Computing Manipulations of Ranking Systems

Ethan Gertle Erika Mackin Malik Magdon-Ismail Lirong Xia Yuan Yi
{gertle,mackie2,yiy2}@rpi.edu, {magdon,xial}@cs.rpi.edu
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY , USA

ABSTRACT

Ranking systems are widely used by agencies to rank agents, for example US News ranks colleges. Such rankings are prone to manipulation by the agency (e.g. US News) for publicity, and also by the agents (e.g. colleges) to get a better rank. We analyze the algorithmic aspects of manipulation in *linear ranking systems* of m agents using n features. Computing optimal manipulation for the ranking agency is NP-hard: we give a mixed integer linear program solution, and also an efficient linear programming heuristic that formulates the problem as a classification task. Computing optimal manipulations for ranked agents subject to a budget constraint is a minimax problem in a *continuous* space: we give a general $O((m+k)^{n^2-n}m^{n-1}\log m)$ algorithm, where k is the number of linear constraints on the weight vector and an $O(m(\log m)^2)$ algorithm for $n = 2$. We also present a large class of heuristic algorithms with approximation ration n.

We tested our algorithms on USNews data from the 2015 college rankings. Our algorithms compute agency and agent manipulation strategies in seconds. We present several interesting experiments on the ranking range of the top-100 colleges, including optimal spending plans for these colleges.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences–Economics; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Economics, Theory

Keywords

ranking systems; manipulation; US News

1. INTRODUCTION

A ranking agency ranks each of m agents based on the values of their n features. A popular approach is the *linear ranking system*, which uses a weighted sum of the features, $\vec{w}\cdot\vec{c} = \sum_{i=1}^{n} w_i c_i$ to rank ($\vec{w}$ is a vector of ranking weights and $\vec{c}$ is a vector of features). An example of a linear ranking system is the U.S. News & World Report (USNews for short) college ranking, which uses a weighted sum of subjective features (such as reputation and high school consular rating) and objective features (such as retention rate, tuition, and number of classes with no more than 20 students).[1] It is well-known that a good college ranking is *"very important"* and *"very profitable"* for a school [21], a phenomenon that has attracted significant attention recently due to Northeastern University's fast rise in the rankings. Ranking systems are everywhere. For example, web search engines rank websites based on features; meta-search engines use search engine results as the features to fine-tune the ranking [14]; linear systems are used to rank applicants for awards, e.g. the Future Fellows Awards [2]; recommender systems rank items based on their features [18]; information retrieval systems rank query results based on relevancy features [22].

Ranking systems provide actionable information, and hence are prone to manipulation by both the ranking agency and the ranked agents. For example, USNews may vary the weights from year to year, for example, in the 2013 rankings (announced in 2012), Harvard dropped from tied-1st to the 2nd place. This has drawn criticism, for it is hard to believe that *"anything changed at Harvard for that to have happened"* [26]. Is USNews being honest or just manipulating the weights to increase its publicity? Similarly, there is evidence that colleges are manipulating their features to improve their USNews ranking, by strategically enlarging the application pool and reducing the size of classes [21], or even falsely reporting their features [23]. Similarly, websites have been manipulating their Google search rank (known as "Google Bombs" [1, 4]), and Baidu, a popular Chinese web search engine, has been accused by manipulating ranks in favor of websites who paid them [10]. There is clear motivation to manipulate, but how can one optimally do it? There is little work on the following question:

"How much can the agency or agent manipulate the rankings?"

If a ranking agency has significant ability to manipulate rankings, then the published rankings start to lose credibility. Given the ranking methodology, ranked agents would like to know how they may best increase their ranking by improving their features. Our contributions are to answer the following questions.

- **Q1 (agency's basic manipulation):** If the agency does not make any commitments on the ranking methodology, how diverse can agents' ranks be?
- **Q2 (agency's manipulation under constraints):** What if the agency commits to (for example) bounds on the weights, or the relative importance of the features?
- **Q3 (agents' manipulation):** What is the optimal spending plan for an agent to improve its features to guarantee a good rank with or without knowing how the ranker will select weights?

[1] USNews applied rounding to the weighted sum, which can be seen as noise and is not studied in this paper.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

1.1 Our contributions

We take an algorithmic approach towards understanding the extent to which linear ranking systems can be manipulated We show that manipulation by ranking agency is NP-hard to compute, and we provide mixed integer linear programs to compute the highest and lowest rank under various constraints, including (1) Top-k constraints, which requires a given set of agents must be ranked within top k. (2) Weight-order constraints, which impose an order over the sizes of the weights based on the relative importance of the features. (3) Weight-range constraints. We also give an efficient linear programming heuristic by formulating the problem as a classification task. For manipulation by ranked agents, we provide an $O((m+k)^{n^2-n}m^{n-1}\log m)$ algorithm, where k is the number of linear constraints on the weight vector, to compute the minimax-optimal spending plan for a budget constrained agent, and improve this to $O(m(\log m)^2)$ when $n=2$. We also provide an efficient heuristic algorithm which computes a spending plan which is at most an n-factor from optimal. Our experiments show that the heuristic is close to optimal in practice.

We tested our algorithms on USNews data for their 2015 college rankings. Our algorithms can compute agency's manipulation and minimax strategies using $n=2$ features in minutes. We show that in these cases our heuristic algorithms achieve competitive accuracy and are much more efficient. More importantly, the heuristic algorithms for computing minimax strategies can find spending plans when there are $n=15$ features (as in the USNews data) within a few hours.

We give experimental results to answer Q1–Q3 for a variety of colleges. For Q1, we show that the range of most agents' ranks can be very diverse. For Q2, if USNews can promise that the weights do not deviate too much or even that the ordering over the weights do not change, then the range is much smaller and manipulation by the agency is less of a concern. For Q3, we give (near) optimal spending plans for a variety of colleges including WUSTL, RPI, and U. New Hampshire (which span a wide range in the top 100 colleges). Our results show that manipulation of rankings for problems on the scale of the USNews data is easily within the reach of current computation capability.

1.2 Related Work and Discussion

The computational complexity of manipulation, bribery, and control in voting has been studied for some time (see for example [7, 8, 17, 16]). Our setting and integer linear problems for basic manipulation problem (Q1) is closely related to the work by [9], where they investigated complexity of and algorithms for computing whether some candidate can win (the *possible winner* problem) and whether some candidate always wins (the *necessary winner* problem) when the chair can control the weights of the voters for various voting rules that are different from the linear system studied in this paper. There are three significant differences between our work and the previous research.

1. Traditional work in voter manipulation considers each voter as an ordering over candidates and the voting rule outputs a winner based on the voter orderings. A voter manipulates his ordering of the candidates to obtain a more desirable winner. We focus on ranking systems, where the inputs are feature vectors (c.f. range voting) and the output is the ordering over agents. An agent's manipulation goal is to obtain as high a rank as possible.
2. In the traditional setting, one tries to come up with a voting rule for which the computational complexity of manipulation is provably high, making the voting rule resistant to strategic manipulations. Our focus is the extent of manipulation and fast algorithms to compute manipulation strategies. We define agent manipulation in terms of minimax strategies which are notoriously hard to compute, and for which we give efficient algorithms and heuristics.
3. Most previous research on manipulation in voting systems is theoretical (see [25] for a recent experimental approach). We tested our algorithms on real USNews college ranking data.

