Betting on Permutations

Yiling Chen
Yahoo! Research
45 W. 18th St. 6th Floor
New York, NY 10011

Lance Fortnow
Department of Computer Science
University of Chicago
Chicago, IL 60637

Evdokia Nikolova*
CS & AI Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

David M. Pennock
Yahoo! Research
45 W. 18th St. 6th Floor
New York, NY 10011

ABSTRACT

We consider a *permutation betting* scenario, where people wager on the final ordering of n candidates: for example, the outcome of a horse race. We examine the *auctioneer problem* of risklessly matching up wagers or, equivalently, finding arbitrage opportunities among the proposed wagers. Requiring bidders to explicitly list the orderings that they'd like to bet on is both unnatural and intractable, because the number of orderings is $n!$ and the number of subsets of orderings is $2^{n!}$. We propose two expressive betting languages that seem natural for bidders, and examine the computational complexity of the auctioneer problem in each case. *Subset betting* allows traders to bet either that a candidate will end up ranked among some subset of positions in the final ordering, for example, "horse A will finish in positions 4, 9, or 13-21", or that a position will be taken by some subset of candidates, for example "horse A, B, or D will finish in position 2". For subset betting, we show that the auctioneer problem can be solved in polynomial time if orders are divisible. *Pair betting* allows traders to bet on whether one candidate will end up ranked higher than another candidate, for example "horse A will beat horse B". We prove that the auctioneer problem becomes NP-hard for pair betting. We identify a sufficient condition for the existence of a pair betting match that can be verified in polynomial time. We also show that a natural greedy algorithm gives a poor approximation for indivisible orders.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

*Part of this work was done while the author was at Yahoo! Research. Supported in part by American Foundation for Bulgaria Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EC'07, June 13–16, 2007, San Diego, California, USA.
Copyright 2007 ACM 978-1-59593-653-0/07/0006 ...$5.00.

General Terms

Economics, Theory

Keywords

Prediction market, expressive betting, order matching, computational complexity

1. INTRODUCTION

Buying or selling a financial security in effect is a wager on the security's value. For example, buying a stock is a bet that the stock's value is greater than its current price. Each trader evaluates his expected profit to decide the quantity to buy or sell according to his own information and subjective probability assessment. The collective interaction of all bets leads to an equilibrium that reflects an aggregation of all the traders' information and beliefs. In practice, this aggregate market assessment of the security's value is often more accurate than other forecasts relying on experts, polls, or statistical inference [16, 17, 5, 2, 15].

Consider buying a security at price fifty-two cents, that pays $1 if and only if a Democrat wins the 2008 US Presidential election. The transaction is a commitment to accept a fifty-two cent loss if a Democrat does not win in return for a forty-eight cent profit if a Democrat does win. In this case of an event-contingent security, the price—the market's value of the security—corresponds directly to the estimated probability of the event.

Almost all existing financial and betting exchanges pair up bilateral trading partners. For example, one trader willing to accept an x dollar loss if a Democrat does not win in return for a y dollar profit if a Democrat wins is matched up with a second trader willing to accept the opposite.

However in many scenarios, even if no bilateral agreements exist among traders, multilateral agreements may be possible. For example, if one trader bets that the Democratic candidate will receive more votes than the Republican candidate, a second trader bets that the Republican candidate will receive more votes than the Libertarian candidate, and a third trader bets that the Libertarian candidate will receive more votes than the Democratic candidate, then, depending on the odds they each offer, there may be a three-way agreeable match even though no two-way matches exist.

We propose an exchange where traders have considerable flexibility to naturally and succinctly express their wagers,

and examine the computational complexity of the auctioneer's resulting *matching problem* of identifying bilateral and multilateral agreements. In particular, we focus on a setting where traders bet on the outcome of a competition among n candidates. For example, suppose that there are n candidates in an election (or n horses in a race, etc.) and thus $n!$ possible orderings of candidates after the final vote tally. Traders may like to bet on arbitrary properties of the final ordering, for example "candidate D will win", "candidate D will finish in either first place or last place", "candidate D will defeat candidate R", "candidates D and R will both defeat candidate L", etc. The goal of the exchange is to search among all the offers to find two or more that together form an agreeable match. As we shall see, the matching problem can be set up as a linear or integer program, depending on whether orders are divisible or indivisible, respectively. Attempting to reduce the problem to a bilateral matching problem by explicitly creating $n!$ securities, one for each possible final ordering, is both cumbersome for the traders and computationally infeasible even for modest sized n. Moreover, traders' attention would be spread among $n!$ independent choices, making the likelihood of two traders converging at the same time and place seem remote.

There is a tradeoff between the expressiveness of the bidding language and the computational complexity of the matching problem. We want to offer traders the most expressive bidding language possible while maintaining computational feasibility. We explore two bidding languages that seem natural from a trader perspective. *Subset betting*, described in Section 3.2, allows traders to bet on which positions in the ranking a candidate will fall, for example "candidate D will finish in position 1, 3-5, or 10". Symetrically, traders can also bet on which candidates will fall in a particular position. In Section 4, we derive a polynomial-time algorithm for matching (divisible) subset bets. The key to the result is showing that the exponentially big linear program has a corresponding separation problem that reduces to maximum weighted bipartite matching and consequently we can solve it in time polynomial in the number of orders.

Pair betting, described in Section 3.3, allows traders to bet on the final ranking of any two candidates, for example "candidate D will defeat candidate R". In Section 5, we show that optimal matching of (divisible or indivisible) pair bets is NP-hard, via a reduction from the unweighted minimum feedback arc set problem. We also provide a polynomially-verifiable sufficient condition for the existence of a pair-betting match and show that a greedy algorithm offers poor approximation for indivisible pair bets.

2. BACKGROUND AND RELATED WORK

We consider *permutation betting*, or betting on the outcome of a competition among n candidates. The final outcome or *state* $s \in \mathcal{S}$ is an ordinal ranking of the n candidates. For example, the candidates could be horses in a race and the outcome the list of horses in increasing order of their finishing times. The state space $\mathcal{S}$ contains all $n!$ mutually exclusive and exhaustive permutations of candidates.

In a typical horse race, people bet on properties of the outcome like "horse A will win", "horse A will *show*, or finish in either first or second place", or "horses A and B will finish in first and second place, respectively". In practice at the racetrack, each of these different types of bets are processed in separate *pools* or groups. In other words, all the "win" bets are processed together, and all the "show" bets are processed together, but the two types of bets do not mix. This separation can hurt liquidity and information aggregation. For example, even though horse A is heavily favored to win, that may not directly boost the horse's odds to show.

Instead, we describe a central exchange where all bets on the outcome are processed together, thus aggregating liquidity and ensuring that informational inference happens automatically.

