a_1 \ge a_2 \ge \ldots \ge a_n$. *Then,*

$$\sum_{i=1}^{n} w_i^1 a_i \ge \sum_{i=1}^{n} w_i^2 a_i.$$

Our final claim is stated in some generality with respect to a greedy (sub-)matching on an arbitrary set of nodes S.

Claim D.3. *Let* $S \subseteq \mathcal{N}$ *be some set of nodes and let* $GR(S)$ *denote the output of Algorithm 1 on the complete edge set on* S *with* $k = \frac{|S|}{2}$. *Then,*

$$w(S) \le w(GR(S)) + \sum_{i=1}^{|S|/2} 2w_i^{GR(S)}\{|S| - 2i\},$$

where $w_i^{GR(S)}$ *denotes the weight of the* i^{th} *heaviest edge in* $GR(S)$.

Proof. Let $|S| = n$. We know that $GR(S)$ contains $\frac{n}{2}$ edges. Let (x_i, y_i) denote the i^{th} largest edge in $GR(S)$, and $S_{-i} := S \setminus \{x_1, y_1, x_2, y_2, \ldots, x_i, y_i\}$. Now, since the greedy algorithm selected the edge (x_i, y_i), it is not hard to deduce that for any $z \in S_{-i}$, we have $w(x_i, y_i) \ge w(x_i, z)$ and $w(x_i, y_i) \ge w(y_i, z)$. Moreover, note that $w(x_i, y_i) = w_i^{GR(S)}$.

Summing the above inequalities for every $z \in S_{-i}$ and adding a trivial inequality on both sides, we get

$$2|S_{-i}|w_i^{GR(S)} + w_i^{GR(S)} = 2w_i^{GR(S)}(n - 2i) + w_i^{GR(S)} \ge \sum_{z \in S_{-i}} (w(x_i, z) + w(y_i, z)) + w(x_i, y_i).$$

Adding these up for all $1 \le i \le \frac{n}{2}$ gives us the claim. □

Now, we are ready to prove the three statements in Proposition 3.8.

(Proof of Equation 6: $w(B_1, B_2) \le 2w(GR(B_1))|B_2|$*)*

The proof is very similar to that of Claim D.3. Since the edges in $GR(B_1)$ have larger weight than those in $GR(B_2)$, we infer that for any $(x, y) \in GR(B_1)$ and $z \in B_2$, we have that

$$w(x, y) \ge w(x, z) \quad \text{and} \quad w(x, y) \ge w(y, z).$$

Equation 6 simply follows from the summing the above inequalities for every $(x, y) \in GR(B_1)$ and $z \in B_2$. □

(Proof of Equation 7: $w(B_2) \le 2w(GR(B_2))\{|B_2| - \frac{w(GR(B_2))}{w^*}\}$)

Recall that $w^* = \frac{w(GR(B_1))}{|B_1|/2} = \frac{2w(GR(B_1))}{\chi N}$. As a consequence of Claim D.3, we have that

$$w(B_2) \le w(GR(B_2)) + \sum_{i=1}^{|B_2|/2} 2w_i^{GR(B_2)}\{|B_2| - 2i\}. \tag{28}$$

Since every edge in $GR(B_1)$ has larger weight than any given edge in $GR(B_2)$, and since w^* denotes the average weight of the edges in $GR(B_1)$, we have that for all $1 \le i \le |B_2|/2$, $w_i^{GR(B_2)} \le w^*$. Letb $t = \lfloor \frac{w(GR(B_2))}{w^*} \rfloor$ be the maximum number of w^* values that fit into $w(GR(B_2))$ and $r = w(GR(B_2)) - tw^*$ be the remainder. Then, we can use a charging argument to transform Equation 28 into one whose right hand side depends on w^*.