Mathemetically speaking, the agency's basic manipulation problem (Q1) is an instance of *maximum feasible subsystems of linear inequalities (MFS)*, also known as the *halfspace depth*, *location depth*, and *Tukey depth* problems [20]. There are heuristic and randomized algorithms for MFS [6, 11, 12, 24, 5]. In MFS, the task is to find a maximum feasible subsystem of a system of linear inequalities. However, we don't see a way to convert the agency's restricted manipulation problem (Q2) to MFS, so we propose a MIP formulation and, for certain restrictions, an efficient LP-heuristic. The agents' manipulation problem (Q3) is a minimax problem with little relationship to MFS.

Previous work on computing optimal decisions in games focused on mixed strategies for a finite minimax problem [13]. Our problem of computing the minimax spending plan (Q3) is significantly different, because we want to compute a pure strategy (a spending plan), where the utility function of the agents (their ranks) cannot be explicitly represented. Our $O((m+k)^{n^2-n}m^{n-1}\log m)$ algorithm leverages a computational geometry algorithm that computes *arrangements* in a Euclidean space [15] (the partition of a hyperplane into lower-dimensional pieces created by a system of other hyperplanes). When $n=2$, we improve our algorithm using a novel binary search method with $O(m(\log m)^2)$ run time. We also develop heuristics for the general case that provide an $O(n)$-approximation to the optimal spending plan, making $O(n)$ calls to our solution of Q2.

There is intense research on analyzing USNews methods to predict rankings (see the work by [19] and reference therein). Our results complement these studies. We are the first to quantitatively study the manipulation capabilities of USNews (Q1 and Q2) and our results on agent manipulation and optimal spending plans (Q3) introduce novel quantitative tools for computing *optimal robust* strategies w.r.t. USNews' ability to change the weights. A different approach to predicting rank change using regression is given in [19], where it is assumed that the weights are unchanged and are known; this is a different problem.

We provide a bridge between the high-stakes manipulation of ranking systems, especially the college rankings, and the multi-agent computational social choice community. Even though our application is college ranking, our methods can be applied to any linear ranking system, and they are efficient.

2. PRELIMINARIES

There are m ranked agents $\{1,\ldots,m\}$. Each agent i is characterized by a n-dimensional feature vector $\vec{c}_i \in [-0.5, 0.5]^n$. (features are bounded between 0 and 1, which is without loss of generality because the ranking is obtained by comparing a weighted sum of the features). Let $P=(\vec{c}_1,\ldots,\vec{c}_n)$ denote *profile* of all feature vectors. A *ranking agency* applies a ranking system that maps each profile to a full ranking with ties over the agents.

We only consider the *linear ranking system* which is characterized by a weight vector $\vec{w}=(w_1,\ldots,w_n)$ that satisfies $w_j \geq 0$ and $\sum_{i=1}^n w_i = 1$ (the non-negative constraint is because we assume every feature is positively correlated with rank). For profile P, the agents are ranked using their scores $\vec{w}\cdot\vec{c} = \sum_{j=1}^n w_j c_j$ (higher score is better). More precisely, agent i is ranked above

agent i' if and only if $\vec{w} \cdot \vec{c}_i > \vec{w} \cdot \vec{c}_{i'}$. The best rank is 1 and the worst is m; agents with the same score get the same rank:

$$\text{Rank}(i) = 1 + |\{i' : \vec{w} \cdot \vec{c}_{i'} > \vec{w} \cdot \vec{c}_i\}|.$$

Example 1 *In the USNews 2015 best college ranking, colleges have* 17 *features, divided into the following seven categories [3]. (Note that the 2015 and 2012 rankings use different weights.)*

- **Undergraduate academic reputation (22.5%):** *"Peer assessment score" (15%) and "High school counselor score" (7.5%).*
- **Retention (22.5%):** *"6-year graduation rate" (18%) and "Average freshman retention rate" (4.5%).*
- **Faculty resources (20%):** *"Classes with* < 20 *students" (6%), "Classes with* ≥ 50 *students" (2%), "Faculty salary" (7%), "Proportion of professors with the highest degree in field" (3%), "Student-faculty ratio (1%)" and "Percent of full-time faculty" (1%).*
- **Student selectivity (12.5%):** *"Fall 2013 acceptance rate" (1.25%), "SAT/ACT 25th-75th percentile" (8.125%), "Freshmen in top 10 percent of high school class" (1.5625%), "Freshmen in top 25 percent of high school class" (1.5625%).*
- **Average spending per student (10%).**
- **Graduation rate (7.5%),** *the over/under-performance predicted by USNews compared to actual graduation rate.*
- **Alumni giving rate (5%).**

The scores (weighted sum of features) are normalized as a percentage of the highest score, and then rounded to the nearest integer. The final ranking is determined by the rounded percentages.

3. MANIPULATION BY THE AGENCY

In the agency's basic manipulation problem, we assume that the ranking agency knows the features of the agents (so the profile P is fixed) and wants to find a weight vector $\vec{w}$ to improve the rank of a distinguished agent as much as possible (constructive manipulation, corresponding to lowering the rank) or worsen the rank as much as possible (destructive manipulation, corresponding to increasing the rank).

Definition 1 (Agency manipulation) *In the basic constructive (resp. destructive) manipulation problem, the agency computes a valid set of weights $\vec{w}$ so that a distinguished agent i has minimum (resp. maximum) rank.*

Recall that valid weights are non-negative and sum to 1. These constraints on $\vec{w}$ are mild: the weights summing to 1 is vacuous since all we care about is the ranking; and, the non-negative constraint simply means the features must positively correlate to ranking. In real life applications, especially in college ranking systems, there are considerably more restrictions on the weights. We consider three common types of additional constraints.

The first requires that a given set of agents is ranked within the top-k for some given k. For example, to make the college ranking appear credible, Harvard, Yale, and Princeton should be ranked within the top-10. The second requires that the relative importance of the features to be fixed. For example, in the 2015 best college ranking, the weight of 6-year graduation rate (18%) is the highest, which means that USNews deem it most important in their ranking. Meanwhile, the weights of High school counselor score and Graduation rate performance are the same. Therefore, it is reasonable to require that for the manipulated weight vector, the weight of 6-year graduation rate is still the highest, and the weights of High school counselor score and Graduation rate performance remain equal. The third requires that the weight vector is in some bounded range. For example, one expects that USNews would not change any weight by more than (say) 10% for next year's ranking.

Definition 2 (Top-k constraint on $\vec{w}$) *Given a subset of agents $\mathcal{A}$ and $k \leq m$, every agent in a subset $\mathcal{A}$ should have rank at most k.*

Definition 3 (Weight-order constraint on $\vec{w}$) *Given a partial ordering $\trianglerighteq$ over $\{1, \ldots, n\}$, $w_j \geq w_{j'}$ if and only if $j \trianglerighteq j'$.*

Definition 4 (Weight-range constraint on $\vec{w}$) *We are given upper and lower bounds $(\underline{W_j}, \overline{W_j})$ and $\underline{W_j} \leq w_j \leq \overline{W_j}$ for $j = 1, \ldots, n$.*

Any of these additional constraints could be added into the basic manipulation problem. Indeed, our formulation can handle any integer-linear constraints, but the ones above are the ones we focus on. The basic manipulation problem without any constraints on $\vec{w}$ asks to compute $\vec{w}$ so that $\vec{w} \cdot (\vec{c}_i - \vec{c}_{i'}) \geq 0$ for as many i' as possible. This is exactly the maximum feasible subsystem problem (MFS), which is NP-hard. After adding the non-negativity constraint, the problem remains NP-hard, which we can prove by reduction from the NP-hard problem CLOSED HEMISPHERE [20]. We state this fact, but defer the proof to a full version.

