Approximability and Computational Feasibility of Dodgson's Rule

Supervisors: Dr A. Slinko, Dr G. Pritchard

John C. M^{c}Cabe-Dansted

June 7, 2006

Abstract

Condorcet[1] proposed that a winner of an election is not legitimate unless a majority of the population prefer that alternative to all other alternatives. However such a winner does not always exist. A number of voting rules have been proposed which select the Condorcet winner if it exists, and otherwise selects an alternative that is in some sense closest to being a Condorcet Winner; a prime example is the rule proposed by Dodgson[2](1876).

Unfortunately, Bartholdi et al. (1989) proved that finding the Dodgson winner is an NP-hard problem. Hemaspaandra et al. (1997) refined this result by proving that it is Θ_2^p-complete and hence is not NP-complete unless Θ_2^p=NP. For this reason, we investigate the asymptotic behaviour of approximations to the Dodgson rule as the number of agents gets large.

Under the assumption that all votes are independent and equiprobable, the probability that the Tideman (1987) approximation picks the Dodgson winner does asymptotically converge to 1, but not exponentially fast. We propose a new approximation that does exhibit exponential convergence, and we can quickly verify that it has chosen the Dodgson winner; this allows us to choose the true Dodgson winner with $\mathcal{O}(\ln n)$ expected running time for a fixed number of alternatives m and n agents.

M$^{\text{c}}$Garvey (1953) proved that all tournaments are the majority relations for some society. We have proved a generalisation of this theorem for weighted tournaments. We find that this generalisation is useful for simplifying proofs relating to rules which use the weighted majority relation.

Bartholdi et al. (1989) found that we can calculate the Dodgson Score using an ILP that requires no more than $m!m$ variables, we present an improved ILP that requires less than $(m-1)!e$ variables ($e \approx 2.71$). We discover that we can solve this ILP in $\mathcal{O}(\ln n)$ arithmetic operations of $\mathcal{O}(\ln n)$ bits in size. Relaxing the integer constraints results in a new polynomial time rule. In 43 million simulations this new rule failed to pick the

[1]Marie-Jean-Antoine-Nicolas de Caritat, Marquis de Condorcet. (1743–1794)
[2]Charles Lutwidge Dodgson (1832–1898), better known as Lewis Carroll.

Dodgson winner only once, and only given many (25) alternatives. Unlike the Dodgson rule, this rule can break ties in favor of alternatives that are in some sense fractionally better.

We show that Dodgson Score admits no constant error approximation unless P=NP, and admits no Polynomial Time Approximation Scheme (PTAS) for Dodgson Score unless W[2]=FPT.

Acknowledgements

I would first of all like to thank my sister, Kim Dansted, for her suggestions for improving the readability of my introduction and for her general support and assistance while I was going through the grueling process of turning a draft into a legible Thesis.

Dr Arkadii Slinko, my supervisor for much assistance with proof reading. Also, unlike most other Lecturers, he helped guide me through my developmental years, due to his work with the pre-tertiary International Mathematical Olympiad training camps. Indeed, this was why I sought him out as a supervisor.

Dr Geoff Pritchard, my co-supervisor, for assistance with the statistics. Thanks to Pritchard, I have a more rigorous proof that Tideman's approximation converges to Dodgson's rule, which makes proper use of the Multivariate Central Limit Theorem.

Contents

List of Tables

Notations

- x^+: 0 if $x < 0$, x otherwise.
- sgn(x): The sign of x, 1 if $x > 0$, -1 if $x < 0$, 0 if $x = 0$.
- $\lfloor x \rfloor$: Floor of x, the largest integer that is less than or equal to x.
- $\lceil x \rceil$: Ceiling of x, the smallest integer that is greater than or equal to x.
- $|x|$: Absolute value of x, that is x if $x > 0$, -x otherwise.
- $\sum_i$: Summation over each i.
- $\prod_i$: Product over each i.
- $m!$: The factorial of m, i.e. $m! = (1)(2)(3)\cdots(m) = \prod_{i=1}^{m} i$.
- $A \wedge B$: A and B, e.g. A and B will occur.
- $A \vee B$: A or B, e.g. A or B will occur.
- $\neg A$: The negation of A, i.e. the statement "A is false".
- $P(E)$: The probability of event E, a real number in $[0, 1]$.
- $P(A|B)$: The probability of event A will occur given B has occurred or will occur.
- var(X): variance of random variable X.
- $E[X]$: mean of random variable X.
- $A \xrightarrow{D} B$: A converges in distribution to B.
- $N(\mu, \Omega)$: A multivariate normal distribution with means of μ and matrix of correlations Ω.

- $\mathcal{L}(\mathcal{A})$: The set of all linear orders on the set of alternatives A.
- $\mathbb{Z}$: The set of integers.
- $2\mathbb{Z}$: The set of even numbers.
- $\mathbb{N}$: The set $\{1, 2, 3, \ldots\}$ of natural numbers. (also known as positive integers)
- $\mathbb{R}$: The set of real numbers, e.g. $0, 1, \pi, \sqrt{2}$.
- $\mathbb{R}^+$: The set of positive real numbers.
- $\mathbb{R}^k$: The set of real valued k-dimensional vectors.
- $[x, y]$: The range of real numbers from x to y inclusive.
- $[x, y)$: The range of real numbers from x to y, including x but not y.
- $a\mathbf{v}b$: Alternative a is ranked before b in linear order $\mathbf{v}$.
- $\bar{\mathbf{v}}$: The opposite of a linear order $\mathbf{v}$, i.e. $a\mathbf{v}b \Leftrightarrow b\bar{\mathbf{v}}a$.
- n_{ba}: The number of linear orders in our profile $\mathcal{P}$ where alternative b is ranked above alternative a.
- $\text{adv}(b, a)$: The advantage of b over a, defined as $(n_{ba} - n_{ab})^+$.
- $\mathbf{1}_k$: A k-dimensional vector with all subscripts equaling 1.
- M^T: The transpose of M, i.e. $M_{ij} \equiv \left(M^T\right)_{ji}$.
- $\#(S)$: The cardinality of the set S, i.e. the number of elements in S.
- $F(x) \in \mathcal{O}(G(x))$: Function F is of order G, i.e. there exists $N \in \mathbb{N}$ and $c > 0$ such that for all n greater than N, $F(x) \leq cG(x)$.
- $\mathbf{x} \leq \mathbf{y}$: vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is less than vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. That is, $x_i \leq y_i$ for all i in $\{1, 2, \ldots, n\}$.
- $\mathbf{x} \geq 0$: vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is positive. That is, $x_i \geq 0$ for all i in $\{1, 2, \ldots, n\}$.
- $A \subseteq B$: A is a subset/submultiset of B.
- x^n: n instances of x, e.g. $\{a, b^2, c\}$ is a multi set with one instance of a and c and two instances of b.
- $\ln x$: The natural log of x, i.e. $\log_e x$.

Chapter 1

Introduction

1.1 Introduction

Taking collective decisions is the mechanism that allows democracy to exist. Furthermore, voting is also useful in other fields such as computer engineering. An example is the use of multiple independent subsystems to solve the same problem and vote on the result, which ensures that the failure of a single subsystem cannot cause the system as a whole to output an incorrect result. NASA is investigating voting in the development of fault tolerant systems, which can reliably use fault prone consumer grade CPUs in high radiation environments (Barry, 2005).

One of the major approaches to voting is to take pairwise elections between each pair of candidates. If we want to know which of two candidates is better, we poll the electorate as to which is better. In the 18^{th} century, Marquis de Condorcet proposed that a winner of an election is not legitimate unless that a majority of the population prefer that alternative to any other alternative. However, Condorcet noted that it is not always possible to pick such an alternative. An example of the Condorcet paradox is that we may have three soldiers marching in a line, each wishing to be as close to the rear as possible. For each possible marching order, two out of the three soldiers would prefer the soldier at the rear to move to the front. Clearly there is no rule which will pick a Condorcet winner for all possible elections.

Under the pairwise comparisons approach, we choose the Condorcet winner when it exists, or seek an alternative which is close to being a Condorcet winner. In this thesis we focus on a rule developed and favored by Dodgson (1876). The Dodgson rule picks the Condorcet winner when it exists, and otherwise allows an administrator to alter the votes, by swapping neighbouring alternatives (i.e. candidates) in agents' rankings, in a minimal way, to create a Condorcet winner. For a given vote, we consider two alternatives to be neighbouring if one is ranked directly above the other in the voters ranking. For

example, if the voters ranking was abc then a and b would be neighbouring alternatives but a and c would not. The Dodgson rule scores each alternative according to the number of neighbouring swaps that would be required to make it a Condorcet winner and picks the alternative with the lowest score as the winner.

The Dodgson rule is one of the most attractive realizations of the Condorcet principle. The Dodgson rule also has some other desirable properties such as monotonicity and satisfying the Pareto criterion[1].

The primary disadvantage of the Dodgson rule is that it can be computationally difficult to find the winner (Bartholdi et al., 1989). For this reason we study methods of approximating the rule.

1.1.1 Borda's Objection to the Condorcet proposal

Borda[2] suggested a different approach to voting than that of Condorcet. Borda suggested that simply choosing a candidate that defeats all others in pairwise elections can lead to tyranny of the majority. There may be a case where a majority of the citizenry slightly prefer a proposal a to proposal b, but a substantial minority strongly prefer proposal b to proposal a. The pairwise approach would select proposal a as the winner.

Borda proposed a rule which assigned a score to each alternative, depending upon the position of the alternative in a voter's preference list. This rule was meant to avoid tyranny of the majority by taking into account the degree by which each voter preferred each alternative.

All non-dictatorial rules[3] are subject to manipulation by voters (Gibbard 1973; Satterthwaite 1974). Thus both Borda's rule and any Condorcet based rule will be subject to manipulation by voters. However, Borda's rule is particularly vulnerable to manipulation. As an example say that there are four alternatives a, b, c and d. It may be that c and d are everybody's least favored alternatives, and a is your most favored alternative. If b looks to win the election by a single point, then ranking b last will cause a to win. Thus, by exaggerating your dislike for alternative b, you may be able to cause a to win. Borda's

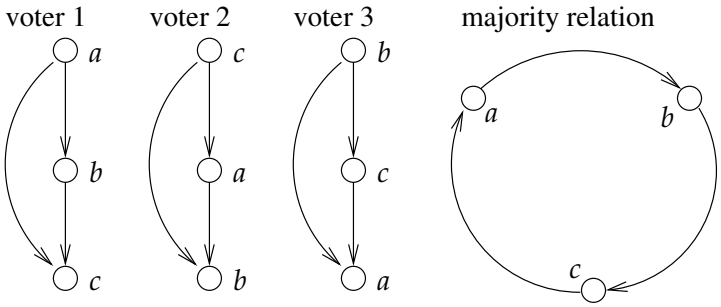
[1] The monotonicity criterion means that if one or more voters change their vote to rank a specific alternative higher, that this cannot cause this alternative to lose. Certain rules such as the Nanson, Hare, Coombs and Plurality runoff rules do not satisfy this property. The Pareto criterion means that if all voters prefer a to b the voting rule should rank a above b. In the case of rules which only select a winner rather than selecting a complete ranking of the alternatives, the Pareto criterion means that the rule will not select b as a winner. There has been considerable research to into what properties various rules satisfy. For a quick summary of which rules satisfy various properties, see the table presented by Nurmi (1983, p. 206).

[2] Jean-Charles de Borda. (1733–1799)

[3] A dictatorial rule is a dictatorship, a rule that picks a candidate based on the preferences of one particular voter, and ignores the other voters' preferences. Borda considers the Condorcet proposal to be a tyranny of the majority; dictatorship is instead a tyranny of an individual dictator.

rule is also vulnerable to manipulation of the agenda. For example, if you propose an alternative that is essentially the same as a but everyone agrees that the modification is not quite as good as the original a, this is also likely to cause a to win. In Borda's own words, the rule he proposed was for "honest gentlemen".

The debate with regard to the relative merits of the Condorcet and Borda approaches still continues, with prominent voting theorists such as Donald Saari arguing against the Condorcet approach. As we discussed previously we may not have a Condorcet winner. Say voter v_1 prefers a to b to c, voter v_2 prefers c to a to b, and voter v_3 prefers b to c to a. Then we have a case where a majority (voters 1,2) prefers a to b, another majority (voters 1,3) prefers b to c and a final majority (voters 2,3) prefers c to a; we call this set of majority preferences a majority relation (see Section 1.2.1 for a definition of a majority relation) We may represent these preferences as graphs below, with arrow pointing from x to y indicating that x is preferred to y.



We see that a cycle appears in the majority relation — even though all individual voters are rational in the sense that none of them have such cycles in their preferences[4]. Saari and Merlin (2000) have noted that under the pairwise approach these cycles can also occur when individual voters have "irrational" cyclic preferences. From the point of view of pairwise elections the above society is equivalent to an irrational voter who preferred a to b, preferred b to c but also preferred c to a. They consider the fact that, under the pairwise approach, a society where all voters individually are rational can be equivalent to a society with irrational voter(s) is a serious flaw in the pairwise approach. This paper suggests that the pairwise approach is flawed because a single voter changing their mind can cause a candidate to change from being ranked from best to being ranked the worst. Of particular relevance is Saari's 2003 paper where he argues against Risse's (2001) claim that both Condorcet and Borda's approaches are reasonable. Saari (2003) expands on his and Merlin's (2000) previous argument, giving the example where we have 8501 voters

[4]Black (1969, pp. 232–234) argues against the use of the term rational to refer to preferences without such cycles. If we define rational in such a way, we would have to state that it is rational to prefer bad to OK, and prefer OK to good, so long as you do not prefer good to bad. For such reasons Black prefers the mathematical term "intransitive" to "irrational".

who prefer a to c to b, and 8500 voters who prefer b to a to c. In pairwise elections, a defeats c unanimously, and c defeats b by one vote, so under the pairwise approach a is the best candidate and b is the worst. However if just one voter reverses their preference from acb to bca then a majority of 8501 voters will prefer b to any other candidate, and so b will be the new Condorcet winner. Thus, under the pairwise approach, changing a single vote can cause b to go from being ranked worst to being ranked best.

However, as noted by Risse (2005), Saari's arguments are not sufficient to sway the defenders of the Condorcet position. For example, Borda's rule does not allow voters to submit irrational preferences (preferences with cycles as above) and so it also does not distinguish between rational and irrational voters. In this thesis we also assume all voters preferences are rational, as does the Borda rule. Also, the fact that even in a large election it is possible for one voter to reverse the outcome of the election is well known by proponents of Condorcet's approach. However, the supporters of the Condorcet approach either do not consider this as a flaw, or believe that the other advantages of the pairwise approach outweigh Saari's objections.

Even if this debate were resolved, and the Condorcet approach to voting was judged obsolete, this would not extinguish interest in the Dodgson rule. The Dodgson rule is an interesting mathematical puzzle worthy of study in its own right, and discoveries which make Dodgson's rule less suitable for use as a voting rule can make it more interesting to mathematicians; as we discuss later, it was discovered that it can be very hard to find the Dodgson winner, but this only made the rule more interesting from the point of view of computer scientists (Hemaspaandra et al., 1997).

Furthermore, the type of rule that is "best" can vary dramatically depending upon to what purpose the rule will be put. The proponents of both Borda's and Condorcet's approaches would consider a dictatorship to be an example of a particularly bad decision procedure. Never-the-less, where decisiveness is the primary requirement, such as in a software development team seeking to rapidly produce a solution to a well understood problem, entrusting the lead programmer to make all design decisions can be the most appropriate decision procedure. (see e.g. McConnell, 1996)

1.1.2 Impossibility Theorems

We have looked at both the Condorcet and Borda's approach to voting theory. Another approach is the axiomatic approach of Kenneth Arrow. A diagram of the relationship between the concepts relevant to this thesis is provided on page 141.

The first important theorem in voting theory which used the axiomatic approach, was Arrow's Impossibility Theorem (1951,1963). Say we have a committee voting on the ranking of the priority of a number of different projects. Arrow proposed a modest set of re-

quirements, unrestricted domain, citizen sovereignty, monotonicity and independence of irrelevant alternatives[5]. Arrow then proved that the only procedure which achieves these requirements is a dictatorship. A similar important result (Gibbard 1973; Satterthwaite 1974) is that it is impossible to find a non-dictatorial rule that is not subject to manipulation by the voters.

Faced with such an impossibility theorem, one may respond by denying that one or more of the suggested criteria is important. From Arrow's point of view, how voters rank a third candidate C, is irrelevant to the question of whether is candidate A is better than candidate B. Therefore one of the criteria that Arrow suggested as desirable was independence of irrelevant alternatives, that the choice of winner should not be affected by this irrelevant information. Dummett (1998) challenges this position.

In an election between only two candidates A and B it may happen that A beats B by a small margin. Knowing only this, Borda's rule selects the majority winner. When a third "irrelevant" alternative C is added, Borda's rule now has some information as to how much the supporters of A dislike B and visa versa. If A is ranked last considerably more often than the other alternatives, then Borda's rule will no longer select the divisive candidate A as winner. Each "irrelevant" alternative added gives a little more information about the strength of the voters' preferences. Dummett (1998) sees the sensitivity of Borda's rule to this information as a strength rather than a weakness.

Another response to an impossibility theorem such as that of Arrow is to find a practical solution that mostly satisfies the criteria. Black (1969) showed that it is possible to find a rule that satisfies all of Arrow's criteria, except in very rare circumstances or for very small committees.

1.1.3 Complexity Classes for Algorithms

Even if a rule exists it may be infeasible to solve it. We have run algorithms on computers, and timed how long it took to compute a result. However these empirical tests have some weaknesses. Firstly, the amount of time depends on factors such as speed of the computer we use to run the algorithm[6] — for this reason we run all such tests on an Intel Xeon

[5] Arrow's criteria mean respectively: for every possible set of votes the rule must uniquely specify a single ranking of the candidates; every ranking of the candidates must be achievable by some set of votes; A citizen choosing to rank a candidate higher must never reduce the collective ranking of the candidate; if we label some arbitrary subset of candidates "irrelevant", then changes in the way the voters order the "irrelevant" candidates in their rankings should not affect the collective ranking of the candidates that are not labelled "irrelevant". For example, if we consider a and b to be our relevant candidates, and the ranking chosen by our rule prefers a to b, then any change to the votes which does not change whether each voter prefers a to b, should not cause the rule to pick a ranking which prefers b to a.

[6] There are many other factors which affect the time it takes to run an algorithm on a particular computer. For example, unlike the 486DX microprocessor, the 486SX microprocessor did not provide instructions for floating point arithmetic. Thus algorithms requiring floating point arithmetic would have

2.8 GHz processor, so while our empirical performance results do not directly apply to other processors they are at least consistent with each other. Secondly, just because an algorithm runs quickly on the input we have tested may not mean that it runs quickly given all possible inputs. In particular, just because the algorithm runs well given small test inputs does not mean that it would scale gracefully as the size of the input grows. For this reason we wish to study the algorithms from a theoretical perspective.

We say that a problem can be solved in polynomial time if the amount of time it takes is a polynomial[7] function of the input size, where the input size is the number of bits[8] required to represent the input with respect to the size of the input. For example we can answer the question "is x divisible by 3" in time that is linear[9] with respect to the number of bits n needed to represent x, so this problem can be solved in polynomial time. We can also add two numbers in time linear with respect the number of digits, as it takes the same amount of time to add each digit.

A decision problem is a problem with a yes-no answer. Some decision problems, of particular relevance to this thesis, are those proposed by Bartholdi et al. (1989):

- *Dodgson Score:* Given a profile $\mathcal{P}$, candidate c and integer k — is the Dodgson score of c less than or equal to k?
- *Dodgson Winner:* Given a profile $\mathcal{P}$ and candidate c — is c the Dodgson winner?
- *Dodgson Ranking:* Given a profile $\mathcal{P}$ and candidates c and d — did c defeat d according to the Dodgson rule, i.e. did c have a lower Dodgson score than d?

Bartholdi et al. (1989) defined the decision problems *Kemeny Score*, *Kemeny Winner* and *Kemeny Ranking* similarly. We use italics to indicate decision problems. For example *Dodgson Score* indicates the decision problem "is the Dodgson score of c less than k in profile $\mathcal{P}$".

Finding the Dodgson score (and winner) is at least as hard as the *Dodgson Score* (and *Winner*) decision problem. If we have found the Dodgson score we may answer whether the score is less than or equal to k easily; the reverse may not be so easy. Likewise, if we have found the Dodgson winner, we may easily determine whether c is that winner; but if we know that c is not the winner we do not necessarily know who the winner is. Thus in this thesis we focus on showing that it is easy to finding Dodgson scores and winners

to use multiple simpler instructions to perform such arithmetic, slowing down the algorithm. Thus a 486SX would usually need more time to complete an algorithm requiring floating point arithmetic than a 486DX, even if the 486SX had a higher clock speed.

[7] A polynomial is a function f of the form $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$, where $a_0, a_1, \ldots, a_n$ are real valued constants, and the integer n is called the order of the polynomial.

[8] A bit is a single "0" or "1".

[9] A linear function is a function a function f of the form $f(x) = a + bx$. These functions are called linear as their graphs are straight lines.

when we fixed the number of alternatives, as in doing so we will also show that *Dodgson Winner* and *Dodgson Score* are easy.

Computer scientists classify problems into classes which represent how difficult the problems are to solve. For example, decision problems may belong to the following classes:

P (Polynomial): The set of problems that can be computed in polynomial time.
Examples: Is x divisible by 3. Given a profile $\mathcal{P}$ and candidate c is c a Condorcet winner?

NP (Nondeterministic-Polynomial[10]): The set of problems for which, if the answer is "yes", there exists a proof of this fact that can be verified in polynomial time.
Example: *Dodgson Score*, because if is the Dodgson score is less than k then there exists some set of less than k neighbouring swaps that make c a Condorcet winner. We may use this set of swaps as a proof that the Dodgson score is less than or equal to k as we may verify that these swaps do indeed make c a Condorcet winner and thus that the minimal set of swaps must contain less than k swaps, i.e. that the Dodgson score is less than k. (Bartholdi et al., 1989)

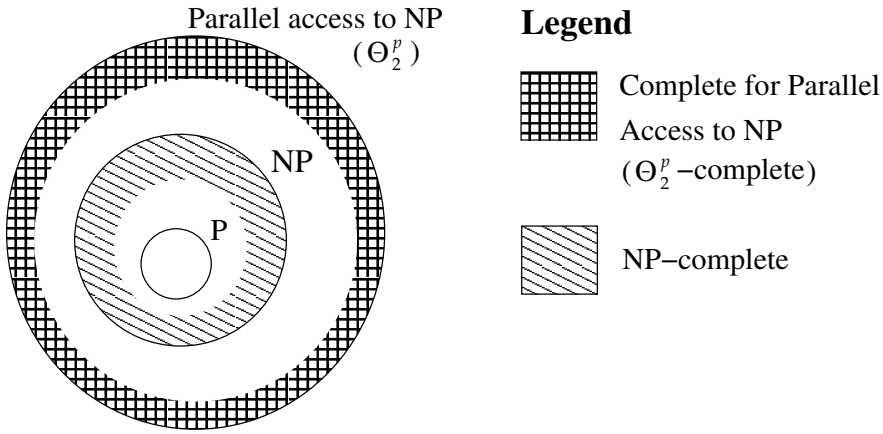
Θ_2^p (Parallel access to NP): The set of problems which could be solved in polynomial time if we had an oracle that offered to answer any number of questions to problems in the class NP, but would only reveal the answers once we had asked the last question. This is called parallel access to NP, because we have access to an oracle that can answer questions, but we essentially have to ask them in parallel, we cannot ask a question and then choose our next question based on the answer to the last. (see e.g.Hemaspaandra et al. 1997)
Example: *Dodgson Winner*, because if we find the Dodgson score for each alternative, we will then know whether c is a Dodgson winner.

All problems in P are in NP and all problems in NP are in Θ_2^p. If a problem is in P we can solve it in polynomial time without a proof, so if we are given a proof we can solve it in polynomial time simply by ignoring the proof, and so the problem is also in NP. If a problem is in NP, then we can solve it by asking a single question of the above oracle, so it is also in Θ_2^p. It is commonly believed that there are problems that are in Θ_2^p that are not in NP, and problems in NP that are not in P, although this has never been proven.

[10]This is called Non-deterministic as the class of problems can be thought of in the following way. Assume that whenever you flip coin, the universe splits into two, such that in one universe the coin lands on its head and in the other the coin lands on its tail. We are allowed to flip coins as many times as we want, each time forking the universe. If the correct answer is "no", we must answer "no" in all the universes; if the correct answer is yes we only need to answer yes in one of the universes and may answer no in the rest.

An NP-hard problem is a problem such that if we had an oracle that offered to give one answer to such a problem, then we could use that oracle to answer any NP problem in polynomial time. For this reason we consider the NP-hard problem to be at least as hard as any NP problem. We call an NP-hard problem that is in NP, an NP-complete problem. Hardness and completeness are defined similarly on the class of Θ_2^p problems. For example the class of NP problems is a subset of the class of Θ_2^p problems, but NP problems are not Θ_2^p-complete unless it turns our that NP is equivalent to Θ_2^p.

We represent the current understanding this as a Venn diagram as below. Problems further out from the center are in some sense harder than those closer to the set of problems in P.



NP-complete problems can take a very long time to solve. There are no known algorithms that can solve NP-complete problems in time that is polynomial with respect to the size of the input, and it is commonly believed that no such algorithm exists. Interestingly, if it is possible to solve a single NP-complete problem in polynomial time, it is possible to solve all NP problems in polynomial time.

One of the reasons the Dodgson rule is so interesting to study is that it is one of the simplest examples of a Θ_2^p-complete problem (Hemaspaandra et al., 1997). Any NP-complete problem can be very hard to solve, and so NP-complete problems are often called infeasible. Problems in the Θ_2^p-complete class are commonly believed to even harder to solve. We consider the Dodgson rule a particularly interesting problem because on the one hand, it is a member of a class of problems usually considered very difficult to solve, on the other hand, as we will discuss later, we prove that it is usually very easy to solve when the number of candidates is fixed.

Just knowing that a problem is in P does not mean that it easy to solve. For example, say that the time required to solve the problems is n^{1000}. This is technically polynomial, but is clearly infeasible even for $n = 2$. For this reason we often wish to classify problems into complexity classes that give us more information about the rate at which the time

required to solve the problem grows with the size of the input, than merely whether it is polynomial or not.

To do this we will use the "big-$\mathcal{O}$" notation. For positive valued functions f and g, the function g is in the class $\mathcal{O}(f(n))$ if there exists positive numbers N and k such that for all n greater than N, the value of $g(n)$ is less than the value of $kf(n)$. If the function g represents the amount of time required to run an algorithm, the function g will depend on the computer that runs the algorithm; however g will belong to the same class $\mathcal{O}(f(n))$ regardless of whether it is run on the very first computer produced by Apple, or the Deep Blue supercomputer that beat the chess master Kasparov. Although one might run the algorithm a million times faster that the other, the rate at which the time grows as n approaches infinity is the same.

For example we can add (and subtract) numbers in linear time with respect to the number of digits d, so addition and subtraction are $\mathcal{O}(d)$ time problems. We can multiply d digit numbers using d additions, each of which require $\mathcal{O}(d)$ time. Thus multiplication is an $\mathcal{O}(d^2)$ time problem. However, just because multiplication is a $\mathcal{O}(d^2)$ problem does not mean that there does not exist a better algorithm. It is known that multiplication cannot be done in time less than linear, but that does not mean that we may not find an algorithm that requires less than $\mathcal{O}(d^2)$ time, in-fact a number of such algorithms have been found. The first such algorithm was found by Karatsuba and Ofman (1962). The best currently known algorithm for multiplication is that of Schönhage and Strassen (1971), which can multiply numbers in $\mathcal{O}(d \log d \log\log d)$ time. Thus multiplication is also a $\mathcal{O}(d \log d \log\log d)$ time problem. Just as the class of NP problems includes all problems in P, the class $\mathcal{O}(d^2)$ contains $\mathcal{O}(f(d))$ for all functions f that grow slower than d^2.

The time required to solve a particular problem may depend upon multiple parameters. For example, the time to find the winner to an election depends upon the number of candidates m and number of voters n. An approach to infeasible problems, advocated for example by Downey (2003), is to see if the problem becomes feasible if we limit one of the parameters. For example, say it takes $\mathcal{O}(n^{m!})$ time to solve a problem. Then if we fix $m = 3$, we may find the problem in $\mathcal{O}(n^6)$ time which is polynomial. However the order of this polynomial grows if we fix m at a higher value, e.g. if we fix $m = 5$ the problem requires $\mathcal{O}(n^{120})$ time to solve.