Specifically, staring with the second term in the right hand side of Equation 28, and applying Claim D.2 on the vectors $(w^*, w^*, \ldots, w^*, r, 0, \ldots, 0)$— w^* appears exactly t times— and $(w_i^{GR(B_2)})_{i=1}^{|B_2|/2}$, we get that

$$\begin{aligned}
\sum_{i=1}^{|B_2|/2} 2w_i^{GR(B_2)}\{|B_2| - 2i\} &\leq \textstyle\sum_{i=1}^{t} 2w^*(|B_2| - 2i) + 2r(|B_2| - 2(t+1)) \\
&= 2w^*(|B_2|t - t(t+1)) + 2r(|B_2| - 2(t+1)) \\
&= 2(w^*t + r)(|B_2| - t - 1) - 2r(t+1) \\
&\leq 2w(GR(B_2))(|B_2| - t - 1) - 2r(t + r/w^*) \\
&= 2w(GR(B_2))(|B_2| - t - \tfrac{r}{w^*} - 1) \\
&= 2w(GR(B_2))(|B_2| - \tfrac{w(GR(B_2))}{w^*} - 1).
\end{aligned}$$

The penultimate and final expressions come from the fact that $t + r/w^* = \frac{w(GR(B_2))}{w^*}$ by definition. We now wrap up the proof of the second statement in Proposition 3.8,

$$w(B_2) \leq w(GR(B_2)) + 2w(GR(B_2))(|B_2| - \frac{w(GR(B_2))}{w^*} - 1) \leq 2w(GR(B_2))(|B_2| - \frac{w(GR(B_2))}{w^*}). \quad \square$$

The proof of the final part of Proposition 3.8 is perhaps the most technically involved of the three statements. $\square$

(Proof of Equation 8: $w(B) \leq 2|B|w(GR(B))(1 - 2\chi)$ as long as $\chi \in [0, \frac{1}{4}]$, $\chi_1 \in [0, \frac{1}{2}]$)

Suppose that $n = |B| = \frac{N}{2}$. From Claim D.3, we know that

$$w(B) \leq w(GR(B)) + \sum_{i=1}^{n/2} 2w_i^{GR(B)}\{n - 2i\}. \tag{29}$$

Our goal for this part of the proof is to show that (over all possible distributions of greedy edge weights satisfying the condition $\chi_1 \in [0, \frac{1}{2}]$), the maximum value of the second term in the above inequality is obtained when the top $2\chi n$ edges of the greedy matching all have the same weight, specifically $\frac{w(GR(B))}{2\chi n}$. First, define $m = \chi n$. Note that $w_m^{GR(B)}$ is the weight of the lightest (lowest-weight) edge in $GR(B_1)$ (since $GR(B)$ has $n/2$ edges and $GR(B_1)$ has χn edges). Our proof will crucially depend on the weight of m^{th} heaviest edge in $GR(B)$, so let us use $\bar{w}$ to denote $w_m^{GR(B)}$. Our first claim on the way to showing Equation 8 is the following:

$$\sum_{i=1}^{m} 2w_i^{GR(B)}\{n - 2i\} \leq 2w_0(n-2) + \sum_{i=2}^{m} 2\bar{w}(n - 2i), \tag{30}$$

where w_0 is defined so that $w_0 + \sum_{i=2}^{m} \bar{w} = \sum_{i=1}^{m} w_i^{GR(B)}$. To see why this is the case, consider the two equal-length vectors $\mathbf{w^1} = (w_0, \bar{w}, \ldots, \bar{w})$ and $\mathbf{w^2} = (w_1^{GR(B)}, \ldots, w_m^{GR(B)})$. Since every entity in $\mathbf{w^2}$ is at least $\bar{w}$ (by definition) and the two vectors sum up to the same quantity, it must be the case that $w_0 \geq w_1^{GR(B)}$. So, we can apply our general Claim D.2 and get Equation 30. Next, we state a similar claim for the rest of the edges in $GR(B)$.

$$\sum_{i=m+1}^{\frac{n}{2}} 2w_i^{GR(B)}\{n - 2i\} \leq \sum_{i=m+1}^{2m} 2\bar{w}(n - 2i) + 2w_f(n - 2(2m+1)), \tag{31}$$

where w_f is defined so that $\sum_{i=m+1}^{2m+1} \bar{w} + w_f = \sum_{i=m+1}^{\frac{n}{2}} w_i^{GR(B)}$. Since $\chi_1 \leq \frac{1}{2}$, w_f must be non-negative. Indeed, we have that $\sum_{i=m+1}^{2m} \bar{w} \leq \sum_{i=1}^{m} w_i^{GR(B)} \leq \sum_{i=m+1}^{n/2} w_i^{GR(B)}$. Therefore, $w_f =$