Theorem 1 *Computing the maximum constructive or destructive basic manipulation weights is NP-hard.*

We present an efficient mixed integer linear program (MILP) in Figure 1 for the agency's manipulation problem that can accomodate any of the weight constraints we have discussed above. The profile P is fixed and for a distinguished agent i^*, the agency wants to compute weights $\vec{w}$ to minimize $\text{Rank}(i^*)$ or maximize $\text{Rank}(i^*)$. In the MILP in Figure 1, the variables are the weights w_j and the binary variables x_{i_1,i_2} for some pairs of agents $i_1, i_2 \leq m$. The binary variable x_{i_1,i_2} is 0 if and only if $\vec{w} \cdot (\vec{c}_{i_1} - \vec{c}_{i_2}) \geq 0$ (for weights $\vec{w}$, i_2 is *not* ranked ahead of i_1, i.e. $\text{Rank}(i_1) \leq \text{Rank}(i_2)$), and $x_{i_1,i_2} = 1$ otherwise. The total number of agents that are ranked strictly ahead of our distinguished agent i^* is therefore $\sum_{i \neq i^*} x_{i^*,i}$, which becomes the objective of the MILP. Minimizing this objective corresponds to the constructive manipulation problem, and maximizing this objective corresponds to the destructive manipulation. For the Top-k constraint, we require that for each agent i' in $\mathcal{A}$, no more than $k-1$ agents are ranked strictly ahead of it, which is guaranteed by the constraint $\sum_{i \neq i'} x_{i',i} < k$. The weight-order constraints and weight-range constraints are standard linear inequality constraints.

4. MANIPULATION BY THE AGENTS

Suppose a distinguished agent has a budget B to improve her features for the next ranking. Let $\vec{c}$ denote the agent's initial feature vector. We focus on *linear budget constraints*. That is, there exists a cost vector $\vec{C} = (C_1, \ldots, C_n)$, where $C_i > 0$. An agent can update its features to $\vec{c} + \vec{\Delta}$ for any $\vec{\Delta} \geq 0$ that satisfies the linear budget constraint $\vec{C} \cdot \vec{\Delta} \leq B$. Any $\vec{\Delta}$ satisfying the budget constraint is a valid *spending plan*, and we collect the set of feature vectors $\vec{c} + \vec{\Delta}$ that are reachable for the agent into the *feasible set* $\text{Feasible}_B(\vec{c})$ (this set depends on $\vec{C}$, but for simplicity, we suppress that dependence).

We further assume that the distinguished agent knows the other agents' features, denoted by P^-, via direct observation or estimation. If the agent also knows the weights $\vec{w}$ that the ranking agency will use, then computing the optimal feasible spending plan is easy and reduces to putting all your money into the feature with biggest payoff per unit of cost.

Theorem 2 *Given the feature vector $\vec{c}$ of a distinguished agent, a budget B, a cost vector $\vec{C}$, other agents' feature vectors P^-, and the weight vector $\vec{w}$, let $j^* = \arg\max_j w_j/C_j$. Then the optimal spending plan is $\vec{\Delta}^*$ where $\vec{\Delta}^*_{j^*} = B/C_{j^*}$ and $\vec{\Delta}^*_j = 0$ for $j \neq j^*$.*

$$
\begin{aligned}
&\min \text{ (max for destructive manipulation) } \sum_{i \neq i^*} x_{i^*,i} \\
s.t.\quad & \left.\begin{array}{l} \forall i \neq i^*,\ x_{i^*,i} - 1 < \vec{w} \cdot (\vec{c}_i - \vec{c}_{i^*}) \leq x_{i^*,i} \\ \forall i \neq i^*, x_{i^*,i} \text{ is binary} \\ \sum_{j \leq n} w_j = 1 \end{array}\right\} \text{Basic manipulation} \\
& \left.\begin{array}{l} \forall i' \in \mathcal{A}, \forall i \neq i',\ x_{i',i} - 1 < \vec{w} \cdot (\vec{c}_i - \vec{c}_{i'}) \leq x_{i',i} \\ \forall i' \in \mathcal{A}, \sum_{i \neq i'} x_{i',i} < k \\ \forall i \neq i', x_{i',i} \text{ is binary} \end{array}\right\} \text{Top-}k \text{ constraints} \\
& \forall j_1 \trianglerighteq j_2, w_{j_1} \geq w_{j_2} \qquad \Longrightarrow \text{Weight-order constraints} \\
& \forall j \leq n, \underline{W_j} \leq w_j \leq \overline{W_j} \qquad \Longrightarrow \text{Weight-range constraints}
\end{aligned}
$$

Figure 1: MILP for agency's manipulation problem w.r.t. the three types of constraints.

PROOF. Since $\vec{w}$ is fixed, the scores of other agents are fixed. To minimize Rank$(\vec{c} + \vec{\Delta})$ we maximize the score $\vec{w} \cdot (\vec{c} + \vec{\Delta})$. That is we maximize $\vec{w} \cdot \vec{\Delta}$ subject to $\vec{C} \cdot \vec{\Delta} \leq B$. The maximum is achieved by a Δ that is positive in only one component, and a quick calculation shows that the component to choose is the one that maximizes w_j / C_j.

When $\vec{w}$ is not known, the optimality of a spending plan needs to be defined. We adopt the celebrated *minimax* principle in decision theory. Let $\mathcal{W}$ be the set of all possible weight vectors that the agency can choose from. A feasible spending plan is minimax optimal, if it offers the best *guarantee* on the rank, against an adversarial ranking agency who can choose any weight vector from $\mathcal{W}$.

Definition 5 (Minimax spending plan) *Given $\mathcal{W}$, the feature vector $\vec{c}$ of the distinguished agent i^*, and the profile P^- for other agents, The* minimax spending plan $\vec{\Delta}^*$ *is the feasible spending plan that minimizes the worst possible rank over $\mathcal{W}$:*

$$\vec{\Delta}^* = \arg \min_{\textit{feasible } \vec{\Delta}} \max_{\vec{w} \in \mathcal{W}} \textit{Rank}(\vec{c} + \vec{\Delta}; \vec{w}),$$

where Rank$(\vec{c} + \vec{\Delta}; \vec{w})$ is the rank of $\vec{c} + \vec{\Delta}$ for the weights $\vec{w}$ given the profile P^- for other agents. $\max_{\vec{w} \in \mathcal{W}}$ *Rank$(\vec{c} + \vec{\Delta}^*; \vec{w})$ is called the* minimax value.

We note that $\max_{\vec{w} \in \mathcal{W}} \text{Rank}(\vec{c} + \vec{\Delta}; \vec{w})$ is exactly the outcome of optimal destructive manipulation by the agency (Definition 1). When $B = 0$, Feasible$_B(\vec{c}) = \{\vec{c}\}$. Therefore, we immediately have the following theorem about the complexity of checking the minimax value.

Theorem 3 *Given $\mathcal{W}$, P^-, Feasible$_B(\vec{c})$ and $T \geq 0$, checking whether the minimax value is smaller than T is coNP-complete.*

The main theoretical contribution of this section is an $O((m + k)^{n^2 - n} m^n)$ algorithm for computing the minimax value and the corresponding minimax plan, when $\mathcal{W}$ can be represented by k linear constraints on $\vec{w}$. More precisely, for any $K \subseteq \{\mathbb{R}\}^n$, we let $\mathcal{W}_K = \{\vec{w} : \forall \vec{a} \in K, \vec{a} \cdot \vec{w} \geq 0\}$.

