# Solving Election Manipulation Using Integer Partitioning Problems

Andrew Lin
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
apl8378@cs.rit.edu

## ABSTRACT

An interesting problem of multi-agent systems is that of voting, in which the preferences of autonomous agents are to be combined. Applications of voting include modeling social structures, search engine ranking, and choosing a leader among computational agents. In the setting of voting, it is very important that each agent presents truthful information about his or her preferences, and not manipulate. The choice of election system may encourage or discourage voters from manipulating. Because manipulation often results in undesirable consequences, making the determination of such intractable is an important goal.

An interesting metric on the robustness of an election system concerns the frequency in which opportunities of manipulations occur in a given election system. Previous work by Walsh has evaluated the frequency of manipulation in the context of very specific election systems, particularly veto, when the number of candidates is limited to at most three, by showing that manipulation problems in these systems can be directly viewed as problems of (Two-Way) Partition, and then using the best known heuristics of Partition. Walsh also claimed similar results hold for $k$-candidate veto election by way of problems involving multi-way partitions.

We show that the results for $k$-candidate veto elections do not follow directly from common versions of partition problems and require non-trivial modifications to Multi-Way Partition. With these modifications, we confirm Walsh's claim that these elections are also vulnerable to manipulation. Our new computational problems also allow one to evaluate manipulation in the general case of $k$-candidate scoring protocols. We investigate the complexity of manipulating scoring protocols using new algorithms we derive by extending the known algorithms of Multi-Way Partition.

It is our conclusion that the problems of manipulation in more general scoring protocols of four or more candidates are not vulnerable to manipulation using extensions of the current known algorithms of Multi-Way Partition. This may be due to weaknesses in these algorithms or complexity in manipulating general scoring protocols.

**Cite as:** Solving Election Manipulation Using Integer Partitioning Problems, Andrew Lin, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 583–590.


## Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

## General Terms

Algorithms, Theory, Experimentation, Performance

## Keywords

computational social choice, manipulation, scoring protocols, partition

## 1. INTRODUCTION

A multi-agent system is composed of multiple interacting intelligent agents, and is used to solve problems that are otherwise difficult or impossible for an individual agent to solve. An interesting problem in multi-agent systems is that of voting [15], in which individual preferences of these agents are to be combined. Voting is used by multi-agent systems in interesting applications such as search engine ranking [17]. To ensure the integrity of the outcome of an election, an important goal in designing election systems is to eliminate or at least limit the opportunity for an agent to have an incentive to report false preferences for personal gain.

An unfortunate result, the Gibbard-Satterthwaite theorem [8, 16], shows that manipulation in inevitable in all reasonable election systems. Manipulation, also known as strategic voting, occurs when one or more voters vote contrary to their true preference ordering. A typical manipulator may bury his or her $2^{nd}$ true preference to give his first preference a larger relative advantage.

Bartholdi, Tovey, and Trick [2] attack this problem by evaluating the worst-case time complexity of computing such manipulations, and show that in many cases this problem is NP-hard, making NP-hardness the standard for worst-case hardness in manipulation problems. It has since been shown that most non-trivial weighted elections, and particularly scoring protocols, are NP-hard to manipulate [4, 9].

Since NP-hardness only demonstrates worst-case hardness and may not fully reflect the difficulty of finding manipulations in typical settings of interest, more recent work has been done to demonstrate the tradeoffs between fairness of elections and frequency of manipulation, as well as hardness of manipulation in random elections. It is now known that in some cases of random elections, as well as elections involving correlated voters such as single-peaked preferences [1, 6], manipulation is easy either in the worst-case or for the nearly all cases asymptotically.

Walsh [18] furthers the study empirically, to investigate issues such as hidden constants in theoretical asymptotic results, as well as the effects of realistic assumptions such as election systems being bounded in size. It is of interest to discover how the known asymptotic behavior applies in practice.

The weighted veto rule of three candidates was studied in Walsh, particularly because it is most directly related to the NP-hard problem of Partition, for which reasonable heuristics, particularly the Complete Karmarkar-Karp (CKK) Algorithm, are known [12]. The runtime of invoking the CKK algorithm, in particular as measured by the number of search-tree branches generated, is thus utilized throughout the study as a metric to evaluate the hardness of finding a manipulation in an election, or showing that none exists. Particularly of interest is the result that hard instances of manipulation for instances of veto elections of three candidates are "rare" and require highly correlated voters. Walsh also remarks that the Partition problem also applies in other cases of 3-candidate scoring systems, as well as veto systems of multiple candidates, without demonstrating the constructions involved.

In Walsh, the frequency of which manipulations occur for this system is determined for varying election sizes, and is shown to exhibit a smooth transition in probability in relation to the number of manipulators. More specifically, it is shown that the probability that a coalition of $m$ manipulators can influence the outcome in an election of $n$ elections is directly correlated with the ratio $\frac{\sqrt{n}}{m}$, and is independent of the problem size. This is known as a phase transition, and the majority of NP-complete problems exhibit their worst-case complexity in instances defined by this region.

It is then shown that in most cases of uncorrelated voters, including uniform votes, as well as when voter weights are normally distributed, on average one may find a manipulation or prove that none exists with very minimal search, averaging slightly more than one search-tree branch in a CKK search. This is true even in the range of the phase transition, and is in direct contrast with other NP-complete problems, in which hard problems are highly associated with this transition.