Ideally, we'd like to allow traders to bet on any property of the final ordering they like, stated in exactly the language they prefer. In practice, allowing too flexible a language creates a computational burden for the auctioneer attempting to match willing traders. We explore the tradeoff between the expressiveness of the bidding language and the computational complexity of the matching problem.

We consider a framework where people propose to buy securities that pay \$1 if and only if some property of the final ordering is true. Traders state the price they are willing to pay per share and the number of shares they would like to purchase. (Sell orders may not be explicitly needed, since buying the negation of an event is equivalent to selling the event.) A *divisible* order permits the trader to receive fewer shares than requested, as long as the price constraint is met; an *indivisible* order is an all-or-nothing order. The description of bets in terms of prices and shares is without loss of generality: we can also allow bets to be described in terms of odds, payoff vectors, or any of the diverse array of approaches practiced in financial and gambling circles.

In principle, we can do everything we want by explicitly offering $n!$ securities, one for every state $s \in \mathcal{S}$ (or in fact any set of $n!$ linearly independent securities). This is the so-called *complete Arrow-Debreu securities market* [1] for our setting. In practice, traders do not want to deal with low-level specification of complete orderings: people think more naturally in terms of high-level properties of orderings. Moreover, operating $n!$ securities is infeasible in practice from a computational point of view as n grows.

A very simple bidding language might allow traders to bet only on who wins the competition, as is done in the "win" pool at racetracks. The corresponding matching problem is polynomial, however the language is not very expressive. A trader who believes that A will defeat B, but that neither will win outright cannot usefully impart his information to the market. The price space of the market reveals the collective estimates of win probabilities but nothing else. Our goal is to find languages that are as expressive and intuitive as possible and reveal as much information as possible, while maintaining computational feasibility.

Our work is in direct analogy to work by Fortnow et. al. [6]. Whereas we explore permutation combinatorics, Fortnow et. al. explore Boolean combinatorics. The authors consider a state space of the 2^n possible outcomes of n binary variables. Traders express bets in Boolean logic. The authors show that divisible matching is co-NP-complete and indivisible matching is Σ_2^p-complete.

Hanson [9] describes a *market scoring rule* mechanism which can allow betting on combinatorial number of outcomes. The market starts with a joint probability distribution across all outcomes. It works like a sequential version of a scoring rule. Any trader can change the probability distribution as long as he agrees to pay the most recent trader

according to the scoring rule. The market maker pays the last trader. Hence, he bears risk and may incur loss. Market scoring rule mechanisms have a nice property that the worst-case loss of the market maker is bounded. However, the computational aspects on how to operate the mechanism have not been fully explored. Our mechanisms have an auctioneer who does not bear any risk and only matches orders.

Research on bidding languages and winner determination in combinatorial auctions [4, 14, 18] considers similar computational challenges in finding an allocation of items to bidders that maximizes the auctioneer's revenue. Combinatorial auctions allow bidders to place distinct values on bundles of goods rather than just on individual goods. Uncertainty and risk are typically not considered and the central auctioneer problem is to maximize social welfare. Our mechanisms allow traders to construct bets for an event with $n!$ outcomes. Uncertainty and risk are considered and the auctioneer problem is to explore arbitrage opportunities and risklessly match up wagers.

3. PERMUTATION BETTING

In this section, we define the matching and optimal matching problems that an auctioneer needs to solve in a general permutation betting market. We then illustrate the problem definitions in the context of the subset-betting and pair-betting markets.

3.1 Securities, Orders and Matching Problems

Consider an event with n competing candidates where the outcome (state) is a ranking of the n candidates. The bidding language of a market offering securities in the future outcomes determines the type and number of securities available and directly affects what information can be aggregated about the outcome. A fully expressive bidding language can capture any possible information that traders may have about the final ranking; a less expressive language limits the type of information that can be aggregated though it may enable a more efficient solution to the matching problem. For any bidding language and number of securities in a permutation betting market, we can succinctly represent the problem of the auctioneer to risklessly match offers as follows.

Consider an index set of bets or orders $\mathcal{O}$ which traders submit to the auctioneer. Each order $i \in \mathcal{O}$ is a triple (b_i, q_i, ϕ_i), where b_i denotes how much the trader is willing to pay for a unit share of security ϕ_i and q_i is the number of shares of the security he wants to purchase at price b_i. Naturally, $b_i \in (0, 1)$ since a unit of the security pays off at most \$1 when the event is realized. Since order i is defined for a single security ϕ_i, we will omit the security variable whenever it is clear from the context.

The auctioneer can accept or reject each order, or in a divisible world accept a fraction of the order. Let x_i be the fraction of order $i \in \mathcal{O}$ accepted. In the indivisible version of the market $x_i = 0$ or 1 while in the divisible version $x_i \in [0, 1]$. Further let $I_i(s)$ be the indicator variable for whether order i is winning in state s, that is $I_i(s) = 1$ if the order is paid back \$1 in state s and $I_i(s) = 0$ otherwise.

There are two possible problems that the auctioneer may want to solve. The simpler one is to find a subset of orders that can be matched risk-free, namely a subset of orders which accepted together give a nonnegative profit to the auctioneer in every possible outcome. We call this problem the *existence of a match* or sometimes simply, the matching problem. The more complex problem is for the auctioneer to find the *optimal match* with respect to some criterion such as profit, trading volume, etc.

DEFINITION 1 (EXISTENCE OF MATCH, INDIVISIBLE ORDERS). *Given a set of orders $\mathcal{O}$, does there exist a set of $x_i \in \{0, 1\}$, $i \in \mathcal{O}$, with at least one $x_i = 1$ such that*

$$\sum_i (b_i - I_i(s)) q_i x_i \geq 0, \quad \forall s \in \mathcal{S}? \tag{1}$$

Similarly we can define the existence of a match with divisible orders.

DEFINITION 2 (EXISTENCE OF MATCH, DIVISIBLE ORDERS). *Given a set of orders $\mathcal{O}$, does there exist a set of $x_i \in [0, 1]$, $i \in \mathcal{O}$, with at least one $x_i > 0$ such that*

$$\sum_i (b_i - I_i(s)) q_i x_i \geq 0, \quad \forall s \in \mathcal{S}? \tag{2}$$

The existence of a match is a decision problem. It only returns whether trade can occur at no risk to the auctioneer. In addition to the risk-free requirement, the auctioneer can optimize some criterion in determining the orders to accept. Some reasonable objectives include maximizing the total trading volume in the market or the worst-case profit of the auctioneer. The following optimal matching problems are defined for an auctioneer who maximizes his worst-case profit.