We first present an $O(m^4)$ algorithm for $n = 2$ and $\mathcal{W} = \mathbb{R}^2_{\geq 0}$ to illustrate the idea, then we improve the theorem in two orthogonal directions: first, we propose a novel binary search method to improve the algorithm for $n = 2$ to $O(m^2 \log m)$; second, we extend the algorithm to arbitrary n and $\mathcal{W}_K$ for arbitrary K.

Our algorithm for $n = 2$ and $\mathcal{W} = \mathbb{R}^2_{\geq 0}$ works as follows. Feasible$_B(\vec{c})$ is a line segment. Let P^- denote the feature vectors of other $m - 1$ agents. We first compute the intersections of Feasible$_B(\vec{c})$ and (1) line segments $[\vec{c}_i, \vec{c}_{i'}]$ between all pairs of points in P^- and (2) all horizontal and vertical rays towards the positive directions in both axes starting from any point in P^-, which can be seen as the line segment connecting any point in P^- and one of $\{(0, +\infty), (+\infty, 0)\}$. These intersections partition Feasible$_B(\vec{c})$ into $O(m^2)$ open intervals and $O(m^2)$ points. Then, for each point and an arbitrary point (which we choose to be the midpoint) in each interval, we use the $O(m^{n-1} \log m)$ enumeration algorithm in [20] to compute the max value. Finally, the algorithm outputs the point with the minimum max value among these intersections and midpoints. The algorithm is illustrated in Algorithm 1.

Algorithm 1: Minimax plan for $n = 2$.

Input: Feasible$_B(\vec{c})$ (which is a line segment) and P^-.

for *Each pair* $\vec{c}_i, \vec{c}_{i'} \in P^-$ **do**
  If both $\vec{c}_i$ and $\vec{c}_{i'}$ are on Feasible$_B(\vec{c})$ then $I_{\vec{c}_i, \vec{c}_{i'}} = \{\vec{c}_i, \vec{c}_{i'}\}$. Otherwise let $I_{\vec{c}_i, \vec{c}_{i'}}$ be the intersection of the line segment $[\vec{c}_i, \vec{c}_{i'}]$ and Feasible$_B(\vec{c})$.
end
for *Each* $\vec{c}_i \in P^-$ **do**
  Compute the set of intersections $I_{\vec{a}_i}$ of Feasible$_B(\vec{c})$ and rays $[\vec{a}_i, (+\infty, 0))$ and $[\vec{a}_i, (0, +\infty))$ respectively.
end
Let I denote all intersections and the two endpoint of Feasible$_B(\vec{c})$.
Order I from left to right as $(\vec{p}_0, p_2 \ldots, \vec{p}_{2t})$ and let $\{\vec{p}_1, \vec{p}_3, \ldots, \vec{p}_{2t-1}\}$ denote the midpoints of $(\vec{p}_0, \vec{p}_2), \ldots, (\vec{p}_{2t-2}, \vec{p}_{2t})$ respectively.
Compute the max value of $\vec{p}_s$ for all $0 \leq s \leq 2t$ by the algorithm in [20].
return the point with the minimum max value.

Example 2 *Suppose there are 4 agents whose feature vectors are $\{\vec{c}, \vec{c}_1, \vec{c}_2, \vec{c}_3\}$ respectively as in Figure 2, where $\vec{c}$ is the feature vector of the distinguished agent. Suppose Feasible$_B(\vec{c})$ is the line segment $[\vec{p}_0, \vec{p}_{10}]$. Algorithm 1 first computes intersections of all line segments connecting points in $P^- = \{\vec{c}_1, \vec{c}_2, \vec{c}_3\}$, and $\{\vec{p}_2, \vec{p}_6\}$ are the intersections. Then Algorithm 1 computes the intersections of Feasible$_B(\vec{c})$ and the rays starting at each $\vec{c}_i$ that goes to positive infinity along the two axes, and $\{\vec{p}_4, \vec{p}_8\}$ are the intersections. The algorithm computes the max values of $\{\vec{p}_0, \vec{p}_2, \ldots, \vec{p}_{10}\}$, and the midpoints of $\{[\vec{p}_0, \vec{p}_2], \ldots, [\vec{p}_8, \vec{p}_{10}]\}$, denoted by $\vec{p}_1, \vec{p}_3, \ldots, \vec{p}_9$ respectively (not shown in the figure). Finally, the algorithm outputs the point with the minimum max value.*

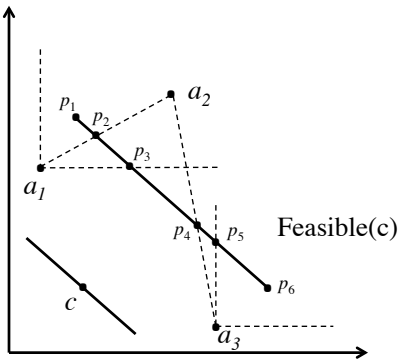


Figure 2: Illustration of Algorithm 1 ($n = 2$).

Theorem 4 *Algorithm 1 computes the minimax plan w.r.t. $\mathcal{W} = \mathbb{R}^2_{\geq 2}$ in $O(m^3 \log m)$ time.*

Proof sketch: To prove the correctness of Algorithm 1, it suffices to prove that all points on the same open line segment $(\vec{p}_s, \vec{p}_{s+1})$ in Feasible$_B(\vec{c})$ have the same max value. For any $\vec{p} \in (\vec{p}_s, \vec{p}_{s+1})$, let $\vec{w}$ denote the weight vector that achieves the max value for $\vec{p}$, let H denote the hyperplane that contains $\vec{p}$ and is perpendicular to $\vec{w}$, and let $S \subseteq P^-$ denote the set of points on the positive side of H. Let $\mathcal{H}$ denote the convex hull of $S \cup \{(0, +\infty), (\infty, 0)\}$. It follows that all points in $\mathcal{H}$ are on the positive side of H, which means that $\vec{p}$ is not contained in $\mathcal{H}$.

For any $\vec{q} \in (\vec{p}_s, \vec{p}_{s+1})$, we first show that $\vec{q}$ is not in $\mathcal{H}$ either. Suppose on the contrary that $\vec{q} \in \mathcal{H}$, then the line segment $[\vec{p}, \vec{q}]$ must intersects a face of $\mathcal{H}$. This contradicts the assumption that $\vec{p}$ and $\vec{q}$ belong to the same line segment on Feasible$_B(\vec{c})$ computed in Step 8 in Algorithm 1. Therefore, $\vec{q} \notin \mathcal{H}$. We note that $\mathcal{H}$ is a closed convex set. Applying the separating hyperplane separation theorem, there exists a hyperplane H' passing $\vec{q}$ such that all points in $\mathcal{H}$ are in one open side of H'. Let $\vec{w}'$ denote the vector that is perpendicular to H' pointing to $\mathcal{H}$. It follows that for all $\vec{q}_1 \in \mathcal{H}$ we have $\vec{w}' \cdot (\vec{q}_1 - \vec{q}) > 0$. In particular, the inequality holds for $\vec{q}_1 = [0, +\infty)$ and $\vec{q}_1 = (+\infty, 0]$, which means that $w'_1 > 0$ and $w'_2 > 0$. Therefore, $\vec{w}' \in \mathcal{W}_{K_{\geq 0}}$. This means that the max value of $\vec{q}$ is at least the max value of $\vec{p}$. Since this holds for any choice of $\vec{p}$ and $\vec{q}$ in $(\vec{p}_s, \vec{p}_{s+1})$, the max value for all points in $(\vec{p}_s, \vec{p}_{s+1})$ must be the same.