Computer scientists (e.g. Downey and Fellows 1995) are interested in whether the order of the polynomial depends upon the value at which we fix m; if not they call the problem Fixed Parameter Tractable (FPT). For example if the problem requires $\mathcal{O}(m!n^2)$ time, then it requires $\mathcal{O}(n^2)$ time for any fixed m; as n^2 is a polynomial, this problem is called Fixed Parameter Tractable. This class of problems is of interest to us as we will show that Dodgson's rule is fixed parameter tractable as it is possible to determine the Dodgson score

using $\mathcal{O}(\log n)$ operations on $\mathcal{O}(\log n)$ digit numbers, for any fixed number of agents m. As $\log n$ grows very slowly, this means that for any fixed m finding the Dodgson score belongs to a class of problems that is considered exceptionally easy to solve.

Another approach to infeasible problems is to attempt to find an approximate solution. A polynomial time approximation scheme is an approximation where for any fixed percentage error, the solution can be found to within that percentage of error. A constant approximation scheme is an approximation which can find the solution in polynomial time for some fixed error. Unfortunately we find that neither type of approximation scheme exists for the Dodgson Quick score. The Dodgson Quick, Dodgson relaxed and R&R approximations, that we will propose later, are a different type of approximation. We prove that our approximations will pick the Dodgson winner almost all of the time for a large number of agents. However, in certain unlikely circumstances, their score from these approximations could differ substantially from the true Dodgson score.

1.1.4 Impracticality Theorems

Finding the winner in an election can be difficult. Young and Levenglick (1978) proved the only neutral, consistent and Condorcet[11] rule was the rule proposed by Kemeny (1959). Bartholdi et al. (1989) proved that *Kemeny Winner* and *Kemeny Ranking* are an NP-hard problems. It can take a very long time to solve NP-hard problems and any voting rule for which we may not discover the winner before their term of office is over is of little use. For this reason, Bartholdi et al. commented that research into "impracticality" theorems, which show that any rule which satisfies a particular set of criteria is impractical to compute, is as important as research into the more traditional impossibility theorems.

Bartholdi et al. also showed that *Dodgson Winner* is an NP-hard problem. This result was refined by Hemaspaandra et al. (1997), who showed that *Dodgson Winner* is complete for parallel access to NP (i.e. Θ_2^p-complete), and hence is not NP-complete unless NP is equivalent to Θ_2^p, which most computer scientists believe it is not. We investigate the possibility of simplifying the problem.

In the same way that Black (1969) found a rule that satisfies Condorcet's criteria except for a very few possible inputs, we may find a procedure that can easily compute the Dodgson winner except for unusual inputs. Indeed, as we discuss later, we find an easy to compute approximation to the Dodgson rule.

[11] Neutral means that the voting rule does not favor any candidate. Consistent means that if the electorate is divided into two parts and the rule picks the same winner for both parts, the rule will pick that same winner for the electorate as a whole. Condorcet means that if a Condorcet winner exists then the rule will select that winner.

1.1.5 Simplifying Assumptions in Voting Theory

If we make assumptions about the voting society we may avoid the implications of some impossibility theorems, such as that of Arrow. That is, if we reject the criteria of unrestricted domain, we may be able to satisfy the other criteria suggested by Arrow.

Black (1948) suggested that if voters' preferences can be represented by a one-dimensional scale, then we may select the position of the median voter. If the voters prefer political positions close to their own (i.e. their preferences are "single-peaked"), then this position will be the Condorcet winner, and thus also the Dodgson winner. Unfortunately, we frequently cannot model voters' preferences on a single dimension; for example parties on either side of the left/right spectrum can be also be militaristic or isolationist and so two or more dimensions may be required. If two or more dimensions are required, we cannot always select the median voter (Black, 1948).

In a similar way, if we make certain assumptions we may avoid the implication the impracticality theorem of Bartholdi et al., that finding the Dodgson winner may be infeasible.

As *Dodgson Winner* is a hard problem, in some sense harder even than the NP-complete problems, we investigate the possibility of simplifying the problem. As discussed above, if voters' preferences can fall on a left/right spectrum we may easily find the Dodgson winner merely by taking the median voter. As many elections cannot be represented in this way, we choose a different assumption. We observe that in national elections there are typically millions of voters, but only a handful of alternative parties. Indeed, if there were millions of candidates it would be hard for the voters to rank them all. In this thesis we first simplify the problem by assuming that there are only a few alternatives.

For a fixed number of alternatives (or voters), Bartholdi et al. (1989) showed that we can find the Dodgson winner in polynomial time. Unfortunately the order of this polynomial may be rather large, and become ever larger as we increase the number of allowed alternatives. Even for mild polynomial complexity,[12] polynomials of low order, it may be infeasible to compute the winner in elections with millions of agents. For these elections we may wish to use a rule that approximates the Dodgson rule, and has very low computational complexity.

1.1.6 Tideman's Approximation to the Dodgson Rule

Tideman (1987, p. 194) suggested a rule which scores each alternative according to the sum of the margins of defeat. For example say 40 voters preferred A to B versus 60 who

[12] That is, if the time required to find the winner is bounded by a polynomial with relatively low order, e.g. n^2.

preferred B to A and 48 preferred C to A versus 52 who preferred A to C. Then the margins of defeat would be 20 and 4 respectively; the Tideman score of A would be 24. The alternative with the lowest Tideman score is the Tideman winner. This rule would pick the same winner as Dodgson's rule if

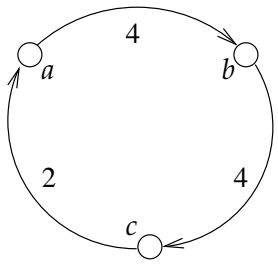
[T1] the number of voters was even[13], and

[T2] "all contending candidates appeared in enough rankings just below the candidates that beat them that they could be made dominant by being advanced above these without having to be advanced above any others to reach these". — (Tideman 1987; p. 143)

Tideman did not present a proof that this is the case, and neither shall we (until later, as Corollary 3.2.3). However we will present an example which satisfies [T1] and [T2] and explain why this means that the Tideman winner is the Dodgson winner in this example.

For example say four voters chose the ranking abc, three chose cab, and three chose bca. In the ranking abc we say that a is ranked above b, which is ranked above c because it is common to represent abc vertically such as in the table below.

4	3	3
a	c	b
b	a	c
c	b	a



The weight on each edge of the graph above represents the margin of defeat of each pair of alternatives. We compute the margins of defeat from the table above as follows: 7 voters prefer a to b, 3 prefer b to a so the margin of defeat of a over b is 4; 7 voters prefer b to c and 3 voters prefer c to b, so the margin of defeat of b over c is 4; 6 voters prefer c to a and 4 voters prefer a to c so the margin of defeat of c over a is 2. Graphs such as these are known as a weighted majority relations.

Each alternative loses to exactly one other alternative, so it is easy to "calculate" the sums of the margins of defeat — we can just read them off the graph. It is clear that alternative a has the lowest Tideman score (i.e. 2) and thus is the Tideman winner. Let us

[13]Tideman's actual criteria was "if a tie could be beaten by an arbitrarily small fraction of a vote". By tie Tideman appears to refer to the case where we cannot pick a Condorcet winner. This amounts to saying that Tideman's rule would choose the Dodgson winner in elections that satisfy [T2], if Dodgson's rule allowed us to split votes into fractional votes and swap neighbouring alternatives in those fractional votes — whereas Dodgson's rule only allows us to swap neighbouring alternatives in whole votes. In any case, we have found that our weaker [T1] condition is sufficient (with [T2]) to guaranty that Tideman's rule picks the same winner as Dodgson's rule.

see how conditions [T1] and [T2] guaranty that the Tideman winner is also the Dodgson winner.

As the number of voters are even [T1], all of the margins of defeat are even (we prove this in Lemma 2.0.12). Recall that a Condorcet winner is a candidate that a majority of the population prefer to any other candidate. Thus, to make a a Condorcet winner we need to reduce the margin of defeat of c over a to zero, so that c no longer defeats a, in a pairwise election. By changing a single vote to rank a above c we both reduce the number of votes where c is ranked above a and increase the number of votes were a is ranked above c, hence decreasing the margin of defeat of c over a by two. Thus we can reduce the margin of defeat of c over a to zero by altering a number of votes equal to half of this margin of defeat 2, in this case 1 vote. Note that in 6 votes c is ranked directly above a. Hence we can swap a above c in up to 6 votes using only one swap of neighbouring alternatives per vote, and we only need to swap a above c in 1 vote to make a a Condorcet winner (i.e. condition [T2] holds). Hence the Dodgson score of a, the minimum number of swaps to make a a Condorcet winner, is 1. This is exactly half the Tideman score of 2. Since the Dodgson scores are exactly half the Tideman scores, and both rules pick the alternative with the lowest score, the Tideman rule will select the Dodgson winner.

Not only does Tideman's rule have low computational complexity, but is simple enough to be computed with pencil and paper. However, Tideman did not prove anything about the effectiveness of this rule as an approximation to Dodgson's rule, nor make any claim about the probability that the Tideman approximation would select the Dodgson winner.

Under the assumption that all possible elections (ordered sets of votes) are equally likely, we show that the Tideman rule converges to the Dodgson winner as we increase the number of voters with a fixed number of alternatives. That is, as the number of voters tends towards infinity, the probability that the Tideman winner is not the Dodgson winner drops to zero.

We show that the probability that the [T2] condition holds converges to one exponentially fast, hence if the number of voters is even the probability that the Tideman winner is the Dodgson winner converges to 1 exponentially fast. However if the number of voters is odd we show that the Tideman winner does not converge exponentially fast. Hence, in the general case where the number of voters may be even or odd, the Tideman winner does not converge to the Dodgson winner exponentially fast.

1.1.7 Our New Approximation, Dodgson Quick.

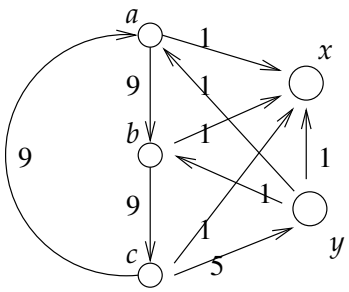
We propose a new rule which we call Dodgson Quick (DQ). Like the Tideman rule our approximation generates scores based on the margins of defeat, and picks the alternative with the lowest score as the winner. The DQ-score was originally intended as a lower

bound[14] for the Dodgson Score which can be calculated very quickly — and can thus improve the performance of an optimized algorithm which calculates the Dodgson winner by helping it to quickly eliminate candidates which cannot possibly be winners. However we noticed that it was a good approximation for the Dodgson rule, and had some convenient theoretical properties, so we decided to study it further.

Note that in the example of the Tideman rule choosing the correct winner the Tideman score was exactly half of the Dodgson scores. However, with an odd number of voters the Tideman score can be odd, but we cannot have a fractional Dodgson score. Recall that the Dodgson score is the minimum number of swaps of neighbouring alternatives needed to make a candidate a Condorcet winner, and the Dodgson rule does not allow us to swap a fractional number of alternatives. Even if the Tideman score of an alternative a is even this may be because it loses to an even number of alternatives by an odd margin of defeat. The Dodgson rule also does not allow us to swap a over some alternative b a fractional number of times, and another alternative c a fractional number of times, even if the total number of swaps is a whole number.

Thus we note that the Dodgson score must be at least the sum of the margins of defeat divided by two, all rounded up. We call this lower bound the Dodgson Quick score, and the candidate with the lowest Dodgson Quick score the Dodgson Quick winner. This leads to a rule with substantially different properties to the Tideman rule.

Below is an example of a weighted majority relation where the Tideman winner is not the Dodgson Quick winner. A weighted majority relation is a directed graph where an arrow from an alternative a to b with a number (weight) w attached indicates that a beat b in a pairwise election with a margin of w (see Section 1.2.1, for a definition of majority relations).



As shown by the following table of scores, x is the Tideman winner, but y is the DQ-winner. We calculate the Dodgson Quick and Tideman scores of x, y as follows. The weights (margins of defeat) of the edges pointing at x are {1,1,1,1}. The sum of these weights is 4, so the Tideman score is 4. For the DQ-score we divide these by 2, so we get $\{\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\}$, however we then round all of these back up to 1. Hence the DQ-score of x is also

[14] A lower bound x for a value y is a number less than or equal to y.

4. The candidate y has only one edge coming in, but it has a weight of 5, so the Tideman score of y is 5. For the DQ-score we divide the 5 by 2 giving 2.5, and then round this up to 3. Hence the DQ-score is 3. We calculate the scores of the candidates a, b and c likewise.

Scores	a	b	c	x	y
Tideman	10	10	9	4	5
DQ	6	6	5	4	3

We initially expected the properties of the DQ-rule to be quite close to the properties of the Tideman rule. Where the number of voters are even they pick the same winner. Even for an odd number of voters, the DQ-winner differs from the Tideman winner only due to rounding. For a large number of voters, the rounding only has a small effect on the DQ-score relative to the magnitude of the score. Although we have presented an example of where the Tideman winner differs from the DQ-winner, it would be intuitive to think that this would happen so rarely as to have little effect.

However, some exploratory numerical results, which we published in our paper (McCabe-Dansted and Slinko, 2006), demonstrated that the Dodgson Quick rule was an exceptionally close approximation to the Dodgson rule indeed. For example, with 5 candidates and 85 voters, and assuming that every possible election (ordered set of votes) is equally likely, we found that the DQ-winner was the Dodgson winner in all of the 1,000,000 simulations. By comparison, the Tideman approximation failed to pick the Dodgson winner in almost one percent of cases.

We investigate the behaviour of the Dodgson Quick approximation and find that as we increase the number of voters not only does our approximation converge to the Dodgson rule, but it also converges exponentially fast. In this respect, our approximation is superior to the one suggested by Tideman; we show that Tideman's approximation does not converge exponentially fast. We suspected that the rule proposed by Simpson (1969) would converge asymptotically to Dodgson's rule despite the fact that it was not intended as an approximation of Dodgson's rule. However we numerically investigate the frequency that the Simpson rule does not pick the Dodgson winner as we increase the number of voters, and find that the frequency does not converge to zero.

A similar result was obtained by Homan and Hemaspaandra (2005). They independently developed an approximation which also exhibits exponential convergence as we increase the number of voters.

Unlike this thesis, Homan and Hemaspaandra approached the problem by looking for a "frequently self-knowingly correct[15]" greedy algorithm. As mentioned previously, we

[15]They define this as an algorithm that reports that it has found the correct answer with probability approaching 1 as the size of the input (in this case an ordered set of votes) to the algorithm increases, and

originally used our approximation to provide lower bounds that we used to increase the speed with which we could find the exact Dodgson scores.

Homan and Hemaspaandra's greedy algorithm assigns a penalty to candidates for which it Tideman's criteria [T2] above does not holds. This means that the scores assigned by their algorithm are not lower bounds. For example consider an election with a single voter whose ranking of three alternatives is abc. Our Dodgson Quick rule notes that to make c a Condorcet winner we need to swap c over a once and over b once, and hence assigns a score of 2 to candidate c. Their greedy algorithm notes that Tideman's criteria [T2] does not hold, i.e. since a and c are not neighbouring preferences, to swap a over c we have to first swap a over b. For this reason their greedy algorithm adds a penalty of one to it's score of c, giving a total score of 3 for candidate c. However in this case, to make c a Condorcet winner, we had to swap a over b anyway. Hence the Dodgson score of c, the minimum number of swaps required to make c a Condorcet winner, is only 2. Hence, unlike our Dodgson Quick scores, the score assigned by Homan and Hemaspaandra (2005)'s greedy algorithm is not always less than or equal to the Dodgson score.

Also our approximation's scores can be calculated from the margin of defeat of each candidate[16], while Homan and Hemaspaandra's penalty cannot.

The simplicity and convenient theoretical properties of our approximation may make our approximation useful in proofs. We found it much easier to prove that our voting rule converged to the Dodgson rule at an exponential rate than to prove that Tideman's voting rule converged at all. When we did prove that Tideman's approximation converged to the Dodgson rule, we found it easier to first prove the Tideman approximation converged to our approximation.

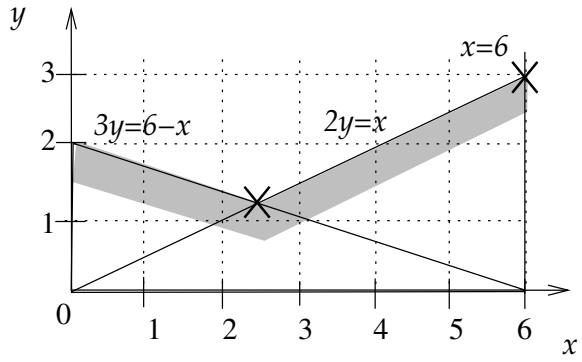
1.1.8 Linear Programs and Integer Linear Programs

A **Linear Program (LP)** is a set of linear constraints and a linear function we wish to maximise or minimise. Simple two dimensional linear programs are often given as math problems at school. For example, given that $2y \geq x$, $x \leq 6$ and $3y > 6 - x$, minimise $f(x, y) = y$. This simple problem can be solved by drawing the graph (shown below), finding the intercepts, finding the value of f for each intercept and finding the minimum

its answer is always correct when it reports that its answer is correct. A greedy algorithm is an algorithm which makes decisions that bring the algorithm closer to its goal in the short term without considering the longer term consequences. Such an algorithm may be chosen because a correct algorithm takes too much time to run or because one proven that the greedy approach will always pick the correct answer anyway.

[16]For those familiar with the classification system proposed by Fishburn (1977), our rule is a C2 rule, the greedy rule is a C3 rule.

value of f.



Intercepts

x	y	$f(x,y)$
6	3	3
2.4	1.2	1.2

In the example above we find two intercepts, the one with the lowest value of f at $(2.4, 1.2)$. If the variables must be integers we call the problem an **Integer Linear Program (ILP)**. In this trivial case it is easier to solve the ILP than the LP, if the variables must be integer is easy to see from the graph above that $f(x, y)$ must be at least 2. However for real world problems, LPs are usually much easier to solve than ILPs.

With respect to LPs and ILPs, the "feasibility problem" refers to the case where we are not interested in the best solution to the problem, and are only interested in whether there exists a solution which satisfies all the constraints.

In the simple example above we only have two variables x and y. However LPs can have thousands of variables, and these problems can be quite difficult to solve.

Dantzig (1963) found an algorithm, usually called the simplex method, that can be used to solve complex LPs. This algorithm usually finds the solution to and LPs quite quickly. However, it can take an exponential amount of time to find the solution. The first algorithm that could solve an LP in polynomial time was found by Khachian (1979), unfortunately this algorithm performed poorly in practice. The Khachian algorithm required $\mathcal{O}(n^6L)$ operations $\mathcal{O}(L)$ bit numbers where n is the number of variables and L is the number of bits required to encode the LP. A better algorithm was found by Karmarkar (1984), that required only $\mathcal{O}(n^{3.5}L)$ operations. The result we will use in this thesis is the discovery by Gonzaga (1989) of an algorithm that can find the solution of an LP using only $\mathcal{O}(n^3L)$ operations. The algorithm independently discovered by Vaidya (1990) is also of interest as it can find the solution of an LP in time better than $\mathcal{O}(n^3L)$ if we limit the number of constraints, although this result is not of use in this thesis.

There is no known algorithm that can solve ILPs (as opposed to LPs) in polynomial time. Indeed, we can transform the problem of finding the Dodgson score into an Integer Linear Problem with $\mathcal{O}(mn)$ variables where m is the number of candidates and n is the number of alternatives (see Chapter 4). If we could solve ILPs in polynomial time we

could find the Dodgson score in polynomial time, which Bartholdi et al. (1989) showed was not possible unless P=NP.

Never-the-less Lenstra, Jr. (1983) found that the amount of time required to solve the integer programming feasibility problem was polynomial if we fix the number of variables in the ILP. Being able to solve the feasibility problem in polynomial time, we may then solve the ILP in polynomial time (for a fixed number of variables) using the binary search algorithm[17]. This result was refined by Eisenbrand (2003), who showed that for a fixed number of variables, the ILP can be solved in $\mathcal{O}(L)$ basic arithmetic operations.

1.1.9 Our New Approximation, Dodgson Relaxed & Rounded.

Under the Dodgson rule, we may only switch neighbouring alternatives in whole votes. The **Dodgson relaxed rule** allows us to split votes into rational[18] fractions of a vote and swap neighbouring alternatives in these fractions of a vote. This provides us with two advantages over the Dodgson rule, first we will show that we may compute the Dodgson relaxed rule in polynomial time, and in logarithmic time if we fix the number of alternatives. Secondly, we will show numerically that when the set of tied winners chosen by this rule differs from the set of tied winners selected by Dodgson's rule, it is usually because the relaxed rule has chosen a subset of the Dodgson winners. We may break ties according to the preferences of the first agent, however it is in some sense more democratic to select an alternative that is fractionally better than the others than to privilege the first agent over the other agents.

By rounding the Dodgson relaxed score up we get a new score which we call the **Dodgson relaxed and rounded (R&R) score**. The R&R rule is exceptionally close to the Dodgson rule. Out of 43 million random simulations under various assumptions we only found a single simulation where the R&R winner was not the Dodgson winner. This simulation had 25 candidates but only 5 agents, a somewhat implausible size for a real world election. The example of a real world election with a huge number of candidates given by Bartholdi et al. (1989), was the election for mayor of Tulsa. Even this election had only 20 candidates, and there were presumably more than 5 residents of Tulsa voting.

[17] The binary search algorithm is an algorithm that can be used to find a number k where k falls with in some known range and we can ask whether "k is greater than x" for any x we choose. Say we know that k falls between 0 and 64. The algorithm involves choosing the midpoint of l and u. An example of the binary search algorithm in practice is if we are asked to guess a number between 0 and 8. We ask if k is greater than 4, if so we ask if k is greater than 6, if not we ask if k is greater than 5; we now know the answer. Note that this algorithm is quite efficient as each time we ask a question we halve the number of possible values of k.

[18] A rational number is a number that can be represented as n/m where n and m are integers. In fact the Dodgson relaxed rule could be equivalently defined as allowing splits into any real fraction of a vote, the Dodgson relaxed score would be the same either way.

The Dodgson Quick score is always less than or equal to the Dodgson relaxed score, which is always less than or equal to the R&R score, which in turn is always less than or equal to the Dodgson score.

1.1.10 A Generalisation of the M^{c}Garvey Theorem

Recall that in the weighted majority relation we presented on page 14, we did not give the set of voters that generated that relation. Can we be sure that such a set of voters really exists? You may like to attempt to find such a set. However finding such a set by trial and error can be time consuming. Even verifying that such a set does actually generate that weighted majority relation can take a while.

M^{c}Garvey (1953) proved that we can find such a set for any ordinary majority relation. However recall that the graphs of ordinary majority relations do not have weights attached to their edges. In this case we also needed the weights to be correct. Thus we cannot use the M^{c}Garvey theorem here.

Fortunately for any weighted tournament, we can find a society with that weighted tournament as its weighted majority relation, if and only if all the weights are even or all the weights are odd. Thus we can check at a glance that all the weights on the weighted tournament are odd. We also check that there are no missing edges, because this is equivalent to an edge with weight of zero, which is even. This can be done easily without even needing to reach for a pencil.

The first paper that mentions this result was probably Debord's PhD thesis (1987) as quoted by Vidu (1999). However this source is inaccessible to the author, and we will give an independent proof in this thesis.

1.2 Social Choice Functions.

In the introduction we described voting procedures which took rankings from each voter and output either a ranking of the candidates or a single winner. Procedures which only output a winner are called social choice functions. These are the procedures we will study in this thesis. Historically SCFs were often called **rules**. Below we will formally define SCFs and, discuss classifications for such rules and give definitions for the SCFs we will study in this thesis.

As voting has a wide variety of uses, we will henceforth avoid the terms voter and candidate. These terms carry the implication that we are only taking about people. In their place we will use the more general terms **agent** and **alternative**.

We assume that agents' preferences are transitive, i.e. if they prefer a to b and prefer b

to c they also prefer a to c. We also assume that agents preferences' are strict, if a and b are distinct they either prefer a to b or b to a. Thus we may consider each agent's preferences to be a ranking of each alternative from best to worst.

Let $\mathcal{A}$ and $\mathcal{N}$ be two finite sets of cardinality[19] m and n respectively. The elements of $\mathcal{A}$ will be called alternatives, the elements of $\mathcal{N}$ agents. We assume that the agents have preferences over the set of alternatives. By $\mathcal{L}(\mathcal{A})$ we denote the set of all linear orders[20] on $\mathcal{A}$; they represent the preferences of agents over $\mathcal{A}$. For example if $\mathcal{A}$ is $\{a, b, c\}$, then $\mathcal{L}(\mathcal{A})$ is the set of all permutations of abc, i.e. $\{abc, acb, bac, bca, cab, cba\}$. The elements of the Cartesian product[21]

$$\mathcal{L}(\mathcal{A})^n = \mathcal{L}(\mathcal{A}) \times \cdots \times \mathcal{L}(\mathcal{A}) \quad (n \text{ times})$$

are called **profiles**, for example, where $\mathcal{A} = \{a, b, c\}$ the ordered set (abc, bca) is a possible profile for a society with two voters. The profiles represent the collection of preferences of an n-element society of agents $\mathcal{N}$. Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be our profile. If a linear order $P_i \in \mathcal{L}(\mathcal{A})$ represents the preferences of the i^{th} agent, then by aP_ib, where $a, b \in \mathcal{A}$, we denote that this agent prefers a to b.

A family of mappings $F = \{F_n\}$, $n \in \mathbb{N}$,

$$F_n \colon \mathcal{L}(\mathcal{A})^n \to \mathcal{A},$$

is called a **social choice function** (SCF).

For a profile $\mathcal{P}$, we define the corresponding **voting situation** $\tilde{\mathcal{P}}$ to be a multiset of linear orders such that for any linear order $\mathbf{v}$ and integer i, if there are i occurrences of $\mathbf{v}$ in $\mathcal{P}$ there are i occurrences of $\mathbf{v}$ in $\tilde{\mathcal{P}}$. A voting situation is similar to a profile, except that voters are anonymous. Most rules only require the information contained in $\tilde{\mathcal{P}}$ to pick their winner.

1.2.1 Fishburn's Classification System for Voting Rules

In 1977, Fishburn proposed a system for classifying rules according to the amount of information they require.

In a tournament, for each pair of alternatives a, b, either a defeats b or b defeats a. Laslier (1997) notes that there are many ways to formally define a tournament, and gives four

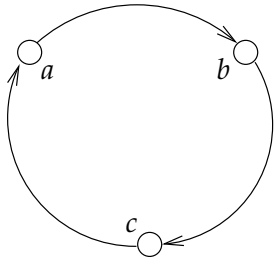
[19] The cardinality of a set is the number of elements in that set.

[20] A linear order should not be confused with a function of linear order. A linear order is essentially a ranking, e.g. $abcde$. A function of linear order is a linear function, e.g. $3x + 8$.

[21] The cartesian product of two sets A and B is the set of pairs (a, b) where a is a member of A and b is a member of B.

examples. We may define a tournament as a complete and asymmetric directed relation. Any such relation may be represented as a complete and asymmetric directed graph, i.e. a graph where the is an arc (directed edge) between every pair of distinct vertices, and for each pair of vertices, if there is an edge from a to b there is no edge from b to a. For example the directed graph below represents the intransitive case where a defeats b, who in turn defeats c who in turn defeats a.

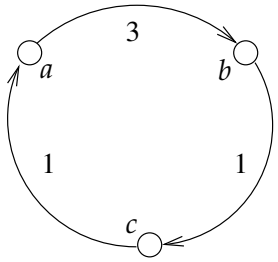
1	1	1
a	c	b
b	a	c
c	b	a



The majority relation is the tournament where a defeats b if and only if a is preferred to b by a majority of the agents in our profile $\mathcal{P}$. For example, the tournament above is a majority relation for the profile above (abc, cab, bca). Fishburn labels all rules which can pick their winner knowing only the majority relation as **C1 rules**.