DEFINITION 3 (OPTIMAL MATCH, INDIVISIBLE ORDERS). *Given a set of orders $\mathcal{O}$, choose $x_i \in \{0, 1\}$ such that the following mixed integer programming problem achieves its optimality*

$$\begin{aligned} \max_{x_i, c} \quad & c & & \\ s.t. \quad & \textstyle\sum_i (b_i - I_i(s)) q_i x_i \geq c, & \forall s \in \mathcal{S} \\ & x_i \in \{0, 1\}, & \forall i \in \mathcal{O}. \end{aligned} \tag{3}$$

DEFINITION 4 (OPTIMAL MATCH, DIVISIBLE ORDERS). *Given a set of orders $\mathcal{O}$, choose $x_i \in [0, 1]$ such that the following linear programming problem achieves its optimality*

$$\begin{aligned} \max_{x_i, c} \quad & c & & \\ s.t. \quad & \textstyle\sum_i (b_i - I_i(s)) q_i x_i \geq c, & \forall s \in \mathcal{S} \\ & 0 \leq x_i \leq 1, & \forall i \in \mathcal{O}. \end{aligned} \tag{4}$$

The variable c is the worst-case profit for the auctioneer. Note that, strictly speaking, the optimal matching problems do not require to solve the optimization problems (3) and (4), because only the optimal set of orders are needed. The optimal worst-case profit may remain unknown.

3.2 Subset Betting

A subset betting market allows two different types of bets. Traders can bet on a subset of positions a candidate may end up at, or they can bet on a subset of candidates that will occupy a particular position. A security $\langle \alpha | \Phi \rangle$ where Φ is a subset of positions pays off \$1 if candidate α stands at a position that is an element of Φ and it pays \$0 otherwise. For example, security $\langle \alpha | \{2, 4\} \rangle$ pays \$1 when candidate α

is ranked second or fourth. Similarly, a security $\langle \Psi | j \rangle$ where Ψ is a subset of candidates pays off \$1 if any of the candidates in the set Ψ ranks at position j. For instance, security $\langle \{\alpha, \gamma\} | 2 \rangle$ pays off \$1 when either candidate α or candidate γ is ranked second.

The auctioneer in a subset betting market faces a nontrivial matching problem, that is to determine which orders to accept among all submitted orders $i \in \mathcal{O}$. Note that although there are only n candidates and n possible positions, the number of available securities to bet on is exponential since a trader may bet on any of the 2^n subsets of candidates or positions. With this, it is not immediately clear whether one can even find a trading partner or a match for trade to occur, or that the auctioneer can solve the matching problem in polynomial time. In the next section, we will show that somewhat surprisingly there is an elegant polynomial solution to both the matching and optimal matching problems, based on classic combinatorial problems.

When an order is accepted, the corresponding trader pays the submitted order price b_i to the auctioneer and the auctioneer pays the winning orders \$1 per share after the outcome is revealed. The auctioneer has to carefully choose which orders and what fractions of them to accept so as to be guaranteed a nonnegative profit in any future state. The following example illustrates the matching problem for indivisible orders in the subset-betting market.

EXAMPLE 1. *Suppose $n = 3$. Objects α, β, and γ compete for positions 1, 2, and 3 in a competition. The auctioneer receives the following 4 orders: (1) buy 1 share $\langle \alpha | \{1\} \rangle$ at price \$0.6; (2) buy 1 share $\langle \beta | \{1,2\} \rangle$ at price \$0.7; (3) buy 1 share $\langle \gamma | \{1,3\} \rangle$ at price \$0.8; and (4) buy 1 share $\langle \beta | \{3\} \rangle$ at price \$0.7. There are 6 possible states of ordering: $\alpha\beta\gamma$, $\alpha\gamma\beta$, $\beta\alpha\gamma$, $\beta\gamma\alpha$, $\gamma\alpha\beta$,and $\gamma\beta\alpha$. The corresponding state-dependent profit of the auctioneer for each order can be calculated as the following vectors,*

$$\begin{aligned} c_1 &= (-0.4, -0.4, \ 0.6, \ 0.6, \ 0.6, \ 0.6) \\ c_2 &= (-0.3, \ 0.7, -0.3, -0.3, \ 0.7, -0.3) \\ c_3 &= (-0.2, \ 0.8, -0.2, \ 0.8, -0.2, -0.2) \\ c_4 &= (\ 0.7, -0.3, \ 0.7, \ 0.7, -0.3, \ 0.7). \end{aligned}$$

6 columns correspond to the 6 future states. For indivisible orders, the auctioneer can either accept orders (2) and (4) and obtain profit vector

$$c = (0.4, \ 0.4, \ 0.4, \ 0.4, \ 0.4, \ 0.4),$$

or accept orders (2), (3), and (4) and has profit across state

$$c = (0.2, \ 1.2, \ 0.2, \ 1.2, \ 0.2, \ 0.2).$$

3.3 Pair Betting

A pair betting market allows traders to bet on whether one candidate will rank higher than another candidate, in an outcome which is a permutation of n candidates. A security $\langle \alpha > \beta \rangle$ pays off \$1 if candidate α is ranked higher than candidate β and \$0 otherwise. There are a total of $N(N-1)$ different securities offered in the market, each corresponding to an ordered pair of candidates.

Traders place orders of the form "buy q_i shares of $\langle \alpha > \beta \rangle$ at price per share no greater than b_i". b_i in general should be between 0 and 1. Again the order can be either indivisible or divisible and the auctioneer needs to decide what fraction x_i of each order to accept so as not to incur any loss, with

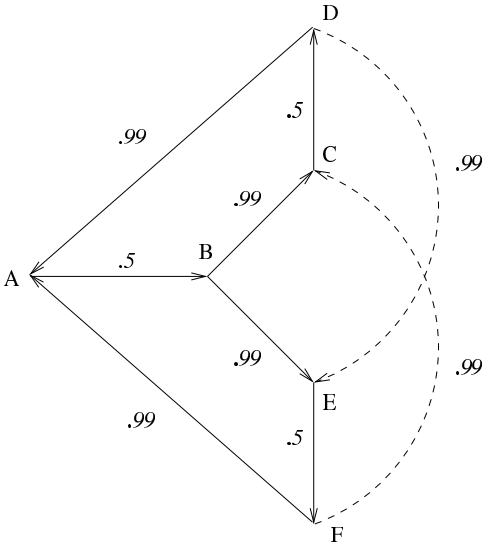

Figure 1: Every cycle has negative worst-case profit of -0.02 (for the cycles of length 4) or less (for the cycles of length 6), however accepting all edges in full gives a positive worst-case profit of 0.44.

$x_i \in \{0, 1\}$ for indivisible and $x_i \in [0, 1]$ for divisible orders. The same definitions for existence of a match and optimal match from Section 3.1 apply.