This proves the correctness of Algorithm 1. Clearly the running time is $O(m^3 \log m)$. □

We can improve step 8 of Algorithm 1 using binary search, where we do not need to compute the max values of all points in $\{\vec{p}_0, \ldots \vec{p}_{2t}\}$. Instead, in each iteration we compute the max value for the point $\vec{p}_s$ in the middle of the list, and use the adversarial weight vector $\vec{w}$ to eliminate half of the points based on the following observation: for any vector $\vec{p}$ with $\vec{w} \cdot (\vec{p} - \vec{p}_s) <= 0$, the max value of $\vec{p}$ is at least the max value of $\vec{p}_s$ at least by using the same $\vec{w}$. Formally, the algorithm is presented as Algorithm 2.

Theorem 5 *Algorithm 1 with step 9 replaced by Algorithm 2 computes the minimax plan w.r.t. $\mathcal{W} = \mathbb{R}^2_{\geq 2}$ in $O(m(\log m)^2)$ time.*

The extension of Algorithm 1 to general n and $\mathcal{W}_K$ follows the same idea and is illustrated in Algorithm 3. We first compute the partition of Feasible$_B(\vec{c})$ by hyperplanes formed by $n-1$ points in P^- and k points representing the constraints in K that are infinitely far away. Such a partition is called an *arrangement*, and we can use the algorithm proposed in [15] to compute a compact representation

Algorithm 2: Binary search for $n = 2$.

```
Let t_min = 0 and t_max = 2t.
while t_min ≤ t_max − 2 do
    Let t* = ⌊(t_min + t_max)/2⌋.
    Compute the max value of p_{t*} by the algorithm in [20] and
    let w denote the weight vector that gives the max value.
    If p_{t_max} − p_{t*} · w ≥ 0, then let t_min = t*; otherwise let
    t_max = t*.
end
return the point in {t_min, t_max} with the minimum max
value.
```

of these sets in $O((m+k)^{n^2-n})$ time. Then, for each set in the partition we compute the max value for an arbitrary point in it by the $O(m^{n-1} \log m)$ algorithm [20], and finally choose the point with the minimum max value.

Algorithm 3: Minimax plan for general n.

```
Input: Feasible_B(c), P⁻, and the constraints K.
for Each combination of n − 1 points in
  P⁻ ∪ {a × ∞ : a ∈ K} do
    Form a hyperplane and compute the arrangement of
    Feasible_B(c) by the algorithm proposed in [15].
end
for Each set in the arrangement do
    Choose any point in it and compute its max value by the
    algorithm in [20].
end
return the point with the minimum max value.
```

Theorem 6 *If no n features in P^- are on the same hyperplane, then Algorithm 3 computes the minimax plan w.r.t. $\mathcal{W}_K$ in $O((m+k)^{n^2-n} m^{n-1} \log m)$ time (k is the number of constraints in K).*

5. APPROXIMATION ALGORITHMS

We give approximation heuristics for manipulation. We give an LP-heuristic for computing best and worst ranks, which we obtain by formulating a classification problem; and, we give a greedy heuristic for computing a spending plan with an approximation guarantee. Let $\vec{c}$ be the current feature vector of the school whose rank we are manipulating (either by choosing weights $\vec{w}$ or a spending plan $\vec{\Delta}$ to update the features), and let $\vec{c}_i$ be the features of the other schools, which are fixed. We will loosely use $\vec{c}$ and $\vec{c}_i$ to refer both to the school and its features.

5.1 Computing Manipulation by Agency

Let $\vec{x}_i = \vec{c} - \vec{c}_i$ be the feature differences, and define $y_i = +1$ if we wish $\vec{c}$ to be ranked above $\vec{c}_i$ and -1 otherwise. So we want $\vec{w} \cdot \vec{x}_i > 0$ if $y_i = +1$ and $\vec{w} \cdot \vec{x}_i < 0$ if $y_i = -1$. That is, after rescaling $\vec{w}$, we can require that

$$y_i(\vec{w} \cdot \vec{x}_i) > 1.$$

We have here a classification problem where we would like to classify the data $(\vec{x}_i, y_i)$ correctly. If the classifications y_i are not all realizable, we would like to realize as many as possible. By choosing all $y_i = +1$, we have the optimal optimal constructive manipulation problem. By choosing all the y_i=-1, we have the optimal destructive manipulation problem.

The plain vanilla classification problem above is the MFS problem. However, we cannot apply standard MFS heuristics when there are additional constraints on $\vec{w}$, so we proceed to describe

a standard heuristic for the classification problem that can easily accomodate additional linear inequality constraints on $\vec{w}$. A common Linear Programming (LP)-heuristic for the classification problem is to introduce slack variables $\xi_i \geq 0$ and only require $y_i(\vec{w} \cdot \vec{x}_i) > 1 - \xi_i$ while at the same time minimizing the sum of the slacks $\sum_i \xi_i$. This is now a linear program,

$$\min_{\vec{w},\vec{\xi}} \vec{1} \cdot \vec{\xi} \quad \text{s.t.} \quad \begin{array}{l} y_i(\vec{w} \cdot \vec{x}_i) > 1 - \xi_i \\ \xi_i \geq 0 \end{array} \qquad (m-1 \text{ constraints}).$$

The nice thing about this LP-relaxation is that we can add linear inequality constraints on $\vec{w}$ and we still have an LP. Examples of linear inequality constraints which are useful are: $\vec{w} \geq \vec{0}$ (natural for the ranking problem); a relative ordering on the weights according to how important the features are; an interesting constraint is that the current top-k schools remain in the top-k, which is a set of linear inequality constraints $\vec{w} \cdot (\vec{c}_i - \vec{c}_j) \geq 0$ for every $\vec{c}_i$ in the top-k and every $\vec{c}_j$ not in the top-k (in general we can implement a constraint that some set of colleges should be ranked above some other set); bound constraints that the weights cannot deviate too much from a reference set of weights (for example the current weights used by USNews). An interesting constraint which cannot be implemented as linear inequality constraints is that the current top-k must remain in the top-ℓ for some $\ell > k$ (this constraint is not even convex; for such constraints, one has to resort to the ILP).

5.2 Approximating Optimal Spending Plans

We first give a simple approximation heuristic for computing an optimal spending plan and show that it achieves an n-factor approximation guarantee on the budget. We then extend this basic heuristic to a class of iterative ones that have at least the same approximation guarantee but a better performance in practice.

To define the simple heuristic, let $\mathcal{W}$ be the set of allowed ranking weights, and partition $\mathcal{W}$ into n disjoint sets, $\mathcal{W} = \mathcal{W}_1 \cup \mathcal{W}_2 \cup \cdots \cup \mathcal{W}_n$. The set $\mathcal{W}_j$ contains the weights $\vec{w}$ whose jth component w_j is maximum (ties can be broken arbitrarily), so $\vec{w} \in \mathcal{W}_j \implies w_j \geq w_k$, for $k = 1, \ldots, n$. Let r^* be the target rank we are trying to achieve, and B^* the minimum budget required to achieve guaranteed rank r^* using optimal spending plan $\vec{\Delta}^*$.