In the extreme case of correlated voters, a case in which all non-manipulative voters veto the candidate of interest, and which the manipulators have a total weight of twice the non-manipulative voters, is considered. In this case, the runtime of invoking the CKK algorithm exhibits a phase transition and grows exponentially in relation to the number of manipulators in the election. However, a further case was studied in which the election is highly correlated as before but with one additional agent who votes at random among the three candidates. Surprisingly, the runtime, as measured in search-tree branches of CKK of manipulating such an election rapidly decreases as the distribution of the weight of this non-correlated voter increases, and exhibits a strong phase transition into a constant as this distribution approaches that of the correlated voters. Based on these results, it is concluded that in the case of 3-candidate weighted veto systems, hard manipulation problems only occur when all of the votes are highly correlated, and are thus exceedingly rare in any reasonable distribution of voters. Unlike common NP-complete problems, hard instances in such manipulation problems are also not directly related to the phase transition of frequency of manipulation.

Although it is remarked in [18] that this technique can be applied to that of all scoring systems of three candidates, and that heuristics of the more general $k$-Way Partition to that of veto systems of more than three candidates, this construction as well as the problem of manipulating other scoring systems of more than three candidates was left open. We propose a solution remedy these open problem. We confirm the results of Walsh for that of $k$-candidate veto elections, but show that this problem does not directly relate to the problem of $(k-1)$-Way Manipulation, and require non-trivial adjustments. We then show how these adjustments allow one to evaluate general scoring protocols of more than three candidates.

We attack the open problem of Walsh by introducing an analogous partition-like problem for the case of general scoring systems, and extending the known algorithms and heuristics of Partition and $k$-Way Partition to that of our new problem in a very natural way. We show how problems involving partitions are related to that of manipulation, as scores given by the voters need to be partitioned among the candidates in a way to ensure a certain desirable outcome. In doing so we are able to investigate whether the known algorithms of these problems will give new results to scoring systems of interest.

Based on our analogous testing of the results of Walsh for these cases, we conclude that the algorithms of Korf, the best-known algorithms for some partition problems, support the results of Walsh for $k$-candidate veto elections of $k > 3$, but require some non-trivial adjustments to the problem instances and algorithms involved.

As our adjustments further allow one to attack the problems of manipulation in more general scoring protocols, we also study the runtime of invoking these new algorithms on such instances. It is our conclusion that the problem of manipulating general scoring protocols, as well as families of scoring protocols, such as veto, are not vulnerable to the best-known algorithms of $k$-Way Partition and analogous extensions thereof.

This may either support the fact that elections of more than three candidates are inherently more resistant to manipulation, or weaknesses in these partition algorithms. We leave these options as an open problem.

We organize the paper as follows. In the preliminaries section, we formally define relevant problems and known algorithms, as well as the connection between partition problems and election manipulation. In Section 3, we introduce our extensions of the known problem of $k$-Way Partition, which we will use to solve some instances of manipulation, and introduce analogous extended algorithms. In the next two sections, we define the connection between the problem of manipulation and the new partition problems, and demonstrate the experimental results graphically.

# 2. PRELIMINARIES

## 2.1 Election Systems

An **election** $E = (C, V)$ consists of a set of **candidates** $C = \{c_1, \ldots, c_m\}$ and **voters** $V = \{v_1, \ldots, v_n\}$. Each voter $v \in V$ presents a **preference ordering** over the candidates $C$, in the form of a complete linear ordering $c_{i_1} > \cdots > c_{i_m}$. An **election system** $\mathcal{E} : E \to C^+$ maps an election to winning candidates(s) based on the preferences of the voters, for which it aggregates. An interesting set of election

systems, scoring protocols, which we will evaluate in this paper, encompasses systems such as plurality, veto, and the Borda count where the number of candidates is fixed.

An $m$-candidate **scoring protocol** is defined as a vector $(\alpha_1, \ldots, \alpha_m)$. In such an election system, each voter contributes $\alpha_i$ points to the $i^{th}$ choice of his or her preference ordering. The candidate with the highest score wins. An election defined by such a scoring protocol can further be weighted, in which each voter $v$ is given a non-negative integer weight $w(v)$. In this case, the vote is counted as $w(v)$ non-weighted votes, and thus the $i^{th}$ choice of candidate $c$ is awarded $w(v)\alpha_i$ points. A prominent example of weighted elections occurs in the U.S. Presidential Elections, in which the Electoral Colleges are given different weights. Weights are also commonly used in computational elections, such as search engine ranking aggregation [17, 5].

A **family of scoring protocols** is an infinite series of scoring protocols $(\alpha^1, \ldots, \alpha^m, \ldots)$ in which $\alpha^m = (\alpha_1^m, \ldots, \alpha_m^m)$ is an $m$-element scoring protocol.

## 2.2 Manipulation

A common problem with many election systems is the incentive for some voters to vote contrary to their true preferences, either individually or collectively in a coalition, manipulating the outcome. This can occur when some voters vote for their $2^{nd}$ preference to avoid "wasting a vote" on their favorite candidate whom is not popular, or bury their $2^{nd}$ preference to give their first preference a larger relative advantage. Unfortunately, several early results have shown that the existence of such strategies in elections is inevitable [8, 16], and an early compromise was made to make the determination of such results at least NP-hard [2]. We define manipulation as a decision problem as follows.

**Name: $\mathcal{E}$-Manipulation**

**Instance:** Candidates $C$, established voters $V$, unestablished voters $V'$, and distinguished candidate $p \in C$.

**Question:** Is there an assignment of preference profiles over $C$ for $V'$ such that $p$ is a winner of the election $(C, V \cup V')$?

Although this problem is indeed NP-hard for some relatively simple election systems, in particular almost all nontrivial weighted scoring protocols [9], more recent papers have focused outside of worst-case complexity. A notable result [18] shows that for some election systems, particularly the veto system and other systems for three candidates, instances where it is easy to find a manipulation, or demonstrate none exists, are very rare. This result was shown using the known algorithms and heuristics for Partition, an NP-hard problem closely related to manipulating weighted three-candidate election systems. We make a definition in the next section.