The orders in the pair-betting market have a natural interpretation as a graph, where the candidates are nodes in the graph and each order which ranks a pair of candidates $\alpha > \beta$ is represented by a directed edge $e = (\alpha, \beta)$ with price b_e and weight q_e. With this interpretation, it is tempting to assume that a necessary condition for a match is to have a cycle in the graph with a nonnegative worst-case profit. Assuming $q_e = 1$ for all e, this is a cycle C with a total of $|C|$ edges such that the worst-case profit for the auctioneer is

$$\Big(\sum_{e \in C} b_e \Big) - (|C| - 1) \geq 0,$$

since in the worst-case state the auctioneer needs to pay \$,1 to every order in the cycle except one. However, the example in Figure 1 shows that this is not the case: we may have a set of orders in which every single cycle has a negative worst-case profit, and yet there is a positive worst-case match overall. The edge labels in the figure are the prices b_e; both the optimal divisible and indivisible solution in this case accept all orders in full, $x_e = 1$.

4. COMPLEXITY OF SUBSET BETTING

The matching problems of the auctioneer in any permutation market, including the subset betting market have $n!$ constraints. Brute-force methods would take exponential time to solve. However, given the special form of the securities in the subset betting market, we can show that the matching problems for divisible orders can be solved in polynomial time.

THEOREM 1. *The existence of a match and the optimal match problems with divisible orders in a subset betting market can both be solved in polynomial time.*

PROOF. Consider the linear programming problem (4) for finding an optimal match. This linear program has $|\mathcal{O}|+1$ variables, one variable x_i for each order i and the profit variable c. It also has exponentially many constraints. However, we can solve the program in time polynomial in the number of orders $|\mathcal{O}|$ by using the ellipsoid algorithm, as long as we can efficiently solve its corresponding *separation* problem in polynomial time [7, 8]. The separation problem for a linear program takes as input a vector of variable values and returns if the vector is feasible, or otherwise it returns a violated constraint.

For given values of the variables, a violated constraint in Eq. (4) asks whether there is a state or permutation s in which the profit is less than c, and can be rewritten as

$$\sum_i I_i(s) q_i x_i < \Big(\sum_i b_i q_i x_i\Big) - c \quad \forall s \in \mathcal{S}. \tag{5}$$

Thus it suffices to show how to find efficiently a state s satisfying the above inequality (5) or verify that the opposite inequality holds for all states s.

We will show that the separation problem can be reduced to the *maximum weighted bipartite matching*[1] problem [3]. The left hand side in Eq. (5) is the total money that the auctioneer needs to pay back to the winning traders in state s. The first term on the right hand side is the total money collected by the auctioneer and it is fixed for a given solution vector of x_i's and c. A weighted bipartite graph can be constructed between the set of candidates and the set of positions. For every order of the form $\langle \alpha | \Phi \rangle$ there are edges from candidate node α to every position node in Φ. For orders of the form $\langle \Psi | j \rangle$ there are edges from each candidate in Ψ to position j. For each order i we put weight $q_i x_i$ on each of these edges. All multi-edges with the same end points are then replaced with a single edge that carries the total weight of the multi-edge. Every state s then corresponds to a perfect matching in the bipartite graph. In addition, the auctioneer pays out to the winners the sum of all edge weights in the perfect matching since every candidate can only stand in one position and every position is taken by one candidate. Thus, the auctioneer's worst-cast state and payment are the solution to the maximum weighted bipartite matching problem, which has known polynomial-time algorithms [12, 13]. Hence, the separation problem can be solved in polynomial time.

Naturally, if the optimal solution to (4) gives a worst-case profit of $c^* > 0$, there exists a matching. Thus, the matching problem can be solved in polynomial time also. □

5. COMPLEXITY OF PAIR BETTING

In this section we show that a slight change of the bidding language may bring about a dramatic change in the complexity of the optimal matching problem of the auctioneer. In particular, we show that finding the optimal match in the pair betting market is NP-hard for both divisible and indivisible orders. We then identify a polynomially-verifiable sufficient condition for deciding the existence of a match.

The hardness results are surprising especially in light of the observation that a pair betting market has a seemingly more restrictive bidding language which only offers $n(n-1)$ securities. In contrast, the subset betting market enables traders to bet on an exponential number of securities and yet had a polynomial time solution for finding the optimal match. Our hope is that the comparison of the complexities of the subset and pair betting markets would offer insight into what makes a bidding language expressive while at the same time enabling an efficient matching solution.

In all analysis that follows, we assume that traders submit unit orders in pair betting markets, that is $q_i = 1$. A set of orders $\mathcal{O}$ received by the auctioneer in a pair betting market with unit orders can be represented by a directed graph, $G(V, E)$, where the vertex set V contains candidates that traders bet on. An edge $e \in E$, denoted (α, β, b_e), represents an order to buy 1 share of the security $\langle \alpha > \beta \rangle$ at price b_e. All edges have equal weight of 1.

We adopt the following notations throughout the paper:

- $G(V, E)$: original equally weighted directed graph for the set of unit orders $\mathcal{O}$.
- b_e: price of the order for edge e.
- $G^*(V^*, E^*)$; a weighted directed graph of accepted orders for optimal matching, where edge weight x_e is the quantity of order e accepted by the auctioneer. $x_e = 1$ for indivisible orders and $0 < x_e \leq 1$ for divisible orders.
- $H(V, E)$: a generic weighted directed graph of accepted orders.
- $k(H)$: solution to the unweighted minimum feedback arc set problem on graph H. $k(H)$ is the minimum number of edges to remove so that H becomes acyclic.
- $l(H)$: solution to the weighted minimum feedback arc set problem on graph H. $l(H)$ is the minimum total weights for the set of edges which, when removed, leave H acyclic.
- $c(H)$: worst-case profit of the auctioneer if he accepts all orders in graph H.
- ϵ: a sufficiently small positive real number. Where not stated, $\epsilon < 1/(2|E|)$ for a graph $H(V, E)$. In other cases, the value is determined in context.

A *feedback arc set* of a directed graph is a set of edges which, when removed from the graph, leave a directed acyclic graph (DAG). *Unweighted minimum feedback arc set* problem is to find a feedback arc set with the minimum cardinality, while *weighted minimum feedback arc set* problem seeks to find a feedback arc set with the minimum total edge weight. Both unweighted and weighted minimum feedback arc set problems have been shown to be NP-complete [10]. We will use this result in our complexity analysis on pair betting markets.

5.1 Optimal Indivisible Matching

The auctioneer's optimal indivisible matching problem is introduced in Definition 3 of Section 3. Assuming unit orders and considering the order graph $G(V, E)$, we restate the auctioneer's optimal matching problem in a pair betting market as picking a subset of edges to accept such that

[1]The notion of perfect matching in a bipartite graph, which we use only in this proof, should not be confused with the notion of matching bets which we use throughout the paper.

worst-case profit is maximized in the following optimization problem,

$$\begin{aligned} \max_{x_e, c} \quad & c & (6) \\ \text{s.t.} \quad & \textstyle\sum_e (b_e - I_e(s)) x_e \geq c, & \forall s \in \mathcal{S} \\ & x_e \in \{0,1\}, & \forall e \in E. \end{aligned}$$

Without lose of generality, we assume that there are no multi-edges in the order graph G.