A weighted tournament is much like an ordinary tournament, but allows weights to be attached to the edges. We typically use W to denote the weight function. A positive weight $W(a, b)$ is the weight of the edge from a to b, and $W(a, b) \equiv -W(b, a)$. For example, below is the graph of the weighted tournament where $W(a, b) = 3$, $W(b, c) = 1$, $W(c, a) = 1$; this could be written equivalently as $W(b, a) = -3$, $W(c, b) = -1$, $W(c, a) = -1$:

2	2	1
a	c	b
b	a	c
c	b	a



The weighted majority relation is the weighted tournament where the weight $W(a, b)$ is equal to the margin by which a would defeat b in a pairwise election[22], according to our profile $\mathcal{P}$. For example, the above weighted tournament is the weighted majority relation for the profile above $(abc, abc, cab, cab, bca)$. We calculate the weight of the edge from a to b as follows: 4 agents prefer a to b and one agent prefers b to a, hence the weight is 3. We calculate the other weights likewise. If we remove the weights from the edges we get an ordinary tournament (see e.g. Laslier 1997). Rules which require more information than the majority relation, but can determine their winner from the weighted majority relation are called **C2 rules**.

[22] A pairwise election is a sub-election where we exclude all but 2 of the m alternatives.

Examples of C2 rules include Simpson's rule, Tideman's rule and the Dodgson quick rule.

Rules which require yet more information are called **C3 rules**. Examples of C3 rules include Dodgson's rule, Kemeny's rule and our approximations: Dodgson relaxed and Dodgson R&R.

With C1 rules, it is useful to know whether a tournament is the majority relation for some society. The M^{c}Garvey Theorem (1953) states that this is true for all tournaments. However the M^{c}Garvey Theorem does not apply to weighted tournaments, and hence is not useful for C2 rules. For this reason we prove a generalization of the M^{c}Garvey Theorem to weighted tournaments.

1.2.2 Advantages

Let $\mathcal{P} = (P_1, P_2, \ldots, P_n)$ be our profile. We define n_{xy} to be the number of linear orders in $\mathcal{P}$ that rank x above y, i.e. $n_{xy} \equiv \#\{i \mid xP_iy\}$. The approximations we consider depend upon the information contained in the matrix $N_{\mathcal{P}}$, where $(N_{\mathcal{P}})_{ab} = n_{ab}$. Many of them use the numbers

$$\text{adv}(a, b) = \max(0, n_{ab} - n_{ba}) = (n_{ab} - n_{ba})^+,$$

which will be called **advantages**. Note that $\text{adv}(a, b) = \max(0, W(a, b)) = W(a, b)^+$ where W is the weighted majority relation on $\mathcal{P}$.

1.2.3 Condorcet Winner

A **Condorcet winner** is an alternative a for which $\text{adv}(b, a) = 0$ for all other alternatives b. A Condorcet winner does not always exist. The rules we consider below attempt to pick an alternative that is in some sense closest to being a Condorcet winner, and will always pick the Condorcet winner when it exists.

1.2.4 Scores

The social choice rules we consider are based on calculating the vector of **scores**. In the rules we describe below, the alternative with the lowest score wins. Let the lowest score be s. It is possible that more than one alternative has a score of s. In this case we may have a set of winners with cardinality greater than one. Strictly speaking, to be a social choice function, a rule has to output a single winner. Rules are commonly modified to achieve this by splitting ties according to the preference of the first voter. However we will usually study the set of tied winners rather than the single winner output from a tie-breaking procedure, as this will give us more information about the rules.

The **Dodgson score** (Dodgson 1876, see e.g. Black 1958; Tideman 1987), which we denote as $\text{Sc}_{\mathbf{D}}(a)$, of an alternative a is the minimum number of neighbouring alternatives that must be swapped to make a a Condorcet winner. For example, say our profile is the single vote cba. Then we swapping a and c would make a a Condorcet winner, however c and a are not neighbouring, so we would have to first swap ba and then swap ca. Hence the Dodgson score of a is 2. We call the alternative(s) with the lowest Dodgson score(s) the **Dodgson winner**(s).

The **Simpson score** (Simpson 1969, see e.g. Laslier 1997) $\text{Sc}_{\mathbf{S}}(a)$ of an alternative a is the maximum advantage of any other alternative b over a:

$$\text{Sc}_{\mathbf{S}}(a) = \max_{b \neq a} \text{adv}(b, a).$$

Although the Simpson rule was not designed to approximate the Dodgson rule, it is slightly simpler than the other approximations we study and so we are interested in whether it approximates the Dodgson rule well for a large number of voters. We call the alternative(s) with the lowest Simpson score(s) the **Simpson winner**(s). That is, the alternative with the smallest maximum defeat is the Simpson winner. This is why the rule is often known as the Maximin or Minimax rule.

The **Tideman score** (Tideman, 1987) $\text{Sc}_{\mathbf{T}}(a)$ of an alternative a is the sum of the advantages of each other alternative b over a:

$$\text{Sc}_{\mathbf{T}}(a) = \sum_{b \neq a} \text{adv}(b, a).$$

We call the alternative(s) with the lowest Tideman score(s) the **Tideman winner**(s). Tideman suggested this approximation as it can be quite hard to compute the Dodgson winner.

The **Dodgson Quick (DQ) score** $\text{Sc}_{\mathbf{Q}}(a)$, which is introduced in this thesis for the first time, of an alternative a is

$$\begin{aligned} \text{Sc}_{\mathbf{Q}}(a) &= \sum_{b \neq a} F(b, a), \\ \text{where } F(b, a) &= \left\lceil \frac{\text{adv}(b, a)}{2} \right\rceil. \end{aligned}$$

That is, the Dodgson Quick score of an alternative a is the sum of the advantage of each other alternative b over a divided by two and rounded up. See Section 1.1.7 for an informal discussion of why this leads to a better approximation than Tideman's rule, and see Chapter 3 for formal proofs. We call the alternative(s) with the lowest Dodgson Quick

score(s) the **Dodgson Quick winner**(s) or **DQ-winner**.

The **Dodgson relaxed (DR) score** $\text{Sc}_{\mathbf{R}}(a)$, which is introduced in this thesis for the first time, is defined similarly to the Dodgson score. Under the Dodgson rule, we may only switch neighbouring alternatives in whole votes. The **Dodgson relaxed rule** allows us to split votes into rational[23] fractions of a vote and swap neighbouring alternatives in these fractions of a vote. However we must swap each other alternative d over a at least $\lceil \text{adv}(a,d)/2 \rceil$ times, even though $\text{adv}(a,d)/2$ times would be enough to made d a Condorcet winner. (Under the Dodgson rule, we must also swap each alternated d over a at least $\lceil \text{adv}(a,d)/2 \rceil$ times, as the Dodgson rule does not allow us to swap alternative in fractional votes). For a more complete definition of the DR score, and discussion of this rule, see Chapter 5.

The **Dodgson Relaxed and Rounded (R&R)** score $\text{Sc}_{\&}(a)$, which is introduced in this thesis for the first time, of an alternative a, is the ceiling[24] $\lceil \text{Sc}_{\mathbf{R}}(a) \rceil$ of the Dodgson relaxed score.

We will not define the Kemeny and Borda rule. Although we mentioned these rules in the introduction, we do not study these rules, and so we do not need a formal definitions of these rules.

1.2.5 Impartial Culture Assumption

The **impartial culture** assumption is that all possible profiles $\mathcal{P}$ are equally likely[25]. This assumption is of course unrealistic. Worse, we have found that the choice of probability model can affect the similarities between approximations to the Dodgson rule (M$^{\text{c}}$Cabe-Dansted and Slinko, 2006). However it is impossible to select an assumption that accurately reflects the voting behaviour of all voting societies. Berg (1985) suggests studying voting properties under a variety of voting assumptions. We have conducted a broader survey of relationships between voting rules M$^{\text{c}}$Cabe-Dansted and Slinko (2006), in this thesis we instead seek to gain an in depth understanding of the Approximability of Dodgson's rule. This requires us to focus on a single assumption of voting behaviour. The impartial culture is the simplest assumption available. As noted by Berg (1985), many voting

[23] A rational number is a number that can be represented as n/m where n and m are integers. In fact the Dodgson relaxed rule could be equivalently defined as allowing splits into any real fraction of a vote, the Dodgson relaxed score would be the same either way.

[24] The ceiling of a number x is the smallest integer that is greater than or equal to x, i.e. x rounded up to the nearest integer.

[25] This is not to be confused with the impartial anonymous assumption that all possible voting situations, are equally likely. With a two alternative, two agent election there are four possible profiles (ab, ab), (ab, ba), (ba, ab) and (ba, ba). However there are only three possible voting situations because $\{ab, ba\} = \{ba, ab\}$. With a large number of agents the amount of support for each alternative will be roughly the same. For example, if we have a million agents we would expect to see half a million, give or take a thousand, supporters of each alternative.

theorists have chosen to focus their research upon the impartial culture assumption. Thus an in depth study of the Approximability of Dodgson's rule under the impartial culture is a natural first step.

We may derive a multinomial distribution from the impartial culture assumption as follows. Let $\mathcal{P}$ be a random profile defined on a set of m alternatives $\mathcal{A}$ and n agents. Let then X be a vector where each X_i represents the number of occurrences of a distinct linear order in the profile $\mathcal{P}$. Then, under the impartial culture assumption, the vector X is $(n, k, \mathbf{p})$-multinomially distributed with $k = m!$ and $\mathbf{p} = \mathbf{1}_k/k = (\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k})$.

An $(n, k, \mathbf{p})$-multinomial distribution is similar to a binomial distribution with n trials. However such a multinomial distribution has k elementary outcomes instead of just "success" and "failure". See Appendix A for a formal definition of an $(n, k, \mathbf{p})$-multinomial distribution, as well as our definition of an (n, p)-binomial distribution and a multivariate $(n, \mathbf{b}, \Omega)$-normal distribution.

1.2.6 Pólya-Eggenberger Urn Model

Although our theoretical results deal only with the impartial culture assumption, we include some numerical results using the Pólya-Eggenberger urn model, to illustrate how differences in the assumption on the voting behaviour of the agents can lead to different results.

Under this model we start with a big urn containing balls, each of a different colour and a non-negative integer a. To generate each random sample, we pull a ball out of the Urn at random and note its colour. After removing each ball, we return the ball that was taken to the urn together with a additional balls of the same colour to the urn.

For generation of random profiles, we replace colours with linear orders. The parameter a characterises homogeneity; for $a = 0$ we obtain the well known Impartial Culture conjecture and for $a = 1$ the so-called Impartial Anonymous Culture conjecture (Berg and Lepelly, 1994). In this thesis we wish the value of the parameter of homogeneity to have the same meaning for different numbers of alternatives. Therefore we use a normalized parameter $b = \frac{a}{m!}$ as our main parameter of homogeneity. For example, if $b = 1$ the second agent copies the first agent approximately half of the time regardless of the number of alternatives.

1.3 Summary

We have introduced a number of major approaches to choosing a winning candidate. One of the major approaches to voting is to consider the result of pairwise elections. Under

this approach we attempt to chose a Condorcet winner. However pairwise elections can have cycles, preventing the occurrence of a Condorcet winner. A number of rules have been suggested that pick the Condorcet winner when it exists, and otherwise picks a candidate that is in some sense the closest to being a Condorcet winner. A particularly interesting example of such a rule is the Dodgson rule.

As it can be computationally hard to find the Dodgson winner, we seek approximations that can be easily computed, yet will frequently pick the Dodgson winner. We propose a new approximation that we call the Dodgson Quick rule. Under the impartial culture assumption, we will prove that the probability that this rule picks the Dodgson winner converges to 1 exponentially fast as we increase the number of voters. We will use this result to prove that the approximation suggested by Tideman also converges to the Dodgson rule. However we prove that this approximation does not converge exponentially fast, in this sense our new approximation is superior.

We also propose the new approximations Dodgson Relaxed and Dodgson Relaxed and Rounded (R&R). The Dodgson Relaxed approximation is defined very similarly to the Dodgson rule, but allows votes to be split into rational fractions of a vote, and neighbouring alternatives to be swapped in these fractional votes. The R&R scores are the Dodgson Relaxed score rounded up. We will show that the scores and winners of these rules can be computed in polynomial time.

We find that the approximations to the Dodgson rule (Tideman, DQ, DR, and R&R) provide increasingly accurate lower bounds to the true Dodgson score. That is, given any profile $\mathcal{P}$ and alternative a

$$1/2\mathrm{Sc}_{\mathbf{T}}(a) \leq \mathrm{Sc}_{\mathbf{Q}}(a) \leq \mathrm{Sc}_{\mathbf{R}}(a) \leq \mathrm{Sc}_{\&}(a) \leq \mathrm{Sc}_{\mathbf{D}}(a).$$

The final approximation R&R is so close that we doubt that we would ever come across a real world example of the R&R rule picking a different winner to the Dodgson rule. Out of 43 million simulations, using different profile sizes and assumptions on voter behaviour, we only found one case were the R&R winner differed from the Dodgson winner. This case was with 25 alternatives and 5 agents, but real world examples usually have many more agents than alternatives. We suggest the R&R rule is in some sense better than the Dodgson rule, because the R&R rule sometimes allows us to break ties in favor of alternatives that are fractionally better according to the relaxed rule[26].

Since real world elections typically only have a limited number of alternatives, we will study how long it takes to compute the Dodgson winner for a fixed number of alternatives. We find that we can find the Dodgson score and Dodgson winner with $\mathcal{O}(\log n)$ ex-

[26]The R&R score is the Dodgson relaxed score rounded up. This when faced with tied R&R winners, one can choose break ties based upon which candidate had a fractionally lower Dodgson relaxed score.

pected time; in, the worst-case we require $\mathcal{O}(\log n)$ basic arithmetic operations of $\mathcal{O}(\log n)$ digit numbers[27]. This result means that finding the Dodgson score and finding the Dodgson winner are Fixed Parameter Tractable (FPT) problems with respect to the parameter m, the number of alternatives. We find it interesting that in the general case *Dodgson Winner* is harder even than NP-complete problems (Hemaspaandra et al., 1997), yet when we make the reasonable assumption that the number of alternatives is fixed, the finding the Dodgson winner is such an easy problem.

An approximation similar to our Dodgson Quick approximation was independently suggested by Homan and Hemaspaandra (2005). However we approached the problem from a different angle. Unlike the algorithm proposed by Homan and Hemaspaandra, our scores provide lower bounds for the Dodgson scores. Also unlike Homan and Hemaspaandra's C3 rule, our approximation is a C2 rule, meaning that it only requires knowledge of the weighted majority relation to find its winner, which can make our approximation easier to study mathematically.

A diagram of the relationship between the concepts relevant to this thesis is provided on page 141. This figure is included solely to assist those who find visual summaries easier to understand. It does not introduce any new material, nor is it otherwise required to understand this thesis.

In our proofs we find it useful to know whether there is a profile for which a particular weighted tournament is a weighted majority relation. M^{c}Garvey's Theorem states that for any tournament there exists a profile for which that tournament is the majority relation. However this result does not extend to weighted tournaments, and so is of no use in our proofs. Fortunately, we know that for any weighted tournament, it is possible to find a profile for which that weighted tournament is the weighted majority relation for that profile. This result was probably first proven by Debord (1987). As we do not have access to this paper, we will present an independent proof.

[27] Note that each basic arithmetic operation (addition, subtraction and multiplication) on d digit numbers requires $\mathcal{O}(d \log d \log \log d)$ time (Schönhage and Strassen, 1971).

Chapter 2

A McGarvey Theorem for Weighted Tournaments

The McGarvey Theorem (1953) is a famous theorem that states that every tournament can be represented as a majority relation for a certain society of voters. We will prove a generalization of the McGarvey Theorem to weighted tournaments and weighted majority relations.

Laslier (1997), calls weighted tournaments "generalized tournaments". However the term "generalized tournament" seems to be less popular with other authors and the term "weighted tournament" gives an indication of how the tournament has been generalized.

Like most other authors, Laslier defines weighted tournaments as matrices and tournaments as (complete and asymmetric) binary relations. However Laslier notes that there are many different equivalent definitions of tournaments, of which Laslier gives four examples. In this Chapter we define both tournaments and weighted tournaments as functions, for consistency. In Chapter 1 we gave a definition of tournaments as complete and asymmetric graphs. This definition is useful for presenting tournaments. However, in proofs, we find it more convenient to define tournaments as functions.

Definition 2.0.1 *Let a **weighted tournament** on A be a function $W : A \times A \to \mathbb{Z}$, such that $W(a,b) = -W(b,a)$ for all a, b. We call $W(a,b)$ a **weight** if $a \neq b$.*

We may equivalently draw a weighted tournament as a weighted graph, i.e. a directed graph with integers (weights) attached to edges. An edge is drawn from a to b, with an arrow pointing to b, if and only if $W(a,b) > 0$.

Note 2.0.2 *Tournaments are not indifferent between any pair of distinct alternatives. For this reason we cannot convert a weighted tournament which contains a 0 weight to an ordinary tournament simply by removing the weights form the edges of the directed graph.*

Definition 2.0.3 *We define the sum $W_1 + W_2$ of two weighted tournaments as a function f, where*

for all alternatives a and b we have:

$$f(a,b) \quad = \quad W_1(a,b) + W_2(a,b).$$

Similarly, we define the difference between two weighted tournaments $W_1 - W_2$ as a function f where for all alternatives a and b we have:

$$f(a,b) \quad = \quad W_1(a,b) - W_2(a,b).$$

Definition 2.0.4 *We define the **weighted majority relation** $W^{\mathcal{P}}$ on a profile $\mathcal{P}$ as the weighted tournament where each weight $W^{\mathcal{P}}(a,b)$ of a pair of alternatives (a,b) equals $n_{ab} - n_{ba}$[1]. We say that a profile $\mathcal{P}$ **generates** a weighted tournament $W^{\mathcal{P}}$ if $W^{\mathcal{P}}$ is the weighted majority relation on $\mathcal{P}$.*

For example, we say that the profile $(abc, abc, cab, cab, bca)$ generates the weighted tournament above.

Note 2.0.5 *$adv(a,b) = W^{\mathcal{P}}(a,b)^+$, where $x^+ = \max(0,x)$. Similarly $W^{\mathcal{P}}(a,b) = adv(a,b) - adv(b,a)$.*

Below we define the tournament and majority relation as function. The definition of tournament below is equivalent to the more traditional definition as a complete and asymmetric binary relation, used by Laslier (1997); we simply write $W(a,b) = 1$ where Laslier would write aWb.

Our definition of majority relation is also equivalent to Laslier's (1997, p. 34, definition 2.1.2). Some other authors include a λ parameter or tie-breaking in their definition of a majority relation. We do not include λ or tie-breaking in our definition; these concepts are not needed to express the M^{c}Garvey theorem.

Definition 2.0.6 *A **tournament** W on A is a weighted tournament where all weights are 1 or -1.*

Definition 2.0.7 *We define the **majority relation** $W^{\mathcal{P}}$ on a profile $\mathcal{P}$ as the tournament where each weight $W^{\mathcal{P}}(a,b)$ of a pair of alternatives (a,b) equals 1 if and only if $n_{ab} - n_{ba}$ is greater than zero. We say that a profile $\mathcal{P}$ **generates** a tournament $W^{\mathcal{P}}$ if $W^{\mathcal{P}}$ is the majority relation on $\mathcal{P}$.*

Definition 2.0.8 *For a weighted tournament W, for which all weights are non-zero, we define the **reduction** of W to be the tournament W_S where:*

$$W_S(a,b) = 1 \quad \Longleftrightarrow \quad W(a,b) > 0.$$

[1] Recall that on page xii we define n_{dc}, for any pair of alternatives d and c, as the number of agents in our profile $\mathcal{P}$ who rank c above d, i.e. $\#\{d\mathcal{P}_i c\}$.

Thus, from the definition of a weighted tournament:

$$W_S(a,b) \;=\; \begin{cases} 1 & \textit{if} \quad W(a,b) > 0 \\ -1 & \textit{if} \quad W(a,b) < 0 \\ 0 & \textit{if} \quad\quad a = b \end{cases} .$$

Note 2.0.9 *If $W^{\mathcal{P}}$ is the weighted majority relation on the profile $\mathcal{P}$, then the reduction $W_S^{\mathcal{P}}$ is the majority relation on the profile $\mathcal{P}$.*

Definition 2.0.10 *The **majority relation** on a profile $\mathcal{P}$ is the reduction $W_S^{\mathcal{P}}$ of $W^{\mathcal{P}}$, where $W^{\mathcal{P}}$ is the weighted majority relation on $\mathcal{P}$.*

We will now state the M^{c}Garvey Theorem in terms of tournaments and majority relations:

Theorem 2.0.11 (M^{c}Garvey 1953) *For every tournament W there exists a profile $\mathcal{P}$ such that W is the majority relation generated by $\mathcal{P}$.*

Lemma 2.0.12 *Let $W^{\mathcal{P}}$ be a weighted majority relation on a profile $\mathcal{P}$ with n agents, then all weights in $W^{\mathcal{P}}$ have the same parity as n. That is, for each pair of distinct alternatives a and b, the weight $W^{\mathcal{P}}(a,b)$ is even if and only if n is even.*

Proof. We know that for all alternatives a and b we have $W^{\mathcal{P}}(a,b) = n_{ab} - n_{ba}$ and $n = n_{ba} + n_{ab}$. Hence $W^{\mathcal{P}}(a,b) + n = 2n_{ab}$ and so $W^{\mathcal{P}}(a,b)$ and n have the same parity. □

Lemma 2.0.13 *For a weighted tournament W with all weights being even, we may construct a profile $\mathcal{P}$ for which W is a weighted majority relation. This profile has exactly $\sum W(a,b)^{+}$ agents.*

Proof. We may construct such a profile as follows:

We start with an empty profile $\mathcal{P}$. For each pair of alternatives (a,b), for which the weight $W(a,b)$ is positive, we let $k = W(a,b)/2$. We take a linear order $\mathbf{v}$, on the set of alternatives $\mathcal{A}$, such that $a\mathbf{v}b$ and $b\mathbf{v}x$ for all $x \neq a,b$. For example, $\mathbf{v} = abcde$. We then reverse the linear order, keeping a ranked above b, in this case producing $\mathbf{w} = edcab$. We add k instances of $\mathbf{v}$ and k instances of $\mathbf{w}$ to the profile $\mathcal{P}$. This ensures that the weight of (a,b) generated by profile $\mathcal{P}$ is equal to $W(a,b)$ without affecting the weight of (x,y) where $(b,a) \neq (x,y) \neq (a,b)$.

For each positive weight $W(a,b)$ we used exactly $W(a,b)$ agents. Thus there are exactly $\sum W(a,b)^{+}$ agents in the constructed profile. □

Note 2.0.14 *In the lemma above we have found an upper bound $\sum W(a,b)^+$ on the number of agents required. Where we have m alternatives and all positive weights are 2 (the smallest positive even number), we require $2\binom{m}{2}$ agents. This is the same upper bound that M^{c}Garvey found for ordinary tournaments. As M^{c}Garvey suspected, there was a tighter upper bound. Erdös and Moser (1964) has shown that for ordinary tournaments we will require no more than $c_1 m/\ln m$ agents, where c_1 is some fixed positive constant. From the previous work by Stearns (1959) we know that it is not possible to find tighter bound than this for ordinary tournaments, as there exists a positive constant c_2 such that for all m there exists a tournament, with only m alternatives, for which more than $c_2 m/\ln m$ agents are required.*

For a weighted tournament with a weight $W(a,b) = w$, we know that the profile will need to contain at least w agents that prefer a to b. Thus $\max_{a,b}(W(a,b))$ provides a lower bound on the number of agents required for weighted tournaments, and so the number of required agents n is unbounded for a fixed number of alternatives m. It follows that Erdös and Moser's bound does not apply to weighted tournaments. Never-the-less, it may be possible to find an equivalent of this bound for weighted tournaments. We have not attempted to do so, as the number of agents required is irrelevant to our proofs in Chapter 3.

Lemma 2.0.15 *For a weighted tournament W with all weights being odd, we may construct a profile which generates this weighted tournament. We will need no more than*

$$\frac{m(m-1)}{2} + \sum_{(a,b)} W(a,b)^+$$

agents to construct this profile.

Proof. Let W_1 be the weighted majority relation of a profile consisting of a single arbitrarily chosen linear order $\mathbf{v}$. Let $W_2 = W - W_1$.

Note that as W_1 is generated from a profile with an odd number (i.e. one) of linear orders, all the weights in W_1 must be odd. Thus all weights in W_2 are the difference between two odd numbers. Hence all weights in W_2 are even and we can construct a profile for which W_2 is the majority relation, as shown by Lemma 2.0.13. Since $W = W_1 + W_2$, joining the profiles that generate W_1 and W_2 constructs a profile that generates W.

Now we shall determine an upper bound on how many agents we will need. Say that $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}$ are the profiles we constructed that generate W_1, W_2, and W respectively. From the construction in Lemma 2.0.13, there will be exactly $\sum_{(a,b)} W_2(a,b)^+$ agents in $\mathcal{P}_2$, and thus exactly $1 + \sum_{(a,b)} W_2(a,b)^+$ agents in $\mathcal{P}$. We may pick $\mathbf{v}$ such that $c\mathbf{v}d$, where c, d are alternatives such that $W(c,d) \geq 1$. As $W_2 = W - W_1$, we have $W_2(c,d)^+ = W(c,d)^+ - 1$.

In a complete graph with m vertices, there are $m(m-1)/2$ edges. Thus, for $m(m-1)/2$ pairs of alternatives $W_1(a,b) = -1$, and hence $W_2(a,b) = W(a,b) + 1$. Thus for no more

than $m(m-1)/2$ pairs (a,b) of alternatives $W_2(a,b) = W(a,b) + 1$. Thus,

$$\sum_{(a,b)} W_2(a,b)^+ \quad \leq \quad (-1) + \frac{m(m-1)}{2} + \sum_{(a,b)} W(a,b)^+.$$

Now we have an upper bound for the number of agents we will need to construct $\mathcal{P}_2$. As there is one more agent in $\mathcal{P}$ that is not in $\mathcal{P}_2$, we know that we need at most

$$\frac{m(m-1)}{2} + \sum_{(a,b)} W(a,b)^+$$

agents to construct $\mathcal{P}$. □

We may now prove our generalisation to the M^{c}Garvey theorem.

Theorem 2.0.16 *There exists a profile that generates a weighted tournament W if and only if all weights in W have the same parity.*

Proof. ($\Longleftarrow$) From the last two lemmas, we know that if all weights are even or if all weights are odd, we can construct a profile that generates W.

($\Longrightarrow$) We know that if n is odd our profile will generate a weighted tournament with all weights odd, if n is even our profile will generate a weighted tournament with all weights even. Thus every profile generates a weighted tournament for which either all weights are even or all weights are odd.

□

Note 2.0.17 *All weights in a tournament W_S are odd (1 or -1). Thus from Theorem 2.0.16, we may find a profile $\mathcal{P}$ such that the weighted majority relation $W^{\mathcal{P}}$ is equal to W_S. As $W^{\mathcal{P}}$ is already an ordinary tournament, its reduction $W_S^{\mathcal{P}}$ is equal to $W^{\mathcal{P}}$, and hence also equal to W_S.*

Hence the M^{c}Garvey theorem, that for all tournaments W_S there exists a profile $\mathcal{P}$ such that the majority relation W_S^P is equal to W_S, is a special case of Theorem 2.0.16.

Chapter 3

Simple Rules that Approximate the Dodgson Rule

3.1 Dodgson Quick, A New Approximation

Definition 3.1.1 *We say that b is **ranked directly above** a in a linear order $\mathbf{v}$ if and only if $a\mathbf{v}b$ and there does not exist c different from a, b such that $a\mathbf{v}c \wedge c\mathbf{v}b$.*

Definition 3.1.2 *Recall that given a profile $\mathcal{P}$, we define $D(b, a)$ as the number of agents who rank b directly above a in their preference list, and we define $F(b, a)$ and the Dodgson Quick (DQ) score $Sc_{\mathbf{Q}}(a)$ of an alternative a as follows*[1]

$$\begin{aligned} F(b,a) &= \left\lceil \frac{adv(b,a)}{2} \right\rceil, \\ Sc_{\mathbf{Q}}(a) &= \Sigma_{b \neq a} F(b,a). \end{aligned}$$

The Dodgson Quick score and the rule Dodgson Quick (DQ) based on that score is introduced in this thesis. Recall also that we define the Dodgson score $\mathrm{Sc}_{\mathbf{D}}(a)$ of an alternative a as the minimum number of neighbouring preferences that must be swapped to make a a Condorcet winner.