Fix $j \in \{1, \ldots, n\}$. For any set of weights $\vec{w} \in \mathcal{W}_j$, some school $\vec{c}^*$ (which depends on $\vec{w}$) achieves rank r^* with score $\vec{w} \cdot \vec{c}^*$. Our target school has score $\vec{w} \cdot \vec{c}$. We define the deficiency of our school with respect to the weights $\vec{w} \in \mathcal{W}_j$, denoted $\Delta(\vec{w})$, to be the *minimum* budget our school needs to expend to raise its rank to r^* with respect to just this one $\vec{w}$. A short calculation verifies that

$$\Delta(\vec{w}) = \max\left(0, \frac{\vec{w} \cdot (\vec{c}^* - \vec{c})}{w_j}\right),$$

because the best way to improve the score of $\vec{c}$ w.r.t. $\vec{w}$ is to improve the jth feature (since w_j is maximum) using the update $c_j \leftarrow c_j + \Delta(\vec{w})$, in which case the score of $\vec{c}$ rises to that of $\vec{c}^*$ (with respect to $\vec{w}$). Define the deficiency w.r.t. the set $\mathcal{W}_j$, denoted Δ_j, to be the maximum deficiency with respect to any weight in $\mathcal{W}_j$, so $\Delta_j = \max_{\vec{w} \in \mathcal{W}_j} \Delta(\vec{w})$. If you add the budget Δ_j to component j of $\vec{c}$ (that is $c_j \leftarrow c_j + \Delta_j$), then *every* weight vector in $\mathcal{W}_j$ is 'satisfied' in the sense that the rank of $\vec{c}$ will become at least r^* with respect to every $\vec{w} \in \mathcal{W}_j$. The next two lemmas are offered without proof. The proofs are immediate from the definition of Δ_j and the fact that for all weights in $\mathcal{W}_j$, the maximum component is the jth component.

Lemma 1 *Increasing c_j by Δ_j is the* minimum *budge required to satisfy* all *the weights in $\mathcal{W}_j$.*

Lemma 2 *Construct the spending plan $\vec{\Delta} = [\Delta_1, \Delta_2, \ldots, \Delta_n]$ using a budget $B = \vec{1} \cdot \vec{\Delta} = \sum_{j=1}^n \Delta_j$. Then the updated features $\vec{c} \leftarrow \vec{c} + \vec{\Delta}$ guarantees a rank of r^* or better.*

We now relate the spending plan $\vec{\Delta} = [\Delta_1, \Delta_2, \ldots, \Delta_n]$ with budget $B = \sum_{j=1}^n \Delta_j$ to the optimal spending plan $\vec{\Delta}^*$ with budget B^*.

Theorem 7 *The budget $B \leq nB$.*

So B guarantees a rank of at least r^* and is not more than an n-factor away from the optimal budget that does so. It is possible to show that this approximation bound for the spending plan $\vec{\Delta}$ is tight in that there are examples for which $B = \Omega(n)B^*$.

PROOF. Since B^* guarantees rank r^*, $B^* \geq \Delta_j$ because Δ_j is the *minimum* budget needed to achieve rank r^* among all the weights in $\mathcal{W}_j$. Thus, $B^* \geq \max_j \Delta_j$. The theorem follows from:

$$B = \sum_{j=1}^n \Delta_j \leq n \max_j \Delta_j \leq nB^*.$$

We can offer an immediate improvement on this algorithm. Compute $\Delta_1, \Delta_2, \ldots, \Delta_n$ *sequentially*: first compute Δ_1 and update the first component of $\vec{c}$ by adding Δ_1; now compute Δ_2 with the updated $\vec{c}$, and so on. Since Δ_j is monotonically decreasing in every component of $\vec{c}$, the sequential algorithm can only lower the budget B. The Δ_j are not trivial to compute. Further, making these big steps may be far from the best we can do in practice. So we offer several extensions of this basic heuristic that are of an iterative flavor. Assume that $\vec{c}$ cannot guarantee a rank r^*. This means that there are some weights $\vec{w}$ in one of the $\mathcal{W}_j$ for which $\vec{w} \cdot \vec{c}^* > \vec{w} \cdot \vec{c}$. We call these weights a *witness* to the fact that $\vec{c}$ cannot guarantee rank r^*. All of these iterative heuristics are of the same flavor.

1: If $\vec{c}$ does not guarantee rank r^*, pick *any* witness $\vec{w} \in \mathcal{W}_j$.
2: Update: $c_j \leftarrow c_j + \delta_j$ for *any* $\delta_j \leq \vec{w} \cdot (\vec{c}^* - \vec{c})/w_j$.

The key property of the algorithm above that is that it only updates the jth component of $\vec{c}$ if the witness is in $\mathcal{W}_j$. Modulo this restriction, the algorithm has flexibility to pick *any* witness and also the size of the update. The next theorem justifies all such heuristics.

Theorem 8 *Any iterative algorithm of the form above uses a budget of at most $B = \sum_{j=1}^n \Delta_j$ to guarantee rank r^*.*

PROOF. The proof of this theorem is immediate from the observation that once a budget of Δ_j has been spent on updating the jth component of $\vec{c}$, there will no longer be any witnesses in $\mathcal{W}_j$.

The theorem allows us to freely explore such heuristics to obtain good experimental performance, all the while enjoying the n-factor approximation guarantee.

Heuristic 1. Pick the witness $\vec{w}$ that maximizes the rank of $\vec{c}$ and update by a small increment δ.

Heuristic 2. Let $\vec{w}^{(j)}$ be the *witness* from $\mathcal{W}_j$ (if it exists) that maximizes the rank of $\vec{c}$. Let $w_k^{(j)}$ be the kth component of $\vec{w}^{(j)}$. If you use witness $\vec{w}^{(j)}$, and update $c_j \leftarrow c_j + \delta$, then the sum of the deficiencies will drop by $\delta \sum_k w_j^{(k)}/w_k^{(k)}$ (the sum is over k for which witnesses exist). Pick the witness for which this drop is largest and update by a small increment δ.

Heuristic 3. Let $\vec{w}^{(j)}$ maximize the rank of $\vec{c}$ in $\mathcal{W}_j$. Select those $\vec{w}^{(j)}$ which give the worst rank and use Heuristic 2 to pick one, and update by a small increment δ.

Heuristics 1 & 3 do not explicitly pay attention to the target rank r^*. They can be applied iteratively and will improve the rank as long as the target rank is better than the current worst rank. So, the following interesting theorem holds, which allows one to compute spending plans with all target ranks efficiently and enjoy the approximation guarantee for every spending plan.

Theorem 9 *Heuristics 1 & 3 incrementally update $\vec{c}$ by a sequence of spending plans $\vec{\Delta}_1, \vec{\Delta}_2, \ldots$ achieving rank guarantees $r_1, r_2, \ldots$. For every r that is greater than the starting rank guarantee, let $\vec{\Delta}_{i_r}$ be the spending plan when rank r is first reached. Then, $\vec{1} \cdot \vec{\Delta}_{i_r} \leq nB^*_r$. That is, the budget spent in getting rank guarantee r is an n-approximation to the minimum budget needed.*

Heuristic 1 only solves one maximum-rank problem, whereas Heuristic 3 solves up to n (one for each $\mathcal{W}_j$). So, Heuristic 1 is $O(n)$-times more efficient but we expect Heuristic 3 to use less budget in practice. One can also run Heuristic 2 with target rank 1. In practice, this should work well, but the approximation guarantee does not hold. This is because the budgets achieving intermediate ranks are not necessarily made with respect to witnesses *for that rank*.

