Picking Winners Using Integer Programming

David Scott Hunter
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, dshunter@mit.edu

Juan Pablo Vielma
Department of Operations Research, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, jvielma@mit.edu

Tauhid Zaman
Department of Operations Management, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, zlisto@mit.edu

We consider the problem of selecting a portfolio of entries of fixed cardinality for a winner take all contest such that the probability of at least one entry winning is maximized. This framework is very general and can be used to model a variety of problems, such as movie studios selecting movies to produce, drug companies choosing drugs to develop, or venture capital firms picking start-up companies in which to invest. We model this as a combinatorial optimization problem with a submodular objective function, which is the probability of winning. We then show that the objective function can be approximated using only pairwise marginal probabilities of the entries winning when there is a certain structure on their joint distribution. We consider a model where the entries are jointly Gaussian random variables and present a closed form approximation to the objective function. We then consider a model where the entries are given by sums of constrained resources and present a greedy integer programming formulation to construct the entries. Our formulation uses three principles to construct entries: maximize the expected score of an entry, lower bound its variance, and upper bound its correlation with previously constructed entries. To demonstrate the effectiveness of our greedy integer programming formulation, we apply it to daily fantasy sports contests that have top heavy payoff structures (i.e. most of the winnings go to the top ranked entries). We find that the entries produced by our approach perform well in practice and are even able to come in first place in contests with thousands of entries. Our approach can easily be extended to other problems with a winner take all type of payoff structure.

Key words: combinatorial optimization, statistics, integer programming, sports analytics
History:

1. Introduction

Imagine one must select a portfolio of entries for a winner-take-all type of contest where all or nearly all of the winnings go to the top performing entry. Because of the top-heavy payoff structure, one natural strategy is to select the entries in such a way that maximizes the probability that at least one of them has exceptional performance and wins the contest. We refer to this as the *picking winners* problem. This framework can be used to model a variety of problems which have a winner-take-all payoff structure. For

instance, pharmaceutical companies want to select a set of drugs in which they invest research resources such that the probability at least one of the drugs is a major success is maximized. Or consider the challenge faced by a venture capital fund. It will invest in a set of start-up companies and if one of the companies becomes a huge success, the fund will earn a huge return. For another example, consider a movie studio that has to select a set of movies to produce and success is achieved if one of the movies is a blockbuster hit. This framework can also model daily fantasy sports contests where people enter lineups of players in a sport (such as hockey or baseball) into daily contests where the top scoring entry wins a huge amount of money. All of these examples require one to select a set of entries (drugs, start-up companies, movies, lineups of players) to maximize the probability that at least one of them has exceptional performance and wins.

To select the entries one requires some predictive model of how well the entries will perform. However, this information alone will not be sufficient to select a good set of entries. The goal is to maximize the probability that at least one entry wins. One may select entries which have a high probability of individually winning. However, if these entries have a high positive correlation, then the probability that at least one of them wins may not be as large as for a set of entries with slightly lower winning probabilities but also lower correlation so they cover more possible outcomes. Therefore, choosing a good set of entries would require information on their joint distribution in addition to their marginal probabilities of winning. Intuitively, a good set of entries will each have a high probability of winning, but also not be too positively correlated in order to diversify the set of entries. What one needs to do is understand the tradeoff between the winning probability of the entries and their correlations in order to select a good set on entries.

There can also be scenarios where the entries are not simply chosen from some fixed set. Instead, they must be constructed from a set of constrained resources. For instance, a movie studio must hire actors for each movie it wishes to produce, and the salary of the actors cannot exceed the movie's budget. Or the studio may want certain types of actors for the movie (a child actor, an older male actor, etc). As another example, in daily fantasy sports contests, one must select players for a lineup, but the players must satisfy budget constraints and position constraints imposed by the company running the contests. For instance, in hockey contests, a lineup can only have one goalie. In these types of situations one must develop a method to construct the entries in a manner which respects the constraints but still maximizes the probability of winning.

The problem of picking winners is very similar to a variety of covering problems. One can view our objective as choosing M entries to a contest so that we cover the most likely winners. We show that in fact under a certain probability distribution on the entries, picking winners reduces to the well known maximum coverage problem where the goal is to pick M subsets of an underlying set such that the union of the subsets has maximum coverage of the underlying set. Because of this fact, a variety of other covering problems are very similar to the picking winners problem. There is the problem of placing a fixed number of sensors in a

region to detect some signal as fast as possible. This has been studied for water networks (Leskovec et al. 2007) and general spatial regions (Guestrin et al. 2005). There is also a problem in information retrieval of selecting a set of documents from a large corpus to cover a topic of interest (Agrawal et al. 2009, Chen and Karger 2006, El-Arini et al. 2009, Swaminathan et al. 2009). Another related problem is selecting a set of users in a social network to seed with a piece of information in order to maximize how many people the message reaches under a social diffusion process (Kempe et al. 2003). Another way to view this problem is that one wants the message to reach a certain target user in the network, but one does not know precisely who this user is, so one tries to cover as much of the network as possible. One can view the sensors, documents, or seed users as entries to the winner-take-all contest. From this perspective, winning means one of the sensors detects the signal, one of the documents covers the relevant topic, or one of the seed users is able to spread the message to the unknown target user.

The similarity between picking winners and these covering problems has important implications. The objective functions in these covering problems are submodular and maximizing them is known to be NP-hard (Karp 1975). The same holds for the picking winners problem. The submodular property is important because it means that one has a lower bound on the performance of a greedy solution (Nemhauser et al. 1978). This fact, plus the simplicity with which one can calculate greedy solutions, motivates us to use a greedy approach to pick winning entries. The similarities between picking winners and these covering problems suggest that we could directly use a greedy solution to pick the entries. However there are some key differences between the problems which complicate this approach. In the covering problems, at each iteration of the greedy approach one simply chooses the entry (a subset, sensor, document, or seed user) from a ground set which maximizes the marginal gain in the objective function. However, we will focus on the situation where the entries must be constructed from constrained resources and cannot simply be chosen from a ground set. Our problem is closely related to the problem of maximizing a stochastic monotone submodular function with respect to a matroid constraint (Asadpour and Nazerzadeh 2015). However in our case the objective function is the probability of at least one entry winning, which many times lacks a closed form expression. Therefore, we cannot even evaluate the exact objective function for a set of entries, which will prevent us from directly applying the greedy approach.

To overcome these challenges, we develop a greedy integer programming formulation for constructing entries from constrained resources for winner-take-all type contests. Our approach is fairly general and can be applied to a variety of problems. To demonstrate the effectiveness of our greedy integer programming formulation, we use it to construct entries for daily fantasy sports competitions. We were able to perform exceptionally well in these contests using our greedy integer programming formulation across multiple sports. To provide more context for our results, we next present a brief overview of the structure of daily fantasy sports contests.

1.1. Daily Fantasy Sports Contests

Daily fantasy sports have recently become a huge industry. For instance, DraftKings, the leading company in this area, reported $30 million in revenues and $304 million in entry fees in 2014 (Heitner 2015). While there is ongoing legal debate as to whether or not daily fantasy sports contests are gambling (Board 2015), there is a strong element of skill required to continuously win contests as evidenced by the fact that the top one percent of players pay 40 percent of the entry fees and gain 91 percent of the profits (Harwell 2015). This space is very competitive, with many users employing advanced analytics to pick winning lineups. Several sites such as Rotogrinders have been created which solely focus on daily fantasy sports analytics (Rotogrinders 2015). Other sites such as Daily Fantasy Nerd even offer to provide optimized contest lineups for a fee (DailyFantasyNerd 2015).

There are a variety of contests in daily fantasy sports with different payoff structures. Some contests distribute all the entry fees equally among all lineups that score above the median score of the contest. These are known as double-up or 50/50 contests. They are relatively easy to win, but do not pay out very much. Other contests give a disproportionately high fraction of the entry fees to the top scoring lineups. We refer to these as *top heavy* contests. Winning a top heavy contest is much more difficult than winning a double-up or a 50/50 contest, but the payoff is also much greater. Typical entry fees range from $3 to $30, while the potential winnings range from thousands to a million dollars (DraftKings 2015). These contests occur nearly everyday and have a near winner-take-all payoff structure. For these reasons, daily fantasy sports provide a great opportunity for us to test our approach to picking winners.

The entry to a daily fantasy sports contest is called a lineup and consists of a set of players in the given sport. One can view the lineups, which are the entries, as being constructed from a set of resources, which are the players. An added complication is the fact that there are constraints on the resources used to build a lineup. A daily fantasy sports company sets a price (in virtual dollars) for each player in a sport. These prices vary day to day and are based upon past and predicted performance (Steinberg 2015). Lineups must then be constructed within a fixed budget. For instance, in DraftKings hockey contests, there is a $50,000 budget for each lineup. There are further constraints on the maximum number of each type of player allowed and the number of different teams represented in a lineup. Finding an optimal lineup is challenging because the performance of each player is unknown and there are hundreds of players to choose from. However, predictions for player points can be found on a host of sites such as Rotogrinders (Rotogrinders 2015) or Daily Fantasy Nerd (DailyFantasyNerd 2015) which are fairly accurate, and many fantasy sports players have a good sense about how players will perform. This suggests that the main challenge is not in predicting player performance, but in using predictions and other information to construct the lineups. In a previous study (Becker and Sun 2014) a mixed integer program is presented which optimizes draft selection and weekly lineup selection for season long fantasy football. This illustrates that optimization can be a useful tool for constructing lineups in fantasy sports. However, that study focuses on season long fantasy contests,

which does not necessarily coincide with the problem of picking winners in daily fantasy contests which we study in this work.

1.2. Our Contributions

Integer programming is a natural tool to use to construct entries because of the different constraints. The challenge comes in understanding how to formulate the integer program to maximize the probability of winning. In this work we present a greedy integer programming formulation that attempts to do this. We begin by finding a tractable approximation for the objective function (the probability of winning) and show that the error of this approximation is small when the underlying distribution of the entries satisfies certain natural properties. From this approximation we are able to extract a set of heuristic principles for constructing entries. First, the entries' scores should have a large expected value and variance. This increases the marginal probability of an entry winning. Second, the entries should also have low correlation with each other to make sure they cover a large number of possible outcomes. These principles form the foundation of our greedy integer programming formulation which sequentially constructs entries with maximal mean subject to constraints on their variance and correlation with previously constructed lineups.

We applied our approach to top heavy hockey and baseball contests in DraftKings. The contests we focused on had thousands of entrants and top payoffs of several thousand dollars. The entries constructed with our greedy integer programming formulation were able to rank in the top ten in these contests multiple times, which is a non-trivial achievement given the large size of the contests. Our performance in hockey contests was especially strong. During the period when this work was conducted, DraftKings allowed a single user to enter 200 lineups into a hockey contest. We began competing with our integer programming approach and performed very well several days in a row. We even placed third or higher in three consecutive contests. We show screenshots of our performance with 200 lineups in Figure 1. A week after we began winning with our integer programming approach, DraftKings reduced the number of multiple entries to 100 in order to make the contests more fair for all participants. This demonstrates the power of our greedy integer programming formulation.

The remainder of the paper is structured as follows. In Section 2 we present the picking winners problem and develop a greedy integer programming formulation for the problem. We present an empirical analysis of hockey data and develop predictive models of player performance in Section 3 for use in our integer programming formulation. In Section 4 we specialize our integer programming formulation for daily fantasy sports contests, with a focus on hockey. We evaluate the performance of our approach in actual fantasy hockey and baseball contests in Sections 5 and 6, respectively. We conclude in Section 7. All proofs can be found in the appendix.