## 2.3 Set Partition

A primitive set partition problem is Two-Way Partition, more simply known as Partition, which is NP-complete. We give the decision version of the problem as follows.

**Name: Partition**[11] (See also [7])

**Instance:** A multi-set of positive integers $S = \{s_1, \ldots, s_n\}$.

**Question:** Is there a subset $A \subseteq S$ such that $\sum A = \sum (S - A)$?

In the optimization version of Partition, we wish to minimize the maximum of the two subsets, namely, $\texttt{Max}(\sum A, \sum (S - A))$. Several heuristics exist to approximate this figure.

We give an example of the connection between election manipulation of three-candidate veto election systems and Partition given in Walsh as follows.

Consider a three-candidate veto election over the candidates $p$, $c_1$, and $c_2$, in which we wish for $p$ to win. Suppose that initially, five voters, of weights 10, 8, 6, 4, and 2 veto $p$, $c_1$, $p$, $c_2$, and $c_2$ respectively. In addition, our coalition consists of four voters of weights 6, 5, 4, and 3.

Without loss of generality, none of the four manipulators will veto $p$, and in this example, we must distribute vetoes of weights $\{6, 5, 4, 3\}$ among candidates $c_1$ and $c_2$, currently with vetoes of total weight 8 and 6, such that each receives vetoes of weight totaling at least 16.

Since $8 - 6 = 2$, this corresponds to a partitioning of the elements $\{6, 5, 4, 3, 2\}$ such that each side has weight at most 10 (or equally, at least 10).

For this application, as well as many others, Partition has been extended to that of $k$-Way Partition, in which the goal is to divide the set into $k$ equal subsets. It is noted in [14] that there are at least three optimization functions of interest: we may wish to minimize the maximum subset sum, maximize the minimum subset sum, or minimize the maximum difference between the sums of each two subsets. It is further demonstrated that all three of these optimization functions can produce different optimal partitions. The first two functions are of interest in our problem. The first optimization is interesting because we want the maximum score given to the non-distinguished candidates not to exceed the final score of our distinguished candidate. The second optimization is of interest in cases where vetoes are counted. We define the problem and these two optimizations as follows.

**Name: k-Way Partition**[11] (See also [7])

**Instance:** A multi-set of positive integers $S = \{s_1, \ldots, s_n\}$.

**Question (decision):** Are there disjoint and covering subsets $S = A_1 \cup \cdots \cup A_k$ such that $\sum A_1 = \cdots = \sum A_k$?

**Question (optimization #1):** Find disjoint and covering subsets $S = A_1 \cup \cdots \cup A_k$ that minimizes

$\texttt{Max}(\sum A_1, \ldots, \sum A_k)$.

**Question (optimization #2):** Find disjoint and covering subsets $S = A_1 \cup \cdots \cup A_k$ that maximizes

$\texttt{Min}(\sum A_1, \ldots, \sum A_k)$.

In the coming sections, we will describe the algorithms for the first optimization. The algorithms for the second optimization follow symmetrically with some minor adjustments.

Although it is mentioned in [18] that manipulation of veto elections of more than three candidates can be resolved as problems of multi-way partition, some adjustments must be made to this problem. We give an example and demonstrate the adjustments as follows.

Consider a four-candidate veto election over the candidates $p$, $c_1$, $c_2$, and $c_3$, in which we wish for $p$ to win. Suppose that initially, four voters, of weights 20, 12, 9, and 7 veto $p$, $c_1$, $c_2$, and $c_3$ respectively. In addition, our coalition consists of six voters of weights 10, 8, 4, 4, 3, and 3.

In this example, we must distribute vetoes of weights $\{10, 8, 4, 4, 3, 3\}$ among candidates $c_1$, $c_2$, and $c_3$, currently with vetoes of total weight 12, 9, and 7, such that each receives vetoes of weight totaling at least 20.

Since $12 - 9 = 3$ and $12 - 7 = 5$, we are interested in partitions of the set $\{10, 8, 4, 4, 3, 3\} \cup \{3, 5\}$. However, in this case, we are interested in partitions in which 3 and 5 are not placed in the same subset, as these two subsets represent the vetoes given to candidates $c_2$ and $c_3$, respectively. This requires some adjustments to the partition problem and algorithms of interest, which our work in later sections encompasses.

Furthermore, as $k$-Way Partition problems partition only individual numbers, the application of this problem and its algorithms in that of manipulation of scoring protocols is limited to cases of plurality and veto, in which each vote is determined by a single candidate. For general scoring protocols such as Borda, we will need to introduce analogous partition problems.

## 2.4 Algorithms of Partition and $k$-Way Partition

Because Partition is NP-complete, several heuristics have been developed to approximate the best partition. Two heuristics of common use are the greedy method, which first sorts the elements of $S$ in non-ascending order, and places each element in the set that minimizes the difference iteratively, and the Karmarkar-Karp heuristic [10], also known as the differencing heuristic, which decides that the two largest elements are in different sets, but defers deciding in which set each element is placed. Both of these heuristics can be modified into pruned exhaustive searches [12].

We give an example of the Karmarkar-Karp heuristic as follows. We are given the multi-set $\{6, 4, 3, 3, 2, 2\}$, which we wish to partition. We place 6 and 4 in opposite subsets. By inserting these two elements in opposite subsets, we effectively create a new element equal to the difference of the two largest elements, since we are only concerned about the total difference. We thus are creating the new element of 2, resulting in multi-set $\{3, 3, 2, 2, 2\}$. We then place each 3 in different subsets, and two elements of 2 in different subsets, resulting in $\{2\}$. In the base case of a single element, we must place it in one of the two subsets. In this case, this algorithm gives a partition of difference 2. Note that in this case the optimal partition has a difference of 0, as $6 + 4 = 3 + 3 + 2 + 2$, and this algorithm is not optimal.