Lemma 3.1.3 *For distinct alternatives $a, b \in \mathcal{A}$, under the impartial culture assumption n_{ba} and $D(b, a)$ are binomial random variables with means of $n/2$ and n/m respectively.*

Proof. For each linear order $\mathbf{v}$, we may reverse the order $\mathbf{v}$ to produce its opposite $\bar{\mathbf{v}}$, i.e. $c\mathbf{v}d \Leftrightarrow d\bar{\mathbf{v}}c$ for all $c, d \in \mathcal{A}$. This operation $v \mapsto \bar{v}$ provides a bijection between linear

[1] Recall that $\lceil x \rceil$ is defined on page xi as the ceiling of x, the smallest integer that is greater than or equal to x. Also recall that n_{ba} was defined as the number of agents who ranked alternative b above alternative a in their preference list.

orders where b is ranked above a and those where b is ranked below a. Hence these two sets of linear orders have the same cardinality. Under the impartial culture assumption, this implies that the probability that any agent ranks b above a is $1/2$.

The number of ways that b can be ranked *directly* above a is easily calculated if we consider the pair ba to be one object. Then we see that this number is equal to the number of permutations of $(m-1)$ objects, i.e. $(m-1)!$. The probability that b is ranked directly above a is $(m-1)!/m!$, which is equal to $1/m$.

Since votes are independent under the impartial culture assumption, n_{ba} and $D(b,a)$ are binomially distributed random variables. The mean of a binomially distributed random variable is np, so the means of n_{ba} and $D(b,a)$ are $n/2$ and n/m, respectively. □

Lemma 3.1.4 *Under the impartial culture assumption, the probability that $D(x,a) > F(x,a)$ for all x converges exponentially fast to 1 as the number of agents n tends to infinity.*

Proof. Under the impartial culture assumption, n_{ba} and $D(b,a)$ are binomially distributed with means of $n/2$ and n/m respectively. From Chomsky's (Dembo and Zeitouni, 1993) large deviation theorem, we know that for a fixed number of alternatives m there exist $\beta_1 > 0$ and $\beta_2 > 0$ such that

$$\begin{aligned} P\left(\frac{D(b,a)}{n} < \frac{1}{2m}\right) &\leq e^{-\beta_1 n}, \\ P\left(\frac{n_{ba}}{n} - \frac{1}{2} > \frac{1}{4m}\right) &\leq e^{-\beta_2 n}. \end{aligned}$$

We can rearrange the second equation to involve $F(b,a)$,

$$\begin{aligned} P\left(\frac{n_{ba}}{n} - \frac{1}{2} > \frac{1}{4m}\right) &= P\left(\frac{2n_{ba}}{n} - 1 > \frac{1}{2m}\right) \\ &= P\left(\frac{2n_{ba} - n}{n} > \frac{1}{2m}\right) \\ &= P\left(\frac{n_{ba} - (n - n_{ba})}{n} > \frac{1}{2m}\right) \\ &= P\left(\frac{n_{ba} - n_{ab}}{n} > \frac{1}{2m}\right) \\ &= P\left(\frac{\text{adv}(b,a)}{n} > \frac{1}{2m}\right). \end{aligned}$$

Since $\text{adv}(b,a) \geq F(b,a)$,

$$P\left(\frac{n_{ba}}{n} - \frac{1}{2} > \frac{1}{4m}\right) \geq P\left(\frac{F(b,a)}{n} > \frac{1}{2m}\right).$$

From this and the law of probability $P(A \vee B) \leq P(A) + P(B)$ it follows that

$$\begin{aligned} P\left(\frac{F(b,a)}{n} > \frac{1}{2m}\right) &\leq e^{-\beta_2 n}, \\ P\left(\frac{D(b,a)}{n} < \frac{1}{2m}\right) &\leq e^{-\beta_1 n}, \end{aligned}$$

and so, where $\beta = \min(\beta_1, \beta_2)$,

$$\begin{aligned} P\left(\frac{F(b,a)}{n} > \frac{1}{2m} \vee \frac{D(b,a)}{n} < \frac{1}{2m}\right) &\leq e^{-\beta_1 n} + e^{-\beta_2 n} \\ &\leq 2e^{-\beta n}. \end{aligned}$$

Hence

$$P\left(\exists_x \frac{F(x,a)}{n} > \frac{1}{2m} \vee \frac{D(x,a)}{n} < \frac{1}{2m}\right) \leq 2me^{-\beta n}.$$

Using $P(\bar{E}) = 1 - P(E)$, we find that

$$P\left(\forall_x \frac{F(x,a)}{n} < \frac{1}{2m} < \frac{D(x,a)}{n}\right) \geq 1 - 2me^{-\beta n}.$$

□

Lemma 3.1.5 *The DQ-score $Sc_{\mathbf{Q}}(a)$ is a lower bound for the Dodgson Score $Sc_{\mathbf{D}}(a)$ of a.*

Proof. Let $\mathcal{P}$ be a profile and $a \in \mathcal{A}$. Suppose we are allowed to change linear orders in $\mathcal{P}$, by repeated swapping neighbouring alternatives. Then to make a a Condorcet winner we must reduce adv(x, a) to 0 for all x and we know that adv$(x, a) = 0$ if and only if $F(x, a) = 0$. Swapping a over neighbouring alternative b will reduce $(n_{ba} - n_{ab})$ by two, but this will not affect $(n_{ca} - n_{ac})$ where $a \neq c$. Thus swapping a over neighbouring b will reduce $F(b, a)$ by one, but will not affect $F(c, a)$ where $b \neq c$. Therefore, making a a Condorcet winner will require at least $\Sigma_b F(b, a)$ swaps. This is the DQ-Score $\mathrm{Sc}_{\mathbf{Q}}(a)$ of a.
□

Lemma 3.1.6 *If $D(x, a) \geq F(x, a)$ for every alternative x, then the DQ-Score $Sc_{\mathbf{Q}}(a)$ of a is equal to the Dodgson Score $Sc_{\mathbf{D}}(a)$.*

Proof. If $F(b, a) \leq D(b, a)$, we can find at least $F(b, a)$ linear orders in the profile where b is ranked directly above a. Thus we can swap a directly over b, $F(b, a)$ times, reducing

$F(b,a)$ to 0. Hence we can reduce $F(x,a)$ to 0 for all x, making a a Condorcet winner, using $\Sigma_x F(x,a)$ swaps of neighbouring preferences. In this case, $\text{Sc}_{\mathbf{Q}}(a) = \Sigma_b F(b,a)$ is an upper bound for the Dodgson Score $\text{Sc}_{\mathbf{D}}(a)$ of a. From Lemma 3.1.5 above, $\text{Sc}_{\mathbf{Q}}(a)$ is also a lower bound for $\text{Sc}_{\mathbf{D}}(a)$. Hence $\text{Sc}_{\mathbf{Q}}(a) = \text{Sc}_{\mathbf{D}}(a)$. □

Corollary 3.1.7 *If $D(x,a) \geq F(x,a)$ for every pair of distinct alternatives (x,a), then the DQ-Winner is equal to the Dodgson winner.*

Theorem 3.1.8 *Under the impartial culture assumption, the probability that the DQ-Score $Sc_{\mathbf{Q}}(a)$ of an arbitrary alternative a is equal to the Dodgson Score $Sc_{\mathbf{D}}(a)$, converges to 1 exponentially fast.*

Proof. From Lemma 3.1.6, if $D(x,a) \geq F(x,a)$ for all alternatives x then $\text{Sc}_{\mathbf{Q}}(a)$= $\text{Sc}_{\mathbf{D}}(a)$ however from Lemma 3.1.4, $P(\forall x\ D(x,a) \geq F(x,a))$ converges exponentially fast to 1 as $n \to \infty$. □

Corollary 3.1.9 *There exists an algorithm that computes the Dodgson score of an alternative a given the frequency of each linear order in the profile $\mathcal{P}$ as input, with expected running time that is logarithmic with respect the number of agents (i.e. is $\mathcal{O}(\ln n)$ for a fixed number of alternatives m).*

Proof. The are at most $m!$ distinct linear orders in the profile. Hence for a fixed number of alternatives the number of distinct linear orders is bounded. Hence we may find the DQ-score and check whether $D(x,a) \geq F(x,a)$ for all alternatives x using a fixed number of additions. The largest number that needs to be added is proportional to the number of agents n. Additions can be performed in time linear with respect to the number of bits - logarithmic with respect to the size of the number. So we have only used an amount of time that is logarithmic with respect to the number of agents.

If $D(x,a) \geq F(x,a)$ for all alternatives x, we know that the DQ-score is the Dodgson score and we do not need to go further. From Lemma 3.1.4 we know that the probability that we need go further declines exponentially fast, and we can still find the Dodgson score in time that is polynomial with respect to the number of agents (Bartholdi et al., 1989). □

Corollary 3.1.10 *There exists an algorithm that computes the Dodgson winner given the frequency of each linear order in the profile $\mathcal{P}$, and has expected running time that is logarithmic with respect the number of agents.*

Corollary 3.1.11 *Under the impartial culture assumption, the probability that the DQ-Winner is the Dodgson winner converges to 1 exponentially fast as we increase the number of agents.*

Obvious from Theorem 3.1.8 above.

We know that the probability that the DQ-Score equals the Dodgson Score converges to 1 at an exponential rate from Theorem 3.1.8 above. Theorem 3.1.13 below does not imply exponential convergence, however this theorem does give a lower bound for this probability given a profile of a particular size. To give an idea as to when the DQ-Score will begin to converge to the Dodgson Score, we will start working towards Theorem 3.1.13.

Lemma 3.1.12 *Let X be a random variable with variance σ^2 and mean of $\mu > 0$, then the probability $P(X \leq 0)$, that X is non-positive, is no greater than $\frac{\sigma^2}{\mu^2}$.*

Proof. Chebyshev's theorem as states (see e.g. Walpole and Myers, 1993, p108) that

$$P(\mu - k\sigma < X < \mu + k\sigma) \quad \geq \quad 1 - \frac{1}{k^2}.$$

Thus

$$P(X > \mu - k\sigma) \quad \geq \quad 1 - \frac{1}{k^2}.$$

As $P(E) = 1 - P(\bar{E})$,

$$P(X \leq \mu - k\sigma) \quad \leq \quad \frac{1}{k^2}.$$

Setting $k = \mu/\sigma$,

$$P(X \leq \mu - \frac{\mu}{\sigma}\sigma) \quad \leq \quad \frac{1}{\left(\frac{\mu}{\sigma}\right)^2},$$

from which, by simplifying both sides, we obtain

$$P(X \leq 0) \quad \leq \quad \frac{\sigma^2}{\mu^2}.$$

□

Theorem 3.1.13 *Under the impartial culture assumption, the probability that the DQ-Score $\Sigma_b F(b, a)$ is not equal to the Dodgson Score of a is less than $\frac{m^3}{n}$.*

Proof. Let $\mathbf{p} = (p_1, p_2, p_3)$ be a vector. Consider the $(n, 3, \mathbf{p})$-multinomial distribution where outcome 1 is "a is ranked over b", outcome 2 is "b is ranked *directly* over a", and outcome 3 is "b is ranked over a, but not directly". Let X be our random vector. From Lemma 3.1.3 the outcomes' probabilities are $p_1 = 1/2$, $p_2 + p_3 = 1/2$, $p_2 = 1/m$. We may deduce that $E[X_1 - X_3] = n/m$, and $\text{var}(X_1 - X_3) \leq n$ as follows,

$$p_3 = \frac{1}{2} - \frac{1}{m},$$

thus,

$$E[X_1 - X_3] = \frac{n}{2} - \left(\frac{n}{2} - \frac{n}{m}\right) = \frac{n}{m}.$$

As X is multinomially distributed, $\text{cov}(X_i, X_j) = -np_ip_j$, so

$$\begin{aligned}
\text{cov}(X_1, -X_3) &= n\left(\frac{1}{2}\right)\left(\frac{1}{2} - \frac{1}{m}\right), \\
\text{var}(X_i + X_j) &= \text{var}(X_i) + \text{var}(X_j) + 2\text{cov}(X_i, X_j), \\
\text{var}(X_i) &= np_iq_i,
\end{aligned}$$

and thus

$$\begin{aligned}
\text{var}(X_1 - X_3) &= n\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + n\left(\frac{1}{2} - \frac{1}{m}\right)\left(\frac{1}{m} - \frac{1}{2}\right) + 2\left(\frac{1}{2}\right)\left(\frac{1}{2} - \frac{1}{m}\right) \\
&\leq n\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + n\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) + 2n\left(\frac{1}{2}\right)\left(\frac{1}{2}\right) = n.
\end{aligned}$$

We may now formulate the upper bound for the probability that the DQ-Score $\text{Sc}_{\mathbf{Q}}(a)$ is not equal to the Dodgson Score $\text{Sc}_{\mathbf{D}}(a)$ in terms of this multinomial distribution, using Lemma 3.1.6.

$$P\left(\text{Sc}_{\mathbf{Q}}(a) \neq \text{Sc}_{\mathbf{D}}(a)\right) \leq P(\exists b : D(b, a) < F(b, a)),$$

as $P(A \vee B) \leq P(A) + P(B)$

$$P\left(\text{Sc}_{\mathbf{Q}}(a) \neq \text{Sc}_{\mathbf{D}}(a)\right) \leq mP(D(b, a) < F(b, a)).$$

Because $n_{ba} - n_{ab} \geq \text{adv}(b, a) \geq F(b, a)$,

$$\begin{aligned}
P\left(\text{Sc}_{\mathbf{Q}}(a) \neq \text{Sc}_{\mathbf{D}}(a)\right) &\leq mP(D(b, a) < n_{ba} - n_{ab}) \\
&= mP(D(b, a) - n_{ba} + n_{ab} < 0),
\end{aligned}$$

and $D(b,a) = X_2$, $n_{ba} = (X_2 + X_3)$, $n_{ab} = X_1$, so

$$\begin{aligned} P\left(\mathrm{Sc}_{\mathbf{Q}}(a) \neq \mathrm{Sc}_{\mathbf{D}}(a)\right) &\leq mP(X_2 - (X_2 + X_3) + X_1 < 0) \\ &= mP(X_1 - X_3 < 0). \end{aligned}$$

We now find an upper bound for $P(X_1 - X_3 < 0)$ using Lemma 3.1.12.

$$\begin{aligned} P(X_1 - X_3 < 0) &\leq \frac{\mathrm{var}(X_1 - X_3)}{E[X_1 - X_3]^2} \\ &\leq \frac{n}{\left(\frac{n}{m}\right)^2} = \frac{m^2}{n}. \end{aligned}$$

Thus the probability that the DQ-score is not the Dodgson Score is less than $\frac{m^3}{n}$. □

Corollary 3.1.14 *Under the impartial culture hypothesis, the probability that the that the Dodgson winner is not the DQ-winner is less than* $\frac{m^4}{n}$.

3.2 Tideman's Rule

Recall that in Section 1.2.4 we defined the Tideman score $\mathrm{Sc}_{\mathbf{T}}(a)$ of an alternative a as

$$\mathrm{Sc}_{\mathbf{T}}(a) = \sum_{b \neq a} \mathrm{adv}(b,a),$$

and that the Tideman winner is the candidate with the lowest score.

Lemma 3.2.1 *Given an even number of agents, the Tideman winner and the DQ-winner will be the same.*

Proof. Since the n is even, we know from Lemma 2.0.12 that all weights in the majority relation W are even. Since the $\mathrm{adv}(a,b) \equiv W(a,b)^+$ it is clear that all advantages will also be even. Since $\mathrm{adv}(a,b)$ will always be even, $\lceil \mathrm{adv}(a,b)/2 \rceil$ will be exactly half $\mathrm{adv}(a,b)$ and so the DQ-score will be exactly half the Tideman score. Hence the DQ-winner and the Tideman winner will be the same. □

Corollary 3.2.2 *Under the impartial culture assumption, for* $2n$ *agents and a fixed number of alternatives* m*, the probability that the Tideman winner is the Dodgson winner converges to 1 exponentially fast as* n *approaches infinity.*

Proof. Obvious, as if there are an even number of agents the Tideman winner equals the DQ-winner (Lemma 3.2.1) and the probability that the DQ-winner is the Dodgson winner converges exponentially fast as $n$approaches infinity (Corollary 3.1.11). □

Corollary 3.2.3 *If the number of agents is even and $D(x, a) \geq F(x, a)$ for every pair of distinct alternatives (x, a), then the Tideman winner is equal to the Dodgson winner.*

Obvious from Lemma 3.2.1 above and Corollary 3.1.7.

Corollary 3.2.4 *If the Tideman winner is not the DQ-winner, all non-zero advantages are odd.*

Proof. As we must have an odd number of agents, from Lemma 2.0.12 all weights in the majority relation W must be odd. Since the $\text{adv}(a, b) \equiv W(a, b)^+$ the advantage $\text{adv}(a, b)$ must be zero or equal to the weight $W(a, b)$. □

Lemma 3.2.5 *There is no profile with three alternatives such that the Tideman winner is not the DQ-winner.*

Proof. The Tideman and Dodgson Quick rules both pick the Condorcet winner when it exists, so if a Condorcet winner exists the Tideman winner and DQ-winner will be the same. It is well known that the absence of a Condorcet winner on three alternatives means that we can rename these alternatives a, b and c so that $\text{adv}(a, b) > 0$, $\text{adv}(b, c) > 0$, and $\text{adv}(c, a) > 0$. These advantages must be odd from the previous corollary. Hence for some $i, j, k \in \mathbb{Z}$ such that $\text{adv}(a, b) = 2i - 1$, $\text{adv}(b, c) = 2j - 1$, and $\text{adv}(c, a) = 2k - 1$. The DQ-Scores and Tideman scores of a, b, c are i, j, k and $2i - 1,\ 2j - 1,\ 2k - 1$ respectively. From here the result is clear, since if $i > j > k$ then $2i - 1 > 2j - 1 > 2k - 1$. □

Lemma 3.2.6 *For a profile with four alternatives there does not exist a pair of alternatives such that a is a DQ-winner but not a Tideman winner, and b is a Tideman winner but not a DQ-winner.*

Proof. By way of contradiction assume that there exist such a, b. Thus there is no Condorcet winner, and so for each alternative c there are one to three alternatives d such that $\text{adv}(c, d) > 0$. Also, since the set of Tideman winners and DQ-winners differ, n must be odd and hence all non-zero advantages must be odd. The relationship between the

Tideman score $\text{Sc}_{\mathbf{T}}(c)$ and the DQ-score $\text{Sc}_{\mathbf{Q}}(c)$ is as follows:

$$\begin{aligned}\text{Sc}_{\mathbf{T}}(c) &= \sum_{d\in\mathcal{A}} \text{adv}(c,d) \\ &= \sum_{d\in\mathcal{A}} \left\lceil \frac{\text{adv}(c,d)}{2} \right\rceil - \#\{c : \text{adv}(c,d) \notin 2\mathbb{Z}\} \\ &= \text{Sc}_{\mathbf{Q}}(c) - (1 \text{ or } 2 \text{ or } 3).\end{aligned}$$

Thus

$$2\text{Sc}_{\mathbf{Q}}(c) - 3 \leq \text{Sc}_{\mathbf{T}}(c) \leq 2\text{Sc}_{\mathbf{Q}}(c) - 1,$$

and so

$$\text{Sc}_{\mathbf{T}}(a) \leq 2\text{Sc}_{\mathbf{Q}}(a) - 1.$$

Given that a is DQ-winner and b is not, we know

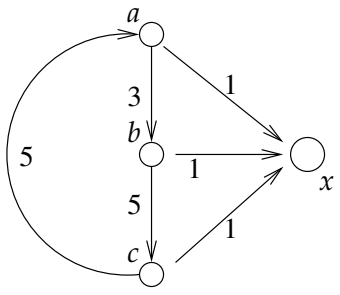
$$\text{Sc}_{\mathbf{Q}}(a) \leq \text{Sc}_{\mathbf{Q}}(b) - 1.$$

Thus by substitution,

$$\begin{aligned}\text{Sc}_{\mathbf{T}}(a) &\leq 2(\text{Sc}_{\mathbf{Q}}(b) - 1) - 1 \\ &= 2\text{Sc}_{\mathbf{Q}}(b) - 3 \\ &\leq \text{Sc}_{\mathbf{T}}(b).\end{aligned}$$

This shows that if b is a Tideman winner, so is a. By contradiction the result must be correct. □

Example 3.2.7 *There do exist profiles with four alternatives where the set of tied Tideman winners differs from the set of tied DQ-winners. By Theorem 2.0.16, we know we may construct a profile with the following advantages.*

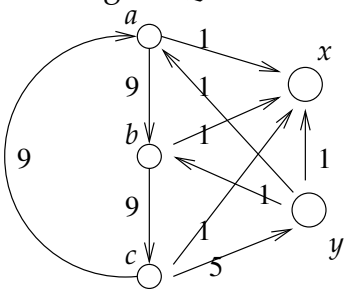


Scores	a	b	c	x
Tideman	5	3	5	3
DQ	3	2	3	3

Here x, b are tied Tideman winners, but b is the sole DQ-winner.

Theorem 3.2.8 *For any $m \geq 5$ there exists a profile with m alternatives and an odd number of agents, where the Tideman winner is not the DQ-winner.*

On page 14 we presented an example of a weighted majority relation where the Tideman winner is not the Dodgson Quick winner. For your convenience, we duplicate it below:



Scores	a	b	c	x	y
Tideman	10	10	9	4	5
DQ	6	6	5	4	3

To extend this example for larger numbers of alternatives, we may add additional alternatives who lose to all of a, b, c, x, y. From Theorem 2.0.16 and Lemma 2.0.12, there exists a profile with an odd number of agents that generates that weighted majority relation.

Theorem 3.2.9 *Under the impartial culture assumption, if we have an even number of agents, the probability that all of the advantages are 0 does not converge to 0 faster than $\mathcal{O}(n^{-\frac{m!}{4}})$.*

Proof. Let $\mathcal{P}$ be a random profile, $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{m!}\}$ be an ordered set containing all $m!$ possible linear orders on m alternatives, and X be a random vector, with elements X_i representing the number of occurrences of $\mathbf{v}_i$ in $\mathcal{P}$. Under the impartial culture assumption, X is distributed according to a multinomial distribution with n trials and $m!$ possible outcomes. Let us group the $m!$ outcomes into $m!/2$ pairs $S_i = \{\mathbf{v}_i, \bar{\mathbf{v}}_i\}$. Denote the number of occurrences of $\mathbf{v}$ as $n(\mathbf{v})$. Let the random variable Y_i^1 be $n(\mathbf{v}_i)$ and Y_i^2 be $n(\bar{\mathbf{v}}_i)$. Let $Y_i = Y_i^1 + Y_i^2$.

From Corollary A.3.7, given $Y_i = y_i$ for all i, each Y_i^1 is independently binomially distributed with $p = 1/2$ and y_i trials. From Corollary A.2.3, if y_i is even then the probability that $Y_i^1 = Y_i^2$ is at least $\frac{1}{2\sqrt{y_i}}$. Combining these results we get

$$\begin{aligned} P(\forall_i Y_i^1 = Y_i^2 | \forall_i Y_i = y_i \in 2\mathbb{Z}) &\geq \prod_i \frac{1}{2\sqrt{y_i}} \\ &\geq \prod_i \frac{1}{2\sqrt{n}} = 2^{-\frac{m!}{2}} n^{-\frac{m!}{4}}. \end{aligned}$$

From Lemma A.3.8, the probability that all X_i are even is at least 2^{-k+1} where $k = m!/2$. Hence

$$P(\forall_i X_{i,1} = X_{i,2}) \geq \left(2^{-\frac{m!}{2}+1}\right)\left(2^{-\frac{m!}{2}} n^{-\frac{m!}{4}}\right) = 2^{1-m!} n^{-\frac{m!}{4}}.$$

If for all i, $X_{i,1} = X_{i,2}$ then for all i, $n(\mathbf{v}_i) = n(\bar{\mathbf{v}}_i)$, i.e. the number of each type of vote is the same as its complement. Thus

$$n_{ba} = \sum_{\mathbf{v}\in\{\mathbf{v}:b\mathbf{v}a\}} n(v) = \sum_{\bar{\mathbf{v}}\in\{\bar{\mathbf{v}}:a\bar{\mathbf{v}}b\}} n(\bar{v}) = \sum_{\mathbf{v}\in\{\mathbf{v}:a\mathbf{v}b\}} n(v) = n_{ab},$$

so $\text{adv}(b, a) = 0$ for all alternatives b and a. □

Corollary 3.2.10 *Under the impartial culture assumption, if we have an even number of agents, the probability that all of the advantages are 0, does not converge to 0 at an exponentially fast rate.*

Lemma 3.2.11 *Under the impartial culture assumption, the probability that the Tideman winner is not the DQ-winner does not converge to 0 faster than $\mathcal{O}(n^{-\frac{m!}{4}})$ as the number of agents n tends to infinity.*

Let $\mathcal{P}$ be our random profile with n agents, for some odd number n. Let $|C|$ be the size of the profile from Theorem 3.2.8. Let us place the first $|C|$ agents from profile $\mathcal{P}$ into sub-profile C and the remainder of the agents into sub-profile D. There is a small but constant probability that C forms the example from Theorem 3.2.8, resulting in the Tideman winner of C differing from its DQ-winner. As n, $|C|$ are odd, $|D|$ is even. Thus from Theorem 3.2.9 the probability that the advantages in D are zero does not converge to 0 faster than $\mathcal{O}(n^{-\frac{m!}{4}})$. If all the advantages in D are zero then adding D to C will not affect the Tideman or DQ-winners. Hence the probability that the Tideman winner is not the DQ-winner does not converge to 0 faster than $\mathcal{O}(n^{-\frac{m!}{4}})$.

Corollary 3.2.12 *Under the impartial culture assumption, the probability that the Tideman winner is not the DQ-winner does not converge to 0 exponentially fast as the number of agents n tends to infinity.*

Theorem 3.2.13 *Under the impartial culture assumption, the probability that the Tideman winner is not the Dodgson winner does not converge to 0 faster than $\mathcal{O}(n^{-\frac{m!}{4}})$ as the number of agents n tends to infinity.*

Proof. From Corollary 3.1.11 the DQ-winner converges to the Dodgson winner exponentially fast. However, the Tideman winner does not converge faster than $\mathcal{O}(n^{-\frac{m!}{4}})$ to the DQ-winner, and hence also does not converge faster than $\mathcal{O}(n^{-\frac{m!}{4}})$ to the Dodgson winner. □

Lemma 3.2.14 *Let S be a subset of $\mathcal{A}$. Let $a_1a_2\ldots a_{|S|}$ and $b_1b_2\ldots b_{|S|}$ be two linear orderings of S. Then the number of linear orders $\mathbf{v}$ in $\mathcal{L}(\mathcal{A})$ where $a_1\mathbf{v}a_2, a_2\mathbf{v}a_3, \ldots, a_{|S|-1}\mathbf{v}a_{|S|}$ is equal to the number of linear orders $\mathbf{v}$ where $b_1\mathbf{v}b_2, b_2\mathbf{v}b_3, \ldots, b_{|S|-1}\mathbf{v}b_{|S|}$, i.e.*

$$\#\{\mathbf{v} : \forall_{i\in[2,|S|]}a_{i-1}\mathbf{v}a_i\} \quad = \quad \#\{\mathbf{v} : \forall_{i\in[2,|S|]}b_{i-1}\mathbf{v}b_i\}.$$

Proof. Let the function f be defined with domain and range $\mathcal{L}(\mathcal{A})$ as follows. If a_i is ranked in position j in $\mathbf{v}$ then b_i is ranked in position j in $f(\mathbf{v})$. If $x \notin S$ is ranked in some position j in v then x is still ranked in position j in $f(\mathbf{v})$.

Clearly, if b_i is ranked in position j in $\mathbf{v}$ then a_i is ranked in position j in all members of $f^{-1}(\mathbf{v})$. If $x \notin S$ is ranked in some position j in $\mathbf{v}$ then x is still ranked in position j in all members of $f^{-1}(\mathbf{v})$. Hence $f^{-1}(\mathbf{v})$ is a function.