6. EXPERIMENTS

Setup. We purchased access to college features on USNews website for their 2015 Best College Ranking. USNews used 17 features. However, USNews was ambiguous about how they normalized scores in some features and deliberately hided the data for "faculty salary" (7%) and "proportion of professors with the highest degree in their fields" (3%), which count for 10% of the total weight. Therefore, in all our experiments we use 15 features whose data are available. For each feature of each college, we first calculate its *Z-score*, which is the deviation from the mean of values of the same feature for all colleges divided by the standard deviation. This is the normalization method used by USNews in their 2011 Best College Ranking [19], but we do not see it mentioned in their 2015 ranking methodology [3]. We then further normalize all feature values to fall into $[-0.5, 0.5]$, which is without loss of generality and is necessary for our MIP (Figure 3). Using the weights provided by USNews and the data described above, we can reconstruct 96% of strict pairwise comparisons in USNews ranking.

Most of the tested algorithms are implemented in Python 2.7.8 and the linear programming solver is GLPK 4.35 vial PuLP interface. The heuristic algorithms in Section 5.1 are implemented and tested in Matlab 2014b. All experiments are tested on a computer with 4 AMD Opteron(TM) Processor 6276 chips that work at 2300 MHz and 504 GB of RAM running CentOS 6.5.

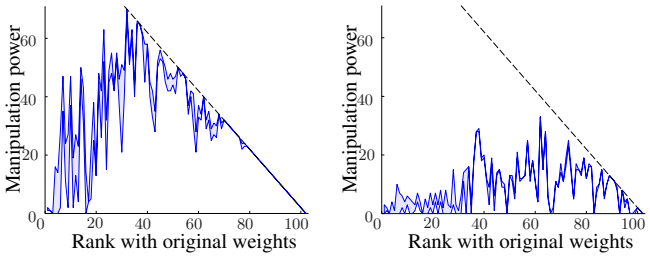
Manipulation by the Ranking Agency. To answer Q1 and Q2 proposed in the Introduction, we run the MIP in Figure 3 on USNews data for the following four natural types of constraints:
1. PosWeight: the only constraint is that all weights are nonnegative. This is required in all experiments.
2. HYP Top-10: Harvard, Yale, Princeton must rank in the top-10.
3. Weight-order: The ranking over the weights must be the same as the ranking over the weights used by US News.
4. Weight-range: weights cannot differ from the original USNews weights by more than 10%, and no single weight exceeds 20%.

We have tested our MIP for all top 102 colleges, and our results are shown in Table 1 (top 10) and Table 2 (40th–47th). We observe that the bottleneck for our MIP is CPU rather than the memory. The average running time for the constructive and destructive manipulation for the 102 colleges is the following:

There a number of interesting findings. With only the PosWeight constraint the range is often very large, for example the worst case rank for Yale is 68, and UCSB has a rank range $[1, 102]$. This is mainly because the value of some feature vectors are not positively correlated with USNews ranking, especially the "Graduate rate Performance Prediction". With the constraint that Harvard, Yale, Princeton must be ranked in top 10, the ranges still look quite large and arbitrary. However, such results are still sometimes very informative. For example, we can say "there is no way RPI can be ranked within top 20 if all features stay the same", and "as long as Harvard, Yale, Princeton are ranked within top-10, UIUC cannot get in top 15 if all features stay the same". Similar observations hold for the Weight-range constraints. The most interesting observation is that for the Weight-order constraints, the ranges are more or less consistent with US News ranking. This suggest that *as long as we use weights whose order is the same as that of US News, we are not too far away from USNews ranking, despite the numbers themselves are different*.

PosWeights	HYP in Top-10	Weight-order	Weight-range
3.16 sec	265.66 sec	1.30 sec	2.03 sec

Table 3: The average running time of the MIP for 102 colleges.



(a)Top-3 stays in top-3 (b) Fixed weight ordering.

Figure 3: Agency's manipulation potential. (a) Top-3 remains in the Top-3 weight constraint. (b) Relative importance of the weights is fixed by the current USNews weights. We show the agency manipulation power for destructive manipulation. The shading illustrates the difference between the exact ILP and the LP-heuristic. The dotted black line shows the maximum possible manipulation.

LP-Classification Heuristic for the Agency. We compare the LP-heuristic with the exact ILP for computing the manipulation power. The results are shown in Figure 3 for destructive manipulation (the results for constructive manipulation are similar). As can be observed from the figure, the size of the shaded region is quite narrow. This indicates only small differences on average between the LP-heuristic and the exact ILP. Comparing Figure 3(a) with (b) we see that the types of weights that USNews would commit to makes a big difference in its manipulation power. Committing to the relative importance of features, but not the exact weights of each feature results in significantly less manipulation ability than a constraint of the form "the top-3 schools will remain in the top-10".

If USNews commits to an ordering for the importance of the features and uses transparent meaningful features then the concern over agency manipulation can be significantly mitigated.

Manipulation by the Ranked Agents. We first use a 2 dimensional dataset from the 15 dimensional USNews by separating into a subjective feature (Peer assessment score (15%), High school counselor score (7.5%), and over/under-performance (7.5%)) and an objective feature, which is the weighted average of the other features w.r.t. their USNews weights. We chose these *two* features

	Princeton	Harvard	Yale	Columbia	Stanford	U. Chicago	MIT	Duke	U. Penn	Caltech
USNews	1	2	3	4	4	4	7	8	8	10
PosWeight	[1, 46]	[1, 45]	[1, 68]	[1, 45]	[1, 51]	[1, 65]	[1, 82]	[1, 65]	[3, 83]	[1, 79]
HYP Top-10	[1, 10]	[1, 10]	[1, 10]	[1, 26]	[1, 33]	[1, 49]	[1, 69]	[1, 49]	[3, 51]	[1, 73]
Weight-order	[1, 4]	[1, 4]	[1, 3]	[3, 8]	[3, 6]	[3, 16]	[5, 14]	[8, 14]	[6, 13]	[5, 16]
Weight-range	[1, 11]	[1, 10]	[1, 15]	[1, 14]	[1, 11]	[1, 44]	[1, 46]	[1, 37]	[3, 39]	[1, 52]

Table 1: Constructive and destructive manipulation for top-10 colleges in USNews 2015 ranking.

	Lehigh	UCSB	Boston U.	NorthEastern	RPI	UCIrvine	UIUC	UWMadison
USNews	40	40	42	42	42	42	42	47
PosWeight	[22, 99]	[1, 86]	[26, 94]	[2, 82]	[21, 97]	[1, 102]	[1, 101]	[21, 99]
HYP Top-10	[23, 99]	[1, 86]	[27, 91]	[2, 81]	[23, 97]	[5, 101]	[19, 101]	[22, 99]
Weight-order	[36, 60]	[35, 53]	[37, 51]	[32, 62]	[36, 50]	[35, 59]	[36, 61]	[30, 58]
Weight-range	[28, 87]	[12, 73]	[30, 82]	[19, 72]	[26, 86]	[18, 96]	[18, 90]	[26, 88]

Table 2: Constructive and destructive manipulation for 40th-47th colleges in USNews 2015 ranking.

because it is an intuitive separation of the 15 features and for two features we have an efficient algorithm to compute the minimax spending plan, so we can test our proposed heuristic algorithms.