Both of these heuristics can be extended to that of a complete algorithm using a depth-first tree search. The construction of the Complete Karmarkar-Karp algorithm is given in [12], which we briefly review. We note that in any given instance of Two-Way Partition, the two largest elements may be in different subsets or the same subset. By the heuristic, we always try the former first, and terminate if the partition is perfect, or within our desired maximum. The search is also pruned if the first element is greater than or equal to the sum of the remaining elements, as the best partition places the first element in one subset and the remaining elements in the other.

An early result by Korf [12] showed how the greedy and Karmarkar-Karp heuristics, as well as the corresponding complete algorithms, can be extended to the case of $k$-Way Partition. In the case of the greedy algorithm, we search a $k$-ary tree in which we try inserting elements into each of the $k$ subsets. In the corresponding CKK algorithm, we will need to try each combination of the largest two tuples, of which there are $k!$.

In [13], the runtime of the greedy and CKK heuristics are evaluated $k$-Way Partition, and two new algorithms of a different type, which utilizes CKK for 2-Way Partition recursively, are introduced. Interestingly, while CKK is still more efficient than the greedy heuristic for 3-Way Partition, the greedy heuristic has a better runtime for $k$-Way Partition for $k \geq 4$, due to the $k!$-ary search tree.

Two new algorithms for $k$-Way Partition, Sequential Number Partitioning (SNP) and Recursive Number Partitioning (RNP) were also introduced in [13]. In Sequential Number Partitioning, one complete subset is first chosen, and the remaining unpartitioned numbers are partitioned recursively using this algorithm for $(k-1)$-Way Partition. In the base case, 2-Way Partition is evaluated using the CKK algorithm. The subsets are generated by inclusion-exclusion tree search with various pruning techniques.

There are several pruning techniques for the search of this first complete subset. First, to avoid symmetry, the first subset sum of this subset is restricted to be no more than $\lfloor \frac{t}{k} \rfloor$, where $t$ is the sum of the elements, and the subsets are subsequently chosen in non-descending order by sum. Also, if $m$ is the maximum subset sum of the best partition found thus far, or our target maximum, we restrict the first subset sum to be at least $t - (k-1)m$, as otherwise the best partition found utilizing this first subset sum cannot beat this difference. To induce as much pruning as early as possible in the search tree, the elements are considered in non-ascending order, and we try including first.

In Recursive Number Partitioning, on an instance of $k$-Way Partition for $k$ even, the set is first divided into two subsets, each of which will be partitioned $\frac{k}{2}$ ways, by a top-level CKK search. In [13, 14], it is shown that both SNP and RNP show a marked improvement over CKK for $k$-Way Partition when $k \geq 3$, with RNP significantly faster than SNP for $k \geq 4$.

We wish to introduce a further extension of Partition in order to investigate the complexity of manipulating a general scoring system of weighted voters. In this paper, we demonstrate a natural extension of the above algorithms to this problem and also how to apply this new problem to the problem of manipulation in scoring systems. In doing so, we wish to gain a better understanding of the complexity of such manipulation problems in relation to the current best known algorithms of such partition-type problems.

# 3. EXTENSIONS OF $k$-WAY PARTITION

In our new problem of $k$-Way Permutation Partition, we are given a multiset of tuples, each of cardinality $k$. We wish to find permutations of each tuple such that, if we take the sum of each of the $k$ positions among the tuples, the $k$ sums are equal. As in the case of $k$-Way Partition, there are a few interesting optimization functions, of which two are of interest to manipulation problems. We give a formal definition as follows.

**Name: $k$-Way Permutation Partition**

**Instance:** A multi-set of $k$-tuples, $S = \{x_1, \ldots, x_n\}$, where $x_i = \{x_i^1, \ldots, x_i^k\}$. It can be assumed without loss of generality that $x_i^k = 0$ for all $1 \leq i \leq n$, as one may normalize the tuple, noting that only the difference among tuple elements is crucial.

**Question (decision):** Is there a mapping $P : \{1, \ldots, n\} \rightarrow S_{\{1,\ldots,k\}}$[1] such that

$$\sum_{1 \leq r \leq n} x_r^{P(r)_1} = \cdots = \sum_{1 \leq r \leq n} x_r^{P(r)_k}?$$

**Question (optimization #1):** Find the mapping $P : \{1, \ldots, n\} \rightarrow S_{\{1,\ldots,k\}}$ that minimizes

$$\texttt{Max}_{1 \leq i \leq k} \sum_{1 \leq r \leq n} x_r^{P(r)_i}.$$

**Question (optimization #2):** Find the mapping $P : \{1, \ldots, n\} \rightarrow S_{\{1,\ldots,k\}}$ that maximizes

$$\texttt{Min}_{1 \leq i \leq k} \sum_{1 \leq r \leq n} x_r^{P(r)_i}.$$

Note that $k$-Way Partition is a special case of this problem, in which each $k$-tuple has the form $(x_i^1, 0, \ldots, 0)$.

In this paper, we will examine extensions of algorithms introduced by Korf in [12, 13, 14] from Partition and $k$-Way Partition to that of $k$-Way Permutation Partition, their relationship to that of more general scoring systems, the distribution of the complexity of instances in evaluating such using these algorithms, and thus the frequency of hard instances of this problem in a system of uniform voters. Using these algorithms, we wish to show whether and when the results of [18] can be extended to that of other scoring systems using these heuristics.

In the rest of this section, we examine how each of the known algorithms for $k$-Way Partition may be extended to that of $k$-Way Permutation Partition. Such algorithms allow us to examine the complexity of manipulating some scoring protocols.