The function f provides a bijection between the sets $\{\mathbf{v} : \forall_{i\in[2,|S|]}a_{i-1}\mathbf{v}a_i\}$, $\{\mathbf{v} : \forall_{i\in[2,|S|]}b_{i-1}\mathbf{v}b_i\}$. Hence the result. □

Corollary 3.2.15 *Let S be a subset of $\mathcal{A}$. The number of linear orders $\mathbf{v}$ where $a_{i-1}\mathbf{v}a_i$ for all $i = 2, 3, \ldots, |S|$ is equal to $n!/|S|!$.*

Definition 3.2.16 *We define the adjacency matrix M, of a linear order $\mathbf{v}$, as follows:*

$$M_{ij} \quad = \quad \begin{cases} 1 & \textit{if} \quad i\mathbf{v}j \\ -1 & \textit{if} \quad j\mathbf{v}i \\ 0 & \textit{if} \quad i = j \end{cases} .$$

Lemma 3.2.17 *Suppose that each linear order is equally likely, then M is an m^2-dimensional random variable satisfying the following equations for all $i, j, r, s \in \mathcal{A}$.*

$$\begin{aligned} E[M] &= 0 \\ = cov(M_{ij}, M_{rs}) &= E[M_{ij}M_{rs}] \\ &= \begin{cases} 1 & \textit{if} \quad i = r \neq j = s \\ 1/3 & \textit{if} \quad i = r, \textit{ but } i, j, s \textit{ distinct } \vee \ j = s, \textit{ others distinct} \\ -1/3 & \textit{if} \quad i = s, \textit{ others distinct } \vee \ j = r, \textit{ others distinct} \\ 0 & \textit{if} \quad i, j, r, s \textit{ distinct } \vee \ i = j = r = s \\ -1 & \textit{if} \quad i = s \neq j = r \end{cases} . \end{aligned}$$

Proof. From Lemma 3.2.14,

$$E[M_{ij}] \quad = \quad \frac{(1) + (-1)}{2} = 0.$$

It is well known that $\text{cov}(X,Y) = E[XY] - E[X]E[Y]$ (see e.g. Walpole and Myers 1993; p97). Thus $\text{cov}(M_{ij}, M_{rs}) = E[M_{ij}M_{rs}] - (0)(0) = E[M_{ij}M_{rs}]$. Note that for all $i \neq j$ we know that $M_{ii}M_{ii} = 0$, $M_{ij}M_{ij} = 1$, and $M_{ij}M_{ji} = -1$. If $i = r$ and i, j, s are all distinct then the sign of $M_{ij}M_{is}$ for each permutation of i, j and s is as shown below.

	i	i	j	j	s	s
	j	s	i	s	i	j
	s	j	s	i	j	i
M_{ij}	$+$	$+$	$-$	$-$	$+$	$-$
M_{is}	$+$	$+$	$+$	$-$	$-$	$-$
$M_{ij}M_{is}$	$+$	$+$	$-$	$+$	$-$	$+$

Thus from Lemma 3.2.14,

$$E[M_{ij}M_{rs}] \;=\; \frac{+1+1-1+1-1+1}{6} = \frac{1}{3}.$$

If i, j, r, s are all distinct then there are six linear orders $\mathbf{v}$ where $i\mathbf{v}j \wedge r\mathbf{v}s$, six linear orders $\mathbf{v}$ where $i\mathbf{v}j \wedge s\mathbf{v}r$, six linear orders $\mathbf{v}$ where $j\mathbf{v}i \wedge r\mathbf{v}s$, and six linear orders $\mathbf{v}$ where $j\mathbf{v}i \wedge s\mathbf{v}r$. Hence from Lemma 3.2.14,

$$E[M_{ij}M_{rs}] \;= \tfrac{6(1)(1)+6(1)(-1)+6(-1)(1)+6(-1)(-1)}{24} = 0 \quad .$$

We may prove the other cases for $\text{cov}(M_{ij}, M_{rs})$ in much the same way. □

Corollary 3.2.18 *As var$(X) =$ cov(X, X) we also have,*

$$\textit{var}(M_{ij}) \;=\; \begin{cases} 1 & if\ i \neq j \\ 0 & if\ i = j \end{cases} \quad .$$

Example 1 *For example, for $m = 4$ the covariances with M_{12} are shown in the matrix*

$$\mathfrak{L} \;=\; \begin{bmatrix} 0 & 1 & 1/3 & 1/3 \\ -1 & 0 & -1/3 & -1/3 \\ -1/3 & 1/3 & 0 & 0 \\ -1/3 & 1/3 & 0 & 0 \end{bmatrix},$$

where $\mathfrak{L}_{ij} =$ cov(M_{ij}, M_{12}).

Definition 3.2.19 *Define Y to be a collection of random normal variables indexed by i, j for*

$1 \leq i < j \leq m$ *each with mean of 0, and covariance matrix* Ω*, where*

$$\Omega_{ij,rs} \quad = \quad cov(Y_{ij}, Y_{rs}) = cov(M_{ij}, M_{rs}),$$

We may use the fact that $i < j$*,* $r < s$ *implies* $i \neq j$*,* $r \neq s$*,* $(s = i \Rightarrow r \neq j)$ *and* $(r = j \Rightarrow s \neq i)$ *to simplify the definition of* Ω *as shown below:*

$$\Omega_{ij,rs} \quad = \quad \begin{cases} 1 & \textit{if} \quad (r,s) = (i,j) \\ 1/3 & \textit{if} \quad r = i, s \neq j \textit{ or } s = j, r \neq i \\ -1/3 & \textit{if} \quad s = i \textit{ or } r = j \\ 0 & \textit{if} \quad i,j,r,s \textit{ are all distinct} \end{cases},$$

i.e. if i, j, r, s *are all distinct then*

$$\begin{aligned} \Omega_{ij,ij} &= 1, \\ \Omega_{ij,rj} = \Omega_{ij,is} &= 1/3, \\ \Omega_{ij,ri} = \Omega_{ij,js} &= -1/3, \\ \Omega_{ij,rs} &= 0. \end{aligned}$$

Theorem 3.2.20 (Multivariate Central Limit Theorem) *Let the* k*-dimensional vectors* $X^1, X^2, \ldots$ *be independently and identically distributed; each with mean of* $\mathbf{b}$*, and covariance matrix of* Ω*. Then*

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} X^i - b \quad \xrightarrow{D} \quad N(0, \Omega),$$

as $n \to \infty$*. (See e.g. Anderson 1984, p. 81)*

Lemma 3.2.21 *As* n *approaches infinity,* $\sum_{i=1}^{n} M_i/\sqrt{n}$ *converges in distribution to*

$$\begin{bmatrix} 0 & Y_{12} & Y_{13} & \cdots & Y_{1m} \\ -Y_{12} & 0 & Y_{23} & \cdots & Y_{2m} \\ -Y_{13} & -Y_{23} & 0 & \cdots & Y_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -Y_{1m} & -Y_{2m} & -Y_{3m} & \cdots & 0 \end{bmatrix},$$

where M_i *is the adjacency matrix for the* i^{th} *linear order in a profile* P*, and recall that* Y *is be a collection of random normal variables indexed by* i, j *for* $1 \leq i < j \leq m$ *each with mean of 0, and*

covariance matrix Ω, where

$$\Omega_{ij,rs} \quad = \quad cov(Y_{ij}, Y_{rs}) = cov(M_{ij}, M_{rs}).$$

Proof. As $M_1, M_2, \ldots, M_n$ are independent identically-distributed (i.i.d.) random variables, we know from the multivariate central limit theorem (see e.g. Anderson, 1984; p81) that $\sum_{i=1}^{n} M_i/\sqrt{n}$ converges in distribution to the multivariate normal distribution with the same mean and covariance as M_1. As $M^T = -M$ and $M_{ii} = 0$, we have the result. □

Lemma 3.2.22 *Ω is non-singular.*

Proof. Consider Ω^2:

$$(\Omega^2)_{ij,kl} \quad = \quad \sum_{1 \le r < s \le m} \Gamma_{ij,kl}(r,s),$$

where $\Gamma_{ij,kl}(r,s) = \Omega_{ij,rs}\Omega_{rs,kl}$.

For i, j, r, s distinct then

$$\begin{aligned}
\Gamma_{ij,ij}(i,j) = \Omega_{ij,ij}\Omega_{ij,ij} &= (1)(1) = 1, \\
\Gamma_{ij,ij}(r,j) = \Omega_{ij,rj}\Omega_{rj,ij} &= (1/3)(1/3) = 1/9, \\
\Gamma_{ij,ij}(i,s) = \Omega_{ij,is}\Omega_{is,ij} &= (1/3)(1/3) = 1/9, \\
\Gamma_{ij,ij}(r,i) = \Omega_{ij,ri}\Omega_{ri,ij} &= (-1/3)(-1/3) = 1/9, \\
\Gamma_{ij,ij}(j,s) = \Omega_{ij,js}\Omega_{js,ij} &= (-1/3)(-1/3) = 1/9, \\
\Gamma_{ij,ij}(r,s) = \Omega_{ij,rs}\Omega_{ij,rs} &= 0.
\end{aligned}$$

Case $(i,j) = (k,l)$:

If $(i,j) = (k,l)$ then

$$\begin{aligned}
\Gamma_{ij,ij}(r,s) &= \Omega_{ij,rs}\Omega_{rs,ij} \\
&= \begin{cases}
(1)^2 & \text{if} \quad (r,s) = (i,j) \\
(1/3)^2 & \text{if} \quad r = i, s \neq j \text{ or } s = j, r \neq i \\
(-1/3)^2 & \text{if} \quad s = i, (r \neq j) \text{ or } r = j, (s \neq i) \\
0 & \text{if} \quad i, j, r, s \text{ are all distinct}
\end{cases},
\end{aligned}$$

Recall that $r < s$, $i < j$ and $r, s \in [1, m]$. Let us consider for how many values of (r,s) each of the above cases occur:

- $(r,s) = (i,j)$: This occurs for exactly one value of (r,s).

- $r = i, s \neq j$: Combining the fact that $r < s$ and $r = i$ we get $i < s$. Thus $s \in (i,j) \cup (j,m]$, and there are $(j-i-1)+(m-j) = (m-i-1)$ possible values of s. As there is only one possible value of r this means that there are also $(m-i-1)$ possible values of (r,s).

- $s = j, r \neq i$: Combining the fact that $r < s$ and $s = j$ we get $r < j$. Thus $r \in [1,i) \cup (i,j)$, and there are $(i-1)+(j-i-1) = (j-2)$ possible values of (r,s).

- $s = i$: Here we want $r \neq j$, however $r < s = i < j$, so explicitly stating $r \neq j$ is redundant. Combining the fact that $r < s$ and $s = i$ we get $r < i$. Hence $r \in [1,i]$ and there are $i-1$ possible values for (r,s).

- $r = j$: Here we want $s \neq i$, however $i < j = r < s$, so explicitly stating that $r \neq j$ is redundant. From here on we will not state redundant inequalities. Combining the fact that $r < s$ and $r = j$ we get $j < s$. Hence $s \in (j,m]$ and there are $m-j$ possible values for (r,s).

hence,

$$\begin{aligned}
\sum_{1 \leq r < s \leq m} \Gamma_{ij,ij}(r,s) &= (1)(1) + ((m-i-1)+(j-2))\left(\frac{1}{3}\right)^2 + ((i-1)+(m-j))\left(\frac{-1}{3}\right)^2 \\
&= 1 + (m+j-i-3)\left(\frac{1}{9}\right) + (m+i-j-1)\left(\frac{1}{9}\right) \\
&= (9 + (m+j-i-3) + (m+i-j-1))/9 \\
&= \frac{2m+5}{9}.
\end{aligned}$$

Case $i = k,\ j \neq l$: then,

$$\begin{aligned}
\Gamma_{ij,il}(r,s) &= \Omega_{ij,rs}\Omega_{rs,il} = \\
&= \begin{cases}
1\Omega_{rs,il} & \text{if} \quad (r,s) = (i,j) \\
{}^1\!/_3\Omega_{rs,il} & \text{if} \quad r = i, s \neq j \text{ or } s = j, r \neq i \\
{}^{-1}\!/_3\Omega_{rs,il} & \text{if} \quad s = i \text{ or } r = j \\
0 & \text{if} \quad i,j,r,s \text{ are all distinct}
\end{cases},
\end{aligned}$$

more precisely,

$$\Gamma_{ij,il}(r,s) \;=\; \begin{cases} (1)(1/3) & = & 1/3 & \text{if} & (i,j)=(r,s) \\ (1/3)(1) & = & 1/3 & \text{if} & r=i, s=l\neq j \\ (1/3)(1/3) & = & 1/9 & \text{if} & r=i, s\neq j, s=\neq l \\ (1/3)(0) & = & 0 & \text{if} & s=j\neq l, r\neq i \\ (-1/3)(-1/3) & = & 1/9 & \text{if} & s=i \\ (-1/3)(1/3) & = & -1/9 & \text{if} & r=j, s=l \\ (-1/3)(0) & = & 0 & \text{if} & r=j, s\neq l \\ 0 & = & 0 & \text{if} & i,j,r,s \text{ are all distinct} \end{cases},$$

hence,

$$\begin{aligned} \sum_{1\leq r<s\leq m}\Gamma(r,s) &= \frac{1}{3}+\frac{1}{3}+\sum_{1\leq r<s\leq m, r=i, s\neq j, s=\neq l}\frac{1}{9}+\sum_{1\leq r<s\leq m, s=i}\frac{1}{9}-\frac{1}{9} \\ &= \frac{1}{3}+\frac{1}{3}+\sum_{i<s\leq m}\frac{1}{9}-\frac{2}{9}+\sum_{1\leq r<i}\frac{1}{9}-\frac{1}{9} \\ &= \frac{1}{3}+(m-i)\frac{1}{9}+(i-1)\frac{1}{9} \\ &= \frac{m+2}{9}. \end{aligned}$$

Similarly for $i\neq k,\ j=l$, we may show $(\Omega^2)_{ij,kj}=\frac{m+2}{9}$. If $j=k$ then

$$\begin{aligned} (\Omega^2)_{ij,kl} &= -\frac{1}{3}-\frac{1}{3}+\frac{1}{9}-\sum_{1\leq r<i, r\neq i}\frac{1}{9}-\sum_{j<s\leq m, s\neq l}\frac{1}{9}, \\ &= -\frac{m+2}{9}, \end{aligned}$$

similarly for $l=i$. If i,j,k,l are all distinct, $(\Omega^2)_{ij,kl}$ equals 0. Consequently

$$\Omega^2 \;=\; \left(\frac{m+2}{3}\right)\Omega-\left(\frac{m+1}{9}\right)I$$

Now, when a matrix Ω satisfies $\Omega^2=\alpha\Omega+\beta I$ with $\beta\neq 0$ it has an inverse as shown below,

$$\Omega\left(\frac{\Omega-\alpha}{\beta}\right) \;=\; I,$$

and hence Ω is not singular. □

Theorem 3.2.23 *If* $f(x) \leq g(x) \leq h(x)$ *for all* x*, and* $\lim_{x\to x_0} f(x) = \lim_{x\to x_0} h(x)$ *then* $\lim_{x\to x_0} f(x) = \lim_{x\to x_0} g(x)$.

This famous theorem is often called the **sandwich theorem**. It is also known as the squeeze theorem or pinch theorem.

Theorem 3.2.24 *The probability that the Tideman winner and Dodgson Quick winner coincide converges asymptotically to* 1 *as* $n \to \infty$.

Proof. The Tideman winner is the alternative $a \in \mathcal{A}$ with the minimal value of

$$G(a) \;=\; \sum_{b\in A} \text{adv}(b,a),$$

while the DQ-winner has minimal value of

$$F(a) \;=\; \sum_{b\in A} \left\lceil \frac{\text{adv}(b,a)}{2} \right\rceil.$$

Let a_T be the Tideman winner and a_Q be the DQ-winner. Note that $G - m \leq 2F \leq G$. If for some b we have $G(b) - m > G(a_T)$, then $2F(b) \geq G(b) - m > G(a_T) \geq 2F(a_T)$ and so b is not a DQ-winner. Hence, if $G(b) - m > G(a_T)$ for all alternatives b distinct from a, then a_T is also the DQ-winner a_Q. Thus,

$$\begin{aligned} P(a_T \neq a_Q) &\leq P(\exists_{a,b} |G(a) - G(b)| \leq 2m \wedge a \neq b) \\ &= P\left(\exists_{a,b} \left|\frac{G(a) - G(b)}{\sqrt{n}}\right| \leq \frac{2m}{\sqrt{n}} \wedge a \neq b\right), \end{aligned}$$

thus for any $\epsilon > 0$ and sufficiently large n, we have

$$P(a_T \neq a_Q) \;\leq\; P\left(\exists_{a,b} \left|\frac{G(a) - G(b)}{\sqrt{n}}\right| \leq \epsilon \wedge a \neq b\right)$$

We will show that the right-hand side of the inequality above converges to 0 as n tends to ∞. All probabilities are non-negative so $0 \leq P(a_T \neq a_Q)$. From these facts and the sandwich theorem it follows that $\lim_{n\to\infty} P(a_T \neq a_Q) = 0$. We let,

$$G_j \;=\; \sum_{i<j} (Y_{ij})^+ + \sum_{k>j} (-Y_{jk})^+,$$

and so,

$$\lim_{n\to\infty} P\left(\exists_{a,b} \left|\frac{G(a) - G(b)}{\sqrt{n}}\right| \leq \epsilon \wedge a \neq b\right) \;=\; P\left(\exists_{i,j} |G_i - G_j| \leq \epsilon \wedge i \neq j\right)$$

Since $\epsilon > 0$ is arbitrary, the infimum is $\epsilon = 0$. Thus,

$$\lim_{n\to\infty} P(a_T \neq a_Q) \quad \leq \quad P(\exists_{i,j} G_i = G_j \wedge i \neq j).$$

For fixed $i < j$ we have

$$G_i - G_j \quad = \quad -Y_{ij} + \sum_{k<i} (-Y_{ki})^+ + \sum_{k>i, k\neq i} (Y_{ik})^+ - \sum_{k<j, k\neq i} (Y_{kj})^+ - \sum_{k>j} (-Y_{jk})^+$$

Define v so that $G_i - G_j = -Y_{ij} + v$. Then $P(G_i = G_j) = P(Y_{ij} = v) = E[P(Y_{ij} = v|v)]$. Since Y has a multivariate normal distribution with a non-singular covariance matrix Ω, we know from Lemma A.4.1 that $P(Y_{ij} = v|v) = 0$. That is, $P(G_i = G_j) = 0$ for any i, j where $i \neq j$. Hence $P(\exists_{i,j} G_i = G_j \wedge i \neq j) = 0$. As discussed previously in this proof, we may now use the sandwich theorem to prove that $\lim_{n\to\infty} P(a_T \neq a_Q) = 0$. □

3.3 Numerical Results

Although we have proven theorems on the rate of convergence, tables of figures can help illustrate the nature of the convergence. In this section we present tables demonstrating the rate fast convergence of our Dodgson Quick rule in comparison to the Tideman rule. We also study the asymptotic limit of the probability that the Simpson winner is the Dodgson winner as we increase the number of agents.

As shown below the convergence of the Tideman winner to the Dodgson Winner occurs much slower than the exponential convergence of the DQ-Winner.

Table 3.1: Number of Occurrences per 1000 Elections with 5 Alternatives that the Dodgson Winner was Not Chosen

Voters	3	5	7	9	15	17	25	85	257	1025
DQ	1.5	1.9	1.35	0.55	0.05	0.1	0	0	0	0
Tideman	1.5	2.3	2.7	3.95	6.05	6.85	7.95	8.2	5.9	2.95
Simpson	57.6	65.7	62.2	57.8	48.3	46.6	41.9	30.2	23.4	21.6

Table 3.1 reports the probability that these rules pick the same winner after we break ties according to the preferences of the first agent. It was generated by averaging 10,000 simulations, so the figures are only approximate, however the trends are clearly significant.

Since the DQ-Winner and Tideman Winner seem to closely approximate Dodgson's Rule we may wish to also look at the probability that these rules pick the same set of tied winners, presented in Table 3.2.

Table 3.2: Number of Occurrences per 1000 Elections with 5 Alternatives that the Set of Tied Dodgson Winners was Not Chosen

Voters	3	5	7	9	15	17	25	33	85	257	1025
DQ-Winners	4.31	4.41	3.21	1.94	0.27	0.08	0.04	0	0	0	0
Tideman	4.31	5.57	7.31	8.43	12.73	13.15	15.46	16.35	15.18	10.2	5.4

From Theorem 3.1.8 we know that the Dodgson Quick winner converges to the Dodgson winner at an exponentially fast rate. These figures confirm that for a large number of agents the simple Dodgson Quick rule provides a very good approximation to the Dodgson rule.

Given that the Dodgson Quick rule is such a good approximation for the Dodgson rule, we may want to compare the other rules to Dodgson Quick in place of the Dodgson rule. This allows us to easily examine the behaviour of the Tideman rule and the Simpson rule with over 100,000 agents. Furthermore, we may use the test developed in Lemma 3.1.6 (i.e. that criteria [T2] from 12 holds) to check that all the Dodgson Quick scores match the Dodgson scores and so Dodgson Quick is picking the correct winner. We have done this and included the output in Appendix B.2.3. We found that as we increased the number of alternatives, the frequency that the quick test was able to verify that the DQ-winner was the Dodgson winner declined. However, even with 8 alternatives, the test was able to demonstrate that the DQ-winner was the Dodgson winner in all 100,000 simulations.

Another question is how well does Dodgson Quick approximate the Dodgson rule with other numbers of alternatives, or if the number of agents is not large in comparison to the number of agents. From 3.3, it appears that our approximation is still reasonably accurate under these conditions. This table was generated by averaging 10,000 simulations, and splitting ties according to the preferences of the first agent.

To give meaning to these figures, let us compare them with the figures in Tables 3.4 and 3.5. We see that even where the number of agents is not very large, the Dodgson Quick rule seems to do a slightly better job of approximating the Dodgson rule than Tideman's approximation. We also see that Simpson's rule does a particularly poor job of approximating frequency of the Tideman approximation picking the Dodgson winner when the number of candidates is large.

As an aside, it would appear that Simpson's rule is not a very accurate approximation of Dodgson's Rule. The probability that the Simpson winner does not equal the Dodgson

Table 3.3: Frequency that the DQ-Winner is the Dodgson Winner

# Alternatives \ # Agents	3	5	7	9	15	25	85
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	0.9984	0.9976	0.9980	0.9992	0.9999	1.0000	1.0000
7	0.9902	0.9875	0.9879	0.9933	0.9980	0.9995	1.0000
9	0.9792	0.9742	0.9778	0.9837	0.9924	0.9978	0.9999
15	0.9468	0.9327	0.9338	0.9412	0.9571	0.9743	0.9988
25	0.8997	0.8718	0.8661	0.8731	0.8971	0.9265	0.9840

Table 3.4: Frequency that the Tideman Winner is the Dodgson winner

# Alternatives \ # Agents	3	5	7	9	15	25	85
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	0.9984	0.9974	0.9961	0.9972	0.9936	0.9917	0.9930
7	0.9902	0.9864	0.9852	0.9868	0.9845	0.9805	0.9847
9	0.9792	0.9730	0.9724	0.9731	0.9718	0.9760	0.9815
15	0.9468	0.9292	0.9263	0.9273	0.9379	0.9485	0.9649
25	0.8997	0.8691	0.8620	0.8625	0.8833	0.9113	0.9534

Table 3.5: Frequency that the Simpson Winner is the Dodgson Winner

# Alternatives \ # Agents	3	5	7	9	15	25	85
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	0.9433	0.9307	0.9339	0.9398	0.9493	0.9575	0.9714
7	0.8734	0.8627	0.8689	0.8786	0.9018	0.9153	0.9404
9	0.8256	0.8153	0.8167	0.8251	0.8562	0.8808	0.9124
25	0.5895	0.5772	0.6147	0.6322	0.7114	0.7529	0.7957

winner is much greater than for Tideman or DQ. We may ask, does the Simpson rule eventually converge to the Dodgson rule as we increase the number of voters, and if not, how close does it get?

3.3.1 Asymptotic Behaviour of Simpson's Rule

From Lemma 3.1.4 and Theorem 3.2.24 we know that the Dodgson winner, Dodgson Quick winner, and Tideman winner all asymptotically converge as we increase the number of agents. Hence we may compute the asymptotic probability that the Simpson winner is equal to the Dodgson winner, by computing the asymptotic probability that the Simpson winner equals the Tideman winner.

From Lemma 3.2.21 we know that the matrix of advantages converges to a multivariate normal distribution as we increase the number of agents. If we had a multivariate normal random vector generator, we could use this model to perform simulations and count in how many simulations the Simpson winner is equal to the Tideman winner. We decided to use a slightly different model so that we could use a univariate normal random number generator.

Table 3.6: The Limit of the Number of Occurrences per 1000 Elections that the Simpson Winner is Not the Dodgson Winner, as $n \to \infty$

#Alternatives	3	4	5	6	7	8
#(DO $\neq$ SI) per 1000	0	6.81	17.18	27	39.33	50.18

Let $\mathcal{P}$ be our profile with m alternatives $\mathcal{A}$ and n agents. Say a and b are two distinct alternatives in the set $\mathcal{A}$. Say $V = (v_1, v_2, \ldots, v_{m!/2})$ is an ordered set of possible linear orders where a is ranked above b. Note that $\{v_1, \bar{v}_1, v_2, \bar{v}_2, \ldots, v_{m!/2}, \bar{v}_{m!/2}\}$ is the set $\mathcal{L}(\mathcal{A})$ of all possible linear orders of $\mathcal{A}$. We define a random vector X on a randomly selected random linear order $\mathbf{v}$ such that

$$X_i = \begin{cases} 1 & \text{if } \mathbf{v} = v_i \\ -1 & \text{if } \mathbf{v} = \bar{v}_i \\ 0 & \text{otherwise} \end{cases}$$

We likewise define an ordered set $\mathcal{X} = \{X^1, X^2, \ldots, X^n\}$, where X^i is the random vector defined on the i^{th} linear order in $\mathcal{P}$. The random vectors are independently identically distributed (i.i.d.) with means of 0, and covariance matrix $\Omega = rI$ where r is some real

number greater than 0 and I is the identity matrix. By the multivariate central limit theorem (3.2.20), we know that $Y = \sum_{i=1}^{n} X^i/\sqrt{n}$ converges to an $N(0, rI)$ multivariate normal distribution. Hence we may easily model $Y_1, Y_2, \ldots, Y_n$ as i.i.d. univariate normally distributed variables.

Using this model we performed 100,000 simulations and generate Table 3.6.

Note that as the number of agents approaches infinity, the probability of a tie approaches 0, and so tie breaking is irrelevant in this table. In Table 3.6, we see that even with an infinite number of voters, the Simpson rule is not especially close to the Dodgson rule.

Chapter 4

Formulation of Dodgson's Rule as an Integer Linear Program

Bartholdi et al. (1989) showed that it is possible to find the Dodgson score of an alternative by using an Integer Linear Program (ILP). In this section we will define term ILP, and show how we may find the Dodgson score with an ILP that has less variables than the ILP suggested by Bartholdi et al. The number of variables is important, as the number of variables is the primary factor in the amount of time required to solve the ILP problem (see e.g. Lenstra, Jr. 1983).

Note 4.0.1 *Recall that, on Page xii, we defined vector inequalities so that for any pair of vectors, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, the statement $\mathbf{x} \leq \mathbf{y}$ means that $x_i \leq y_i$ for all i in $\{1, 2, \ldots, n\}$, and $\mathbf{x} \geq 0$ means that $x_i \geq 0$ for all i in $\{1, 2, \ldots, n\}$.*

Definition 4.0.2 *A **Linear Program (LP)** is an ordered set $(M, N, A, \mathbf{b}, \mathbf{c})$ where M and N are positive integers; A is an $M \times N$ matrix; $\mathbf{b}$ and $\mathbf{c}$ are M-dimensional vectors. The **optimal value** of a linear program is the maximum possible value of $\mathbf{c}^T\mathbf{x}$ where $\mathbf{x}$ is an arbitrary M-dimensional non-negative vector (i.e. $\mathbf{x} \geq 0$) and $A\mathbf{x} \leq \mathbf{b}$.*

Note 4.0.3 *A **Linear Program (LP)** may also be defined as an ordered set (V, C, f), where V is a finite set of real valued variables, C is a finite set of linear constraints on those variables, and f is a linear function on those variables. The **optimal value** of a linear program is then the maximum possible value of f subject to the constraints in C.*

Definition 4.0.4 *An **Integer Linear Program (ILP)** is the same as an LP, but with an additional constraint that all variables must be integers.*

Lemma 4.0.5 *Let $\mathcal{P}$ be a profile. Suppose that $\mathfrak{S}$ is a set of swaps that makes an alternative d a Condorcet winner. Then there exists a set of swaps $\mathfrak{T}$ such that $\mathfrak{T}$ also makes d a Condorcet winner, $\mathfrak{T}$ contains only swaps of d over other alternatives and $|\mathfrak{T}| = |\mathfrak{S}|$.*

Proof. In every linear order v in the profile $\mathcal{P}$, element d has a deficit function $h_d(a)$ the number of swaps required to move d above a (i.e. one greater than the number of alternatives ranked between a and d if a is ranked above d, and 0 otherwise). Now suppose that k swaps in $\mathfrak{S}$ reduce to zero the deficits $h_d(b)$ for $b \in B \subseteq \mathcal{A}$, where B is the set of alternatives whose deficits are reduced to zero by $\mathfrak{S}$. Then any other set of swaps that reduce to zero the same set of deficits will also make d a Condorcet winner.