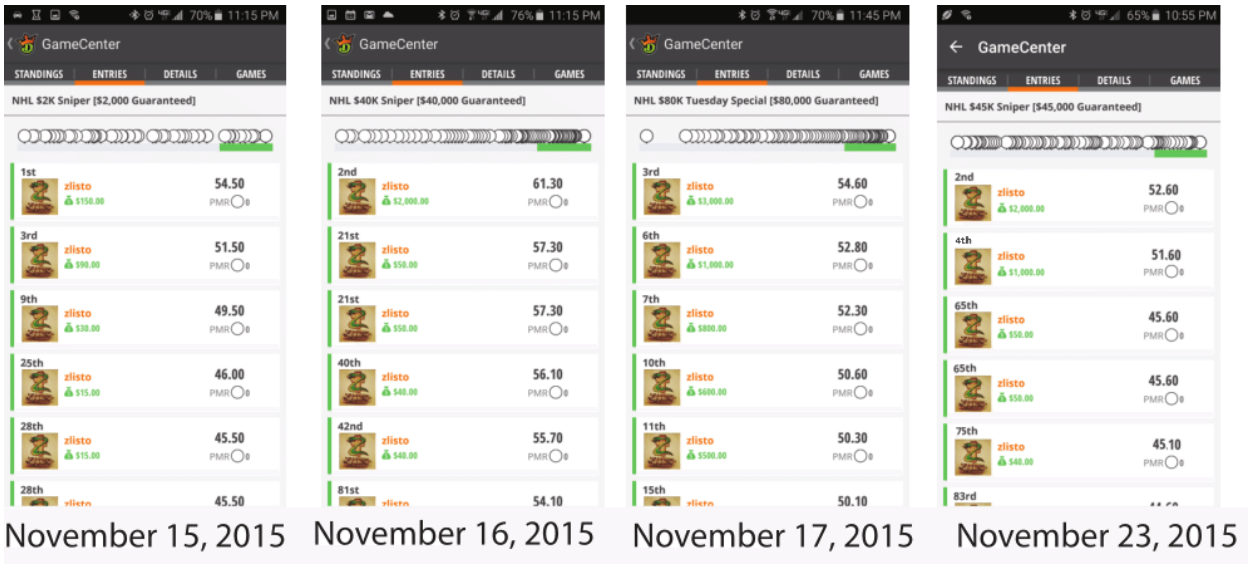

Figure 1 **Screenshots of the performance of our top integer programming lineups in real DraftKings hockey contests with 200 linueps entered.**

2. Problem Formulation

We consider a set of events $\mathcal{E} = \{E_i\}_{i=1}^N$ defined on a probability space $(\Omega, \mathcal{F}, \mathbf{P})$, with sample space Ω, σ-algebra $\mathcal{F}$, and probability measure $\mathbf{P}$. Each event E_i corresponds to an entry in a contest winning. The aim is to select a subset of events $\mathcal{S} \subseteq \mathcal{E}$ of size M such that we maximize the probability that at least one event occurs. For notational convenience, we will denote $U(\mathcal{S}) = \mathbf{P}\left(\bigcup_{E_i \in \mathcal{S}} E_i\right)$ as the probability that at least one event occurs, i.e. at least one entry wins. The optimization problem we want to solve is

$$\max_{\mathcal{S} \subseteq \mathcal{E}, |\mathcal{S}| = M} U(\mathcal{S}) \tag{2.1}$$

In general, solving this optimization problem can be difficult. In fact, we have the following result.

Theorem 2.1 *Maximizing $U(\mathcal{S})$ is NP-hard.*

Despite the computational complexity in maximizing $U(\mathcal{S})$, the objective function possesses some nice properties which we can use to obtain good solutions efficiently, which are given by the following theorem.

Theorem 2.2 *The function $U(\mathcal{S})$ is non-negative, non-decreasing and submodular.*

Monotonically non-decreasing submodular functions have an important property which is that one can bound the sub-optimality of a greedy solution. Such a solution is obtained by sequentially adding elements to the set $\mathcal{S}$ such that each new addition results in the largest marginal increase in the objective function. Formally, let $\mathcal{S}_g = (F_1, F_2, ..., F_M)$ be the greedy solution of the optimization problem. The greedy solution is an ordered set, with element F_i being added on the ith step of the greedy optimization. Let $S_g^i =$

$(F_1, F_2, ..., F_i)$ be the solution obtained after the ith element is added to S_g, with $\mathcal{S}_g^0 = \emptyset$. Then the ith event added to S_g is given by

$$F_i = \arg \max_{F \in \mathcal{E}/S_g^{i-1}} U\left(S_g^{i-1} \bigcup F\right) - U\left(S_g^i\right), \quad 1 \leq i \leq M. \tag{2.2}$$

For submodular functions, we have the following well known result.

Theorem 2.3 (Nemhauser et al. (1978)) *Let $U(\mathcal{S})$ be a non-decreasing submodular function. Let $\mathcal{S}_g$ be the greedy maximizer of $U(\mathcal{S})$ and let $\mathcal{S}^*$ be the maximizer of $U(\mathcal{S})$. Then*

$$\frac{U(\mathcal{S}_g)}{U(\mathcal{S}^*)} \geq 1 - e^{-1}. \tag{2.3}$$

Therefore, greedy solutions have performance guarantees for submodular maximization problems. In addition, many times they can be computed very quickly, which is important as time can be a limiting factor in many applications. For these reasons, we will use a greedy approach for the picking winners problem.

2.1. Independent Events

We first consider the simplest case where all E_i are independent and let $p_i = \mathbf{P}(E_i)$. Let E_i^c denote the complement of E_i. Because of the independence of the events, we can rewrite the objective function as

$$\begin{aligned} U(\mathcal{S}) &= 1 - \mathbf{P}\left(\bigcap_{E_i \in S} E_i^c\right) \\ &= 1 - \prod_{E_i \in \mathcal{S}} (1 - p_i). \end{aligned} \tag{2.4}$$

The optimization problem can then be written as

$$\max_{\mathcal{S} \subseteq \mathcal{E}, |\mathcal{S}| = K} U(\mathcal{S}) = \min_{\mathcal{S} \subseteq \mathcal{E}, |\mathcal{S}| = K} \prod_{E_i \in S} (1 - p_i). \tag{2.5}$$

The form of the objective function for independent events leads to a simple solution, given by the following theorem.

Theorem 2.4 *For a set of independent events $\mathcal{E} = \{E_1, E_2, ..., E_N\}$ let $p_i = \mathbf{P}(E_i)$ and without loss of generality assume that $p_1 \geq p_2 \geq ... \geq p_N$. Let the M element subset $\mathcal{S}^* \subset \mathcal{E}$ maximize $U(\mathcal{S})$. Then $\mathcal{S}^*$ consists of the M most likely events, i.e. $\mathcal{S}^* = \{E_1, E_2, ..., E_M\}$.*

The proof of this result is obvious from the form of $U(\mathcal{S})$ in equation (2.4). From Theorem 2.4 we see that if the events are independent, then one simply chooses the most likely events in order to maximize the probability that at least one event occurs. This simple solution relies crucially upon the independence of the events. Another useful property when the events are independent concerns the greedy solution.

Lemma 1 *For a set of independent events $\mathcal{E} = \{E_1, E_2, ..., E_N\}$, let the M element greedy maximizer of $U(\mathcal{S})$ be $\mathcal{S}_g$. Then $\mathcal{S}^* = \mathcal{S}_g$.*

In this case we have that the greedy solution is also an optimal solution. Indeed, independent events is an almost trivial case to solve. This also suggests when the events have low dependence, greedy solutions should perform well.

2.2. Dependent Events

When events are not independent, a simple, exact solution is generally not achievable. In fact, it can be non-trivial to even evaluate the objective function for dependent events. In this section we provide a tractable approximation to the objective function when the events are dependent and characterize the quality of the approximation as a function of properties of the underlying distribution. We will see that when the events are not too dependent upon each other, our approximation will have low error.

We start with an approximation for the objective function. Recall that our objective function is defined as $U(\mathcal{S}) = \mathbf{P}\left(\bigcup_{E\in\mathcal{S}} E\right)$. The difficulty of optimizing this objective function comes from the lack of independence in the events which require us to understand their entire joint distribution. Many times this distribution will not be available to us, or if it is, evaluating the required probability will be difficult. To obtain something more tractable, we define the following objective function which only utilizes pairwise correlations:

$$U_2(\mathcal{S}) = \sum_{E\in\mathcal{S}} \mathbf{P}(E) - \frac{1}{2} \sum_{E,F\in\mathcal{S}, E\neq F} \mathbf{P}\left(E \bigcap F\right). \tag{2.6}$$

This objective function is obtained by expanding $U(\mathcal{S})$ using the inclusion-exclusion principle and then only keeping terms which involve the intersection of two or fewer events. The benefit of $U_2(\mathcal{S})$ is that it only requires us to know the probability of intersections of pairs of events, which is many times much easier to calculate than probabilities of higher order intersections. In order for this objective function to act as a good surrogate for $U(\mathcal{S})$, we must show that the difference $U(\mathcal{S}) - U_2(\mathcal{S})$ is small. If we impose certain conditions on the joint density of the events, we can bound the difference. To do this we define a few new terms.

We assume we have chosen a subset $\mathcal{S}$ of the events. Let $p_0(\mathcal{S}) = \mathbf{P}(\bigcap_{E\in\mathcal{S}} E^c)$ be the probability that no event in $\mathcal{S}$ occurs. For $1 \leq l \leq M$, let $\mathcal{S}_l$ be the set of all l-element subsets of $\mathcal{S}$. For instance, if $\mathcal{S} = \{a, b, c\}$ then $\mathcal{S}_1 = \{\{a\}, \{b\}, \{c\}\}$ and $\mathcal{S}_2 = \{\{a, b\}, \{a, c\}, \{b, c\}\}$. For an element $T \in \mathcal{S}_l$, let $p'_T = \mathbf{P}\left(\left(\bigcap_{E\in T} E\right) \bigcap \left(\bigcap_{F\in\mathcal{S}/T} F^c\right)\right)$ be the probability that all events in T occur but no other events in $\mathcal{S}$ occur. We then have the following result regarding the difference between $U(\mathcal{S})$ and $U_2(\mathcal{S})$.

Theorem 2.5 *Let $\mathcal{E}$ be a set of events on a probability space and assume $|\mathcal{E}| = N$. Assume there are constants $0 < p < 1$, $\delta > 0$, and $c_1 > 0$ such that for any $\mathcal{S} \subset \mathcal{E}$ with $|\mathcal{S}| = M < N$, $p_0(\mathcal{S}) = (1-p)^M$, $pM < N^{-\delta} < 1/3$, and for any $1 \leq l \leq M$ and $T \in S_l$, $p'_T \leq c_1 p^l$. Then*

$$\frac{U(\mathcal{S}) - U_2(\mathcal{S})}{U(\mathcal{S})} \leq 24 c_1 N^{-\delta}. \tag{2.7}$$

This result imposes two conditions on the joint distribution of the events. First, if N is large, the probability of no event in $\mathcal{S}$ occurring is large. This is not an unreasonable condition because we are typically dealing with a large set of events, each of which we expect to have a low probability of occurring. Second, there is an upper bound on the dependence between all events. For instance, if the events are independent and have the same probability of occurring, then the condition is satisfied with $c_1 = 1$. More generally, the condition says that the joint probability of l events must decay exponentially in l. This limits the joint dependence of the events. Under these two conditions, $U_2(\mathcal{S})$ will be a good approximation to $U(\mathcal{S})$. We will use $U_2(\mathcal{S})$ instead of $U(\mathcal{S})$ in the remainder of this work because $U_2(\mathcal{S})$ is much easier to work with than $U(\mathcal{S})$, and it is a good approximation to $U(\mathcal{S})$.

2.3. Multivariate Gaussian Random Variables

We now consider a probability model that captures a fairly general class of events. Let $\{X_i\}_{i=1}^N$ be jointly normal random variables with mean vector μ and covariance matrix Σ. We define the events $E_i = \{X_i \geq t\}$ for some constant t. If we imagine each X_i as the return of a company we invest in, then the event says that company i has a return greater than t. For fantasy sports competitions, X_i can be the points of lineup i and E_i corresponds to the lineup exceeding t points. We further assume that $t > \max_{1 \leq i \leq N} \mu_i$ because we are interested in the unlikely event that one of the X_i exceed their mean by a large amount. For instance, in daily fantasy hockey contests, the maximum mean of a lineup is around 30 points, while the winning threshold t is near 60 points.

We want to choose a set of events $\mathcal{S} \subseteq \mathcal{E}$ to maximize $U_2(\mathcal{S})$. To do this, we must first calculate the marginal and joint probabilities of the events. Since no closed form expression exists for these probabilities for multivariate Gaussian random variables, we instead resort to approximations. We have the following result.

Theorem 2.6 *Let $\{X_i\}_{i=1}^N$ be jointly normal random variables. Let the marginal mean and variance of X_i be μ_i and σ_i^2 and let the correlation coefficient of X_i and X_j be ρ_{ij}. Also, let $z_i = (t - \mu_i)/\sigma_i$. Define the function $U_2^l(\mathcal{S})$ as*

$$U_2^l(\mathcal{S}) = \sum_{i:E_i \in \mathcal{S}} \frac{1}{z_i + 1/z_i} \exp\left(-\frac{z_i^2}{2}\right) - \frac{1}{2} \sum_{i,j:E_i,E_j \in \mathcal{S}, i \neq j} \exp\left(-\frac{(2t - \mu_i - \mu_j)^2}{2\left(\sigma_i^2 + \sigma_j^2 + \sqrt{(\sigma_i^2 - \sigma_j^2)^2 + 4\rho_{ij}\sigma_i^2\sigma_j^2}\right)}\right). \tag{2.8}$$

Then $U_2(\mathcal{S}) \geq U_2^l(\mathcal{S})$.