Results for Heuristics 1 & 2, and Algorithm 1 on the 2D dataset are in Figure 4, for three colleges that are ranked at 14th, 44th, and 102nd by USNews with alphabetical tie-breaking. The x-axis is the target rank and the y-axis is the minimum total budget to guarantee that rank for the unit cost function for both features. For example, for college 14 to guarantee the first place, it needs at least 0.3 total improvement in both features. Both heuristic algorithms and Algorithm 1 run in seconds for each college. Surprisingly, the optimal budget computed by both heuristic algorithms are very close to the optimal solution despite the guaranteed 2-approximation ratio.

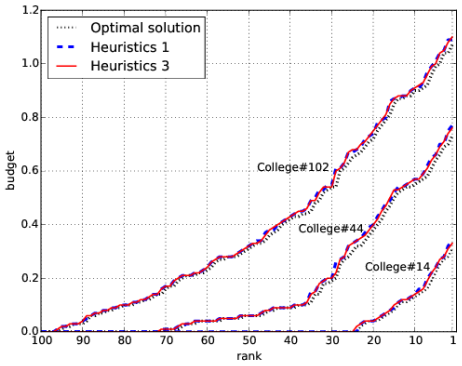


Figure 4: Heuristics 1 & 3 and optimal for 2D data.

We have also tested Heuristics 1 & 3 on the 15 dimensional USNews dataset under the same Weight-range constraint in our previous experiments, for which the exact algorithm (Algorithm 3) is impractical due to high complexity. The results are in Figure 5. Surprisingly, we observe that the outcomes of the two heuristic algorithms are very close. We conjecture that they are also very close to the optimal solution as for $n = 2$, despite the guaranteed approximation ratio of 15. The running time of the two heuristic algorithms for colleges ranked at $\{4, 14, 24, ..., 94, 101\}$ are shown in Figure 6. We note that Heuristic 1 is much more computationally efficient than Heuristic 3. Due to their close performance in Figure 5, we believe that Heuristic 1 is a good algorithm in practice.

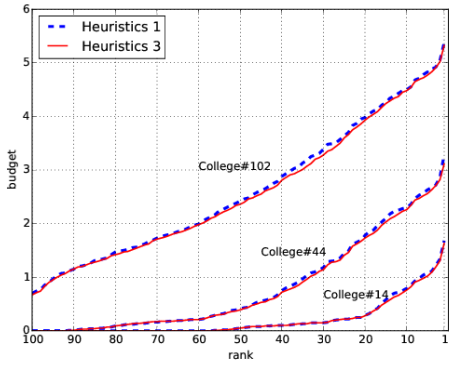


Figure 5: Heuristics 1 & 3 for 15D USNews data.

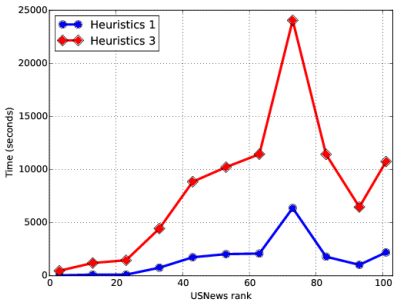


Figure 6: Running time of Heuristics & 3 for 15D USNews data.

7. FUTURE WORK

There are many interesting open questions for future research. Can we improve the worst-case $O(n)$ guarantee for some of the proposed heuristic algorithms? How to efficiently compute a spending plan when the budget constraint is not linear? Can we efficiently compute a spending plan for other notions of optimality, e.g. the minimax regret? How should we model multiple colleges strategically improving their features as a game and what are the equilibria of such games? What if there are multiple ranking agencies? Experimentally we plan to apply our methodologies to other ranking systems including Shanghai ranking, QS ranking, etc.

REFERENCES

[1] http://en.wikipedia.org/wiki/Google_bomb.

[2] http://www.arc.gov.au/pdf/FT14/FT14\%20Funding\%20Rules.PDF, 2014.

[3] http://www.usnews.com/education/best-colleges/articles/2014/09/08/how-us-news-calculated-the-2015-best-colleges-rankings, September 2015.

[4] S. Adali, T. Liu, and M. Magdon-Ismail. An analysis of optimal link bombs. *Theoretical Computer Science*, 437:1–20, 2012.

[5] E. Amaldi, P. Belotti, and R. Hauser. Randomized Relaxation Methods for the Maximum Feasible Subsystem Problem. In *Integer Programming and Combinatorial Optimization*, volume 3509 of *Lecture Notes in Computer Science*, pages 249–264, 2005.

[6] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1–2):181–210, 1995.

[7] J. Bartholdi, III, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[8] J. Bartholdi, III, C. Tovey, and M. Trick. How hard is it to control an election? *Math. Comput. Modelling*, 16(8-9):27–40, 1992. Formal theories of politics, II.

[9] D. Baumeister, M. Roos, J. Rothe, L. Schend, and L. Xia. The possible winner problem with uncertain weights. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, pages 133–138, 2012.

[10] L. Chao. China Shutters 6,600 Websites for Manipulating Information Online. http://blogs.wsj.com/chinarealtime/2011/08/29/china-shutters-6600-websites-for-manipulating-information-online/, August 2011.

[11] J. W. Chinneck. Computer Codes for the Analysis of Infeasible Linear Programs. *The Journal of the Operational Research Society*, 47(1):61–72, 1996.

[12] J. W. Chinneck. Fast Heuristics for the Maximum Feasible Subsystem Problem. *INFORMS Journal on Computing*, 13(3):210–223, 2001.

[13] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 82–90, Ann Arbor, MI, USA, 2006.

[14] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th World Wide Web Conference*, pages 613–622, 2001.

[15] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing Arrangements of Lines and Hyperplanes with Applications. *SIAM Journal on Computing*, 15(2):341–363, 1986.

[16] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Using complexity to protect elections. *Communications of the ACM*, 53:74–82, 2010.

[17] P. Faliszewski and A. D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[18] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: the anatomy of a recommender system. In *Proceedings of the third annual conference on Autonomous Agents*, pages 434–435, 1999.

[19] S. L. Gnolek, V. T. Falciano, and R. W. Kuncl. Modeling Change and Variation in U.S. News & World Report College Rankings: What would it really take to be in the Top 20? *Research in Higher Education*, 55(8):761–779, 2014.

[20] D. S. Johnson and F. P. Preparata. The densest hemisphere problem. *Theoretical Computer Science*, 6(1):93–107, 1978.

[21] M. Kutner. How to Game the College Rankings. http://www.bostonmagazine.com/news/article/2014/08/26/how-northeastern-gamed-the-college-rankings/, September 2014.

[22] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[23] R. Pérez-Peña and D. E. Slotnik. Gaming the College Rankings. http://www.nytimes.com/2012/02/01/education/gaming-the-college-rankings.html, January 2012.

[24] M. E. Pfetsch. Branch-and-Cut for the Maximum Feasible Subsystem Problem. *SIAM Journal on Optimization*, 19(1):21–38, 2007.

[25] J. Rothe and L. Schend. Control Complexity in Bucklin, Fallback, and Plurality Voting: An Experimental Approach. *Experimental Algorithms*, 7276:356–368, 2012.

[26] V. Strauss. Why U.S. News college rankings shouldn't matter to anyone. http://www.washingtonpost.com/blogs/answer-sheet/wp/2013/09/10/why-u-s-news-college-rankings-shouldnt-matter-to-anyone/, September 2013.