# 4. $K$-WAY PERMUTATION PARTITION AS A RESTRICTED $K$-WAY PARTITION PROBLEM

Our extensions of the SNP and RNP algorithms to that of $k$-Way Permutation Partition stem from the observation that $k$-Way Permutation Partition is a restricted version of the $k$-Way Partition problem. More specifically, given tuples $S = \{x_1 = \{x_1^1, \ldots, x_1^k\}, \ldots, x_n = \{x_n^1, \ldots, x_n^k\}\}$, we wish to find a k-Way Partition of the multiset of the union of all elements, $\{x_1^1, \ldots, x_1^k, \ldots, x_n^1, \ldots, x_n^k\}$, excluding zeros, with the additional constraint that each of the $k$ subsets contains at most one element from each of the $n$ tuples. As both the SNP and RNP algorithm work in a divide-and-conquer manner in which subpartitions are generated in sequence, this constraint can be implemented by restrictions in branching in each node of the search tree. We give a brief description of how these restrictions are implemented.

[1] $S_A$ is the set of all permutations over $A$.

In the SNP algorithm, we wish to choose a set of elements for our first subset. As in $k$-Way Partition, we consider the elements in non-ascending order in an inclusion-exclusion search tree. In the algorithm for $k$-Way Permutation Partition, each element is labeled with its corresponding tuple, and in the top-level inclusion-exclusion search, if an element from this tuple has already been chosen, we must exclude this element from the set. On the other hand, if this is the last element from its tuple in the elements to be considered, we must include it. We then partition the remaining elements $k-1$ ways, normalizing the tuples if necessary. Note that since zero elements are excluded, this algorithm is also consistent with the SNP algorithm given by Korf.

There also exists a similarly well-defined extension of the RNP algorithm proposed by Korf. In our extension of RNP on $k$-Way Permutation Partition, we perform the CKK algorithm on the elements, which are labeled with their corresponding tuples, under the constraint that each of the two subsets receive no more than $\frac{k}{2}$ elements from each tuple. As each of the two branches in a CKK search node entail combining two subpartitions, we achieve this by refraining from combining subpartitions in which one or both subsets exceeds $\frac{k}{2}$ elements from any tuple. There also exists similar simple pruning techniques, that exclude search nodes in which we cannot attain a subset minimum we used in pruning, or that cannot beat the best partition found thus far.

In both algorithms, the base case of 2-Way Permutation Partition can be solved as a case of 2-Way Partition, as we may normalize so that there is only one non-zero element in each 2-tuple.

We consider an example as follows: partition the tuples $\{\{8,0,0,0\},\{6,2,0,0\},\{5,5,2,0\},\{5,5,0,0\},\{5,4,3,0\}\}$ optimally. In this algorithm we sort the union of the elements of each tuple, excluding zero, arriving at the multiset $\{8_1, 6_2, 5_3, 5_3, 5_4, 5_4, 5_5, 4_5, 3_5, 2_2, 2_3\}$. We note the tuple each element originates from as a subscript. The first branch of the top-level CKK search combines elements $8_1$ and $6_2$, adding a new element of net weight 2 to this set. Since we are interested in the further partitioning of the 2-Way Partition of this set, we must keep track of how we arrive at our net elements. We could represent this resulting multiset as follows: $\{5_3, 5_3, 5_4, 5_4, 5_5, 4_5, 3_5, 2 = 8_1 - 6_2, 2_2, 2_3\}$. Skipping a few steps, eventually this top-level CKK algorithm produces a partition of sets $\{6_2, 5_3, 5_4, 4_5, 3_5, 2_3\}$ and $\{8_1, 5_3, 5_4, 5_5, 2_2\}$ at its first left-hand-side leaf. This corresponds to two instances of 2-Way Permutation Partition: $\{\{0,0\},\{6,0\},\{5,2\},\{5,0\},\{4,3\}\}$ and $\{\{8,0\},\{2,0\},\{5,0\},\{5,0\},\{5,0\}\}$. In this case, since we are interested in 4-Way Permutation Partition, we evaluate two instances of 2-Way Permutation Partition for each leaf of the top-level CKK instance.

We give some pseudocode for an implementation of such an algorithm. In our code below, the $\texttt{RNP}_{\texttt{inner}}$ function takes a multi-set of weights, each associated with a difference between the sums of two sets of elements. In each of these two sets of elements, each element is associated with the tuple it originated from. For example, in the example above, in the first node, we formed an element 2, which was derived from the difference of element 8 and 6 from tuples 1 and 2 respectively. In the code below, recall that we are minimizing the maximum subset sum. We store this figure in the variable `best`.

$\mathbf{RNP_{inner}}(S = \{(s_1, A_1, B_1), (s_2, A_2, B_2), \ldots, (s_n, A_n, B_n)\}, k)$, where $s_1 \geq \cdots \geq s_n$, $s_i = \sum A_i - \sum B_i$. $A_i$ and $B_i$ are multisets of elements, each element labeled with the tuple it originated from.

**if** $n = 1$ [Base case.]
  [Recursively partition the tuples corresponding to sets $A_1$ and $B_1$ $\frac{k}{2}$ ways, after normalizing.]
  **return** $\texttt{Max}(\texttt{RNP}(\texttt{Normalize}(A_1)), \texttt{RNP}(\texttt{Normalize}(B_1)))$

**else**
  [Try difference if possible.]
  **if** each of $A_1 \cup B_2$ and $B_1 \cup A_2$ sum to at most $k(\texttt{best}-1)$ and contain at most $\frac{k}{2}$ elements from each tuple
    $\texttt{best} \leftarrow \texttt{Min}(\texttt{best}, \texttt{RNP}_{\texttt{inner}}(\{$
      $(s_1 - s_2, A_1 \cup B_2, B_1 \cup A_2), \ldots,$
      $(s_n, A_n, B_n)\}, k))$

  [Try sum if possible.]
  **if** each of $A_1 \cup A_2$ and $B_1 \cup B_2$ sum to at most $k(\texttt{best}-1)$ and contain at most $\frac{k}{2}$ elements from each tuple
    $\texttt{best} \leftarrow \texttt{Min}(\texttt{best}, \texttt{RNP}_{\texttt{inner}}(\{$
      $(s_1 + s_2, A_1 \cup A_2, B_1 \cup B_2), \ldots,$
      $(s_n, A_n, B_n)\}, k))$

  **return** best

At the top level, we break the tuples into individual elements, with corresponding trivial element sets.