The function $U_2^l(\mathcal{S})$ is a lower bound on $U_2(\mathcal{S})$ and has a closed form expression. We use this function to gain insights on good entries. From the expression for $U_2^l(\mathcal{S})$ we observe three important features that

characterize a subset $\mathcal{S}$ which will have a high chance of winning. We assume that $z_i \geq 1$ which can be done by having t be large enough. In this case, the contribution of the first summation to $U(\mathcal{S})$ increases for large μ_i and large σ_i. The contribution of the second summation increases for small ρ_{ij}. There is an intuitive explanation for how μ_i, σ_i, and ρ_{ij} impact the objective function. First, the means of the chosen entries should be large so that there is a greater probability that they exceed t. Second, the variance of the entries should be large as well. This is less obvious, but the intuition here is that all one cares about is having some entry's score exceed t. If entries with large variances are chosen, this increases the probability of exceeding t. While this also increases the probability that some of the entries will have a very low value, this is not a concern as the goal is to have at least one exceed t. Third, the correlation between each pair of entries should be small. What this does is diversify the selected entries so there is a better chance that one of them exceeds t. In short, one would like $\mathcal{S}$ to consist of entries with high means and high variance which have low correlation with each other.

2.4. Picking Winners with Constrained Resources

We now consider a case where entries are constructed from a set of resources $\{Y_k\}_{k=1}^P$ which are jointly Gaussian random variables. We define the mean and standard deviation of Y_k as μ'_k and σ'_k, respectively. We define the correlation coefficient of Y_k and Y_l as ρ'_{kl}. Each entry is given by the sum of K resources. Let us define x_{ik} as variables which are one if resource k is in entry i and zero otherwise. Then we can write $X_i = \sum_{k=1}^P Y_k x_{ik}$. We assume that there are certain constraints on the resources such that not every possible subset of K resources is a feasible entry. In this scenario, we want to construct M feasible entries from the given resources to maximize the probability of winning. To do this we will develop an integer programming formulation which builds upon the intuitions gained in Section 2.3. The three insights we saw there for constructing a set of winning entries were to make sure each entry had a high mean, a high variance, and low correlation with other entries. We can write the mean of X_i as

$$\mu_i = \sum_{k=1}^{P} \mu'_k x_{ik}. \tag{2.9}$$

The variance of X_i is

$$\sigma_i^2 = \sum_{k=1}^{P} (\sigma'_k)^2 x_{ik} + \sum_{k=1}^{P} \sum_{l=1, l\neq k}^{P} \rho'_{kl} \sigma'_k \sigma'_l x_{ik} x_{il}, \tag{2.10}$$

and the covariance of X_i and X_j is

$$\text{Cov}(X_i, X_j) = \sum_{k=1}^{P} (\sigma'_k)^2 x_{ik} x_{jk} + \sum_{k=1}^{P} \sum_{l=1, l\neq k}^{P} \rho'_{kl} \sigma'_k \sigma'_l x_{ik} x_{jl}, \tag{2.11}$$

With these expressions we want to understand how to construct a set of entries that will have a high probability of winning. To achieve a high mean, each entry should be chosen to maximize the sum of its resources'

means. This insight is fairly obvious and very natural. The more interesting insights come from the variances and covariances of the entries. To obtain a more intuitive understanding of what entries have good properties with respect to their variance and covariance, we set the variance of each resource equal to one and constrain the correlation coefficient to be either zero, δ, or $-\delta$ for some $\delta > 0$. We can then characterize the variance of an entry by the number of pairs of resources that have a positive or negative correlation and the covariance of a pair of entries by the number of their shared resources and the number of pairs of resources with positive and negative correlations. Let us define for X_i the number of positively and negatively pairs of correlated resources as n_i^+ and n_i^-. Let us also define for X_i and X_j the number of positively and negatively pairs of correlated resources as n_{ij}^+ and n_{ij}^- and the number of shared resources as n_{ij}. We will also refer to n_{ij} as the *overlap* of entries X_i and X_j because this is the number of resources they have in common. With this notation, the variances and covariances can be written as

$$\sigma_i^2 = K + \delta\left(n_i^+ - n_i^-\right) \tag{2.12}$$

$$\text{Cov}(X_i, X_j) = n_{ij} + \delta\left(n_{ij}^+ - n_{ij}^-\right) \tag{2.13}$$

From these expressions we gain some key insights concerning the variances and covariances. A high variance can be achieved if the entry has a large n_i^+ and a small n_i^-. That is, the entry's resources should be chosen so many of them are positively correlated and few of them are negatively correlated. By avoiding negative correlations, we are making it more likely that either the resources all have a high value or all have a low value. The covariance of a pair of entries can be reduced in two ways. First, reduce the overlap n_{ij}. If the entries share fewer resources, they will be clearly be less correlated. Second, increase the number of pairs of resources between the lineups that have a negative correlation and decrease the number of pairs of resources with positive correlation. Limiting the number of positively correlated resources between lineups reduces their covariance and subsequent correlation coefficient.

2.5. Greedy Integer Programming Formulation with Constrained Resources

Constructing a set of entries that optimizes the $U_2^l(\mathcal{S})$ is highly non-trivial. The function is non-linear and non-convex. Therefore, we will not directly maximize $U_2^l(\mathcal{S})$. Instead, we will use the three properties discussed about $U_2^l(\mathcal{S})$ to guide the development of a greedy integer programming formulation for constructing a set of entries from constrained resources which have a high probability of winning. We choose a greedy approach for three important reasons. First, from the submodular property of the original objective function we have a lower bound on the performance of a greedy solution. The objective function we will use in our integer programming formulation is different from the original objective function, but shares some of its key properties, so we assume the loss in optimality with our greedy approach will not be significant. Second, the greedy integer programming formulation is linear and can be solved very quickly, which will be important in applications we consider later. Third, and most importantly, we will show in Sections 5 and 6 that this

greedy integer programming formulation performs very well on actual data. In particular, we will see that with this approach we are able to consistently perform well daily fantasy sports contests.

We consider the situation described in Section 2.4 where we want to construct M entries consisting of K resources each from a set of P resources. As previously stated, our decision variable is x_{ik} which is 1 if resource k is in entry i. Let $\{x^*_{ik}\}_{k=1}^P$ correspond to the ith greedy entry. This is the solution to our greedy integer programming formulation in the ith iteration. We will construct the entries sequentially in the following manner. We choose a set of positive constants $\{\epsilon_i\}_{i=1}^M$ and $\{\gamma_{ij}\}_{i,j=1}^M$. Assume we have obtained the first $i-1$ entries. We then construct the ith entry to maximize its mean subject to its variance being larger than ϵ_i and its covariance with entry j being less than γ_{ij} for $1 \le j \le i-1$. For $2 \le i \le M$, our greedy integer programming formulation for the ith entry is

$$\begin{aligned}
\underset{x}{\text{maximize}} \quad & \sum_{k=1}^{P} \mu'_k x_{ik} \\
\text{subject to} \quad & \sum_{k=1}^{P} (\sigma'_k)^2 x_{ik} + \sum_{k=1}^{P} \sum_{l=1, l\neq k}^{P} \rho'_{kl}\sigma'_k\sigma'_l x_{ik} x_{il} \ge \epsilon_i \\
& \sum_{k=1}^{P} (\sigma'_k)^2 x_{ik} x^*_{jk} + \sum_{k=1}^{P} \sum_{l=1, l\neq k}^{P} \rho'_{kl}\sigma'_k\sigma'_l x_{ik} x^*_{jl} \le \gamma_{ij}, \quad 1 \le j \le i-1 \\
& \sum_{k=1}^{P} x_{ik} = K
\end{aligned} \tag{2.14}$$

For the first entry we do not have the covariance constraint and our greedy integer programming formulation is simply

$$\begin{aligned}
\underset{x}{\text{maximize}} \quad & \sum_{k=1}^{P} \mu'_k x_{1k} \\
\text{subject to} \quad & \sum_{k=1}^{P} (\sigma'_k)^2 x_{1k} + \sum_{k=1}^{P} \sum_{l=1, l\neq k}^{P} \rho'_{kl}\sigma'_k\sigma'_l x_{1k} x^*_{1l} \ge \epsilon_1 \\
& \sum_{k=1}^{P} x_{1k} = K
\end{aligned} \tag{2.15}$$

The greedy approach only constrains the covariance of a lineup with the previous lineups, whereas a non-greedy approach would limit all pairwise covariances. By only constraining the covariance with previous lineups, the number of overlap constraints is linear in the number of previous entries. The variance constraints are non-linear in this formulation. We can introduce auxiliary variables to make this formulation linear as is standard in many integer programming formulations. However, we take a different approach when we apply this approach to fantasy sports contests. We will make certain assumptions on the underlying distribution of the resources which allow us to makes these constraints linear. We will see that this allows the integer program to be solved very quickly and still perform well in real applications.

Our greedy integer programming formulation gives one the flexibility to choose the parameters ϵ_i and γ_{ij}. We may want the first few entries' covariances to be lower in order to force exploration of the space of entries, while for later entries this constraint can be relaxed to avoid excluding entries with large means. However, we will see that using a single value for γ_{ij} and for ϵ_i works well in several applications. In addition, we will also see that the choice of the value for γ_{ij} depends upon the size of the space of resources.

3. Hockey Data Analysis

To demonstrate the effectiveness of our greedy integer programming formulation for picking winners, we apply it to top heavy daily fantasy sports hockey contests as described in Section 1. The entries into these contests are known as lineups and consist of resources which are players in the National Hockey League (NHL), which is the main hockey league in North America. The daily fantasy contests we focus on are from the site DraftKings. We begin by describing the basic elements of hockey and hockey daily fantasy sports contests. To apply our greedy integer programming formulation, we need to know the mean and correlation of the resources, which are the hockey players. In this section we present models and analysis for these inputs. Section 3.2 presents predictive models for players points using publicly available data and Section 3.3 studies the structural correlations between hockey players. While our analysis focuses on DraftKings, we note that other daily fantasy sports sites have very similar contests and point scoring systems. All data in this section was collected between October 21, 2015 and December 31, 2015.

3.1. Hockey Basics

A hockey team consists of six players: one center, two wingers, two defensemen, and a goalie. The centers, wingers, and defensemen are known as skaters because they are free to skate on the ice rink, whereas the goalie primarily stays in front of the goal. In hockey the aim is for a team to shoot a puck into the goal of the opposing team and the goalie's task is to protect the goal from any opponent scores.

In daily fantasy sports hockey contests there are constraints on the types of players that must be present in a lineup. For DraftKings, each lineup must consist of two centers, three wingers, two defensemen, one goalie, and one utility player which is any skater, resulting in a lineup of nine players. We show a screenshot of one possible DraftKings lineup in Figure 2. The figure also shows a column labeled FPPG, which stands for fantasy points per game. This is the average number of fantasy points per game the player has earned throughout the current NHL season. DraftKings provides this information to help users construct their lineups. The total fantasy points for a lineup is the sum of each player's fantasy points. The lineup shown has 34.4 fantasy points per game. This score is in the typical range of NHL lineups in DraftKings and gives a sense of the value of different scoring actions. In a daily fantasy sports contest, there is a specific way in which players score fantasy points. Because we test our approach in DraftKings, we will focus on its points scoring system[1]. We summarize the details of the DraftKings scoring system in Table 1. In Figure 2 we

[1] Full scoring rules can be found at www.draftkings.com/rules.