We show that the optimal matching problem for indivisible orders is NP-hard via a reduction from the unweighted minimum feedback arc set problem. The latter takes as input a directed graph, and asks what is the minimum number of edges to delete from the graph so as to be left with a DAG. Our hardness proof is based on the following lemmas.

LEMMA 2. *Suppose the auctioneer accepts all edges in an equally weighted directed graph $H(V,E)$ with edge price $b_e = (1-\epsilon)$ and edge weight $x_e = 1$. Then the worst-case profit is equal to $k(H) - \epsilon|E|$, where $k(H)$ is the solution to the unweighted minimum feedback arc problem on H.*

PROOF. If the order of an edge gets \$1 payoff at the end of the market we call the edge a winning edge, otherwise it is called a losing edge. For any state s, all winning edges necessarily form a DAG. Conversely, for every DAG there is a state in which the DAG edges are winners (though the remaining edges in G are not necessarily losers).

Suppose that in state s there are w_s winning edges and $l_s = |E| - w_s$ losing edges. Then, l_s is the cardinality of a feedback arc set that consists of all losing edges in state s. The edges that remain after deleting the minimum feedback arc set form the maximum DAG for the graph H. Consider the state s_{max} in which all edges of the maximum DAG are winners. This gives the maximum number of winning edges w_{max}. All other edges are necessarily losers in the state s_{max}, since any edge which is not in the max DAG must form a cycle together with some of the DAG edges. The number of losing edges in state s_{max} is the cardinality of the minimum feedback arc set of H, that is $|E| - w_{max} = k(H)$.

The profit of the auctioneer in a state s is

$$\begin{aligned} profit(s) &= \Big(\sum_{e \in E} b_e\Big) - w \\ &= (1-\epsilon)|E| - w \\ &\geq (1-\epsilon)|E| - w_{max}, \end{aligned}$$

where equality holds when $s = s_{max}$. Thus, the worst-case profit is achieved at state s_{max},

$$profit(s_{max}) = (|E| - w_{max}) - \epsilon|E| = k(H) - \epsilon|E|.$$

□

Consider the graph of accepted orders for optimal matching, $G^*(V^*, E^*)$, which consists of the optimal subset of edges E^* to be accepted by the auctioneer, that is edges with $x_e = 1$ in the solution of the optimization problem (6). We have the following lemma.

LEMMA 3. *If the edge prices are $b_e = (1-\epsilon)$, then the optimal indivisible solution graph G^* has the same unweighted minimum feedback arc set size as the graph of all orders G, that is $k(G^*) = k(G)$. Furthermore, G^* is the smallest such subgraph of G, i.e., it is the subgraph of G with the smallest number of edges, that has the same size of unweighted minimum feedback arc set as G.*

PROOF. G^* is a subgraph of G, hence the minimum number of edges to break cycles in G^* is no more than that in G, namely $k(G^*) \leq k(G)$.

Suppose $k(G^*) < k(G)$. Since both $k(G^*)$ and $k(G)$ are integers, for any $\epsilon < \frac{1}{|E|}$ we have that $k(G^*) - \epsilon|E^*| < k(G) - \epsilon|E|$. Hence by Lemma 2, the auctioneer has a higher worst-case profit by accepting G than accepting G^*, which contradicts the optimality of G^*. Finally, the worst-case profit $k(G) - \epsilon|E^*|$ is maximized when $|E^*|$ is minimized. Hence, G^* is the smallest subgraph of G such that $k(G^*) = k(G)$. □

The above two lemmas prove that the maximum worst-case profit in the optimal indivisible matching is directly related to the size of the minimum feedback arc set. Thus computing each automatically gives the other, hence computing the maximum worst-case profit in the indivisible pair betting problem is NP-hard.

THEOREM 4. *Computing the maximum worst-case profit in indivisible pair betting is NP-hard.*

PROOF. By Lemma 3, the maximum worst-case profit which is the optimum to the mixed integer programming problem (6), is $k(G) - \epsilon|E^*|$, where $|E^*|$ is the number of accepted edges. Since $k(G)$ is integer and $\epsilon|E^*| \leq \epsilon|E| < 1$, solving (6) will automatically give us the cardinality of the minimum feedback arc set of G, $k(G)$. Because the minimum feedback arc set problem is NP-complete [10], computing the maximum worst-case profit is NP-hard. □

Theorem 4 states that solving the optimization problem is hard, because even if the optimal set of orders are provided computing the optimal worst-case profit from accepting those orders is NP-hard. However, it does not imply whether the optimal matching problem, i.e. finding the optimal set of orders to accept, is NP-hard. It is possible to be able to determine which edges in a graph participating in the optimal match, yet unable to compute the corresponding worst-case profit. Next, we prove that the indivisible optimal matching problem is actually NP-hard. We will use the following short fact repeatedly.

LEMMA 5 (EDGE REMOVAL LEMMA). *Given a weighted graph $H(V,E)$, removing a single edge e with weight x_e from the graph decreases the weighted minimum feedback arc set solution $l(H)$ by no more than x_e and reduces the unweighted minimum feedback arc set solution $k(H)$ by no more than 1.*

PROOF. Suppose the weighted minimum feedback arc set for the graph $H - \{e\}$ is F. Then $F \cup \{e\}$ is a feedback arc set for H, and has total edge weight $l(H-\{e\}) + x_e$. Because $l(H)$ is the solution to the weighted minimum feedback arc set problem on H, we have $l(H) \leq l(H-\{e\}) + x_e$, implying that $l(H - \{e\}) \geq l(H) - x_e$.

Similarly, suppose the unweighted minimum feedback arc set for the graph $H - \{e\}$ is F'. Then $F' \cup \{e\}$ is a feedback arc set for H, and has set cardinality $k(H-\{e\})+1$. Because $k(H)$ is the solution to the unweighted minimum feedback arc set problem on H, we have $k(H) \leq k(H - \{e\}) + 1$, giving that $k(H - \{e\}) \geq k(H) - 1$. □

THEOREM 6. *Finding the optimal match in indivisible pair betting is NP-hard.*

PROOF. We reduce from the unweighted minimum feedback arc set problem again, although with a slightly more complex polynomial transformation involving multiple calls to the optimal match oracle. Consider an instance graph G of the minimum feedback arc set problem. We are interested in computing $k(G)$, the size of the minimum feedback arc set of G.

Suppose we have an oracle which solves the optimal matching problem. Denote by $optimal_match(G')$ the output of the optimal matching oracle on graph G' with prices $b_e = (1-\epsilon)$ on all its edges. By Lemma 3, on input G', the oracle $optimal_match$ returns the subgraph of G' with the smallest number of edges, that has the same size of minimum feedback arc set as G'.