$\mathbf{RNP}(S = \{(s_1^1, \ldots, s_k^1), \ldots, (s_1^m, \ldots, s_k^m)\})$

  $\texttt{best} \leftarrow \infty$
  [Each element by itself, labeled with its originating tuple.]
  **return** $\texttt{RNP}_{\texttt{inner}}(\{(s_1^1, \{(s_1^1, 1)\}, \emptyset), \ldots,$
    $(s_k^m, \{(s_k^m, m)\}, \emptyset)\}, k)$

The main difference between the implementation of RNP for this restricted RNP problem and that of Korf's implementation for k-Way Partition is the extra restriction for combining weights, as the subsets involved must not exceed $\frac{k}{2}$ elements from any tuple. The SNP algorithm can be implemented.

# 5. MANIPULATION AS A PARTITION PROBLEM

We resolve manipulation in scoring systems into instances of $k$-Way Permutation Partition. There are two relevant cases.

THEOREM 1. *Consider an election of the scoring system $(\alpha_1, \ldots, \alpha_k)$ and candidates $c_1, \ldots, c_k$ with initial scores $s_1, \ldots, s_k$. Let the set of manipulative voters be $V'$ of cardinality $||V'|| = m$ with weights $w_1, \ldots, w_m$.*

*There exists a manipulation ensuring the victory of $c_1$ iff there exists a $(k-1)$-way permutation partition of the tuples $\{\{s_2, \ldots, s_k\}\} \cup \{\{w_1\alpha_2, \ldots, w_1\alpha_k\}, \ldots, \{w_m\alpha_2, \ldots, w_m\alpha_k\}\}$ with maximum subset sum of at most $s_1 + \alpha_1 \sum_{1 \leq i \leq m} w_i$.*

*It should be noted that in order to apply this reduction, it is only necessary that the each voter's contribution to the scores of each candidate depend only upon his or her position in the voter's preference ordering. It is not necessary for the system to have a fixed scoring vector. Such elections may or may not be of interest to computational social choice theory, and will not be evaluated in this paper.*

PROOF. Without loss of generality, each voter from $V'$ will choose $c_1$ as his or her first preference, giving it a final score of $s_1 + \alpha_1 \sum_{1 \leq i \leq m} w_i$. To ensure the victory of $c_1$, none of the final scores of candidates $c_2, \ldots, c_k$ can exceed this figure. As each voter of weight $w_i$ is free to choose a permutation of scores $w_i\alpha_2, \ldots, w_i\alpha_k$ for candidates $c_2, \ldots, c_k$, this corresponds to the problem of $k$-Way Permutation Partition containing a vector containing these scores for each voter in $V'$, as well as a vector representing the current scores of the $k-1$ non-distinguished candidates.

A manipulation ensuring the victory of $c_1$ thus exists iff a such a $(k-1)$-way permutation partition not exceeding the final score of $c_1$ mentioned earlier exists. □

THEOREM 2. *Consider a $q$-veto election of candidates $c_1, \ldots, c_k$ with initial scores $s_1, \ldots, s_k$. Let the set of manipulative voters be $V'$ of cardinality $||V'|| = m$ with weights $w_1, \ldots, w_m$. Without loss of generality, suppose $s_k \geq s_2, \ldots, s_{k-1}$.*

*There exists a manipulation ensuring the victory of $c_1$ iff there exists a $(k-1)$-way permutation partition of the tuples $\{\{s_k - s_2, \ldots, s_k - s_k\}\} \cup \{\{\underbrace{w_1, \ldots, w_1}_{q}, 0, \ldots, 0\}, \ldots, \{\underbrace{w_m, \ldots, w_m}_{q}, 0, \ldots, 0\}$ with minimum subset sum of at least $s_k - s_1$.*

PROOF. In this case, we are counting the total weight of the vetoes each non-distinguished candidates receives. Without loss of generality, no vetoes are given to the distinguished candidate. With some modification, this reduction also applies to cases of scoring protocols of the form $(\underbrace{\alpha, \ldots, \alpha}_{k-q}, \alpha_{k-q+1}, \ldots, \alpha_k)$ for $\alpha > \alpha_{k-q+1} \geq \ldots \geq \alpha_k$.

□

# 6. EXPERIMENTAL RESULTS

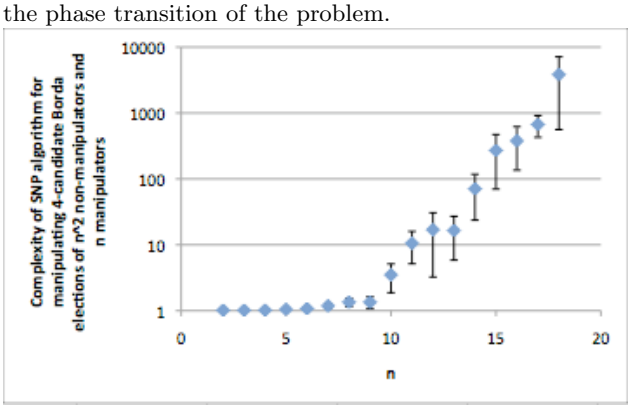
We test the feasibility of using algorithms of $k$-Way Permutation Partition for solving manipulation problems of various election systems using the connections in the previous section. To eliminate the issues of computer architecture and focus on the algorithm computationally, in each case, as a benchmark, we count the number of branches evaluated in invoking the algorithms in question, as opposed to runtime, for cases of interest. We also evaluate the standard error to provide 95% confidence intervals on the experimental data, to ensure statistical significance.