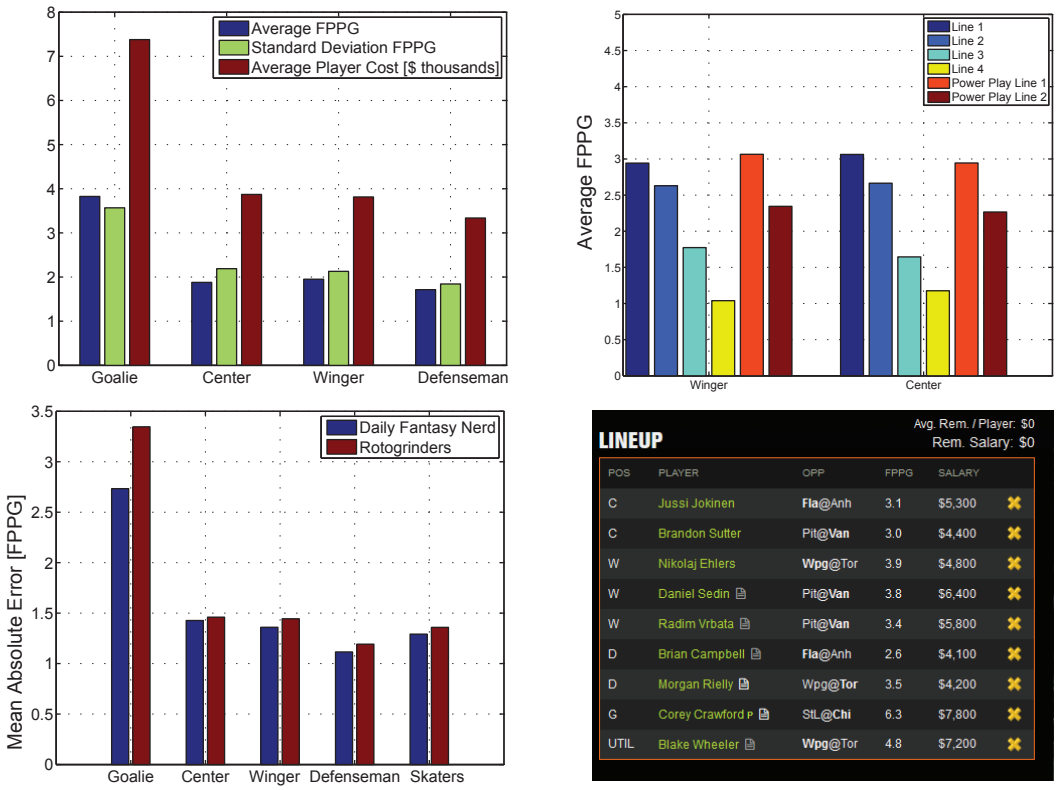

Figure 2 **(top left) Mean and standard deviation of fantasy points per game (FPPG) and average player cost in DraftKings for each hockey position. (bottom left) Mean absolute error of NHL player FPPG predictions of Daily Fantasy Nerd and Rotogrinders for each hockey position. (top right) Mean FFPG of offensive skaters in different types of lines. (bottom right) Screenhsot of an NHL lineup from DraftKings.**

show the basic statistics of points per game for each position averaged over all NHL players. We also show the average DraftKings cost per player in the figure. As can be seen, on average goalies tend to score almost twice as many points as all other players and also cost twice as much. Also, the standard deviation of each position's points per game is roughly the same size as the average value.

The centers and wingers are known as offensive players. In the NHL each hockey team plays its offensive players in fixed sets known as *lines*. This means that if one player from the line, such as the center, is on the ice, then so are the other two players from the line. Each team has four standard lines and two power play lines, which only play when there is a power play as a result of a penalty. We show the average points per game for players based upon their line in Figure 2. The rank of the line typically suggests how well it performs. As can be seen, line one is generally the highest scoring line.

The points system of DraftKings is very similar to most daily fantasy hockey points systems. It gives more points for players that perform better, but it also has some important properties which we use to construct our lineups. The first property concerns assists, which is passing the puck to a player who either

Score type	Points
Goal	3
Assist	2
Shot on Goal	0.5
Blocked Shot	0.5
Short Handed Point Bonus (Goal/Assist)	1
Shootout Goal	0.2
Hat Trick Bonus	1.5
Win (goalie only)	3
Save (goalie only)	0.2
Goal allowed (goalie only)	-1
Shutout Bonus (goalie only)	2

Table 1 Points system for NHL contests in DraftKings.

scores or who passes the puck to the player that scores. Each assist is worth two points and DraftKings awards points for up to two assists per goal. This is an important aspect of the scoring system because it means that for each goal scored, it is possible for a single lineup to receive seven points (three points for the goal, and four points for the two assists that led to the goal). However, such a situation can only occur if all players involved are on a single lineup. This suggests that to score many points, a lineup should contain all players from a single line. By putting all players from a single line in a lineup, we are essentially increasing the variance of the lineup. This is a popular fantasy sports tactic known as *stacking*. Many expert daily fantasy sports players use stacking to win in a variety of sports contests (Drewby 2015), (Bales 2015b), (Bales 2015a). The second property of the points system concerns goalies. If a goalie's team wins the game, an additional three points is awarded to the goalie. This is important because as we will see, it is possible to predict who will win an NHL hockey game with relatively good accuracy, so by choosing goalies on winning teams, an extra three points can be obtained, which is a significant amount in hockey contests given that lineups average between 30 to 40 points.

3.2. Player Point Predictions

Many fantasy sports players devote a substantial amount of effort in developing elaborate prediction models for players' performance. Their advantage comes mainly from the power of their predictive models. However, we do not have the domain knowledge required to improve upon the prediction models of the top fantasy players. Therefore, we take a different approach. Publicly available player point predictions are provided by the websites Rotogrinders (Rotogrinders 2015) and Daily Fantasy Nerd (DailyFantasyNerd 2015). We find the predictions from these sites to be relatively accurate, but the accuracy varies by position. We plot in Figure 2 the mean absolute error in points of the two sites' predictions for different hockey positions using our data from the 2015 NHL season. We see that the prediction errors of both sites are very similar. Therefore, it does not seem that either one has a superior prediction model.

We build a predictive model for the fantasy points for each player in the NHL using these prediction sites. For a player k, let f_{k1} and f_{k2} be the predictions by Rotogrinders and Daily Fantasy Nerd, respectively. For

	(1) Skater points	(2) Goalie points	(3) Goalie points	(4) Goalie points	(5) Goalie points	(6) Goalie points
β_1 (Rotogrinders)	0.634*** (0.0203)	0.628*** (0.0864)	0.203 (0.144)		0.203 (0.144)	
β_2 (Daily Fantasy Nerd)	0.282*** (0.0242)	-0.0173 (0.106)	-0.0309 (0.113)	-0.0301 (0.113)		
β_3 (Win probability)			4.310* (2.123)	5.304** (2.005)	4.176* (2.064)	5.172** (1.941)
β_0	1.334 (0.0314)	1.686 (0.549)	(1.032)	(1.003)	(1.013)	(0.983)
R^2	0.238	0.0858	0.0179	0.0140	0.0178	0.0139
Samples	10825	565	506	506	506	506

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 2 Regression coefficients, R squared, and sample count for different prediction models of skater and goalie points.

skaters, we perform a linear regression on the player's points using these predictions as features. We denote μ_k as the predicted points of player k using our model. For skater k our prediction is

$$\mu_k = \beta_0 + \beta_1 f_{1k} + \beta_2 f_{2k}. \tag{3.1}$$

For goalies, we must also factor in the outcome of the game since there is a three point bonus for victory. Daily Fantasy Nerd provides win probabilities for each goalie, and we have found these estimates to be relatively accurate. We perform a linear regression on the goalie points using as predictor variables the predictions of Rotogrinders, Daily Fantasy Nerd, and the win probability of the goalie's team. For a goalie k, let this win probability by p_k. Then the prediction for goalie k is

$$\mu_k = \beta_0 + \beta_1 f_{1k} + \beta_2 f_{2k} + \beta_3 p_k. \tag{3.2}$$

We compare different prediction models in Table 2. For skaters, both Rotogrinders and Daily Fantasy Nerd predictions are significant. For the skaters, Rotogrinders and Daily Fantasy Nerd predictions are both significant. For goalies, something different occurs. When we do not include the win probability we find that only Rotogrinders is significant. When the win probability is included, it is the only significant feature. We will test these different models on real hockey contests in Section 5.

3.3. Player Correlations

To create high variance lineups we must first understand the correlations between players. There are two important correlations we look at. The first concerns players in lines. The second concerns offensive players and goalies.

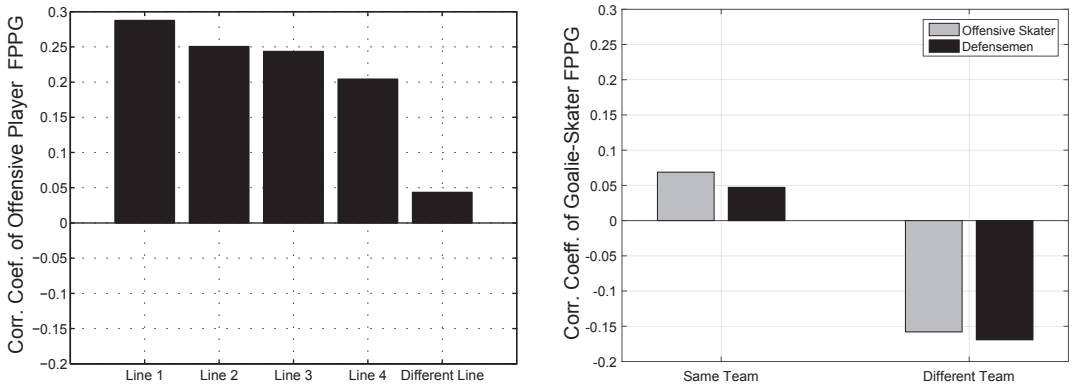

Figure 3 **(left) Average correlation coefficient of FPPG for offensive players in the same and different lines. (right) Average correlation coefficient of FPPG for goalies and skaters in the same and opposing teams.**

We first begin by examining the correlations of FPPG between players in the same line versus those in different lines. We calculate the Pearson correlation coefficient between the FPPG for every pair of NHL offensive players (centers and wingers) in the same line and in the same team but different line. We plot the resulting average correlation coefficients for each group in Figure 3. We see that the players on the same line have a much higher correlation than players on different lines. This is not surprising given the way fantasy points are awarded. This correlation is most likely due to the assist/goal combinations that give points to multiple players who are simultaneously on the ice. This correlation should be included in a lineup to increase the variance.

We next look at the correlation between goalies and skaters in the same game. We consider the correlation of FPPG for offensive players and defensemen with goalies on the same and opposing team in a game. We plot the resulting average Pearson correlation coefficient of the FPPG for these groups in Figure 3. We find that the correlation is negative for goalies and skaters on opposing teams. For the same team the correlation is near zero. The negative correlation is due to the fact that if a skater scores fantasy points, it is typically because a goal was scored, which gives negative fantasy points to the goalie. On the same team the only structural condition that would correlate skaters and goalies is the win bonus and points for scoring goals. However, this is much weaker than the negative correlation of goalies opposing skaters. The negative correlation is something we want to avoid within a lineup as it would reduce the overall variance. Therefore, to increase lineup variance, we will avoid putting goalies and skaters on opposing teams in a single lineup.

4. Fantasy Hockey Greedy Integer Programming Formulations

We now present our greedy integer programming approach for selecting multiple lineups for hockey contests. The integer programming is based upon the formulation in Section 2.5. For the mean points of a player

we user our predictive regression models. We do not explicitly use the correlations and variances of the player's points. Instead we assume that the players all have the same variance and that the magnitude of the covariance is much smaller than the variance. This will lead to simple constraints for the variance and covariance. We will construct M lineups from a set of P NHL players and we let the variable x_{ik} equal one if player k is selected for lineup i and zero otherwise. Each player k is predicted to achieve μ_k fantasy points from our predictive model.

4.1. Feasibility Constraints

We begin by formulating the basic feasibility constraints of a lineup. Daily fantasy sports contests all impose a budget B constraint on a lineup. For DraftKings, this budget is $B = 50,000$ fantasy dollars. There is also a constraint on the number of players of each type of position. In DraftKings, a hockey lineup consists of nine players: two centers, three wingers, two defensemen, a goalie, and a utility player who can be any skater type. We let the cost of player k be c_k. There are N_T NHL teams playing in a contest, and we denote the set of players on team l by T_l. Also, we denote the set of players who are centers, wingers, defenseman, and goalies by C, W, D, and G, respectively. The DraftKings position constraints for the ith lineup are given by

$$\begin{aligned} 2 \leq \sum_{k \in C} x_{ik} \leq 3, \\ 3 \leq \sum_{k \in W} x_{ik} \leq 4, \\ 2 \leq \sum_{k \in D} x_{ik} \leq 3, \\ \sum_{k \in G} x_{ik} = 1, \end{aligned} \tag{4.1}$$

DraftKings also requires each lineup to have players that come from at least three different NHL teams. The team constraints for lineup i are

$$\begin{aligned} & t_{il} \leq \sum_{k \in T_l} x_{ik}, \quad 1 \leq l \leq N_T \\ & \sum_{l=1}^{N_T} t_{il} \geq 3, \\ & t_{il} \in \{0,\, 1\}, \quad 1 \leq l \leq N_T. \end{aligned} \tag{4.2}$$

We put these constraints together to obtain the feasibility constraints for a lineup:

$$\begin{aligned} & \sum_{k=1}^{P} c_k x_{ik} \leq B, \quad \text{(budget constraint)} \\ & \sum_{k=1}^{P} x_{ik} = 9, \quad \text{(lineup size constraint)} \\ & \text{Equations (4.1)}, \quad \text{(position constraint)} \\ & \text{Equations (4.2)}, \quad \text{(team constraint)} \\ & x_{ik} \in \{0,1\}, \quad 1 \leq k \leq P. \end{aligned} \tag{4.3}$$

4.2. Stacking Lineups Using Structural Correlations

We now present the integer programming constraints on the lineup variance. The constraints we use are simpler than the general form in equation (2.14). Rather than force the variance of the lineup to be larger than a lower bound, we constrain the structure of the lineup to ensure that its variance will be large. We assume equal variance of all players and covariances that are much smaller in magnitude than the variances. This allows us to approximate the variance of a lineup using only the number of pairs of positively and negatively correlated players. The constraints we present next, which we refer to as different types of stacking, exclude any negatively correlated players and try to have a large number of positively correlated players.