The following procedure finds $k(G)$ by using polynomially many calls to the optimal match oracle on a sequence of subgraphs of G.

```
set G' := G
iterations := 0
while (G' has nonempty edge set)
    reset G' := optimal_match(G')
    if (G' has nonempty edge set)
        increment iterations by 1
        reset G' by removing any edge e
    end if
end while
return (iterations)
```

This procedure removes edges from the original graph G layer by layer until the graph is empty, while at the same time computing the minimum feedback arc set size $k(G)$ of the original graph as the number of iterations. In each iteration, we start with a graph G' and replace it with the smallest subgraph G' that has the same $k(G')$. At this stage, removing an additional edge e necessarily results in $k(G'-\{e\}) = k(G')-1$, because $k(G'-\{e\}) < k(G')$ by the optimality of G', and $k(G'-\{e\}) \geq k(G')-1$ by the edge-removal lemma. Therefore, in each iteration the cardinality of the minimum feedback arc set gets reduced exactly by 1. Hence the number of iterations is equal to $k(G)$.

Note that this procedure gives a polynomial transformation from the optimal matching problem to the unweighted minimum feedback arc set problem, which calls the optimal matching oracle exactly $k(G) \leq |E|$ times, where $|E|$ is the number of edges of G. Hence the optimal matching problem is NP-hard. □

5.2 Optimal Divisible Matching

When orders are divisible, the auctioneer's optimal matching problem is described in Definition 4 of Section 3. Assuming unit orders and considering the order graph $G(V, E)$, we restate the auctioneer's optimal matching problem for divisible orders as choosing quantity of orders to accept, $x_e \in [0, 1]$, such that worst-case profit is maximized in the following linear programming problem,

$$\begin{aligned} \max_{x_e, c} \quad & c & (7)\\ \text{s.t.} \quad & \textstyle\sum_e (b_e - I_e(s))x_e \geq c, & \forall s \in \mathcal{S} \\ & x_e \in [0,1], & \forall e \in E. \end{aligned}$$

We still assume that there are no multi-edges in the order graph G.

When orders are divisible, the auctioneer can be better off by accepting partial orders. Example 2 shows a situation when accepting partial orders generates higher worst-case profit than the optimal indivisible solution.

EXAMPLE 2. *We show that the linear program (7) sometimes has a non-integer optimal solution.*

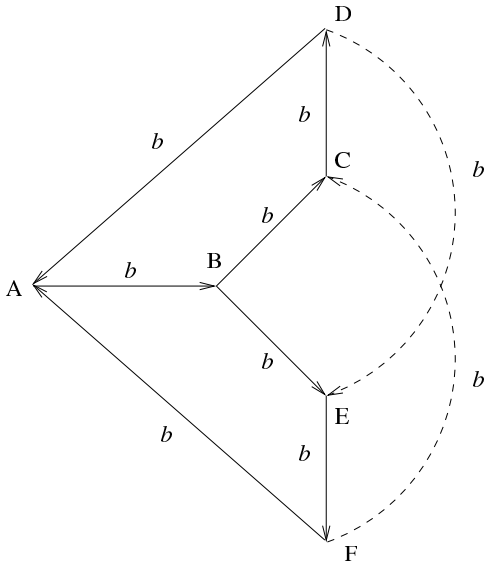

Figure 2: An order graph. Letters on edges represent order prices.

Consider the graph in Figure 2. There are a total of five cycles in the graph: three four-edge cycles ABCD, ABEF, CDEF, and two six-edge cycles ABCDEF and ABEFCD. Suppose each edge has price b such that $4b-3 > 0$ and $6b-5 < 0$, namely $b \in (.75, .80)$, for example $b = .78$. With this, the optimal indivisible solution consists of at most one four-edge cycle, with worst case profit $(4b-3)$. On the other hand, taking $\frac{1}{2}$ fraction of each of the three four-edge cycles would yield higher worst-case profit of $\frac{3}{2}(4b-3)$.

Despite the potential profit increase for accepting divisible orders, the auctioneer's optimal matching problem remains to be NP-hard for divisible orders, which is presented below via several lemmas and theorems.

LEMMA 7. *Suppose the auctioneer accept orders described by a weighted directed graph $H(V, E)$ with edge weight x_e to be the quantity accepted for edge order e. The worst-case profit for the auctioneer is*

$$c(H) = \sum_{e \in E} (b_e - 1)x_e + l(H). \quad (8)$$

PROOF. For any state s, the winning edges form a DAG. Thus, the worst-case profit for the auctioneer achieves at the state(s) when the total quantity of losing orders is minimized. The minimum total quantity of losing orders is the solution to weighted minimal feedback arc set problem on H, that is $l(H)$. □

Consider the graph of accepted orders for optimal divisible matching, $G^*(V^*, E^*)$, which consists of the optimal subset of edges E^* to be accepted by the auctioneer, with edge weight $x_e > 0$ getting from the optimal solution of the linear program (7). We have the following lemmas.

LEMMA 8. $l(G^*) \leq k(G^*) \leq k(G)$.

PROOF. $l(G^*)$ is the solution of the weighted minimum feedback arc set problem on G^*, while $k(G^*)$ is the solution of the unweighted minimum feedback arc set problem on G^*. When all edge weights in G^* are 1, $l(G^*) = k(G^*)$. When x_e's are less than 1, $l(G^*)$ can be less than or equal to $k(G^*)$. Since G^* is a subgraph of G but possibly with different edge weights, $k(G^*) \leq k(G)$. Hence, we have the above relation. □

LEMMA 9. *There exists some ϵ such that when all edge prices b_e's are $(1-\epsilon)$, $l(G^*) = k(G)$.*

PROOF. From lemma 8, $l(G^*) \leq k(G)$. We know that the auctioneer's worst-case profit when accepting G^* is

$$c(G^*) = \sum_{e \in E^*} (b_e - 1)x_e + l(G^*) = l(G^*) - \epsilon \sum_{e \in E^*} x_e.$$

When he accepts the original order graph G in full, his worst-case profit is

$$c(G) = \sum_{e \in E} (b_e - 1) + k(G) = k(G) - \epsilon|E|.$$

Suppose $l(G^*) < k(G)$. If $|E| - \sum_{e \in E^*} x_e = 0$, it means that G^* is G. Hence, $l(G^*) = k(G)$ regardless of ϵ, which contradicts with the assumption $l(G^*) < k(G)$. If $|E| - \sum_{e \in E^*} x_e > 0$, then when

$$\epsilon < \frac{k(G) - l(G^*)}{|E| - \sum_{e \in E^*} x_e},$$

$c(G)$ is strictly greater than $c(G^*)$, contradicting with the optimality of $c(G^*)$. Because x_e's are less than 1, $l(G^*) > k(G)$ is impossible. Thus, $l(G^*) = k(G)$. □

THEOREM 10. *Finding the optimal worst-case profit in divisible pair betting is NP-hard.*

PROOF. Given the optimal set of partial orders to accept for G when edge weights are $(1-\epsilon)$, if we can calculate the optimal worst-case profit, by lemma 9 we can solve the unweighted minimum feedback arc set problem on G, which is NP-hard. Hence, finding the optimal worst-case profit is NP-hard. □

Theorem 10 states that solving the linear program (7) is NP-hard. Similarly to the indivisible case, we still need to prove that just finding the optimal divisible match is hard, as opposed to being able to compute the optimal worst-case profit. Since in the divisible case the edges do not necessarily have unit weights, the proof in Theorem 6 does not apply directly. However, with an additional property of the divisible case, we can augment the procedure from the indivisible hardness proof to compute the unweighted minimum feedback arc set size $k(G)$ here as well.