Now a swap of d upwards reduces all deficits by 1. Any other swap cannot reduce a deficit by more than one. Hence swapping d upwards k times will reduce to zero all deficits $h_d(b)$ for $b \in B$ (and maybe some additional deficits, but this will only further benefit d, and thus will not prevent d being a Condorcet winner). □

From Lemma 4.0.5 above, we may assume without loss of generality, that to make some fixed alternative d a Condorcet winner using a minimal number of swaps, we may only use swaps of d over other alternatives. Thus, given some fixed profile $\mathcal{P}$, we may describe our set of swaps by the number of times that d is swapped above other alternatives in each linear order in $\mathcal{P}$. That is, we may uniquely determine our set of swaps based upon a vector $\mathfrak{S} = (s_1, s_2, \ldots, s_n)$ where s_i is the number of times that d is swapped over some other alternative in the i^{th} linear order in $\mathcal{P}$.

4.1 Discussion of Variables

Bartholdi et al. (1989) developed an Integer Linear Program (ILP) problem with no more than $m!m$ variables. In this section we will informally discuss how this was achieved. We will then informally discuss how we will form an ILP with less than $(m-1)!e$ variables, where $e = 2.71\cdots$ is the exponential constant.

Say that $\mathcal{P}$ is a fixed profile, d is one of the alternatives in $\mathcal{P}$, and we are interested in the Dodgson score of d.

As discussed in the previous section, when attempting to make d a Condorcet winner with a minimal number of swaps, we need only consider the vector $\mathfrak{S} = (s_1, s_2, \ldots, s_n)$, where for each i in $\{1, 2, \ldots, n\}$ the value s_i is the number of times d is swapped up the preference list of the i^{th} agent. It would be tempting to use $s_1, s_2, \ldots, s_n$ as the variables of the Integer Linear Programming (ILP) problem. These variables are integer, unfortunately their effect is not linear. Recall that s_i is the number of times that d is swapped up the preference list of the i^{th} agent. The second time that d is swapped up this preference list d will be swapped over a different agent, and so the effect of increasing s_i is not linear.

Let us consider the 0–1 (binary) $m \times n$ matrix $A^{\mathcal{P}}$ where $A^{\mathcal{P}}_{ij}$ is 0 if and only if d is one of the top i preferences of the j^{th} agent. We have mn variables in this ILP. Below is an

example of a profile $\mathcal{P}$ and the corresponding matrix $A^{\mathcal{P}}$.

Profile $\mathcal{P}$

a	a	b	b	d
c	c	d	a	a
d	d	a	c	b
b	b	c	d	c

$\rightarrow$

Matrix $A^{\mathcal{P}}$

1	1	1	1	0
1	1	0	1	0
0	0	0	1	0
0	0	0	0	0

Let $\mathcal{Q}$ be the profile $\mathcal{P}$ after our swaps $\mathfrak{S}$ are applied. We will define the matrix $A^{\mathcal{Q}}$ similarly to $A^{\mathcal{P}}$. Then the matrix $A^{\mathcal{Q}}$ will be an $m \times n$ matrix of binary variables. We may use these mn variables in the matrix $A^{\mathcal{Q}}$ as the variables in our ILP. Clearly we cannot choose $A^{\mathcal{Q}}$ to be any $m \times n$ binary matrix; the matrix $A^{\mathcal{Q}}$ is subject to two further constraints for each i, j. Firstly, $A^{\mathcal{Q}}_{i(j-1)} \geq A^{\mathcal{Q}}_{ij}$, because if d is ranked is one the top $j-1$ preferences of the i^{th} agent then it is clearly also one of the top j preferences of that agent. Secondly, $A^{\mathcal{Q}}_{ij} \leq A^{\mathcal{P}}_{ij}$ because we only swap d up preference lists, not down. We note that each time $\mathfrak{S}$ swaps d over a neighbouring alternative in a linear order of $\mathcal{Q}$, this changes single element of $A^{\mathcal{Q}}$ from one to zero. Thus $\sum_{ij} A^{\mathcal{P}}_{ij} - A^{\mathcal{Q}}_{ij}$ is the number of swaps used. Thus, if we add constraints to ensure that d is a Condorcet winner in $\mathcal{Q}$, then the minimum value of $\sum_{ij} A^{\mathcal{P}}_{ij} - A^{\mathcal{Q}}_{ij}$ will be the Dodgson score of d.

We may reduce the number of variables by throwing away some information from the profile $\mathcal{P}$. The Dodgson rule only requires the information contained in the voting situation $\tilde{\mathcal{P}}$. Let $\tilde{n}$ be the number of unique linear orders in $\tilde{\mathcal{P}}$. We may represent a voting situation using an ordered set $S = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\tilde{n}}\}$ of unique linear orders, and an ordered set $F = \{f_1, f_2, \ldots, f_{\tilde{n}}\}$ where f_i is the number of agents who chose the linear order $\mathbf{v}_i$. Let us consider the integer $m \times \tilde{n}$ matrix $A^{\tilde{\mathcal{P}}}$ where $A^{\tilde{\mathcal{P}}}_{ij}$ is 0 if d is one of the top i preferences in $\mathbf{v}_j$, and is f_j otherwise.

Voting Situation $\tilde{\mathcal{P}}$

2	1	1	1
a	b	b	d
c	d	a	a
d	a	c	b
b	c	d	c

$\rightarrow$

Matrix $A^{\tilde{\mathcal{P}}}$

2	1	1	0
2	0	1	0
0	0	1	0
0	0	0	0

We now define $A^{\tilde{\mathcal{Q}}}$ so that $A^{\tilde{\mathcal{Q}}}_{ij}$ is the number of agents who chose the linear order $\mathbf{v}_j$ as their preference list in $\mathcal{P}$ and also do not rank d in their top i^{th} preferences after the swaps in swaps in $\mathfrak{S}$ are applied. Thus $A^{\tilde{\mathcal{P}}} = A^{\tilde{\mathcal{Q}}}$ if no swaps are applied, i.e. if $\mathfrak{S} = \mathbf{0}_n$. Say that $\tilde{\mathcal{P}}$ is as shown in the above table, and $\mathfrak{S}$ describes a single swap; and say also that this swap changes one of the two $acdb$ linear orders in $\mathcal{P}$ to $adcb$. Then the resulting matrix $A^{\tilde{\mathcal{Q}}}$ is as shown below.

2	1	1	0
1	0	1	0
0	0	1	0
0	0	0	0

We may use the $m\tilde{n}$ elements of the matrix $A^{\tilde{\mathcal{Q}}}$ as the variables of our ILP. We note that the number of unique linear orders $\tilde{n}$ is less than or equal to $m!$. Thus this representation of the Dodgson score problem as an ILP has no more than $m!m$ variables. Bartholdi et al. (1989) used the principle that $\tilde{n}$ is no more than $m!$ to define an ILP that also had $m!m$ variables. We will show how we can reduce the number of variables further, and present an ILP that has only $\mathcal{O}[(m-1)!]$ variables, reducing the number of variables by m^2. So far we have reduced the number of variables by joining together agents who chose the same linear order, converting our profile to a voting situation. We will show how we may join together more variables.

First we note that if an agent ranks two (or more) alternatives below d, the order of those alternatives do not matter when finding the Dodgson score of d, as $\mathfrak{S}$ never swaps d *down* the preference list. For example, for the purposes of calculating the Dodgson score of d, the orders $adbc$ and $adcb$ are equivalent. We may discard the alternatives following d, so instead of having $\{adbc, adcb\}$ we have $\{ad^2\}$ (the ^{2} means that the multiplicity of ad is 2). For example,

1	1	1
a	a	a
d	d	b
b	c	c
c	b	d

becomes

2	1
a	a
d	b
	c
	d

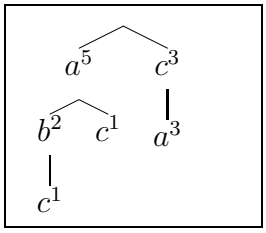
.

We now have only $\mathcal{O}[(m-1)!]$ unique sequences, and thus only $\mathcal{O}(m!)$ variables, although we will not prove this as we will find yet better formulation.

Say we have two sequences abd and acd, and that we have already applied one swap to each sequence. Then the sequences both become ad, and it makes no difference whether we apply and additional swap to the first sequence or the second. Thus we may join the a in abd and acd. Similarly if we have $abcd$ and abd, we may join the ab in each sequence. Doing so results in a tree as shown below.

2	1	1	1	3
a	a	a	a	c
d	b	b	c	a
	c	d	d	d
	d			

becomes



We will show later that there are less than $(m-1)!e$ nodes in this tree and thus there are less than $(m-1)!e$ variables in the corresponding ILP. However, we will first formally define the ILP.

4.2 Preliminary Definitions

Note 4.2.1 *We have defined a voting situation $\tilde{\mathcal{P}}$ as a multiset, much like a profile but with anonymous agents, on page 20. We have defined multisets and submultisets in Appendix A.5.*

Definition 4.2.2 *For any linear order $\mathbf{v}$ in our profile $\mathcal{P}$ and alternative d, we define $\mathfrak{A}_{\mathbf{v}}(d)$ to be the number of alternatives ranked above d in $\mathbf{v}$, and $\mathfrak{B}_{\mathbf{v}}(d)$ to be the number of alternatives ranked below d in $\mathbf{v}$. Let be $\tilde{\mathcal{P}} = (A, f)$ be our voting situation. Recall that A is the set of unique alternatives in the multiset $\tilde{\mathcal{P}}$ and $f\colon A \to \mathbb{N}$ is the frequency function which determines the frequency of each member of A. We define $\mathfrak{A}_{\tilde{\mathcal{P}}}(d)$, to be $\sum_{v \in A} \mathfrak{A}_{\mathbf{v}}(d)\, f(\mathbf{v})$. For example if our voting situation is $\{abcd^2, adcb\}$ then $\mathfrak{A}_{\tilde{\mathcal{P}}}(d)$ is 7, and the score of d according to the famous Borda rule would be $\mathfrak{B}_{\tilde{\mathcal{P}}}(d)$ using this notation.*

We have developed a formulation of the Dodgson score as an ILP which requires less variables than the formulation by Bartholdi et al. (1989). We now define some concepts required to describe this ILP.

Let $\mathcal{P}$ be a profile. Let d be the alternative for which we wish to find the Dodgson Score. Let $\mathcal{A}$ be the set of all alternatives. Let $\mathcal{C}$ be the set of all alternatives except d, i.e. $\mathcal{C} = \mathcal{A} - \{d\}$.

Definition 4.2.3 *For any set X, we define $Z(X)$ to be the set of all ordered subsets of X, and $M(X)$ to be the set of all possible multisets whose unique elements are a subset of X. Thus a voting situation is an element of $M(\mathcal{L}(\mathcal{A}))$.*

Definition 4.2.4 *We define a **sequence** of alternatives to be a (possibly empty) ordered set of alternatives. We write the sequence (a, b, c) as abc. A sequence of alternatives is a member of the set $Z(\mathcal{A})$.*

Definition 4.2.5 *For any sequence $\mathbf{c} = c_1 \cdots c_{|\mathbf{c}|}$ and alternative a we define $\mathbf{c}a$ to be a appended to $\mathbf{c}$, i.e. $c_1 \cdots c_{|\mathbf{c}|} a$.*

Definition 4.2.6 *We define the function $S_{\mathcal{P}}\colon Z(\mathcal{A}) \to M(\mathcal{L}(\mathcal{A}))$, such that for all $\mathbf{c} = c_1 \cdots c_{|\mathbf{c}|}$ in $Z(\mathcal{A})$ the multiset $S_{\mathcal{P}}(\mathbf{c})$ is a submultiset of $\tilde{\mathcal{P}}$, containing only those linear orders whose $|\mathbf{c}|$ highest ranked preferences are $c_1 \cdots c_{|\mathbf{c}|}$ in that order.*

4.3 Definition of Integer Linear Program

Given that we have fixed $\mathcal{P} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$, we know that $S_\mathcal{P}$ is likewise fixed. Let $\mathcal{Q}$ be the profile $\mathcal{P}$ after the swaps in $\mathfrak{S}$ have been applied. Let us consider the possible functions $S_\mathcal{Q}$, given that $\mathcal{P}$ is fixed but we may choose $\mathfrak{S} = (s_1, s_2, \ldots, s_n)$ to be any n-dimensional vector of non-negative integers such that $s_i \leq \mathfrak{A}_d(\mathbf{v}_i)$ for all integers i in $\{1, 2, \ldots, n\}$.

Given $a, b \in \mathbf{c}$ with $b \neq a$ we know that if the $(|\mathbf{c}| + 1)^{\text{th}}$ alternative in a linear order is a it cannot also be b hence we have:

$$S_\mathcal{P}(\mathbf{c}a) \cap S_\mathcal{P}(\mathbf{c}b) \quad = \quad \emptyset.$$

As we have assumed (without loss of generality) that we only swap d above other candidates, $S_\mathcal{Q}(\mathbf{c})$ is always a subset of $S_\mathcal{P}(\mathbf{c})$, and so we also have,

$$S_\mathcal{Q}(\mathbf{c}a) \cap S_\mathcal{Q}(\mathbf{c}b) \quad = \quad \emptyset.$$

Consider some linear order $\mathbf{v}$ in $\mathcal{P}$. Let $C_\mathbf{v}(S_\mathcal{P})$ be the set of sequences $\mathbf{c}$ such that $\mathbf{v}$ is a member of $S_\mathcal{P}(\mathbf{c})$, and similarly let $C_\mathbf{v}(S_\mathcal{Q})$ be the set of sequences $\mathbf{c}$ such that $\mathbf{v}$ is a member of $S_\mathcal{Q}(\mathbf{c})$.

Say, for example, that the linear order $\mathbf{v}$ is $abcd$. We see that in this case $C_\mathbf{v}(S_\mathcal{P})$ is $\{abc, ab, a\}$. We may swap d over 0,1,2 or 3 alternatives in $\mathbf{v}$, giving us 4 choices for $C_\mathbf{v}(S_\mathcal{Q})$: $\emptyset, \{a\}, \{a, ab\}, \{a, ab, abc\}$. We see that, in general, we may choose $C_\mathbf{v}(S_\mathcal{Q})$ to be any subset of $C_\mathbf{v}(S_\mathcal{P})$ such that, if for any sequence $\mathbf{c}$ and alternative a we have $\mathbf{v} \in S_\mathcal{Q}(\mathbf{c}a)$, then we also have $\mathbf{v} \in S_\mathcal{Q}(\mathbf{c})$. Equivalently, since $\mathfrak{S}$ is arbitrary, $S_\mathcal{Q}$ can be any function of the form $Z(\mathcal{A}) \to M(\mathcal{L}(\mathcal{A}))$ which satisfies the following criteria for all $\mathbf{c} \in Z_C$ and $a \notin \mathbf{c}$:

$$\begin{aligned} S_\mathcal{Q}(\mathbf{c}) \quad &\subseteq \quad S_\mathcal{P}(\mathbf{c}), \\ S_\mathcal{Q}(\mathbf{c}) \quad &\supseteq \quad S_\mathcal{Q}(\mathbf{c}a), \end{aligned}$$

Recall that we define $\mathbf{c}a$ as a appended to $\mathbf{c}$, i.e. $\mathbf{c}a$ represents $c_1 \cdots c_{|\mathbf{c}|}a$. The first statement is the parallel of the requirement that $C(S_\mathcal{Q})$ be a subset of $C(S_\mathcal{P})$, i.e. that because we only swap d over other alternatives, if a sequence $\mathbf{c} \in Z(\mathcal{C})$ represents the top $|\mathbf{c}|$ alternatives in an agent's preference list after applying the swaps, then the top $|\mathbf{c}|$ alternatives will also be $\mathbf{c}$ before the swaps were applied. The second requirement $S_\mathcal{Q}(\mathbf{c}) \supseteq S_\mathcal{Q}(\mathbf{c}a)$ is the parallel of the requirement that if the top $|\mathbf{c}| + 1$ alternatives are $\mathbf{c}a$ then the top $|\mathbf{c}|$ alternatives must be $\mathbf{c}$.

When trying to find the Dodgson score we are only interested in the number of swaps

required to make d a Condorcet winner, not which swaps are required. When we only consider the cardinality the two requirements above become, $\mathbf{c} \in Z(\mathcal{C})$ and $a \notin \mathbf{c}$:

$$\begin{aligned} 0 \leq |S_{\mathcal{Q}}(\mathbf{c})| &\leq |S_{\mathcal{P}}(\mathbf{c})|, \\ |S_{\mathcal{Q}}(\mathbf{c})| &\geq \sum_{a \neq \mathbf{c}} |S_{\mathcal{Q}}(\mathbf{c}a)|, \end{aligned}$$

The first requirement is easy to understand. Given the requirement that $S_{\mathcal{Q}}(\mathbf{c})$ must be a subset of $S_{\mathcal{P}}(\mathbf{c})$, we see that the requirement allows the cardinality of $S_{\mathcal{Q}}(\mathbf{c})$ to be any value between 0 and the cardinality of $S_{\mathcal{P}}(\mathbf{c})$. The second requirement follows from the fact that $S_{\mathcal{Q}}(\mathbf{c}a) \cap S_{\mathcal{Q}}(\mathbf{c}b) = \emptyset$ and $S_{\mathcal{Q}}(\mathbf{c}) \supseteq S_{\mathcal{Q}}(\mathbf{c}a)$. Let $U_{\mathcal{Q}}(\mathbf{c})$ be the cardinality of the set $S_{\mathcal{Q}}(\mathbf{c})$, and $U_{\mathcal{P}}(\mathbf{c})$ be the cardinality of the set $S_{\mathcal{P}}(\mathbf{c})$. We may now express these constraints in terms of U:

$$\begin{aligned} 0 \leq U_{\mathcal{Q}}(\mathbf{c}) \leq U_{\mathcal{P}}(\mathbf{c}), \\ U_{\mathcal{Q}}(\mathbf{c}) \geq \textstyle\sum_{a \neq \mathbf{c}} U_{\mathcal{Q}}(\mathbf{c}a), \end{aligned}$$

Now wish to require that $\mathfrak{S}$ makes d a Condorcet winner, i.e. d is a Condorcet winner in $\mathcal{Q}$. We know d will be a Condorcet winner in $\mathcal{Q}$ if and only if d is ranked below a in at most half of the linear orders in $\mathcal{P}$ for all a in C. The corresponding constraints on U are, for all a in C:

$$\sum_{\mathbf{c} \subseteq Z(C - \{a\})} U_{\mathcal{Q}}(\mathbf{c}a) \leq \frac{n}{2}.$$

To provide tighter constraints, we make use of the fact that d must be ranked below a in an integer number of linear orders, and modify the above constraints to be:

$$\sum_{\mathbf{c} \subseteq Z(C - \{a\})} U_{\mathcal{Q}}(\mathbf{c}a) \leq \left\lfloor \frac{n}{2} \right\rfloor.$$

We assert that $\mathfrak{A}_{\tilde{\mathcal{P}}}(d) = \sum_{\mathbf{c} \in Z_C, c \neq \emptyset} U_{\mathcal{P}}(\mathbf{c})$. This is clearly true if our profile $\mathcal{P}$ has no agents, as there are no agents to rank alternatives above d and also $U_{\mathcal{P}}(\mathbf{c}) = 0$. Assume that our assertion is true for all voting situations with n agents. Say that we add a linear order $\mathbf{v}$ to our voting situation $\tilde{\mathcal{P}}$. We choose $\mathbf{a} \in Z(\mathcal{C})$ so that $\mathbf{v}$'s top $|\mathbf{a}| + 1$ preferences are $a_1 \cdots a_{|\mathbf{a}|} d$ in that order. Then $\mathfrak{A}_v(d) = |\mathbf{a}|$. Adding $\mathbf{v}$ to $\tilde{\mathcal{P}}$ will add an additional member to $S(a_1)$, $S(a_1 a_2)$, ..., $S(a_1 \cdots a_{|\mathbf{a}|-1})$ and $S(a_1 \cdots a_{|\mathbf{a}|})$. Thus adding $\mathbf{v}$ to P will increase $U_{\mathcal{P}}(a_1)$, $U_{\mathcal{P}}(a_1 a_2)$, ..., $U_{\mathcal{P}}(a_1 \cdots a_{|\mathbf{a}|-1})$ and $U_{\mathcal{P}}(a_1 \cdots a_{|\mathbf{a}|})$ by one. It follows that adding $\mathbf{v}$ to $\tilde{\mathcal{P}}$ will increase both $\mathfrak{A}_{\tilde{\mathcal{P}}}(d)$ and $\sum_{\mathbf{c} \in Z_C, c \neq \emptyset} U_{\mathcal{P}}(\mathbf{c})$ by $|\mathbf{a}|$.

We know that

$$\mathfrak{A}_{\tilde{\mathcal{P}}}(d) = \sum_{\mathbf{c}\in Z_C, c\neq\emptyset} U_{\mathcal{P}}(\mathbf{c}),$$

similarly,

$$\mathfrak{A}_{\tilde{\mathcal{Q}}}(d) = \sum_{\mathbf{c}\in Z_C, c\neq\emptyset} U_{\mathcal{Q}}(\mathbf{c}).$$

The total number of swaps $\Sigma\mathfrak{S}$ is equal to the total number of times d has been swapped up linear orders in $\mathcal{P}$, hence

$$\begin{aligned}\Sigma\mathfrak{S} &= \mathfrak{A}_{\tilde{\mathcal{P}}}(d) - \mathfrak{A}_{\tilde{\mathcal{Q}}}(d)\\ &= \sum_{\mathbf{c}\in Z_C,\mathbf{c}\neq\emptyset} U_{\mathcal{P}}(\mathbf{c}) - \sum_{\mathbf{c}\in Z_C,\mathbf{c}\neq\emptyset} U_{\mathcal{Q}}(\mathbf{c})\\ &= \sum_{\mathbf{c}\in Z_C} U_{\mathcal{P}}(\mathbf{c}) - U_{\mathcal{Q}}(\mathbf{c})\end{aligned}$$

The Dodgson score of d is the minimum number of swaps required to make d a Condorcet winner, i.e. the minimum value of $\Sigma\mathfrak{S}$ given the constraints discussed above. Thus to find the Dodgson Score we wish to maximise $\sum_{\mathbf{c}\in Z_C,\mathbf{c}\neq\emptyset} U_{\mathcal{Q}}(\mathbf{c})$ subject to the conditions above, and take $\sum_{\mathbf{c}\in Z_C,\mathbf{c}\neq\emptyset} U_{\mathcal{P}}(\mathbf{c}) - U_{\mathcal{Q}}(\mathbf{c})$ as the Dodgson Score. We will do this by representing this as an ILP, and solving this ILP.

Note that we are no longer interested in $\mathfrak{S}$, since the Dodgson score can be calculated directly from $U_{\mathcal{Q}}$. Note that $0 \leq U_{\mathcal{Q}}(\mathbf{c}) \leq U_{\mathcal{P}}(\mathbf{c})$ and so $U_{\mathcal{Q}}(\mathbf{c})$ is constant (i.e. zero) if $U_{\mathcal{P}}(\mathbf{c}) = 0$. Also we are not interested in $U_{\mathcal{Q}}(\emptyset)$ as it is a trivial case and $U_{\mathcal{Q}}(\emptyset) \equiv U_{\mathcal{P}}(\emptyset)$ so $U_{\mathcal{Q}}(\emptyset)$ is thus not a variable.

Let $V = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{|V|})$ be an ordered set such that V contains a sequence $\mathbf{c}$ if and only if $\mathbf{c} \neq \emptyset$, and $U_{\mathcal{P}}(\mathbf{c}) > 0$, and $\mathbf{c}$ is a sequence of alternatives that does not contain d (i.e. $\mathbf{c} \in Z(\mathcal{C})$).

Let us consider the ILP $(M, N, A, \mathbf{b}, \mathbf{c})$. Let the variable vector of the ILP $\mathbf{x}$ represent $(U_{\mathcal{Q}}(\mathbf{c}_1), U_{\mathcal{Q}}(\mathbf{c}_2), \ldots, U_{\mathcal{Q}}(\mathbf{c}_{|V|}))^T$.

For each constraint of the form $U_{\mathcal{Q}}(\mathbf{c}_i) \leq U_{\mathcal{P}}(\mathbf{c}_i)$, we add a row to the matrix A. Say that this row is the j^{th} row. Then $A_{j,i} = 1$ and $A_{j,k} = 0$ for each $k \neq i$. We then set $b_j = U_{\mathcal{P}}(\mathbf{c}_i)$.

For each constraint of the form $U_{\mathcal{Q}}(\mathbf{c}_i) \geq \sum_{a\notin\mathbf{c}_i} U_{\mathcal{Q}}(\mathbf{c}_i a)$. Say that this row is the j^{th} row.

Then,

$$A_{j,k} = \begin{cases} -1 & \text{if} \quad k = i \\ 1 & \text{if} \quad \exists_{a \notin \mathbf{c}_i} \mathbf{c}_i a = \mathbf{c}_k \\ 0 & \text{otherwise} \end{cases},$$

and $b_j = 0$.

For each constraint of the form $\sum_{\mathbf{c} \subseteq Z(C-\{a\})} U_{\mathcal{Q}}(\mathbf{c}a) \leq \lfloor \frac{n}{2} \rfloor$, we add a row to the matrix A. Say this row is the j^{th} row. Then,

$$A_{j,k} = \begin{cases} 1 & \text{if} \quad \exists_{\mathbf{c} \subseteq Z(C-\{a\})} \mathbf{c}_i a = \mathbf{c}_k \\ 0 & \text{otherwise} \end{cases},$$

and $b_j = \lfloor \frac{n}{2} \rfloor$.

We let M be $|V|$, and let N be the number of rows we have added to the matrix. We let $\mathbf{c} = \mathbf{1}_M$.

Note 4.3.1 *We do not need to include constraints of the form $U_{\mathcal{Q}}(\mathbf{c_i})$, as in our definition of an LP, the variables are all non-negative.*

4.4 Number of Variables

The primary factor in the complexity of solving LPs and ILPs is the number of variables. We investigate the number of variables used in the ILP developed in Section 4.

Lemma 4.4.1 *There are no more than $(m-1)n$ non-empty sequences $\mathbf{c}$ for which $S_{\mathcal{P}}(\mathbf{c})$ is non-empty.*

Proof. We see that for each agent $\mathbf{v}$ there are at most $m-1$ alternatives ranked above d. Thus $\mathbf{v}$ is a member of $S_{\mathcal{P}}(\mathbf{c})$ for at most $m-1$ non-empty sequences $\mathbf{c}$. Thus given the n agents in profile $\mathcal{P}$, there are at most $(m-1)n$ non-empty sequences $\mathbf{c}$ for which $S_{\mathcal{P}}(\mathbf{c})$ is non-empty. □

Recall that on the facing page we defined $V = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{|V|})$ as some ordered set such that V contains a sequence $\mathbf{c}$ if and only if $\mathbf{c} \neq \emptyset$, and $U_{\mathcal{P}}(\mathbf{c}) > 0$, and $\mathbf{c}$ is a sequence of alternatives that does not contain d (i.e. $\mathbf{c} \in Z(\mathcal{C})$). Recall also that $U_{\mathcal{P}}(\mathbf{c}) > 0$ is the cardinality of the set $S_{\mathcal{P}}(\mathbf{c})$ of linear orders in $\mathcal{P}$ such that the $|\mathbf{c}|$ highest ranked alternatives are $\mathbf{c} = c_1 \cdots c_{|\mathbf{c}|}$ in that order. Recall that $\mathcal{Q}$ is our profile after the swaps $\mathfrak{S}$ are applied.