Goalie Stacking. As discussed in Section 3.3, goalies are negatively correlated with the skaters that they are opposing. Therefore, to increase the variance of a lineup, we would like to avoid having this negative correlation by making sure all of the lineup's skaters are not opposing its goalie. We will refer to this constraint as *goalie stacking*. To model this constraint in our integer programming formulation, we denote the set of skaters who are opposing goalie k by O_k. Due to the team constraint given by (4.2), the maximum number of players we can have in one lineup from a given NHL team is six. For lineup i and goalie k, the goalie stacking constraint is

$$6x_{ik} + \sum_{l \in O_k} x_{il} \leq 6, \quad \forall k \in G. \tag{4.4}$$

This constraint will force the goalie variable x_{ik} to be zero if the lineup has any skater opposing goalie k.

Line Stacking. Additionally, in Section 3.3 we found that players on the same line are positively correlated. Therefore, to increase the lineup variance, we would like to impose additional constraints on our formulation that will create lineups with players on the same line. We will refer to this as *line stacking*. There are eight possible skaters on a team and a line consists of three skaters. Therefore, the maximum number of lines we can include in a lineup is two. To include as much positive correlation as possible amongst the players, we can make a lineup to have one complete line of three players, and two partial lines with at least two players each. To formulate the line stacking constraints, we introduce some notation. Let N_L denote the total number of lines, and let L_l denote the set of players on line l. With this notation we can formulate the constraint of having at least one complete line in lineup i as

$$\begin{aligned} 3v_{il} &\leq \sum_{k \in L_l} x_{ik}, \quad 1 \leq l \leq N_L \\ \sum_{l=1}^{N_L} v_{il} &\geq 1 \\ v_{il} &\in \{0,1\}, \quad 1 \leq l \leq N_L. \end{aligned} \tag{4.5}$$

Under these constraints v_{il} is one if lineup i has all three players from line l, and zero otherwise, and we require at least one of the v_{il} to be one. To formulate the constraint of having at least two players from two distinct lines, we use the following constraints:

$$\begin{aligned} 2w_{il} &\leq \sum_{k \in L_l} x_{ik}, \quad 1 \leq l \leq N_L \\ \sum_{l=1}^{N_L} w_{il} &\geq 2 \\ w_{il} &\in \{0,1\}, \quad 1 \leq l \leq N_L. \end{aligned} \tag{4.6}$$

Similar to the complete line constraints, this constraint sets w_{il} to one if at least two out of three players of line l are in lineup i, and there must be at least two w_{il} equal to one.

Defensemen Stacking. Lastly, we have found that defensemen that are on the first power play line score substantially higher than defensemen that are not. Therefore, to increase a lineup's average points, one could choose defensemen that are on the first power play line. We will refer to this as *defensemen stacking*. To model this constraint, let P_D denote the set of defensemen that are on any team's first power play line. Then, to use only these defensemen in lineup i we add the following constraints to our integer programming formulation:

$$x_{ik} = 0 \quad \forall k \in D \setminus P_D. \tag{4.7}$$

This constraint makes sure that any defenseman that is chosen for the lineup is also in the first power play line of some team.

There are many other constraints that could be added to the model, such as forcing a lineup to have every player on a team's first power play, imposing constraints to have seven players from one game, etc. We did not investigate all the possible types of stacking, but they can all be naturally incorporated into our integer programming formulation. In this work we focus on the constraints presented because they capture the major structural correlations in hockey.

4.3. Overlap Constraints

Our greedy integer programming formulation in Section 2.5 had a lower bound on the covariance of lineup i with the previous $i-1$ lineups. We assume that the player variance is much larger in magnitude than the player correlations. Using this assumption, we approximate the correlation of lineups i and j as simply the overlap of the two lineups. We can then add constraints which lower bound this overlap with all previous $i-1$ lineups. We will refer to these as overlap constraints. We let x^*_{lk} be the optimal value of the variable for player k in lineup l. We define the overlap constraints for lineup i as

$$\sum_{k=1}^{P} x^*_{lk} x_{ik} \leq \gamma, \quad l = 1, \ldots, i-1. \tag{4.8}$$

These constraints make sure that lineup i will have no more than γ players in common with any previously constructed lineup. We refer to γ as the maximum lineup overlap. Decreasing γ forces the lineups to be more diverse, whereas for larger γ the lineups will share more players. We will see in Section 5 how the choice of γ is impacted by the number of NHL games being played on a given night.

4.4. Complete Greedy Integer Programming Formulation

We now present the final greedy integer programming formulation for constructing M lineups. We obtain lineup i ($1 \leq i \leq M$) by solving

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{j=1}^{N} \mu_j x_{ij} \\ \text{subject to} \quad & \text{Equation (4.3)} \quad \text{(feasibility constraints)} \\ & \text{Equation (4.8)} \quad \text{(multiple lineup constraints)} \\ & \text{Any subset of equations (4.4), (4.5), (4.6), (4.7)} \quad \text{(stacking constraints).} \end{aligned} \tag{4.9}$$

By solving this integer programming formulation iteratively M times, each time updating the multiple lineup constraints with the previously found solutions, one obtains the M greedy lineups. This formulation allows flexibility in the choice of stacking constraints, prediction model, maximum lineup overlap, and number of linueps constructed. We will see next how to select the best settings using data from real fantasy hockey contests.

5. Performance

We now evaluate the performance of our greedy integer programming formulation on real daily fantasy sports hockey contests. To obtain data on the performance of the population lineups in several actual DraftKings hockey contests we needed to enter the contests. As entrants, we are able to download the lineups and points of every lineup in the contest. In total we have data for 38 different top heavy contests which contain several thousand entrants. We evaluate how much profit our lineups would have made in each contest competing against the actual population lineups. We investigate a variety of settings, including the number of lineups, type of stacking, prediction model, and maximum overlap.

We combine the different constraints from Section 4.2 to create six different types of stacking which are summarized in Table 3. The first type is no stacking, which only uses the basic constraints and none of the stacking constraints. All the other types of stacking use goalie stacking (equation (4.4)) as this is a very natural negative correlation we wish to avoid. Also, all the other types of stacking require at least one complete line (equation (4.5)), and at least two partial lines (equation (4.6)). Only two defensemen are allowed in Type 2, the logic being that the utility player should be an offensive skater that will score more points, as was seen in Figure 2. Types 3 and 4 use defensemen stacking (equation (4.7)). Types 4 and 5 require exactly three different teams to be represented on a lineup. This constraint increases the lineup variance by having more players on the same team in the lineup.

No stacking	Type 1	Type 2	Type 3	Type 4	Type 5
$\mathcal{C}_k$	$\mathcal{C}_k$	$\mathcal{C}_k$	$\mathcal{C}_k$	$\mathcal{C}_k$	$\mathcal{C}_k$
	Equations (4.4)	Equations (4.4)	Equations (4.4)	Equations (4.4)	Equations (4.4)
	Equations (4.5)	Equations (4.5)	Equations (4.5)	Equations (4.5)	Equations (4.5)
	Equations (4.6)	Equations (4.6)	Equations (4.6)	Equations (4.6)	Equations (4.6)
		$\sum_{j \in D} x_{ij} = 2$	Equations (4.7)	Equations (4.7)	$\sum_{k=1}^{N_T} t_{ik} = 3$
				$\sum_{k=1}^{N_T} t_{ik} = 3$	$\sum_{k \in T_l} x_{ik} \leq 6t_{il},$
				$\sum_{k \in T_l} x_{ik} \leq 6t_{il},$	$1 \leq l \leq N_T$
				$1 \leq l \leq N_T$	

Table 3 Summary of the constraints for different types of stacking in our integer programming formulation for constructing hockey lineups.

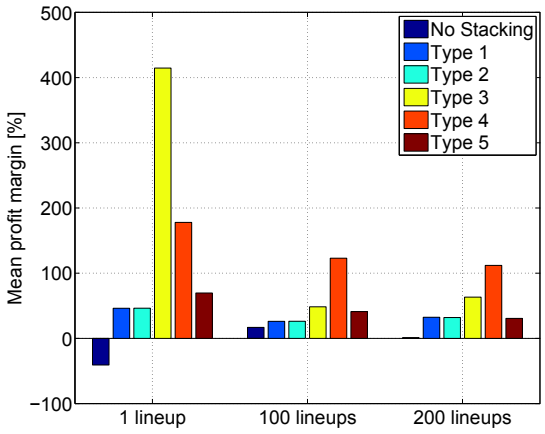

Figure 4 Plot of the mean profit margin in DraftKings hockey contests versus the number of integer programming lineups. The plots are for different types of stacking constraints. The maximum lineup overlap allowed is seven.

5.1. Impact of Stacking and Number of Lineups

We begin by investigating the performance of each type of stacking versus the number of lineups entered. We create different numbers of lineups for each type of stacking with a maximum lineup overlap of seven and calculate the mean profit margin across all contests. We plot the results in Figure 4. We observe that the lowest profit margin is achieved by no stacking. This shows the importance of stacking lineups to increase lineup variance. Type 4 stacking achieves the highest profit margin for 100 and 200 lineups. For one lineup, Type 3 stacking has a higher mean profit margin due to a single contest where the profit margin is 124%. Excluding this single outlier, Type 4 stacking has the highest mean profit margin for one lineup. Recall that Type 4 stacking includes goalie stacking, line stacking (complete and partial), defensemen stacking, and at most three different teams. It has more stacking than all other types, which we expect to give its lineups

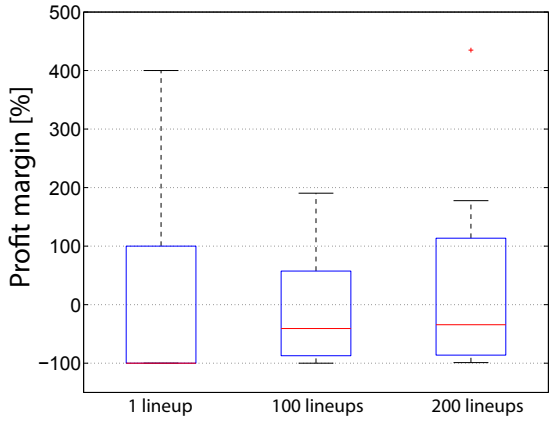

Figure 5 **Boxplot of the profit margin in DraftKings hockey contests versus the number of integer programming lineups. The lineups use Type 4 stacking and a maximum lineup overlap of seven.**

more variance. We see from the data that this maximizing variance strategy is effective for winning top heavy contests.

It seems from Figure 4 that the mean profit margin is largest for one lineup. However, this is a deceiving statistic because the mean is dictated essentially by a single outlier for these top heavy contests. A more accurate picture is provided by the boxplot in Figure 5. Here we show the profit margin for Type 4 stacking with a maximum lineup overlap of seven. We see that with one lineup, the median profit margin is -100%. However, with more lineups, the median profit margin becomes larger. In addition, the 75th percentile is larger for 200 lineups than for 100 lineups. This shows that with more lineups the distribution of the profit margin is shifted upward and given a slightly fatter tail. Therefore, to win more money consistently, it is advantageous to use more lineups.

5.2. Impact of Prediction Models

We next investigate the impact of different prediction models on our performance. We consider Type 4 stacking, a maximum lineup overlap of seven, and 200 lineups. The prediction models we consider use Rotogrinders predictions, Daily Fantasy Nerd predictions, both sites' predictions, and both sites plus the win probability. We show in Figure 6 the mean profit margin of different prediction models for different numbers of integer programming lineups. We see here that the model with all three features does the best or near the best for each number of lineups. The reason here may be that by including the win probability, the integer programming lineups are more biased towards goalies expected to win the games, resulting in a better chance of obtaining the game win bonus of three points.