$\sum_{i=m+1}^{n/2} w_i^{GR(B)} - \sum_{i=m+1}^{2m} \bar{w} \geq 0$. Once again, Equation 31 can be shown via an application of Claim D.2, since for every $i > m$, $w_i^{GR(B)} \leq \bar{w}$, so we are simply transferring the weights to the smaller indices. Combining Equations 30 and 31, we get the following intermediate inequality, from which it is more convenient to arrive at the required claim.

$$\sum_{i=1}^{n/2} 2w_i^{GR}\{n-2i\} \leq 2w_0(n-2) + 2\sum_{i=2}^{2m} \bar{w}(n-2i) + 2w_f(n-2(2m+1)).$$

Moreover, since $w_0 + \sum_{i=2}^{m} \bar{w} = \sum_{i=1}^{m} w_i^{GR(B)} \leq \sum_{i=m+1}^{n/2} w_i^{GR(B)} = \sum_{i=m+1}^{2m} \bar{w} + w_f$, it must be the case that $w_0 \leq \bar{w} + w_f$. Let $\epsilon = w_0 - \bar{w}$. Clearly, $\epsilon \leq w_f$ and therefore, the following inequality holds:

$$\epsilon(n-2) + w_f(n-2(2m+1)) \leq (\epsilon + w_f)\frac{n-2+n-2(2m+1)}{2} = (\epsilon + w_f)(n-2m-2). \quad (32)$$

The rest of the proof follows from direct algebraic manipulation and an application of Equation 32.

$$\begin{aligned}
\sum_{i=1}^{n/2} 2w_i^{GR}\{n-2i\} &\leq 2(\bar{w}+\epsilon)(n-2) + 2\sum_{i=2}^{2m} \bar{w}(n-2i) + 2w_f(n-2(2m+1)) \\
&= \sum_{i=1}^{2m} \bar{w}(n-2i) + 2\epsilon(n-2) + 2w_f(n-2(2m+1)) \\
&\leq 2\bar{w}\cdot 2m(n-2m-1) + 2(\epsilon+w_f)(n-2m-2) \\
&\leq 2\bar{w}\cdot 2m(n-2m-1) + 2(\epsilon+w_f)(n-2m-1) \\
&= 2(2m\bar{w}+\epsilon+w_f)(n-2m-1) \\
&= 2w(GR(B)))(n-2m-1).
\end{aligned}$$

Note that by definition, $w(GR(B)) = w_0 + (2m-1)\bar{w} + w_f = \epsilon + 2m\bar{w} + w_f$. Of course, we can now complete the proof by plugging this back into Equation 29, i.e., we have that

$$\begin{aligned}
w(B) &\leq w(GR(B)) + 2w(GR(B))(n-2m-1) \\
&\leq w(GR(B))(n-2m) \\
&= nw(GR(B))(1-2m/n) = nw(GR(B))(1-2\chi) \quad \square
\end{aligned}$$

This completes the proof of the final part of Proposition 3.8. $\square$

E 8-APPROXIMATE TRUTHFUL ALGORITHM FOR DENSEST K-SUBGRAPH

E.1 Approximation Algorithm when $k \geq \frac{N}{2}$

Claim E.1. *(Trivial Algorithm) Suppose that $k \geq \frac{N}{2}$, then the algorithm that selects a set of size k uniformly at random from $\mathcal{N}$ is a universally truthful 8-approximation algorithm for Densest k-subgraph.*

Proof. The truthfulness of this algorithm is quite obvious so we show the approximation factor. A trival upper bound for OPT is $OPT \leq w(\mathcal{N})$, where $w(T)$ denotes the total weight of the graph induced by T.

Let S be the random set returned by the above algorithm. Then, for some $i, j \in \mathcal{N}$, what is the probability that $i, j \in S$: this probability is exactly $\frac{\binom{N-2}{k-2}}{\binom{N}{k}}$. As expected, the worst case occurs when $k = \frac{N}{2}$, giving us $Pr(i, j \in S) \geq \frac{N/2-1}{2(N-1)} \geq \frac{1}{6}$ for $N > 3$. Therefore, we have that $E[w(S)] \geq \frac{1}{6}\sum_{i,j\in\mathcal{N}} w(i,j) \geq \frac{1}{8}\sum_{i,j\in\mathcal{N}} w(i,j)$. $\square$

Received February 2007; revised March 2009; accepted June 2009