We choose voter weights uniformly from the range [0, 65536), and voter preference orderings uniformly as a random permutation. This is consistent with the definition of random voters in [18].

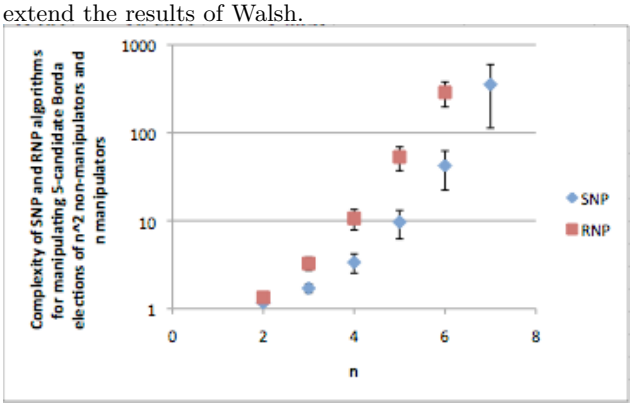
Our first test is for the case of manipulating weighted 2-approval and veto elections of a fixed number, $k$, of candidates, as the $(k-1)$-Way Permutation Problem instances for these cases are relatively similar to that of $(k-1)$-Way Partition. We found that, for these simple cases, the runtime

is similar to that of the 3-candidate cases tested by Walsh, that most cases can be solved in an average of about one branch. Particularly of interest, hard instances are not directly related to the phase transition of this problem. However, these results require the use of $(k-1)$-Way Permutation Partition, due to the arbitrary nature of the initial scores. Due to the nature of the reductions involved, we presume that this result also applies to scoring protocols of the form $(\alpha_1, \alpha_2, 0, \ldots, 0)$ for $\alpha_1 > \alpha_2$.

We test the case of 4-candidate Borda elections, which is the simplest case of a scoring protocol not of the form mentioned above. As with all of the plots in this section, we plot the 95% confidence interval of the mean of our testing. As demonstrated in [18] for elections of three candidates, the phase-transition of this problem also occurs for $m = cn^2$ for some constant $c > 0$. In the following chart, we plot the runtime of invoking the SNP algorithm for an instances of manipulating 4-candidate Borda elections of $m = n^2$ non-manipulative voters and $n$ manipulators, as this is within the phase transition of the problem.

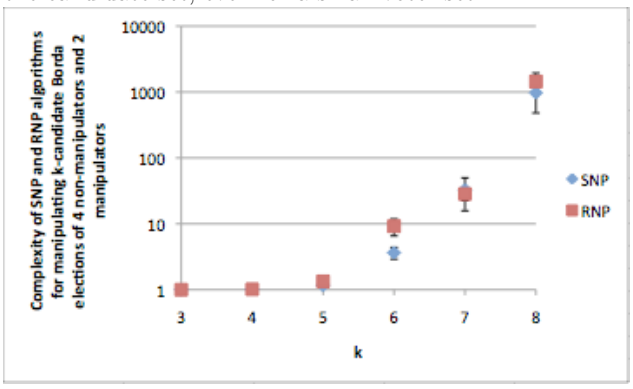



As we can see in this plot, the runtime of evaluating uniformly random instances of manipulating weighted 4-candidate Borda elections soars exponentially for instances near the phase transition of this problem. This result is in direct contrast to the results in Walsh for 3-candidate elections. Our next test case involves 5-candidate Borda elections, for which we have a choice between the SNP and RNP algorithms. As seen below, we observe that the RNP algorithm is significantly slower than that of SNP, contrary to the results in Korf in the general setting. Neither algorithms extend the results of Walsh.

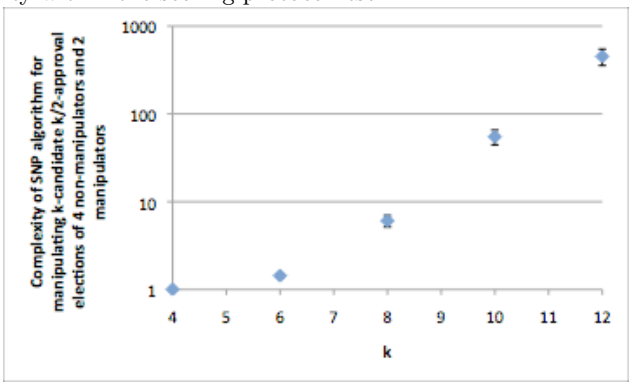



Our next two tests involves scaling the number of candidates in the election. We wish to investigate how the runtime complexity of the algorithms in question scale with the number of candidates in the election.

Below, we are plotting the runtimes of our algorithms on the case of manipulating $k$-candidate Borda elections for a fixed number of non-manipulators and manipulators, 4 and 2 respectively. The exponential results demonstrate that the SNP and RNP algorithms do not scale well with the size of the candidate set, even for a small voter set.




In our last test case below, we test a simplier family of scoring protocols, that of $k$-candidate $\frac{k}{2}$-approval elections, using the SNP algorithm (the runtime of the RNP algorithm is similar). This case is of interest as it is the most complex approval case for our algorithms, as one may choose to partition approvals or vetoes. The exponential nature of these results also show that the runtime of these algorithms stem from the number of candidates, as opposed to the complexity within the scoring protocol itself.