5.3. Impact of Maximum Lineup Overlap

One parameter we can adjust in our model is the maximum lineup overlap. By decreasing the allowed overlap, we force the lineups to be more diverse. We find that the degree of diversity we want depends

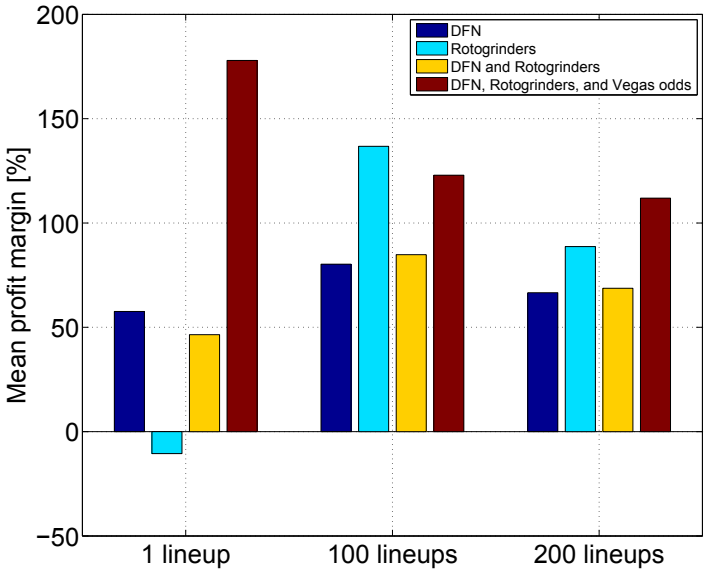

Figure 6 **Plot of the mean profit margin of our integer programming approach in DraftKings hockey contests with different prediction models. The models considered are Daily Fantasy Nerd predictions (DFN), Rotogrinders predictions, DFN and Rotogrinders predictions, and DFN and Rotogrinders predictions with Vegas odds. The maximum lineup overlap allowed is seven and constraint type four is used for the integer programming lineups.**

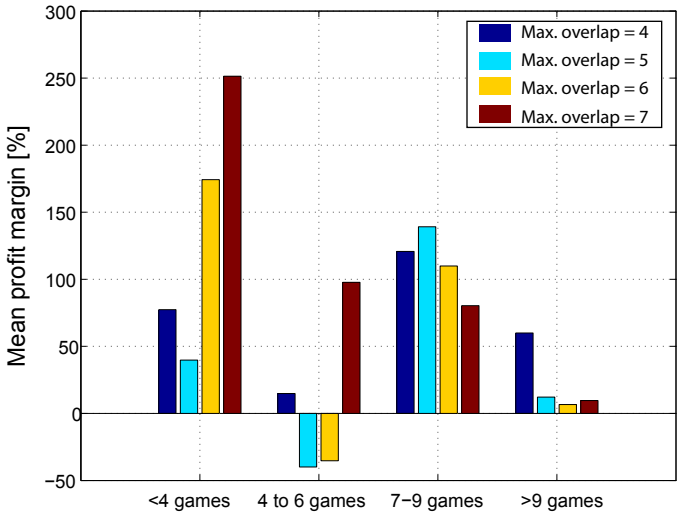

Figure 7 **Plot of the mean profit margin 200 integer programming lineups with Type 4 stacking in DraftKings hockey contests versus maximum lineup overlap for different number of NHL hockey games being played.**

upon how large the space of lineups is. The size of the lineup space on a given night is larger if there are more NHL games being played. We plot the mean profit margin for 200 integer programming lineups as a function of the number of games played in a night for different values of the maximum lineup overlap

in Figure 7. These lineups were created using Type 4 stacking. For nights with less than four games, a maximum lineup overlap of seven does best. For a larger number of games, it seems that decreasing the maximum lineup overlap improves performance. For instance, for nights with more than nine games, an overlap of four does best. The reason for this is that when there are few games being played, the lineup space is smaller and probably has fewer good lineups. By reducing the maximum lineup overlap, we end up not selecting these good lineups and instead choose many poor lineups. However, on nights with many games, there is a huge lineup space that has several different good lineups. A smaller maximum lineup overlap causes the integer program to choose several of these good lineups, increasing the chance that one of them will achieve a high rank. For maximum lineup overlaps smaller than four we found that it becomes difficult to obtain 200 feasible lineups. Maximum lineup overlaps larger than seven produce lineups that are too similar and do not provide enough diversity. Therefore, the maximum lineup overlap should be tuned between four and seven depending on how many games are being played on a given night.

5.4. Lineup Creation Order Versus Lineup Performance Rank

The greedy integer programming formulation is solved iteratively to create multiple lineups. One question to ask is do the lineups which are created earlier perform better? To investigate this, we set the integer programming approach to create 200 linueps with Type 4 stacking and a maximum overlap of seven. We then evaluate their performance rank relative to each other in the contests. This allows us to look at several statistics relating the creation rank and the performance rank of the lineups. We first investigate the Spearman rank correlation coefficient between these two rankings. We find that the average value of the correlation coefficient across the different contests is 0.09 with a standard deviation of 0.1. This suggests that there is very little correlation between the creation rank and performance rank of the lineups. To dive deeper into the analysis, we plot in Figure 8 two different sets of data. First, we show a boxplot of the creation rank of the best performing lineup. Second, we show the performance rank of the first created lineup. We see that the median performance rank of the first created lineup is 74.5. This shows that the first created lineup has a slightly, but not substantially improved performance. For the best performing lineup, the median creation rank is 124.5. Therefore, while the first created lineup is slightly better, the winning lineup is generally one of the later created lineups. This supports our finding of a very low correlation between the creation and performance rank of the lineups.

5.5. Mean and Standard Deviation of Lineup Points Versus Profit Margin

One way to characterize the lineups produced by our greedy integer programming formulation is by the mean and standard deviation of their points. We can compare these with the mean and standard deviation of the points of all lineups in a contest. Let us define the mean and standard deviation of our optimized lineups as μ_1 and σ_1 and of the population lineups as μ_0 and σ_0. We will look at the difference in these parameters for our lineups and the population lineups for each contest. We use 200 integer programming lineups with

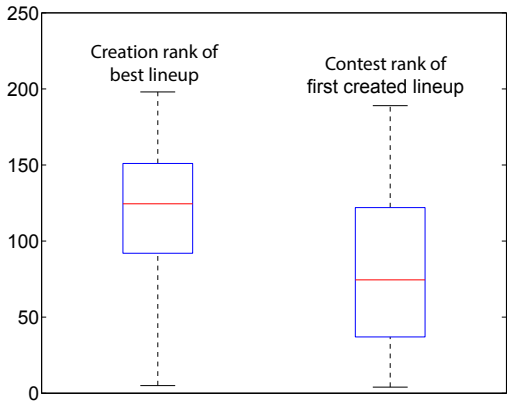

Figure 8 **Plot of the performance rank of the first created lineup and the creation rank of the best performing lineup. There were 200 integer programming lineups created with Type 4 stacking and a maximum overlap of seven.**

Type 4 stacking and a maximum overlap of seven and plot the profit margin versus the difference in the mean and standard deviation in Figure 9. One can see that there is a substantial variation in mean difference, which ranges from -10 to 20 points. When it is below zero, no profit is made, but when it is above zero, a substantial profit is made. This is a result of the fact that the lineups must satisfy the stacking constraints which increase their individual variance. The result is that when the actual points are realized, we are either far above or far below the population mean. The standard deviation of the integer programming lineups is generally below the population standard deviation. This is because we are choosing lineups that have some correlation because they are designed to maximize points, even though the individual lineups are designed to have a high variance. From Figure 9 one can see that we are equally likely to be either above or below the population mean. However, because the contest payoff structure is so asymmetric, with a maximum loss of 100%, but a maximum gain which is at least an order of magnitude larger, our linueps end up being profitable.

5.6. Implementation and Runtime

All formulations were constructed using the JuMP algebraic modeling language (Dunning et al. 2015, Lubin and Dunning 2015), which is written in the Julia programming language (Bezanson et al. 2012). This allowed us to test the runtime of our approach for various integer programming solvers. In practice, we must wait for information about which goalies are being played to be posted before we construct our lineups. Goalies do not play every night, so if we do not wait for this information, we may put goalies in lineups who are not playing and receive zero points for them. This goalie information is generally posted on public websites about 30 minutes before the games begin. We must be able to solve the greedy integer programming formulation in this time to be able to enter the contest. We show in Figure 10 the

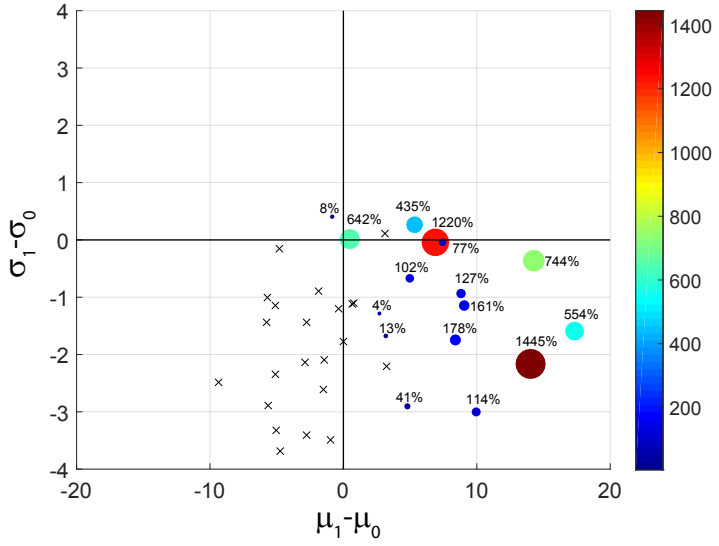

Figure 9 **Plot of profit margin versus $\mu_1 - \mu_0$ and $\sigma_1 - \sigma_0$ for DraftKings hockey contests with 200 integer programming lineups created using Type 4 stacking and a maximum overlap of seven. The profit margin for each contest is indicated through the color and size of the markers and also listed next to the marker. The x markers indicate contests where the profit margin is not positive.**

time needed to solve the greedy integer programming formulation for 100 lineups in each of our contests using different integer programming solvers. We consider the free solvers CBC (COIN-OR 2016) and GLPK (GNU 2016) and the commercial solver Gurobi (Gurobi Optimization 2016). All computations were done using a Intel Core i5-4570 3.20GHz with 8GB of RAM. We find that all solvers are able to solve the integer programming formulations in under four minutes, giving sufficient time to enter the lineups into the contest. The complete software used to construct the integer programming lineups is available at `https://github.com/dscotthunter/Fantasy-Hockey-IP-Code`.

6. Baseball

Having demonstrated the success of our greedy integer programming formulation in hockey contests, our next goal was to see how much our approach generalizes. To do this, we applied our approach to top heavy daily fantasy baseball contests in DraftKings. We have less data for baseball than we did for hockey because the season had just begun. Also, the structure of baseball contests is similar to hockey contests. Therefore, we will present here a brief summary of our baseball greedy integer programming formulation and its performance on real contest data. In short, we have found that in our limited number of baseball contests, the method does indeed perform very well and is able to place in the top ten in contests with tens of thousands of entrants multiple times.

We begin with a brief overview of how baseball is played and how fantasy points are scored. A baseball team consists of pitchers and hitters. The goal is for the hitters to hit the ball pitched by the opposing pitcher

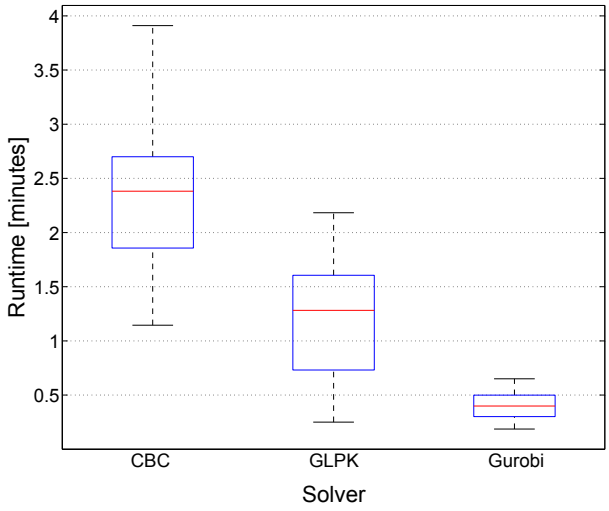

Figure 10 **The runtime of our algorithm for generating 100 lineups with different integer programming software.**

and then run to four bases to score runs. In fantasy sports contests a pitcher scores points for striking out batters, pitching for more innings in the game, and winning the game. A hitter scores points by hitting the pitch, scoring a run by reaching home base, or having other hitters score a run as a result of his hit, also known as a run batted in or RBI.