Corollary 4.4.2 *We need no more than $(m-1)n$ variables in our ILP.*

Proof. Where $S_{\mathcal{P}}(\mathbf{c}) = \emptyset$ we have $U_{\mathcal{P}}(\mathbf{c}) = 0$. As discussed on page 65 we know $0 \leq U_{\mathcal{Q}}(\mathbf{c}) \leq U_{\mathcal{P}}(\mathbf{c})$ and $\mathbf{c}$ is not a member of V. Also recall that $\emptyset$ was not a member of V. Hence, from the previous lemma, $M = |V| \leq (m-1)n$. □

We now have a reasonable upper bound on the number of variables in our ILP. However in this thesis we are primarily interested in the case where $n \gg m$. Thus we will show that the number of variables is $\mathcal{O}(1)$ if the number of alternatives m is fixed.

Lemma 4.4.3 *There are less than $(m-1)!e$ ordered sets of unique alternatives in $Z(\mathcal{C})$, i.e. $|Z(\mathcal{C})| < (m-1)!e$, where $e = 2.71\cdots$ is the exponential constant.*

Proof. We have m alternatives, so we have $m-1$ alternatives in C. Hence we have $(m-1)$ orders sets of length one, $(m-1)(m-2)$ orders sets of length 2, and $(m-1)(m-2)\ldots(m-k)$ orders sets of length k where $k \leq |\mathcal{C}| = m-1$. Thus we know,

$$\begin{aligned}
\#\text{ sequences} &= 1 + (m-1) + (m-1)(m-2) + \cdots \\
&= \frac{(m-1)!}{(m-1)!} + \frac{(m-1)!}{(m-2)!} + \frac{(m-1)!}{(m-3)!} + \cdots + \frac{(m-1)!}{1!} + \frac{(m-1)!}{0!} \\
&= (m-1)!\left(\frac{1}{(m-1)!} + \frac{1}{(m-2)!} + \cdots + \frac{1}{0!}\right) \\
&< (m-1)!\left(\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \ldots\right) \\
&= (m-1)!e, \quad \text{where } e = 2.718282\ldots
\end{aligned}$$

Hence we have less than $(m-1)!e$ ordered sets $\mathbf{c}$ in $Z(\mathcal{C})$ □

Corollary 4.4.4 *This formulation of the Dodgson Score as an ILP has less than $(m-1)!e$ variables.*

Proof. For each ordered set of unique alternatives $\mathbf{c}$ that does not include d, i.e member of $Z(\mathcal{C})$, there exists no more than a single variable $U_{\mathcal{Q}}(\mathbf{c})$. From the previous lemma we know that the number of ordered sets of unique alternatives is less than $(m-1)!e$. □

4.5 Size of the Linear Program

A second factor in the complexity of solving LPs and ILPs is the number of bits required to encode the ILP. We investigate the number of bits required to encode the ILP

$(M, N, A, \mathbf{b}, \mathbf{c})$, that we developed to find the Dodgson score in Section 4, in terms of the number of variables M in the ILP.

Corollary 4.5.1 *There are no more than $3M$ non-zero entries in the matrix A.*

Proof. We have a constraint of the form $U_{\mathcal{Q}}(\mathbf{c}_i) \leq U_{\mathcal{P}}(\mathbf{c}_i)$, for each $\mathbf{c}$ in $|V|$. Recall that for each such constraint we added a single non-zero entry $(A_{j,i})$ to A.

We also add constraints of the form $U_{\mathcal{Q}}(\mathbf{c}_i) \geq \sum_{a \notin \mathbf{c}_i} U_{\mathcal{Q}}(\mathbf{c}_i a)$ for each a in $\mathcal{C} = \mathcal{A} - \{d\}$. For each of these entries we added a non-zero entry to A for each sequence $\mathbf{c}_i a \in V$ that ended in a for $\mathbf{c}_i a$. Since we have such a constraint for each a in $\mathcal{C}$, in total we added a single non-zero entry to A for each element $\mathbf{c}_i a$ of V, i.e. exactly $M = |V|$ non-zero entries.

Similarly, for the constraints of the form $\sum_{\mathbf{c} \subseteq Z(C-\{a\})} U_{\mathcal{Q}}(\mathbf{c}a) \leq \lfloor \frac{n}{2} \rfloor$ we also added a single non-zero entry to A for each element of $|V|$.

Hence in total there are $3M$ non-zero entries in A. □

Corollary 4.5.2 *It is possible to encode A using $\mathcal{O}(M \ln M)$ bits.*

Proof. In LP problems A is typically a sparse matrix, and is encoded in a way that makes use of this. For each row we may identify which columns are non-zero. Identifying a column (i.e. variable) requires $\mathcal{O}(\ln M)$ bits. Since the non-zero entries are -1 or 1 we may encode which type of non-zero entry they are using a single bit. Thus encoding the $3M$ non-zero entries requires $\mathcal{O}(M \ln M)$ bits.

Since no rows are empty we do not need to worry about using bits to encode empty columns. Thus it is possible to encode A in $\mathcal{O}(M \ln M)$ bits. □

Lemma 4.5.3 *It is possible to encode* $\mathbf{b}$ *using $\mathcal{O}(M \ln n)$ bits.*

Proof. Each non-zero entry in $\mathbf{b}$ represents the cardinality of a submultiset of our voting situation $\tilde{\mathcal{P}}$. Thus each entry of $\mathbf{b}$ is at most n, and hence can be encoded in $\mathcal{O}(\ln n)$ bits. $\mathbf{b}$ has $\mathcal{O}(M)$ entries, and so can be encoded $\mathcal{O}(M \ln n)$ bits. □

Lemma 4.5.4 *It is possible to encode* $\mathbf{c}$ *using $\mathcal{O}(M)$ bits.*

Proof. We have set $\mathbf{c} = \mathbf{1}_M$. Since 1 is a constant, it is possible to encode $\mathbf{c}$ using $\mathcal{O}(1)$ bits. Since $\mathbf{c}$'s M entries are all 1, it can be encoded using $\mathcal{O}(\ln M)$ bits. □

Theorem 4.5.5 *It is possible to encode our LP* $(M, N, A, \mathbf{b}, \mathbf{c})$*, for finding Dodgson's score of* d*, using* $\mathcal{O}(M \ln M + M \ln n)$ *bits.*

Proof. As N is $\mathcal{O}(M)$, we may encode M and N in $\mathcal{O}(\ln M)$ bits. Thus from the last three results, encoding $(M, N, A, \mathbf{b}, \mathbf{c})$ requires $\mathcal{O}(\ln M + M + M \ln M + M \ln n)$ bits. This simplifies to $\mathcal{O}(M \ln M + M \ln n)$. □

Corollary 4.5.6 *It is possible to encode our LP with* $\mathcal{O}\left[((m-1)!) \ln ((m-1)!n)\right]$ *bits.*

Proof. From Theorem 4.5.5, we may encode our LP in $L \in \mathcal{O}(M \ln M + M \ln n)$ bits. Thus, $L \in \mathcal{O}[M(\ln M + \ln n)]$, and thus $\mathcal{O}[M \ln(Mn)]$ bits. Thus,

$$L \in \mathcal{O}\left[((m-1)!e) \ln ((m-1)!en)\right],$$

which we may simplify by removing the constants to be:

$$L \in \mathcal{O}\left[(m-1)! \ln ((m-1)!n)\right].$$

□

Chapter 5

Dodgson Relaxed & Rounded

In Chapter 3 we studied the asymptotic properties of very simple approximations to the Dodgson rule, including a new rule we proposed called the Dodgson Quick rule. In this chapter we propose another approximation to the Dodgson rule. Although this approximation is not as simple as the approximations in Chapter 3, we will show that it can be computed in polynomial time. Not only is this approximation even closer than the Dodgson Quick rule, which exhibited exponential convergence, but this rule is defined in a similar way.

In the previous chapter we showed that it is possible to find the Dodgson score of an alternative using an ILP. Unfortunately it can be difficult to find the optimal value of an ILP quickly. We define the Dodgson relaxed score in terms of the optimal value of a relaxation of the Integer Linear Program (ILP) to a Linear Problem (LP). A number of algorithms have been found which can find the optimal value of an LP in polynomial, the first was found by Khachian (1979).

Under the Dodgson rule, we may only switch neighbouring alternatives in whole votes. By relaxing the requirement that variables in the LP be integer, the Dodgson relaxed rule allows us to split votes into rational[1] fractions of a vote and swap neighbouring alternatives in these fractions of a vote. This provides us with two advantages over the Dodgson rule, first we will show that we may compute the Dodgson relaxed rule in polynomial time, and in logarithmic time if we fix the number of alternatives. Secondly, we will show numerically that when the set of tied winners chosen by this rule differs from the set of tied winners selected by Dodgson's rule, it is usually because the relaxed rule has chosen a subset of the Dodgson winners. We may break ties according to the preferences of the first agent, however it is in some sense more democratic to select an alternative that is fractionally better than the others than to privilege the first agent over the other agents.

[1] We could equivalently say that the Dodgson rule allows us to split votes into portions of any real size; due to the nature of LPs,these portions will always have a rational size.

By rounding the Dodgson relaxed score up we get a new score which we call the **Dodgson relaxed and rounded (R&R) score**. The R&R rule is exceptionally close to the Dodgson rule. Out of 43 million random simulations under various assumptions we only found a single simulation where the R&R winner was not the Dodgson winner. This simulation had 25 candidates but only 5 agents, a somewhat implausible size for an election. The example of a real world election with a huge number of candidates given by Bartholdi et al. (1989), was an election for mayor of Tulsa. Even this election had only 20 candidates, and there were presumably more than 5 residents of Tulsa voting.

5.1 Definition of Dodgson Relaxed Score

Definition 5.1.1 *Take the ILP defined in Chapter 4, that is used find the Dodgson score of some arbitrary alternative d. Recall that the Dodgson Score of d is the minimum value of $\sum_{\mathbf{c}\in Z_C,\mathbf{c}\neq\emptyset} U_{\mathcal{P}}(\mathbf{c})-U_{\mathcal{Q}}(\mathbf{c})$ subject to the constraints of this ILP. If we relax the integer constraints we get an LP, and we call the minimum value of $\sum_{\mathbf{c}\in Z_C,\mathbf{c}\neq\emptyset} U_{\mathcal{P}}(\mathbf{c}) - U_{\mathcal{Q}}(\mathbf{c})$ subject to these new constraints the **Dodgson relaxed score** of the alternative d. We may also equivalently define the Dodgson relaxed score as a modification of the Dodgson score where we are allowed to split votes and swap alternatives in those fractions of a vote. However note that the ILP requires that we swap each other alternative d over a at least $\lceil adv(a,d)/2\rceil$ times, even though $adv(a,d)/2$ times would be enough to made d a Condorcet winner. The alternative with the lowest Dodgson relaxed score is called the **Dodgson relaxed winner**.*

Example 5.1.2 *Say we have a profile with a single agent and two alternatives $\{a,b\}$, say that an agent votes ab. The Dodgson rule does not allow us to split fractional numbers of votes and so we have to switch b over a once, so the Dodgson score of b is 1. We might think that the Dodgson relaxed score of b would be 0.5 as the Dodgson relaxed rule does allow us to switch a fractional number of votes. However note that the Dodgson relaxed rule requires that not only we make b a Condorcet winner, but also that we swap b above a at least $F(a,b) = \lceil adv(a,b)/2\rceil$ times. Thus the Dodgson relaxed score is also 1.*

Note 5.1.3 *The requirement that we swap b above a at least $F(a,b) = \lceil adv(a,b)/2\rceil$ times makes the Dodgson relaxed rule a more accurate approximation to the Dodgson rule than if we had to swap b above a only $adv(a,b)/2$ times. Also, as we discuss below, this means that the Dodgson relaxed score of an alternative is always between the DQ-score and the Dodgson score.*

Example 5.1.4 *We present a voting situation in Table 5.1L. This voting situation has 6 alternatives and 5 agents, the Dodgson relaxed score of alternative d differs from the Dodgson score of d.*

The voting situation on the left is an ordinary voting situation where all frequencies are integers. Table 5.1R has split three votes into halves, as allowed by the Dodgson relaxed rule. Note that,

$$\begin{aligned} F(a,d) = F(b,d) = F(c,d) = F(g,d) &= 1, \\ F(f,d) &= 0. \end{aligned}$$

Thus under both the Dodgson and Dodgson relaxed rules we will have to swap d above a, b, c, and g at least once each. The minimal set of swaps is shown on each subtable. On the left subtable we see that we need 5 swaps to make d a Condorcet winner, and so the Dodgson score of d is 5. On Table 5.1R we see that we need 9 half-swaps to swap d above a, b, c, g at least once. Thus the Dodgson relaxed score of d is 4.5.

Table 5.1: Example of Dodgson Score of d Differing from the Relaxed Score

Table 5.1L: No Splits (Dodgson Score =5)

1	1	1	1	1
b	a	a	d	a
c	b	c	g	b
g	g	g	f	c
d	d	d	a	f
a	c	b	b	d
f	f	f	c	g

Table 5.1R: Splits Allowed (Dodgson Relaxed Score=$9/2$)

$1/2$	$1/2$	$1/2$	$1/2$	$1/2$	$1/2$	1	1
b	b	a	a	a	a	d	a
c	c	b	b	c	c	g	b
g	g	g	g	g	g	f	c
d	d	d	d	d	d	a	f
a	a	c	c	b	b	b	d
f	f	f	f	f	f	c	g

Note 5.1.5 *Although the Dodgson relaxed score differs from the Dodgson score, the R&R score is $\lceil 4.5 \rceil = 5$ which equals the Dodgson score. We will show later that the Dodgson relaxed score is always less than or equal to the Dodgson score. As the R&R score is the relaxed score rounded up, it is clear that the R&R score will differ from the Dodgson score if and only if the relaxed score differs from the Dodgson score by at least one. With millions of random simulations, with 3 or 5 alternatives, various numbers n of agents and various assumptions of voter behaviour, we found no cases where the Dodgson relaxed score differed from the Dodgson score for any pair of alternatives; with 7, 9 or 15 alternatives the Dodgson score differed from the relaxed score by no more than 0.5; with 25 alternatives the relaxed score differed from the Dodgson score by as much as 2. Thus we only found cases where the R&R score differed from the Dodgson score in the random simulations with 25 alternatives.*

Conjecture 5.1.6 *For profile with less than 6 alternatives or less than 5 agents, the Dodgson score of any of the alternatives equals the Dodgson relaxed score for that alternative.*

Note 5.1.7 *It appears that the Dodgson relaxed winner differs from the Dodgson winner only for large and complicated profiles. The smallest profile we found where the Dodgson relaxed winner (after tie breaking) differed from the Dodgson winner had 15 alternatives, although we expect that there are profiles with substantially less alternatives where the Dodgson relaxed winner differs from the Dodgson winner.*

Lemma 5.1.8 *For any profile $\mathcal{P}$ and alternative a, The Dodgson relaxed score of a is equal to or greater than the Dodgson Quick score of a.*

Proof. The Dodgson relaxed rule requires us to swap alternative a over each alternative b distinct from a at least $F(b, a)$ times. Hence the Dodgson relaxed score of a is at least

$$\sum_{b \neq a} F(b, a),$$

which is the Dodgson quick score. □

Lemma 5.1.9 *For any profile $\mathcal{P}$ and alternative a, the Dodgson relaxed score of a is less than or equal to the Dodgson score of a.*

Proof. As with the Dodgson rule, for each alternative b we must raise a above b at least $F(b, a)$ times. The minimum number of swaps is the Dodgson score. Any set of swaps that is valid under the Dodgson rule is also valid under the Dodgson relaxed rule, so the Dodgson relaxed score can be no greater than the Dodgson score. □

Corollary 5.1.10 *The Dodgson relaxed score is as close or closer to the Dodgson score than the DQ-score.*

Obvious from the previous two lemmas.

Corollary 5.1.11 *Under the impartial culture assumption, the probability that the Dodgson relaxed score of a is the Dodgson score converges to 1 as we increase the number of agents.*

Obvious from the previous corollary, and Theorem 3.1.8 which states that under the impartial culture assumption the probability that the DQ-score equals the Dodgson score converges to 1 as we increase the number of agents.

Corollary 5.1.12 *Under the impartial culture assumption, the probability that the Dodgson relaxed winner is the Dodgson winner converges to 1 as we increase the number of agents.*

We have compared the Dodgson rule and the Dodgson relaxed rule numerically. As the impartial culture assumption may not be realistic, we have aggregated the results using 8 values of the parameter of homogeneity b (i.e. $b \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 2\}$) under the Pólya-Eggenberger urn model. Recall that with $b = 0$, this urn model is equivalent to the impartial culture assumption. For each value of b we did 10,000 simulations, thus the results in Tables 5.2, 5.3 and 5.4 are aggregated from a total of 80,000 simulations for each profile size.

Table 5.2: Occurrences out of 80,000 that the Dodgson Relaxed Winner Differed from the Dodgson Winner after Tie-Breaking.

	# agents						
# alternatives	3	5	7	9	15	25	85
7	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
15	0	0	0	1	0	0	0
25	0	3	5	0	3	5	1

From Table 5.2 we see that the Dodgson relaxed rule is such a close approximation to the Dodgson rule that we may want to pick a more demanding definition of equality. Dodgson's rule may pick a set of tied winners. In Table 5.2, this tie is broken by the preferences of the first agent. Instead of considering just the cases where Dodgson's rule picks a different winner, we also consider cases where the approximation did not pick the same set of tied winners, in Table 5.3.

Table 5.3: Occurrences out of 80,000 that the Set of Tied Dodgson Relaxed Winners Differed from the Set of Tied Dodgson Winners.

	# agents						
# alternatives	3	5	7	9	15	25	85
7	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
15	0	0	0	1	2	0	0
25	0	3	9	0	4	5	1

Another interesting question is whether the winner chosen by the Dodgson relaxed rule is a Dodgson winner. If the set of tied winners selected by the Dodgson relaxed rule is a subset of the set of tied Dodgson winners then clearly the winner chosen by the Dodgson relaxed rule, after tie-breaking, will be a Dodgson winner. The Dodgson relaxed scores can be fractional scores, and the difference between the true Dodgson score and the Dodgson relaxed score is usually less than one. Thus we expected that the set of tied

Dodgson relaxed winners would be a subset of the Dodgson winners. We see in Table 5.4 that this is almost always the case.

Table 5.4: Occurrences out of 80,000 that the Set of Tied Dodgson Relaxed Winners were not a Subset of the Set of Tied Dodgson Winners.

		# agents						
		3	5	7	9	15	25	85
# alternatives	7	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0
	25	0	0	2	0	0	0	0

From the discussed tables, we see that the main difference between the Dodgson relaxed rule and the Dodgson rule is that the relaxed rule usually picks a subset of the Dodgson winners. This behaviour may actually be superior to the behaviour of the Dodgson rule. Selecting a set of tied winners means that the rule is unable to distinguish between the different alternatives. It is common to break ties according to the preferences of the first agent, however if the rule picks a large set of tied winners this tie breaking procedure makes the rule close to a dictatorship of the first agent (see e.g. M^{c}Cabe-Dansted and Slinko 2006). Though Dodgson died in 1898, it seems likely that, if he were alive, he would agree that it is better to choose a winner that is in some sense fractionally closer to being a Condorcet winner than the other alternatives, than to privilege the first agent over the other agents.

If we do not want to break ties by fractional amounts, then we may create a closer approximation to the Dodgson scores. The normal Dodgson scores are integers, and from Lemma 5.1.9 the Dodgson relaxed scores are lower bounds for the normal Dodgson scores. Thus for a Dodgson relaxed score r and an exact score e, not only is $e \geq r$ but also $e \geq \lceil r \rceil$. This $\lceil r \rceil$ is the R&R score. The R&R winner was the Dodgson winner in all 43 million simulations except one. That exception was with a population with 25 alternatives and 5 agents, generated according to the impartial culture assumption. There were 10,000 such simulations performed so we are confident (to a 99.9% degree of confidence) that even given this unusual profile size the probability that the R&R and Dodgson winners differ is less than 0.001.

As proven by Lemma 5.2.3 the Dodgson relaxed approximation resulting by relaxing the integer constraints can be computed with $\mathcal{O}\left(m^4 n^4 \ln(mn)\right)$ arithmetic operations on $\mathcal{O}\left(mn \ln(mn)\right)$ bit numbers. Thus the time required to solve the LP is polynomial with respect to the number of alternatives m and number of agents n. Together with the fact that the Dodgson relaxed winner is in some sense more legitimate than the Dodgson winner

chosen after tie-breaking, this may make the Dodgson relaxed rule a good replacement for the Dodgson rule.

5.2 Complexity

Theorem 5.2.1 *Given a pair of numbers both of which can be encoded in n bits, we may perform the basic arithmetic operations, add, subtract and multiply on these numbers in $\mathcal{O}(n \ln n \ln \ln n)$ time.*

Proof. We may perform additions and subtractions in $\mathcal{O}(n)$ time the obvious way, adding (or subtracting) each digit and carrying the overflow. As shown by Schönhage and Strassen (1971), we may perform multiplications in $\mathcal{O}(n \ln n \ln \ln n)$ time. □

Theorem 5.2.2 *It is possible to solve LPs in $\mathcal{O}(M^3 L)$ operations with $\mathcal{O}(L)$ bits of precision, where M is the number of variables and L is the number of bits required to encode the LP.*

The Gonzaga (1989) algorithm can solve an LP in $\mathcal{O}(M^3 L)$ time.

Lemma 5.2.3 *We may find the Dodgson relaxed score in $\mathcal{O}\left(m^4 n^4 \ln(mn)\right)$ arithmetic operations with $\mathcal{O}(mn \ln(mn))$ bits of precision, where m is the number of alternatives and n is the number of agents.*

Proof. From Corollary 4.4.2, our LP has $M = (m-1)n$ variables, which is in $\mathcal{O}(mn)$. From Theorem 4.5.5, we may encode our LP in $L \in \mathcal{O}(M \ln M + M \ln n)$ bits. Thus L is in $\mathcal{O}(mn \ln(mn))$. We may use Gonzaga's algorithm to solve the LP in $\mathcal{O}(M^3 L)$ time. Thus we may solve this LP using $\mathcal{O}(m^4 n^4 \ln(mn))$ arithmetic operations requiring $\mathcal{O}(mn \ln(mn))$ bits of precision. □

Note 5.2.4 *Although we have shown that the time required to calculate the Dodgson relaxed score is polynomial, we are particularly interested in the case where $n \gg m$. Thus we will study the case where the number of alternatives m is fixed and the number of agents n tends to infinity.*

Lemma 5.2.5 *We may solve our LP with $\mathcal{O}\left[((m-1)!)^4 \ln\left((m-1)!n\right)\right]$ operations of*

$$\mathcal{O}\left[(m-1)! \ln\left((m-1)!n\right)\right]$$

bits of precision.

Proof. From Lemma 4.4.3 we know that the number M of variables is less than $(m-1)!e$, where $e = 2.71\ldots$ is the exponential constant. From Corollary 4.5.6, the number of bits L needed to encode our LP is in $\mathcal{O}\left[(m-1)!\ln\left((m-1)!n\right)\right]$.

Using Gonzaga's algorithm we may solve LPs in $\mathcal{O}(M^3L)$ arithmetic operations of $\mathcal{O}(L)$ bits of precision. Thus we may solve this ILP with $\mathcal{O}\left[\left((m-1)!\right)^4\ln\left((m-1)!n\right)\right]$ such operations. □

Corollary 5.2.6 *We may find the Dodgson relaxed score using $\mathcal{O}(\ln n)$ operations of $\mathcal{O}(\ln n)$ bits, if the number of alternatives m is fixed.*

Chapter 6

Feasibility of Dodgson's Rule

One may find the Dodgson winner by simply computing the Dodgson score for each alternative, and finding which alternative has the lowest score. This method is inefficient; computing the Dodgson score requires an integer linear programming problem to be solved as discussed in Chapter 4.

Instead we start with the quick polynomial lower bounds discussed in Lemma 3.1.5. We replace the lowest lower bound with the exact score until the lowest exact score is lower than the lowest lower bound. We then know that the alternative with the lowest exact score is the Dodgson winner.

Algorithm 1 FindDodgsonWinner(Profile $\mathcal{P}$)

1: Calculate the Dodgson Quick scores, and use these as lower bounds on the Dodgson scores. (See Lemma 3.1.5)
2: Replace lowest lower bound with the exact Dodgson score.
3: Repeat step 2 until the lowest exact Dodgson score is lower than the lowest lower bound.
4: The winner is (are) the alternatives(s) with the lowest Dodgson scores.

In practice as well as in theory, the lower bound usually is the same as the Dodgson score, and this improved algorithm will rarely have to solve more than one or two integer linear programs. As we can see from Table 6.1, the Dodgson winner is quite easy to calculate for the moderate sizes. The Dodgson winner can also be computed quite easily for a large number of agents if we limit the number of alternatives. Calculating the Dodgson winner with 1025 agents and 5 alternatives takes about 10 milliseconds, on a 2.8GHz Xeon. A homogeneous population[1] reduces the time required to find the Dodgson winner, with increasing b to one almost halving the time required to calculate the Dodgson winners for a population with 85 agents and 25 alternatives.

[1] Homogeneity was modeled using the Pólya-Eggenberger Urn Model (see e.g. Norman and Samuel, 1969)

6.1 Theoretical Worst-Case Results

Theorem 6.1.1 (Eisenbrand 2003) *An ILP with a fixed number of variables and constraints can be solved with $\mathcal{O}(L)$ basic arithmetic operations of $\mathcal{O}(L)$ bits, where L is the number of bits required to encode the LP.*

Theorem 6.1.2 *For a fixed number of alternatives m, our ILP for the Dodgson Score can be computed in $\mathcal{O}(\ln n)$ basic arithmetic operations on $\mathcal{O}(\ln n)$ bits.*

Proof. From Corollary 4.4.4, we have less than $(m-1)!e$ variables. Likewise we have $\mathcal{O}[(m-1)!]$ constraints. Thus for a fixed number of alternatives m, we have a fixed number of variables and constraints. From Corollary 4.5.6 we may encode the ILP using $\mathcal{O}(\ln n)$ bits for fixed m. Thus from Eisenbrand's result, we may solve this ILP using $\mathcal{O}(\ln n)$ basic arithmetic operations in of $\mathcal{O}(\ln n)$ bits. □

We have shown that for fixed m we require only $\mathcal{O}(\ln n)$ operations to solve the ILP. Thus for some positive real valued function f, we require $\mathcal{O}(f(m)\ln n)$ operations to solve this ILP, and so this problem is Fixed Parameter Tractable (FPT) with respect to the parameter m. In practice, knowing that a problem is FPT may be of little practical use if f grows exceedingly quickly, e.g. if f is impossibly large even for $m \geq 2$. For this reason we prove some results to give an idea as to what f is. We will show that solving the ILP is FPT for a function f that is reasonable for $m \leq 3$, although it does grow very rapidly.

Corollary 6.1.3 *For a fixed number of alternatives m, our ILP for the Dodgson Score can be computed in $\mathcal{O}(\ln^2 n \ln n \ln\ln n)$ time.*

Theorem 6.1.4 (Lenstra, Jr. 1983) *The integer programming feasibility problem can be solved with $\mathcal{O}(p^{\frac{9p}{2}}L)$ arithmetic operations of $\mathcal{O}(p^{2p}L)$ bits in size, where p is the number of variables, and L the number of bits of input.*

Theorem 6.1.5 *The Dodgson score can be computed in time of order*

$$\mathcal{O}\left[f(m)\ln^3 n \ln\ln n \ln\ln\ln n\right]$$

where $f(m) = ((m-1)!)^{7(m-1)!}$.