First, note that the optimal divisible subgraph G^* of a graph G is the weighted subgraph with minimum weighted feedback arc set size $l(G^*) = k(G)$ and smallest sum of edge weights $\sum_{e \in E^*} x_e$, since its corresponding worst case profit is $\left(k(G) - \epsilon \sum_{e \in E^*} x_e\right)$ according to lemmas 7 and 9.

LEMMA 11. *Suppose graph H satisfies $l(H) = k(H)$ and we remove edge e from it with weight $x_e < 1$. Then, $k(H - \{e\}) = k(H)$.*

PROOF. Assume the contrary, namely $k(H - \{e\}) < k(H)$. Then by Lemma 5, $k(H - \{e\}) = k(H) - 1$. Since removing a single edge cannot reduce the minimum feedback arc set by more than the edge weight,

$$l(H) - x_e \leq l(H - \{e\}). \tag{9}$$

On the other hand $H - \{e\} \subset H$ so we have,

$$l(H - \{e\}) \leq k(H - \{e\}) = k(H) - 1 = l(H) - 1. \tag{10}$$

Combining (9) and (10), we get $x_e \geq 1$. The contradiction arises. Therefore, removing any edge with less than unit weight from an optimal divisible graph does not change $k(H)$, the minimal feedback arc set size of the unweighted version of the graph. □

We now can augment the procedure for the indivisible case in Theorem 6, to prove hardness of the divisible version, as follows.

THEOREM 12. *Finding the optimal match in divisible pair betting is NP-hard.*

PROOF. We reduce from the unweighted minimum feedback arc set problem for graph G. Suppose we have an oracle for the optimal divisible problem called *optimal_divisible_match*, which on input graph H computes edge weights $x_e \in (0, 1]$ for the optimal subgraph H^* of H, satisfying $l(H^*) = k(H)$. The following procedure outputs $k(G)$.

```
set G' := G
iterations := 0
while (G' has nonempty edge set)
    reset G' := optimal_divisible_match(G')
    while (G' has edges with weight < 1)
        remove an edge with weight < 1 from G'
        reset G' by setting all edge weights to 1
        reset G' := optimal_divisible_match(G')
    end while
    if (G' has nonempty edge set)
        increment iterations by 1
        reset G' by removing any edge e
    end if
end while
return (iterations)
```

As in the proof of the corresponding Theorem 6 for the indivisible case, we compute $k(G)$ by iteratively removing edges and recomputing the optimal divisible solution on the remaining subgraph, until all edges are deleted. In each iteration of the outer while loop, the minimum feedback arc set is reduced by 1, thus the number of iterations is equal to $k(G)$.

It remains to verify that each iteration reduces $k(G)$ by exactly 1. Starting from a graph at the beginning of an iteration, we compute its optimal divisible subgraph. We then keep removing one non-unit weight edge at a time and recomputing the optimal divisible subgraph, until the latter contains only edges with unit weight. By Lemma 11 throughout the iteration so far the minimum feedback arc set of the corresponding unweighted graph remains unchanged.

Once the oracle returns a graph G' with unit edge weights, removing any edge would reduce the minimum feedback arc set: otherwise G' is not optimal since $G' - \{e\}$ would have

the same minimum feedback arc set but smaller total edge weight. By Lemma 5 removing a single edge cannot reduce the minimum feedback arc set by more than one, thus as all edges have unit weight, $k(G')$ gets reduced by exactly one. $k(G)$ is equal to the returned value from the procedure. Hence, the optimal matching problem for divisible orders is NP-hard. □

5.3 Existence of a Match

Knowing that the optimal matching problem is NP-hard for both indivisible and divisible orders in pair betting, we check whether the auctioneer can identify the existence of a match with ease. Lemma 13 states a sufficient condition for the matching problem with both indivisible and divisible orders.

LEMMA 13. *A sufficient condition for the existence of a match for pair betting is that there exists a cycle C in G such that,*

$$\sum_{e \in C} b_e \geq |C| - 1, \tag{11}$$

where $|C|$ is the number of edges in the cycle C.

PROOF. The left-hand side of the inequality (11) represents the total payment that the auctioneer receives by accepting every unit orders in the cycle C in full. Because the direction of an edge represents predicted ordering of the two connected nodes in the final ranking, forming a cycle meaning that there is some logical contradiction on the predicted orderings of candidates. Hence, whichever state is realized, not all of the edges in the cycle can be winning edges. The worst-case for the auctioneer corresponds to a state where every edge in the cycle gets paid by \$1 except one, with $|C| - 1$ be the maximum payment to traders. Hence, if inequality (11) is satisfied, the auctioneer has non-negative worst-case profit by accepting the orders in the cycle. □

It can be shown that identifying such a non-negative worst-case profit cycle in an order graph G can be achieved in polynomial time.

LEMMA 14. *It takes polynomial time to find a cycle in an order graph $G(V, E)$ that has the highest worst-case profit, that is*

$$\max_{C \in \mathcal{C}} \left(\sum_{e \in C} b_e - (|C| - 1) \right),$$

where $\mathcal{C}$ is the set of all cycles in G.

PROOF. Because

$$\sum_{e \in C} b_e - (|C| - 1) = \sum_{e \in C} (b_e - 1) + 1 = 1 - \sum_{e \in C} (1 - b_e),$$

finding the cycle that gives the highest worst-case profit in the original order graph G is equivalent to finding the shortest cycle in a converted graph $H(V, E)$, where H is achieved by setting the weight for edge e in G to be $(1 - b_e)$.