There is a great deal of similarity between the structure of hockey and baseball lineups, so we are able to almost directly use the hockey integer programming formulation with some slight modifications. We do not go through the details of the baseball formulation here, but instead present the basic constraints. First, our objective is to maximize the expected fantasy points of the lineups. Here we simply use the projections from Daily Fantasy Nerd for the expected players' points. Second, there are feasibility constraints related to the players' position, team, and salary. A baseball lineup consists of ten players: two pitchers and eight hitters (catcher, first baseman, second baseman, third baseman, shortstop, and three outfielders). The lineup must have players from at least two different baseball games and the budget for a lineup is 50,000 fantasy dollars, just as in hockey. Finally, a lineup cannot have more than five hitters from one team.

We have stacking and overlap constraints for baseball. As in hockey, we want to stack the baseaball lineups to increase their variance. We do this by adding constraints which remove negative correlations and increase positive correlations within a lineup. We use two different stacking types to do this. First, similar to goalie stacking in hockey, we do not have a pitcher and any opposing players on the same lineup. Pitchers and opponents have negatively correlated points, just as in hockey with goalies and opposing skaters. Second, similar to line stacking in hockey, we use consecutive hitters on a lineup. Consecutive hitters have positively correlated points because of the structure of baseball. For instance, if the first three batters can

Date	Best Rank	Total Number of Entries
5/25/2016	1	47,916
5/26/2016	3	38,333
5/27/2016	1,342	57,500
5/29/2016	7	38,333
5/30/2016	213	38,333
5/31/2016	146	47,916
6/1/2016	400	47,916
6/2/2016	17	46,000
6/3/2016	376	53,666
6/5/2016	2	38,333

Table 4 The date, our top ranked lineup, and number on entries in DraftKings top heavy baseball contests.

get on base and the fourth batter hits a home run, then the first three batters get points for runs and the fourth batter gets points for his hit plus the three RBIs. In our formulation, we use five consecutive batters in each lineup, allowing for cyclic orders such as batters in positions $(8, 9, 1, 2, 3)$. We choose five consecutive batters because this is the maximum allowed by DraftKings. The overlap constraint has the same structure for both baseball and hockey except that we set the maximum overlap parameter γ to six for baseball.

Baseball contests allow 200 entries per person, but are typically larger than hockey contests, which makes them more difficult to win. While top heavy hockey contests never have more than 24,000 entrants, top heavy baseball contests always have more than 38,000 entrants. Nonetheless, our approach has proven successful . We show the rank achieved by our best lineup (out of 200) in historic DraftKings daily fantasy baseball competitions and the number of entrants in the competitions in Table 4. As can be seen, our greedy integer programming heuristic is able to place in the top ten in four competitions, and even comes in first place in one of the competitions. This success in baseball is significant because we use the same general principles presented in Section 2.5 which were used for hockey. This suggests that our approach to picking winners is very general and has applicability to a variety of domains.

7. Conclusion

We have presented here a greedy integer programming formulation for constructing entries from a set of constrained resources which have a large probability of winning winner-take-all contests. Our approach is developed from insights gained from analyzing properties of the probability of winning, which is many times an intractable function. We have shown that our approach can consistently win top heavy daily fantasy hockey and baseball contests. While we applied our approach to one application domain, it is very general and can be applied in many other settings. At a high level, our approach can be summarized as follows.

1. Build simple prediction models for the resources' means and understand their pairwise correlations.
2. Formulate an integer program that constructs entries with maximal mean while lower bounding their variance and upper bounding their correlation with previously constructed entries.
3. Solve the integer programs to sequentially generate entries in a greedy manner.

The success we were able to achieve for both hockey and baseball given the relative simplicity of our approach shows its power. We do not use the actual value of any correlations, just a basic understanding of their sign. Our predictions can even be noisy. It is the manner in which the entries are constructed which allows us to pick winners despite noise in the prediction models.

An interesting question is how easily can this approach be ported over to other domains? We expect our approach to work in winner-take-all contests where one can obtain predictions for the entries and also where one understands the basic correlation structure of the entries. Such situations can include movie studios trying to select films to produce, pharmaceutical companies trying to select drugs to develop, or a venture capital fund trying to select start-up companies in which to invest. In these situations experts will have some predictions for success of different entries, whether they be movies, drugs, or companies. Also, correlations may be understood because of similarities between the entries. For instance, the success of companies developing similar products may be correlated. Therefore, one should be able to apply our approach to picking winners in these and many other similar situations.

8. Proofs

8.1. Proof of Theorem 2.1

We show that maximizing $U(\mathcal{S})$ is NP-hard by reducing it to the maximum coverage problem. In the maximum coverage problem one is given a set $\mathcal{U}$ of n elements and a collection $\mathcal{E} = \{E_i\}_{i=1}^N$ of N subsets of $\mathcal{U}$ such that $\bigcup_{E\in\mathcal{E}} E = \mathcal{U}$. The goal is to select M sets from $\mathcal{E}$ such that their union has maximum cardinality. This is known to be an NP-hard optimization problem. To show that this is an instance of maximizing $U(\mathcal{S})$ we assume that the sample space Ω is countable and finite with R elements. We also assume that each element $\omega \in \Omega$ has equal probability, i.e. $\mathbf{P}(\omega) = R^{-1}$. Let $\mathcal{F}$ be the σ-algebra of Ω. For any set $\mathcal{S} \in \mathcal{F}$, we can write $U(\mathcal{S}) = R^{-1}|\bigcup_{\omega\in\mathcal{S}} \omega|$. Then we have

$$\begin{aligned}\max_{\mathcal{S}\subseteq\mathcal{E},|\mathcal{S}|=M} U(\mathcal{S}) &= \max_{\mathcal{S}\subseteq\mathcal{E},|\mathcal{S}|=M} R^{-1}\left|\bigcup_{\omega\in\mathcal{S}}\omega\right| \\ &= \max_{\mathcal{S}\subseteq\mathcal{E},|\mathcal{S}|=M}\left|\bigcup_{\omega\in\mathcal{S}}\omega\right|.\end{aligned}$$

Therefore, maximizing $U(\mathcal{S})$ is equivalent to the maximum coverage problem.

8.2. Proof of Theorem 2.2

The function $U(\mathcal{S})$ is non-negative and non-decreasing because it is the probability of a set of events. We must show that it is also submodular. A submodular function f satisfies

$$f\left(\mathcal{S}\bigcup v\right) - f\left(\mathcal{S}\bigcup v\right) \geq f\left(\mathcal{T}\bigcup v\right) - f\left(\mathcal{T}\bigcup v\right) \tag{8.1}$$

for all elements v and pairs of sets $\mathcal{S}$ and $\mathcal{T}$ such that $\mathcal{S} \subseteq \mathcal{T}$. We show that the function $U(\mathcal{S})$ is submodular as follows. We let the σ-algebra of the probability space be $\mathcal{F}$. Consider sets $\mathcal{S}, \mathcal{T}, v \in \mathcal{F}$ such that $\mathcal{S} \subseteq \mathcal{T}$.

We can write $v = v_{\mathcal{S}} \bigcup v_{\mathcal{T}} \bigcup v_0$ where we define $v_{\mathcal{S}} = v \bigcap \mathcal{S}$, $v_{\mathcal{T}} = v \bigcap \mathcal{T} \bigcap \mathcal{S}^c$, and $v_0 = v \bigcap \mathcal{T}^c$. Then we have

$$U\left(\mathcal{T} \bigcup v\right) - U\left(\mathcal{T}\right) = \mathbf{P}\left(v_0\right)$$

and

$$\begin{aligned} U\left(\mathcal{S} \bigcup v\right) - U\left(\mathcal{S}\right) &= \mathbf{P}\left(v_{\mathcal{T}} \bigcup v_0\right) \\ &\geq \mathbf{P}\left(v_0\right) \\ &\geq U\left(\mathcal{T} \bigcup v\right) - U\left(\mathcal{T}\right), \end{aligned}$$

thus showing that $U(\mathcal{S})$ satisfies the submodularity condition.

8.3. Proof of Theorem 2.5

The proof of Theorem 2.5 relies upon the following lemma.

LEMMA 1. *For a set of events $\mathcal{S}$, let $M = |\mathcal{S}|$. Then*

$$U_2(\mathcal{S}) = \frac{1}{2} \sum_{l=1}^{M} l(3-l) \sum_{T \in \mathcal{S}_l} p'_T. \tag{8.2}$$

With this lemma we can rewrite $U_2(\mathcal{S})$ as

$$\begin{aligned} U_2(\mathcal{S}) &= 1 - p_0 - (1 - p_0) + \frac{1}{2} \sum_{l=1}^{M} l(3-l) \sum_{T \in S_l} p'_T \\ &= U(\mathcal{S}) - \sum_{l=1}^{M} \sum_{T \in S_l} p'_T + \frac{1}{2} \sum_{l=1}^{M} l(3-l) \sum_{T \in S_l} p'_T \\ &= U(\mathcal{S}) + \frac{1}{2} \sum_{l=1}^{M} (-l^2 + 3l - 2) \sum_{T \in S_l} p'_T \\ &= U(\mathcal{S}) + \frac{1}{2} \sum_{l=3}^{M} (-l^2 + 3l - 2) \sum_{T \in S_l} p'_T. \end{aligned}$$

Above we have used the fact that $1 - p_0 = \sum_{l=1}^{M} \sum_{T \in S_l} p'_T$. We also adjusted the limits of the final sum using the fact that the terms $-l^2 + 3l - 2$ are zero for $l = 1, 2$. We will next upper bound the difference $U(\mathcal{S}) - U_2(\mathcal{S})$ and lower bound $U(\mathcal{S})$ in order to upper bound $(U(\mathcal{S}) - U_2(\mathcal{S}))/U(\mathcal{S})$. We first consider $U(\mathcal{S})$. We have that $p_0 = (1-p)^M$ and $pM < N^{-\delta} < 1/4$ for some $\delta > 0$. Using this we obtain

$$\begin{aligned} U(\mathcal{S}) &= 1 - p_0 \\ &= 1 - (1-p)^M \end{aligned} \tag{8.3}$$

To obtain a lower bound, we must upper bound $(1-p)^M$. This is done as follows.

$$\begin{aligned}(1-p)^M &= \sum_{i=0}^{M} \binom{M}{i} (-p)^i \\ &\leq 1 - Mp + \frac{(Mp)^2}{2} + \sum_{i=3}^{M} (Mp)^i \\ &\leq 1 - Mp + \frac{(Mp)^2}{2} + \frac{(Mp)^3}{1-Mp} \\ &\leq 1 - Mp + (Mp)^2.\end{aligned}$$

Above we have used the fact that $Mp < 1/3$. With this bound we have

$$\begin{aligned}U(\mathcal{S}) &\geq 1 - (1 - Mp + (Mp)^2) \\ &\geq Mp(1-Mp) \\ &\geq \frac{2Mp}{3}.\end{aligned}$$

We next upper bound the difference $U(\mathcal{S}) - U_2(\mathcal{S})$. Recall that we assumed for any $1 \leq l \leq M$ and any $T \in \mathcal{S}_l$ that $p_T \leq c_1 p^l$. With this assumption we have

$$\begin{aligned}U(\mathcal{S}) - U_2(\mathcal{S}) &= \frac{1}{2} \sum_{l=3}^{M} (l^2 - 3l + 2) \sum_{T \in \mathcal{S}_l} p_T \\ &\leq \frac{c_1}{2} \sum_{l=3}^{M} (l^2+2) \binom{M}{l} p^l \\ &\leq \frac{c_1}{2} \sum_{l=3}^{M} (l^2+2) (Mp)^l \\ &\leq \frac{c_1}{2} \sum_{l=3}^{\infty} l^2 (Mp)^l + c_1 (Mp)^3 \sum_{l=0}^{\infty} (Mp)^l \\ &\leq \frac{c_1}{2} \left(\frac{Mp(1+Mp)}{(1-Mp)^3} - Mp - 4(Mp)^2 \right) + \frac{c_1 (Mp)^3}{1-Mp} \\ &\leq \frac{c_1}{2} \frac{13(Mp)^3}{(1-Mp)^3} + \frac{c_1 (Mp)^3}{1-Mp} \\ &\leq 8 c_1 (Mp)^2 \left(\frac{Mp}{(1-Mp)^3} + \frac{Mp}{1-Mp} \right) \\ &\leq 16 c_1 (Mp)^2.\end{aligned}$$

Combining the two bounds we obtain

$$\begin{aligned}\frac{U(\mathcal{S}) - U_2(\mathcal{S})}{U(\mathcal{S})} &\leq \frac{16 c_1 (Mp)^2}{2Mp/3} \\ &\leq 24 c_1 Mp \\ &\leq 24 c_1 N^{-\delta}.\end{aligned}$$

8.4. Prof of Lemma 1

We assume without loss of generality that $\mathcal{S} = \{E_i\}_{i=1}^M$. Then we can write $U_2(\mathcal{S})$ as

$$\begin{aligned} U_2(\mathcal{S}) &= \sum_{i=1}^{M}\sum_{l=1}^{M}\sum_{T\in\mathcal{S}_l, E_i\in T} p'_T - \frac{1}{2}\sum_{i,j=1,j\neq i}^{M}\sum_{l=2}^{M}\sum_{T\in\mathcal{S}_l, E_i, E_j\in T} p'_T \\ &= \sum_{l=1}^{M}\binom{l}{1}\sum_{T\in\mathcal{S}_l} p'_T - \sum_{l=2}^{M}\binom{l}{2}\sum_{T\in\mathcal{S}_l} p'_T \\ &= \frac{1}{2}\sum_{l=1}^{M} l(3-l)\sum_{T\in\mathcal{S}_l} p'_T. \end{aligned}$$

Above we have used the convention $\binom{l}{k} = 0$ for $l < k$ and the relation

$$\binom{l}{1} - \binom{l}{2} = \frac{l(3-l)}{2}.$$

8.5. Proof of Theorem 2.6

We define the marginal mean and variance of X_i as μ_i and σ_i^2 and we define the correlation coefficient of X_i and X_j as ρ_{ij}. We define $p_i = \mathbf{P}(E_i)$ and $p_{ij} = \mathbf{P}(E_i \bigcap E_j)$. We will use the following bounds for Gaussian random variables.