## 7. CONCLUSIONS AND FUTURE WORK

These new algorithms show how the problems of k-Way Partition and the known algorithms of such can be applied to the seemingly unrelated problem of manipulating weighted scoring systems. These new findings allow one to develop a better understanding of the inner workings of these manipulation problems. Known algorithms of partition-type problems, including Sequential Number Partitioning (SNP) and Recursive Number Partitioning (RNP) have the property of finding a good partition relatively quickly when one exists, and this property appears to extend to that of our new problems. Although RNP has been demonstrated to be faster than SNP for the special case of $k$-Way Partition, this simplification does not appear to extend more generally here, particularly for more complex systems such as Borda.

In each case, the best-known algorithms for Korf, and very natural extensions and generalizations thereof, do not validate the conclusions of Walsh and Nisan for elections of more than three candidates, but instead show that scoring protocols in general are not directly vulnerable to manipu-

lation via the known algorithms of partition-like problems. However, this may be due to either the strength of elections having more than three candidates, or weakness of the best-known algorithms for $k$-Way Partitioning and the extensions given in this paper. We leave this as an open problem.

As an open problem, we note that there are also inherent weaknesses of the known algorithms of partition. The obvious weakness of RNP, for instance, is that it can only be applied to cases of $k$-Way Permutation Partition for $k$ even. This weakness is demonstrated clearly in Korf, in which it is shown that 3-Way Partition is almost as slow as 4-Way Partition, and by extension, similar results hold for $k$-Way Permutation Partition. In the cases where RNP would otherwise be more efficient than SNP, it is not available as an option for $k$ odd as the CKK algorithm used at the top-level division is designed to enumerate partitions which are very close to even.

A second weakness of RNP relates to the top-level CKK division, which divides the initial set of labeled elements into two. Recall that in the CKK algorithm, the algorithm tracks a list of 2-way partitions of subsets of the original set. If either subset in any of the 2-way partitions in such a list cannot be furthered partitioned $\frac{k}{2}$ ways in a way better than the best partition found so far, it is clearly fruitless to continue on this search branch, as the final 2-way partition cannot yield a better overall $k$-way partition. One way to capitalize on this insight is to perform the algorithm recursively on the instances of $\frac{k}{2}$-Way Permutation Partition at the search nodes of interest. Unfortunately, this will only prune the top-level search tree only if a better partition does not exist, and may be expensive, as the number of such nodes, absent pruning, may be exponential. A possible open problem of interest is thus a study of the tradeoffs in making such prunings, determining fast heuristics of when to perform this test within the tree.

Resolving these two weaknesses of the RNP algorithm can help explain the anomaly of why despite being faster than SNP in $k$-Way Partition [13, 14], it behaved unusually slow in general for $k$-Way Permutation Partition.

Another problem of interest involves using variations of these new algorithms for polynomial-time approximation algorithms, a problem studied in the context of different election systems in [3]. Since it is also known that the CKK algorithm, and possibly the extensions thereof, are especially fast at finding a good partition when one exists, approximation and probabilistic algorithms may exist to capitalize on the tradeoffs in performing an incomplete search. Interesting optimization and approximation functions may include minimizing the number of manipulators needed, or relaxing the restrictions of the manipulation problem. The problem Partition has a polynomial-time approximation scheme (PTAS), which may also be extended to some of the extensions used here.

## 8. ACKNOWLEDGEMENTS

We wish to offer our special thanks to Dr. E. Hemaspaandra and three anonymous referees for their input.

This work is supported in part by NSF grant IIS-0713061

## 9. REFERENCES

[1] M. Ballester and G. Haeringer. A characterization of single-peaked preferences. *UFAE and IAE Working Papers*, 2006.

[2] J. Bartholdi, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, pages 227–241, 1989.

[3] E. Brelsford, P. Faliszewski, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Approximability of manipulating elections. *Proceedings of the* $23^{rd}$ *AAAI Conference*, pages 44–49, 2008.

[4] V. Conitzer, J. Lang, and T. Sandholm. When are elections with few candidates hard to manipulate? *Journal of the ACM, Volume 54, Issue 3, Article 14*, pages 1–33, 2002.

[5] C. Dwork, R. Numar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. *Proceedings of WWW10*, pages 613–622, 2001.

[6] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. The shield that never was: societies with single-peaked preferences are more open to manipulation and control. *Proceedings of TARK*, pages 118–127, 2009.

[7] M. Garey and D. Johnson. Computers and intractability: A guide to the theory of NP-completeness. *W.H. Freeman and Company*, 1979.

[8] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, pages 587–601, 1973.

[9] E. Hemaspaandra and L. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, pages 73–83, 2007.

[10] K. Karmarkar and R. Karp. The differencing method of set partitioning. *Technical Report, Computer Science Division, University of California, Berkeley*, 1982.

[11] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[12] R. Korf. From approximate to optimal solutions: A case study of number partitioning. *Proceedings of the* $14^{th}$ *IJCAI Conference*, pages 266–272, 1995.

[13] R. Korf. Multi-way number partitioning. *Proceedings of the* $28^{th}$ *IJCAI Conference*, 2009.

[14] R. Korf. Objective functions for multi-way number partitioning. *Third Annual Symposium on Combinatorial Search*, 2010.

[15] J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Voting in multi-agent systems. *The Computer Journal, 10.1093/comjnl/bxh164*, 2006.

[16] M. Satterthwaite. Vote elicitation: Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory 10 (April 1975)*, pages 187–217, 1975.

[17] L. Wai and L. Ho. Rank aggregation for meta-search engines. *Proceedings of the* $13^{th}$ *international World Wide Web conference on Alternate track papers and posters*, pages 384–385, 2004.

[18] T. Walsh. Where are the really hard manipulation problems? the phase transition in manipulating the veto rule. *Proceedings of the* $28^{th}$ *IJCAI Conference*, pages 324–329, 2009.