Finding the shortest cycle in graph H can be done within polynomial time by resorting to the *shortest path* problem. For any vertex v in V, we consider every neighbor vertex w such that $(v, w) \in E$. We then find the shortest path from w to v, denoted as $path(w, v)$. The shortest cycle that passes vertex v is found by choosing the w such that $e_{(v,w)} + path(w, v)$ is minimized. Comparing the shortest cycle found for every vertex, we then can determine the shortest overall cycle for the graph H. Because the short path problem can be solved in polynomial time [3], we can find the solution to our problem in polynomial time. □

If the worst-case profit for the optimal cycle is non-negative, we know that there exists a match in G. However, the condition in lemma 13 is not a necessary condition for the existence of a match. Even if all single cycles in the order graph have negative worst-case profit, the auctioneer may accept multiple interweaving cycles to have positive worst-case profit. Figure 1 exhibits such a situation.

If the optimal indivisible match consists only of edge disjoint cycles, a natural greedy algorithm can find the cycle that gives the highest worst-case profit, remove its edges from the graph, and proceed until no more cycles exist. However, we show that such greedy algorithm can give a very poor approximation.

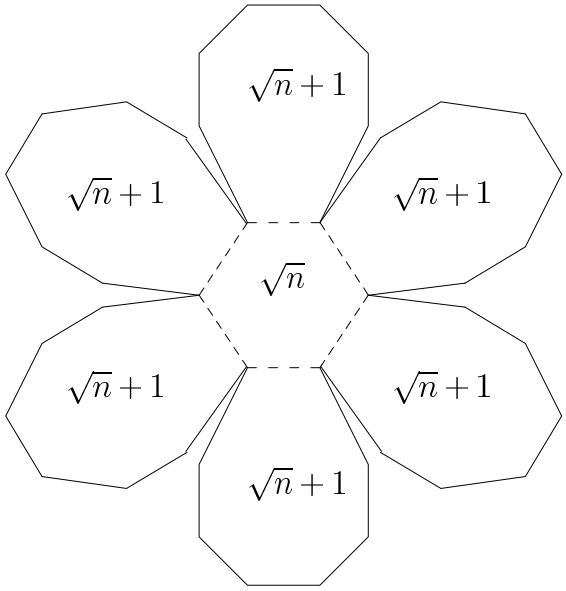

Figure 3: Graph with n vertices and $n + \sqrt{n}$ edges on which the greedy algorithm finds only two cycles, the dotted cycle in the center and the unique remaining cycle. The labels in the faces give the number of edges in the corresponding cycle.

LEMMA 15. *The greedy algorithm gives at most an $O(\sqrt{n})$-approximation to the maximum number of disjoint cycles.*

PROOF. Consider the graph in Figure 3 consisting of a cycle with $\sqrt{n}$ edges, each of which participates in another (otherwise disjoint) cycle with $\sqrt{n} + 1$ edges. Suppose all edge weights are $(1 - \epsilon)$. The maximum number of disjoint cycles is clearly $\sqrt{n}$, taking all cycles with length $\sqrt{n} + 1$.

Because smaller cycles gives higher worst-case profit, the greedy algorithm would first select the cycle of length $\sqrt{n}$, after which there would be only one remaining cycle of length n. Thus the total number of cycles selected by greedy is 2 and the approximation factor in this case is $\sqrt{n}/2$. □

In light of Lemma 15, one may expect that greedy algorithms would give $\sqrt{n}$-approximations at best. Approxima-

tion algorithms for finding the maximum number of edge-disjoint cycles have been considered by Krivelevich, Nutov and Yuster [11, 19]. Indeed, for the case of directed graphs, the authors show that a greedy algorithm gives a $\sqrt{n}$-approximation [11]. When the optimal match does not consist of edge-disjoint cycles as in the example of Figure 3, greedy algorithm trying to finding optimal single cycles fails obviously.

6. CONCLUSION

We consider a permutation betting scenario, where traders wager on the final ordering of n candidates. While it is unnatural and intractable to allow traders to bet directly on the $n!$ different final orderings, we propose two expressive betting languages, subset betting and pair betting. In a subset betting market, traders can bet either on a subset of positions that a candidate stands or on a subset of candidates who occupy a specific position in the final ordering. Pair betting allows traders bet on whether one given candidate ranks higher than another given candidate.

We examine the auctioneer problem of matching orders without incurring risk. We find that in a subset betting market an auctioneer can find the optimal set and quantity of orders to accept such that his worst-case profit is maximized in polynomial time if orders are divisible. The complexity changes dramatically for pair betting. We prove that the optimal matching problem for the auctioneer is NP-hard for pair betting with both indivisible and divisible orders via reductions to the minimum feedback arc set problem. We identify a sufficient condition for the existence of a match, which can be verified in polynomial time. A natural greedy algorithm has been shown to give poor approximation for indivisible pair betting.

Interesting open questions for our permutation betting include the computational complexity of optimal indivisible matching for subset betting and the necessary condition for the existence of a match in pair betting markets. We are interested in further exploring better approximation algorithms for pair betting markets.

7. ACKNOWLEDGMENTS

We thank Ravi Kumar, Yishay Mansour, Amin Saberi, Andrew Tomkins, John Tomlin, and members of Yahoo! Research for valuable insights and discussions.

8. REFERENCES

[1] K. J. Arrow. The role of securities in the optimal allocation of risk-bearing. *Review of Economic Studies*, 31(2):91–96, 1964.

[2] J. E. Berg, R. Forsythe, F. D. Nelson, and T. A. Rietz. Results from a dozen years of election futures markets research. In C. A. Plott and V. Smith, editors, *Handbook of Experimental Economic Results (forthcoming)*. 2001.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition)*. MIT Press and McGraw-Hill, 2001.

[4] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial Auctions*. MIT Press, Cambridge, MA, 2005.

[5] R. Forsythe, T. A. Rietz, and T. W. Ross. Wishes, expectations, and actions: A survey on price formation in election stock markets. *Journal of Economic Behavior and Organization*, 39:83–110, 1999.

[6] L. Fortnow, J. Kilian, D. M. Pennock, and M. P. Wellman. Betting boolean-style: A framework for trading in securities based on logical formulas. *Decision Support Systems*, 39(1):87–104, 2004.

[7] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[8] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin Heidelberg, 1993.

[9] R. D. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.

[10] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y.)*, pages 85–103. Plenum, New York, 1972.

[11] M. Krivelevich, Z. Nutov, and R. Yuster. Approximation algorithms for cycle packing problems. *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 556–561, 2005.

[12] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[13] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[14] N. Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00)*, Minneapolis, MN, 2000.

[15] D. M. Pennock, S. Lawrence, C. L. Giles, and F. A. Nielsen. The real power of artificial markets. *Science*, 291:987–988, February 2002.

[16] C. Plott and S. Sunder. Efficiency of experimental security markets with insider information: An application of rational expectations models. *Journal of Political Economy*, 90:663–98, 1982.

[17] C. Plott and S. Sunder. Rational expectations and the aggregation of diverse information in laboratory security markets. *Econometrica*, 56:1085–1118, 1988.

[18] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.

[19] R. Yuster and Z. Nutov. Packing directed cycles efficiently. *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2004.