LEMMA 2. *(Gordon 1941) Let X be a Gaussian random variable with mean μ and positive standard deviation σ. For any value $t > \mu$ let $z = (t-\mu)/\sigma$. Then we have*

$$\frac{\exp\left(-z^2/2\right)}{\sqrt{2\pi}\sigma(z+1/z)} \leq \mathbf{P}(X \geq t) \leq \frac{\exp\left(-z^2/2\right)}{\sqrt{2\pi}\sigma z}. \tag{8.4}$$

We will also use the following lemma to obtain an upper bound on the joint probability p_{ij}.

LEMMA 3. *Let (X_1, X_2) be a pair of jointly Gaussian random variables with means μ_1, μ_2, positive standard deviations σ_1, σ_2, and correlation coefficient ρ. For any value $t > \max(\mu_1, \mu_2)$, we have*

$$\mathbf{P}(\min(X_1, X_2) > t) \leq \exp\left(-\frac{(2t-\mu_1-\mu_2)^2}{2\left(\sigma_1^2+\sigma_2^2+\sqrt{(\sigma_1^2-\sigma_2^2)^2+4\rho\sigma_1^2\sigma_2^2}\right)}\right). \tag{8.5}$$

With these bounds for the relevant probabilities, we can obtain a lower bound on the objective function. Let $z_i = (t-\mu_i)/\sigma_i$. Then using Lemmas 2 and 3 we have

$$\begin{aligned} U_2(S) &= \sum_{i:E_i\in\mathcal{S}} p_i - \frac{1}{2}\sum_{i,j:i:E_i,E_j\in\mathcal{S},i\neq j} p_{ij} \\ &\geq \sum_{i:E_i\in\mathcal{S}} \frac{1}{z_i+1/z_i}\exp\left(-\frac{z_i^2}{2}\right) \\ &\quad - \frac{1}{2}\sum_{i,j:i:E_i,E_j\in\mathcal{S},i\neq j} \exp\left(-\frac{(2t-\mu_i-\mu_j)^2}{2\left(\sigma_i^2+\sigma_j^2+\sqrt{(\sigma_i^2-\sigma_j^2)^2+4\rho_{ij}\sigma_i^2\sigma_j^2}\right)}\right) \\ &= U_2^l(\mathcal{S}). \end{aligned}$$

8.6. Proof of Lemma 3

We define the two dimensional vectors $\mathbf{t} = [t, t]$, $\mu = [\mu_1, \mu_2]$, and $\mathbf{1} = [1, 1]$ and we define the covariance matrix of X_1 and X_2 as

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}. \tag{8.6}$$

To obtain the upper bound on the probability of interest we use a careful application of the Chernoff bound. For any $\theta > 0$, we can upper bound the probability of interest as

$$\begin{aligned}
\mathbf{P}(\min(X_1, X_2) > t) &= \int_t^\infty \int_t^\infty \frac{|\Sigma|^{-1/2}}{2\pi} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right) dx_1 dx_2 \\
&\leq \int_{-\infty}^\infty \int_{-\infty}^\infty \frac{|\Sigma|^{-1/2}}{2\pi} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right) \exp\left(\theta \mathbf{1}^T(\mathbf{x}-\mathbf{t})\right) dx_1 dx_2.
\end{aligned}$$

To perform the integral above, we transform to a basis where Σ is diagonal. Let the eigenvalues of Σ be λ_1 and λ_2 with $\lambda_1 \geq \lambda_2 > 0$. It can be shown that these eigenvalues are

$$\lambda_1 = \frac{\sigma_1^2 + \sigma_2^2}{2} + \frac{\sqrt{\left(\sigma_1^2 - \sigma_2^2\right)^2 + 4\rho\sigma_1^2\sigma_2^2}}{2} \tag{8.7}$$

$$\lambda_2 = \frac{\sigma_1^2 + \sigma_2^2}{2} - \frac{\sqrt{\left(\sigma_1^2 - \sigma_2^2\right)^2 + 4\rho\sigma_1^2\sigma_2^2}}{2} \tag{8.8}$$

We can write $\Sigma = V^{-1}\Lambda V$ where V is an orthonormal matrix and Λ is a diagonal matrix with elements λ_1 and λ_2 along its diagonal. We perform a coordinate transformation in the integral to the variables $\mathbf{u} = V^{-1}(\mathbf{x} - \mu)$. Using this change of variables we obtain

$$\begin{aligned}
\mathbf{P}(\min(X_1, X_2) > t) &\leq \int_{-\infty}^\infty \int_{-\infty}^\infty \frac{|\Sigma|^{-1/2}}{2\pi} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T V\Lambda^{-1}V^{-1}(\mathbf{x}-\mu)\right) \exp\left(\theta \mathbf{1}^T(\mathbf{x}-\mathbf{t})\right) dx_1 dx_2 \\
&\leq \exp\left(-\theta(2t - \mu_1 - \mu_2)\right) \prod_{i=1}^2 \int_{-\infty}^\infty \frac{1}{\sqrt{2\pi\lambda_i}} \exp\left(-\frac{u_i^2}{2\lambda_i} + \theta u_i \sum_{j=1}^2 V_{ij}\right) du_i \\
&\leq \exp\left(-\theta(2t - \mu_1 - \mu_2) + \frac{\theta^2}{2}\sum_{i=1}^2 \lambda_i \left(\sum_{j=1}^2 V_{ij}\right)^2\right)
\end{aligned}$$

We choose θ to minimize the upper bound. Let the value of θ which achieves this minimum be θ^*. It can easily be shown that

$$\theta^* = \frac{2t - \mu_1 - \mu_2}{\sum_{i=1}^2 \lambda_i \left(\sum_{j=1}^2 V_{ij}\right)^2}.$$

Using this value for θ^* we obtain the tightest upper bound

$$\begin{aligned}\mathbf{P}(\min(X_1, X_2) > t) &\leq \exp\left(-\theta^*(2t - \mu_1 - \mu_2) + \frac{(\theta^*)^2}{2}\sum_{i=1}^{2}\lambda_i\left(\sum_{j=1}^{2}V_{ij}\right)^2\right) \\ &\leq \exp\left(-\frac{(2t - \mu_1 - \mu_2)^2}{2\sum_{i=1}^{2}\lambda_i\left(\sum_{j=1}^{2}V_{ij}\right)^2}\right).\end{aligned}$$

Finally, we note that since V is an orthonormal two by two matrix, we have that $V_{ij} = -V_{ij}$ for $i \neq j$ and the inner product of any two distinct rows is zero. Using this we obtain the following useful property.

$$\begin{aligned}\sum_{i=1}^{2}\left(\sum_{j=1}^{2}V_{ij}\right)^2 &= \sum_{i=1}^{2}\sum_{j=1}^{2}V_{ij}^2 + \sum_{i=1}^{2}\sum_{j,k=1,j\neq k}^{2}V_{ij}V_{ik} \\ &= \sum_{i=1}^{2}1 + \sum_{i=1}^{2}\sum_{j,k=1,j\neq k}^{2}V_{ji}V_{ki} \\ &= 2.\end{aligned}$$

Using this plus the expression for λ_1, we obtain

$$\begin{aligned}\mathbf{P}(\min(X_1, X_2) > t) &\leq \exp\left(-\frac{(2t - \mu_1 - \mu_2)^2}{2\sum_{i=1}^{2}\lambda_i(\sum_{j=1}^{2}V_{ij})^2}\right) \\ &\leq \exp\left(-\frac{(2t - \mu_1 - \mu_2)^2}{2\lambda_1\sum_{i=1}^{2}(\sum_{j=1}^{2}V_{ij})^2}\right) \\ &\leq \exp\left(-\frac{(2t - \mu_1 - \mu_2)^2}{4\lambda_1}\right) \\ &\leq \exp\left(-\frac{(2t - \mu_1 - \mu_2)^2}{2\left(\sigma_1^2 + \sigma_2^2 + \sqrt{(\sigma_1^2 - \sigma_2^2)^2 + 4\rho\sigma_1^2\sigma_2^2}\right)}\right).\end{aligned}$$

References

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Management Science*, 2015.

Jonathan Bales. The art of stacking (or not) in daily fantasy baseball. http://playbook.draftkings.com/mlb/the-art-of-stacking-or-not-in-daily-fantasy-baseball/, June 2015a.

Jonathan Bales. Pairing a QB with his receiver(s). https://rotogrinders.com/articles/pairing-a-qb-with-his-receiver-s-481544, December 2015b.

A. Becker and A. Sun. An analytical approach for fantasy football draft and lineup management. *Journal for Quantitative Analysis in Sports*, 2014.

Jeff Bezanson, Stefan Karpinski, Viral Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

New York Times Editorial Board. Rein in online fantasy sports gambling. http://www.nytimes.com/2015/10/05/opinion/rein-in-online-fantasy-sports-gambling.html, October 2015.

Harr Chen and David R Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 429–436. ACM, 2006.

COIN-OR. CBC: COIN-OR Branch and Cut, 2016. `https://projects.coin-or.org/Cbc`.

DailyFantasyNerd. Daily fantasy nerd. http://dailyfantasynerd.com, December 2015.

DraftKings. Draftkings. http://draftkings.com, December 2015.

Drewby. Rackem stackem: Dfs hockey. http://dailyroto.com/nhl-dfs-stacking-strategy-draftkings-fanduel, October 2015.

Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *arXiv:1508.01982 [math.OC]*, 2015.

Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–298. ACM, 2009.

GNU. GLPK: GNU Linear Programming Kit, 2016. `https://www.gnu.org/software/glpk/`.

Robert D Gordon. Values of mills' ratio of area to bounding ordinate and of the normal probability integral for large values of the argument. *The Annals of Mathematical Statistics*, 12(3):364–366, 1941.

Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.

Gurobi Optimization. The Gurobi Optimizer, 2016. `http://www.gurobi.com`.

Drew Harwell. Why you (probably) won't win money playing Draftkings, FanDuel. http://www.dailyherald.com/article/20151012/business/151019683/, October 2015.

Darren Heitner. Draftkings reports $304 million of entry fees in 2014. http://www.forbes.com/sites/darrenheitner/2015/01/22/draftkings-reports-304-million-of-entry-fees-in-2014/, January 2015.

Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

Miles Lubin and Iain Dunning. Computing in operations research using Julia. *INFORMS Journal on Computing*, 27 (2):238–248, Spring 2015.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

Rotogrinders. Rotogrinders. https://rotogrinders.com/, December 2015.

Daniel Steinberg. How daily fantasy football pricing algorithms work. https://dailyfantasywinners.com/fantasy-categories/featured/daily-fantasy-football-pricing-algorithms-work/, 2015.

Ashwin Swaminathan, Cherian V Mathew, and Darko Kirovski. Essential pages. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 173–182. IEEE Computer Society, 2009.