Proof. From Corollary 4.4.4, we have less than $(m-1)!e$ variables. Likewise we have $\mathcal{O}[(m-1)!]$ constraints. Thus for a fixed number of alternatives m, we have a fixed number of variables and constraints. From Corollary 4.5.6 we may encode the ILP using

$\mathcal{O}\left[(m-1)!\ln\left((m-1)!n\right)\right]$ bits. Thus from Lenstra's result, we may solve the ILP feasibility problem using

$$\mathcal{O}\left[\left((m-1)!e\right)^{\frac{9(m-1)!e}{2}}(m-1)!\ln\left((m-1)!n\right)\right]$$

operations with

$$b \;=\; \mathcal{O}\left[\left((m-1)!e\right)^{2(m-1)!e}(m-1)!\ln\left((m-1)!n\right)\right]$$

bits of precision. This only solves the ILP feasibility problem, i.e. it only answers the question "Is the Dodgson Score of d less than k", for any arbitrary integer k. As the Dodgson Score is never greater than mn we can find the Dodgson score using a binary search of $\mathcal{O}(\ln(mn))$ steps. Thus we may find the Dodgson score in

$$\mathcal{O}\left[\left((m-1)!e\right)^{\frac{9(m-1)!e}{2}}(m-1)!\ln\left((m-1)!n\right)\ln n\ln m\right]$$

arithmetic operations.

From Schönhage and Strassen (1971), we know that it is possible to perform the basic arithmetic operations $\{+,-,\times\}$ in $\mathcal{O}(b\ln b\ln\ln b)$ time where b is the number of bits. Using these results we find that we may solve the ILP problem in

$$\mathcal{O}\left[\left((m-1)!e\right)^{\frac{13(m-1)!e}{2}}\left((m-1)!\right)^3\ln^4\left((m-1)!\right)\ln^2 n\ln m\times\ln\ln\left((m-1)!n\right)\times\ln\ln(b)\right]$$

time. Rather than expand this further, let us just note that this grows slower than,

$$\mathcal{O}\left[\left((m-1)!\right)^{7(m-1)!}\ln^3 n\ln\ln n\ln\ln\ln n\right],$$

and hence the result. □

Note 6.1.6 *This function $f(m)$ grows quickly with the number of alternatives; $f(3)\approx 10^4$ is moderate but $f(4)\approx 10^{32}$ is huge. A modern PC can perform roughly one billion operations per second; performing 10^{32} operations would take roughly a million billion years, which is vastly greater than the age of the universe.*

6.2 Approximability Classes

Definition 6.2.1 *Let g be any function. If f is a function that can be computed in polynomial time, and $g(X)-\epsilon\leq f(X)\leq g(X)+\epsilon$ for all inputs X and some constant ϵ, we call f a* ***constant***

approximation scheme *for* g.

Theorem 6.2.2 *There is no constant approximation scheme for the Dodgson Score unless P=NP.*

Proof. Assume that there exists a constant approximation scheme for the Dodgson score. Let us call this scheme "ApproxDodgsonScore". Then it is possible to find the exact Dodgson score using the following algorithm:

Algorithm 2 DodgsonScore(Profile $\mathcal{P}$, alternative d)

1: **for** each alternative $c \neq d$ in $\mathcal{P}$ **do**
2: Replace c with 5ϵ clones of c
3: **end for**
4: **return** Round(ApproxDodgsonScore$(\mathcal{P}, d)/(5\epsilon)$)

As each alternative other than d has been replaced with 5ϵ clones, where before we would have to swap d over s other alternatives, in the modified profile we have to swap d over $5\epsilon s$ alternatives. Hence in the modified profile, d has exactly 5ϵ times the original Dodgson score. Finding the approximate Dodgson score of this modified profile and dividing by 5ϵ gives the original Dodgson score of d to within ± 0.2. As we know the Dodgson score is an integer, we can find the exact Dodgson score simply by rounding this value.

Since Bartholdi et al. (1989) has shown that there is no polynomial algorithm to find the Dodgson Score unless P=NP, we can say by contradiction that there is no constant approximation scheme for Dodgson Score.

□

Lemma 6.2.3 *For any graph* $G = (V, E)$ *with* k *vertices, there is a profile* $\mathcal{P}$ *with a special alternative* q *such that the size of the Minimum Dominating Set of* G *is* $\lfloor k^{-2}s \rfloor$*, where* s *is the Dodgson score* $Sc_{\mathbf{D}}(q)$ *of* q*. This profile contains* n *agents and* m *alternatives with* $n \in \Theta(k)$ *and* $m \in \Theta(k^4)$*, and can be constructed in time that is polynomial with respect to* k*.*

.

Proof. For each $i \in 1, 2, \ldots, k$, let V_i be the set of vertices adjacent to v_i and $\bar{V}_i$ be the set of vertices not adjacent to v_i. Thus $V_i \cup \bar{V}_i = V$.

To construct the profile $\mathcal{P}$, we will first construct the set of alternatives $\mathcal{A}$.

- For each $i \in 1, 2, \ldots, k$
  - Define a set of alternatives D_i=$\{D_{i1}, D_{i2}, \ldots, D_{i(k^2-1)}\}$.

  - Define a set of alternatives R_i=$\{R_{i1}, R_{i2}, \ldots, R_{i(2k^3)}\}$

- Let R as the union of all R_i and D be the union of all D_i.
- Define $\mathcal{A}$, the set of all alternatives, as $V \cup R \cup D$. Note that V is the set of all vertices in G, and so all vertices in G are included as alternatives in $\mathcal{A}$.

Consider the following profile on $n = 2k + 1$ agents.

V_1	V_2	$\ldots$	V_k	$\bar{V_1}$	$\bar{V_2}$	$\ldots$	$\bar{V_k}$	V
D_1	D_2	$\ldots$	D_k	R_1	R_2	$\ldots$	R_k	R
q	q	$\ldots$	q	q	q	$\ldots$	q	q

For conciseness we have not listed the alternatives ranked below q; all alternatives not ranked explicitly above q are ranked below q. Also, when we were not interested in the order that an agent ranked members of some set of alternatives, we represented that set of alternatives as a single cell on the table. For example, in the first column of the table below, the set V_1 is in the top cell. This means that the top $|V_1|$ preferences of the first agent are from V_1, ranked in some arbitrary order. Note that in the last column the top cell is V, the next cell down is R and the third cell down is q. This means that the last agent ranks all members of V over all alternatives that are not members of V; ranks all members of $V \cup R$ over all alternatives that are not members of $V \cup R$, and finally ranks all members of $V \cup R \cup \{q\}$ over all alternatives that are not members of $V \cup R \cup \{q\}$.

Let $\mathfrak{S} = \{s_1, s_2, \ldots, s_n\}$ be a vector representing a minimal set of swaps that make q a Condorcet winner. That is, s_i is the number of swaps applied to the i^{th} agent's preference list, and we minimise the total number $\Sigma\mathfrak{S}$ of swaps used. From Lemma 4.0.5, we can assume without loss of generality that all s_i swaps will be used to swap q up the preference list.

Let $D_{\mathfrak{S}}$ be the set of D_i which $\mathfrak{S}$ has swapped q over all elements of. For example. from the profile above, we see that if $s_1 \geq |D_1|$, then D_1 will be a member of $D_{\mathfrak{S}}$. Thus $|D_{\mathfrak{S}}|$ is the number of values of i for which $\mathfrak{S}$ has swapped q over all elements of D_i. We will show later, in (4) below, that if $\mathfrak{S}$ swaps q over part an element of D_i it will swap q over all elements of D_i.

Note that to make q a Condorcet Winner in the smallest number of swaps:

1. $\mathfrak{S}$ *will swap q over each element of V at least once.* Each element a of V is either in V_i or $\bar{V_i}$, so we see from the profile above that $\text{adv}(a, q) = 1$. Thus we need to swap q over each alternative of V at least once to make q a Condorcet winner.

2. $\mathfrak{S}$ *will not swap* q *over any* D_i *or* R_i*, except as needed to swap over elements of* V. All the elements of R and all D_i are ranked above q at most twice, so q already wins over these alternatives.

3. $\mathfrak{S}$ *will not swap over* R_i *for any* $i \in \{1, 2, \ldots, n\}$. One way to make q a Condorcet winner is to swap q to the top of the first n agents preferences. We see from the profile above that the set of agents ranked above q in the i^{th} agent's preference list is $D_i \cup V_i$. Thus swapping q to the top i^{th} agent's preference requires $\mathfrak{S}$ to swap q over $|D_i \cup V_i|$ other alternatives. We have defined D_i, V_i such that $|D_i| = (k^2 - 1)$ and $|V_i| \leq |V| = k$, so $|D_i \cup V_i| < 2k^2$. Thus swapping q over the first k agents will take less than $2k^3$ swaps. Thus $\mathfrak{S}$ will never include the $2k^3$ swaps required to swap over some whole R_i.

4. $\mathfrak{S}$ *will not swap* q *over only part of a* D_i or R_i. We have to swap q over all of a D_i or R_i to be able to swap over elements of V, and from (2) we only swap of elements of D and R as required to swap over elements of V.

5. *If* $\mathfrak{S}$ *swaps* q *over a* D_i*, then* $\mathfrak{S}$ *will swap* q *over at least one of, and no more than* n *of, the elements of* V_i. From (2), if $\mathfrak{S}$ does not swap over at least one element of V_i then the swaps over D_i were wasted. Also, $|V_i| \leq k$, so there are no more than k elements of V_i to swap over.

6. $|D_{\mathfrak{S}}| = \lfloor k^{-2}\Sigma\mathfrak{S} \rfloor$, as $\mathfrak{S}$ does not swap q over any elements of R, it will only swap q over alternatives in the first k agents. From (3) and (4), if $\mathfrak{S}$ swaps over an element of D_i it must also swap over the complete D_i and 1 to k members of V_i. For each i, the cardinality of $|D_i|$ is $k^2 - 1$. Thus if the number of swaps s_i applied to the i^{th} agents preference list is non-zero, we know

$$k^2 \leq s_i \leq k^2 - 1 + k.$$

For each $i \in \{1, 2, \ldots, k\}$ we have $s_i > 0$ if and only if $D_i \in D_{\mathfrak{S}}$. From (3) we know $\mathfrak{S}$ does not swap q over any elements of R, so for each $i \in \{k+1, k+2, \ldots, n\}$ we have $s_i = 0$. Thus,

$$|D_{\mathfrak{S}}|k^2 \leq \Sigma\mathfrak{S} \leq |D_{\mathfrak{S}}|(k^2 - 1 + k),$$

which simplifies to,

$$|D_{\mathfrak{S}}|k^2 \leq \Sigma\mathfrak{S} \leq |D_{\mathfrak{S}}|k^2 + |D_{\mathfrak{S}}|(k - 1),$$

and as $|D_{\mathfrak{S}}| \leq k$,

$$|D_{\mathfrak{S}}|k^2 \leq \Sigma\mathfrak{S} < |D_{\mathfrak{S}}|k^2 + k^2,$$

thus,

$$|D_{\mathfrak{S}}| \leq k^{-2}\Sigma\mathfrak{S} < |D_{\mathfrak{S}}| + 1,$$

and so,

$$|D_{\mathfrak{S}}| \quad = \quad \left\lfloor k^{-2}\Sigma\mathfrak{S} \right\rfloor$$

7. *The vertices corresponding to the members $D_{\mathfrak{S}}$ form a minimum dominating set, where we say that a vertex v_i corresponds to D_j if and only if $i = j$.* To make q a Condorcet winner we must swap q over every element in V at least once. Each element in V is a vertex. If $D_{\mathfrak{S}}$ is fixed, we can only swap over a vertex $v_j \in V$ if and only if there is a $D_i \in D_{\mathfrak{S}}$ such that v_j is in V_i, i.e. v_j is adjacent to v_i. Thus the vertices corresponding to the elements of $D_{\mathfrak{S}}$ form a dominating set. As $|D_{\mathfrak{S}}| = \lfloor k^{-2}\Sigma\mathfrak{S} \rfloor$ minimising $\Sigma\mathfrak{S}$ will minimise $|D_{\mathfrak{S}}|$ and so $D_{\mathfrak{S}}$ is minimal.

8. Thus $\mathrm{Sc}_{\mathbf{D}}(q)$, the minimal value of $\Sigma\mathfrak{S}$, satisfies $|D_S| = \lfloor k^{-2}\mathrm{Sc}_{\mathbf{D}}(q) \rfloor$.

As the size of the profile we have generated is only $\Theta(k^5)$, this profile can be constructed in time which is polynomial with respect to k, and as shown above, the size of the minimum dominating set is $\lfloor sk^{-2} \rfloor$, where s is the Dodgson score.

□

Note 6.2.4 *We have given a reduction of a W[2]-hard problem (Dominating set) to Dodgson score. It would be intuitive to believe that this means that Dodgson score is W[2]-hard. However this reduction transforms the parameter $m(G)$ into the rather uninteresting parameter $\lfloor n^{-2}Sc_{\mathbf{D}}(q) \rfloor$. The Dodgson score is Fixed Parameter Tractable (FPT) with respect to the more interesting parameter m, and so is not W[2]-hard with respect to this parameter. It is unknown whether Dodgson score is W[2]-hard with respect to the parameters n and $Sc_{\mathbf{D}}(q)$.*

Definition 6.2.5 *Let f be a function $f\colon X \to \mathbb{R}^+$, and g be a function $f\colon X \times \mathbb{R}^+ \to \mathbb{R}^+$. We say g is a **Polynomial Time Approximation Scheme (PTAS)** for f if for all $x \in X$ and $\epsilon \in \mathbb{R}^+$ the equality $f(x)(1-\epsilon) \leq f(x) \leq g(x)(1+\epsilon)$ holds and there is an algorithm that can find $g(x, \epsilon)$ in polynomial time for fixed ϵ.*

Algorithm 3 k-Dominating Set(V,E)

```
1: s ← k + 1
2: for all |V|^k choices for a sequence of vertices v_1, v_2, ..., v_k do
3:    if v_1, v_2, ..., v_k form a dominating set then
4:       s ← min(s, number of unique vertices in v_1, v_2, ..., v_k)
5:    end if
6: end for
```

Lemma 6.2.6 *There is no PTAS for Dominating Set unless W[2]=FPT.*

Proof. As shown by Downey and Fellows (1995), Dominating Set is W[2]-complete. As shown by Bazgan (1995) PTAS is in FPT. Hence there is no PTAS for Dominating Set unless $W[2] = FPT$. □

Theorem 6.2.7 *Dodgson Score does not admit a Polynomial Time Approximation Scheme (PTAS).*

Proof. Assume there is a PTAS for Dodgson Score. Now we will show there must also be a PTAS for Dominating Set.

For any real number ϵ greater than zero we may choose an integer k such that $k > \lceil \epsilon^{-1} \rceil$. Let us define $m(G)$ as the size of the minimum dominating set in a graph G.

Case 1: $m(G) \leq k$. We may first use Algorithm 3, to determine whether $m(G) \leq k$ and if so the exact value for $m(G)$, in polynomial time for fixed k. If we have not yet found $m(G)$ then $m(G) > k$.

Case 2: From Lemma 6.2.3 We may construct a profile P with an alternative q such that $m(G) = \left\lfloor \frac{\mathrm{Sc}_{\mathbf{D}}(q)}{|V|^2} \right\rfloor$. We have assumed that there is a PTAS for Dodgson Score. Thus we may find in polynomial time, for any fixed $\epsilon > 0$, a number D such that $D(1-\epsilon) < Sc_{\mathbf{d}}(q) < D(1+\epsilon)$. Since, in Case 2, we have $m(G) > k$,

$$\left\lfloor \frac{\mathrm{Sc}_{\mathbf{D}}(q)}{|V|^2} \right\rfloor \quad > \quad k.$$

Thus, $Sc_{\mathbf{d}}(q) > |V|^2 k$. We let $\bar{D} = \max(D, |V|^2 k)$. Thus $\bar{D} > |V|^2 k$, and so,

$$\frac{\bar{D}}{|V|^2} \quad > \quad k = \left\lceil \frac{1}{\epsilon} \right\rceil.$$

It follows that $\bar{D}/|V|^2 > 1/\epsilon$, and by taking the reciprocal of each side we get $|V|^2/\bar{D} < \epsilon$. As

$$m(G) \quad = \quad \left\lfloor \frac{\mathrm{Sc}_{\mathbf{D}}(q)}{|V|^2} \right\rfloor,$$

we have the inequality

$$\frac{\mathrm{Sc}_{\mathbf{D}}(q)}{|V|^2} - 1 \;\leq\; m(G) \;\leq\; \frac{\mathrm{Sc}_{\mathbf{D}}(q)}{|V|^2}.$$

Using $D(1-\epsilon) < Sc_{\mathbf{d}}(q) < D(1+\epsilon)$, the inequalities above become

$$\frac{D(1-\epsilon)}{|V|^2} - 1 \;\leq\; m(G) \;\leq\; \frac{D(1+\epsilon)}{|V|^2}.$$

We now wish to show that a similar inequality holds for $\bar{D}$:

$$\frac{\bar{D}(1-\epsilon)}{|V|^2} - 1 \;\leq\; m(G).$$

As $\bar{D} = \max(D, |V|^2 k)$, either $\bar{D} = D$ or $\bar{D} = |V|^2 k$. If $\bar{D} = D$ the above inequality clearly follows from the previous inequalities. Suppose $\bar{D} = |V|^2 k$. We know $m(G) > k$. Thus

$$\frac{\bar{D}}{|V|^2} = k \;<\; m(G).$$

It follows that

$$\frac{\bar{D}(1-\epsilon)}{|V|^2} - 1 \;<\; m(G).$$

Thus we have shown that the following inequality holds whether $\bar{D} = D$ or $\bar{D} = |V|^2 k$:

$$\frac{\bar{D}(1-\epsilon)}{|V|^2} - 1 \;\leq\; m(G) \;\leq\; \frac{D(1+\epsilon)}{|V|^2},$$

rearranging the left side,

$$\frac{\bar{D}(1-\epsilon) - |V|^2}{|V|^2} \;\leq\; m(G) \;\leq\; \frac{D(1+\epsilon)}{|V|^2},$$

and again,

$$\frac{\bar{D}\left(1 - \epsilon - \frac{|V|^2}{\bar{D}}\right)}{|V|^2} \;\leq\; m(G) \;\leq\; \frac{D(1+\epsilon)}{|V|^2},$$

recall that $|V|^2/\bar{D} < \epsilon$, and so

$$\frac{\bar{D}(1-2\epsilon)}{|V|^2} \;<\; m(G) \;\leq\; \frac{D(1+\epsilon)}{|V|^2},$$

Table 6.1: Time in Milliseconds to Compute the Dodgson Winner (#Alter/#Voter,b=0)

alternatives

a\v	3	5	7	9	15	25	85
3	1.270	1.400	1.310	1.460	1.465	1.805	1.483
5	1.479	1.579	1.675	1.747	2.196	2.137	2.795
7	1.743	1.901	1.987	2.030	2.206	2.799	6.607
9	1.816	2.006	2.171	2.555	2.717	3.952	11.631
15	2.528	2.897	3.666	4.087	5.920	8.468	33.316
25	3.668	5.346	8.097	8.804	13.978	22.431	102.189

Agents

Table 6.2: Time in Milliseconds to Compute the Dodgson Winner (5 alternatives,b=0)

a\v	3	5	9	17	33	65	129	257	513	1025
5	2.223	2.314	2.921	2.769	3.184	4.450	5.787	8.185	12.592	20.564

finally, $D \leq \bar{D}$, so

$$\frac{\bar{D}(1-2\epsilon)}{|V|^2} \quad < \quad m(G) \quad < \quad \frac{\bar{D}(1+2\epsilon)}{|V|^2},$$

Thus we can find $m(G)$ to within our desired error of 2ϵ in polynomial time, and so we have a PTAS for Dominating Set. But from Lemma 6.2.6 Dominating Set does not admit a PTAS unless FPT=W[2]. Hence, by contradiction, Dodgson score does not admit a PTAS, unless FPT=W[2]. □

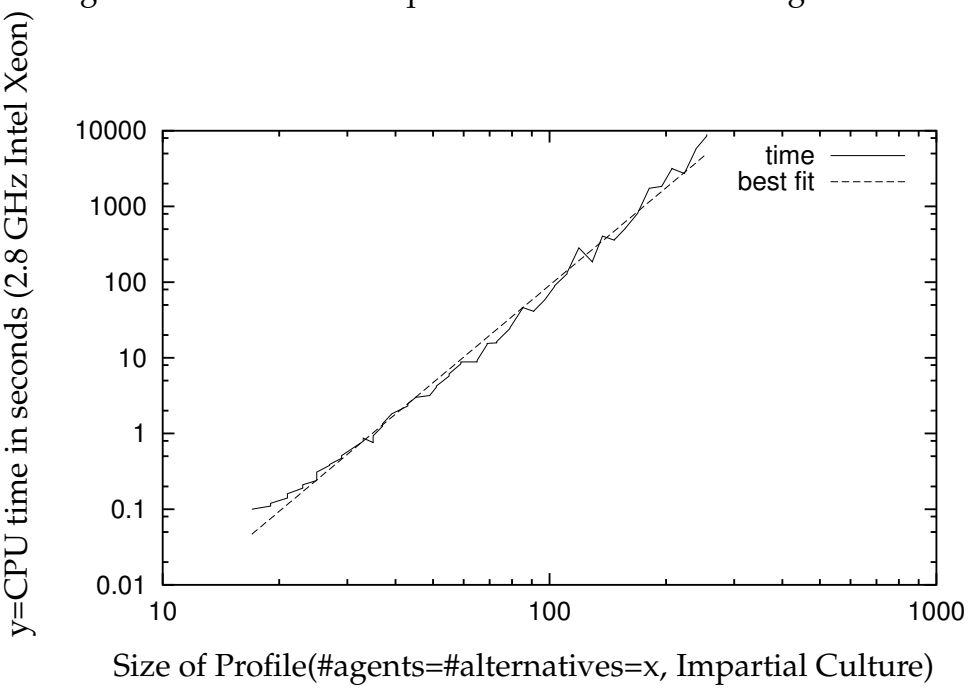
6.3 Empirical Performance of Dodgson's Rule.

Let t be CPU time and s be size. We use time $= e^a s^b$ as the formula for best fit. The best fit line for Figure 6.1 has $a = -15.177$, $b = 4.275$. These values are calculated using the gnuplot fit function. For other populations other than impartial culture, the graph looks much the same, but the best fit parameters a and b change a little.

Table 6.3: CPU Time in Seconds to Calculate the Dodgson Winner in Impartial Culture for Square Profile ($n = m = s$).

Size s	5	15	25	35	45	55	65	85	111	195	255
CPU Time	0.001	0.007	0.024	0.076	0.3	0.613	0.923	446	12.9	185.5	853

Figure 6.1: CPU Time Required to Calculate the Dodgson Winner for 10 Profiles



In the final row of the table we consider even sizes, all other rows consider only odd sizes. For most of the rows of the table, the number of agents and number of alternatives are both equal to the "size" s of the profile. In one case row we have used three times more agents than alternatives; in the row underneath we have set the number of agents to be the number of alternatives squared. The Homogeneity is the parameter of Homogeneity b defined in Section 1.2.6

Homogeneity	sizes: s	#Voters	#alternatives	a	b
0	odd	s	s	-15.18	4.275
0.1	odd	s	s	-13.36	3.717
0.5	odd	s	s	-14.11	3.822
2	odd	s	s	-14.3	3.877
0	odd	3s	s	-13.90	4.36
0	odd	s^2	s	-16.89	6.592
0	even	s	s	-15.08	4.262

From this table it would appear that the expected running time of the Dodgson winner is roughly $\mathcal{O}\left(\max(m,n)^4\right)$. However it is likely that rare occurrences of infeasible problems would affect the overall running time of the algorithm.

Chapter 7

Conclusion

In Chapter 1 we discussed the history of voting theory and various important theorems. One of these theorems, the M^{c}Garvey theorem is very useful in the study of C1 voting rules. Unfortunately this theorem was of no use to us since we consider only C2 and C3 rules.

In Chapter 2 we presented a proof of a M^{c}Garvey theorem for weighted tournaments; we proved that a weighted tournament is the weighted majority relation of some society if and only if all the weights have the same parity. Although this result was probably originally proven by Debord (1987), we presented an independent proof as we did not have access to Debord's thesis. This generalisation was very useful in simplifying the proofs in the next chapter. It appears that this generalisation is as useful in the study of C2 voting rules as the original M^{c}Garvey theorem is in studying C1 voting rules.

In Chapter 3 we presented our new approximation, Dodgson Quick, and proved various results regarding rules which approximate the Dodgson rule, particularly when the number of voters become large. Under the assumption that all elections are equally likely, we proved that for both our Dodgson Quick approximation and Tideman's approximation, the probability that the approximation chooses the Dodgson winner approaches one as the number of voters tends to infinity. In practice this algorithm is very effective for moderately large numbers of agents n under the impartial culture assumption, picking the correct winner in all 1,000,000 random simulations with 5 alternates, 85 agents.

However, out of these two approximations, only our approximation converges exponentially fast. As this approximation can be computed in a fixed number of additions for fixed m, it can be computed in $\mathcal{O}(\ln n)$ time for fixed m. We used this to show that we may find the Dodgson winner in $\mathcal{O}(\ln n)$ time for a fixed number of alternatives.

We also studied the asymptotic behaviour of Simpson's rule[1] as we increase the number of voters. Although Simpson's rule does not converge to the Dodgson rule as the number

[1] Simpson's rule is also known as the Minimax/Maximin rule.

of voters increases, it does come quite close for a small number of candidates.

In Chapter 4 we presented an ILP that had no more than $(m-1)!e$ variables, and can be used to find the Dodgson score. This is an improvement of order m^2 over the ILP suggested by Bartholdi et al. (1989), for which the upper bound was $m!m$. The number of variables is the primary factor in the theoretical worst case performance of algorithms for solving ILPs.

In Chapter 5 we proposed another new rule, the Dodgson Relaxed and Rounded (R&R) rule. Unlike the previously considered approximations, this rule is similar not only in outcomes but also in definition. We suggest that this rule could be considered superior to the original, since in practice the primary difference between the original Dodgson rule and the Dodgson relaxed rule is that relaxed rule allows us to break ties in favor of candidates that are in some sense fractionally better than the other candidates that would have be tied under the original rule. If we do not favor candidates that are fractionally better, i.e. if we round up the Dodgson Relaxed scores, then the Dodgson Relaxed winner would be the Dodgson winner in all 43 million simulations except one. This simulation had 25 candidates and 5 agents, an unrealistic size for real world election. Also this approximation is far closer than the other approximations studied under this assumption. By comparison, the Dodgson Quick and Tideman approximations only picked the correct winner in 87% of random simulations with 25 candidates and 5 agents generated under the impartial culture assumption.

Unlike the Dodgson winner, the R&R winner can be computed in polynomial time, using at most $\mathcal{O}\left(m^4n^4\ln(mn)\right)$ arithmetic operations on $\mathcal{O}\left(mn\ln(mn)\right)$ bit numbers.

It is interesting to note that each of the rules we studied provided a successively tighter lower bound to the Dodgson rule, i.e. given any profile $\mathcal{P}$ and alternative a,

$$1/2\mathrm{Sc}_{\mathbf{S}}(a) \leq 1/2\mathrm{Sc}_{\mathbf{T}}(a) \leq \mathrm{Sc}_{\mathbf{Q}}(a) \leq \mathrm{Sc}_{\mathbf{R}}(a) \leq \mathrm{Sc}_{\&}(a) \leq \mathrm{Sc}_{\mathbf{D}}(a).$$

The approximation independently discovered by Homan and Hemaspaandra (2005) does not fit on this hierarchy, as its score is not a lower bound (or upper bound) for the Dodgson score.

In Chapter 6 we discuss the parametrized complexity of Dodgson's rule. For a fixed number of alternatives, we may compute the Dodgson winner in $\mathcal{O}(\ln n)$ operations of $\mathcal{O}(\ln n)$ bits in size. That is, there is some real valued function f such that the number of operations required is of order $\mathcal{O}(f(m)\ln(n))$. Unfortunately, the theoretical bound on the complexity may grow very quickly with the number of alternatives; it may become infeasible with only 4 alternatives. Furthermore we show that there exists no constant approximation scheme for the Dodgson score unless P=NP, and no Polynomial Time Approximation Scheme (PTAS) unless W[2]=FPT.

In practice the Dodgson winner turns out to be easy to compute for the small numbers of candidates and alternatives that are studied by voting theorists. When studying voting rules numerically, voting theorists typically limit themselves to 25 candidates and 100 agents (see e.g. M^{c}Cabe-Dansted and Slinko 2006; Shah 2003; Nurmi 1983).

It appears that except for a very few tricky cases the Dodgson winner is easy to compute. In most elections, we may verify that our Dodgson Quick scores match the scores of the original Dodgson rule; this is how we proved that the expected time required to find the Dodgson winner is $\mathcal{O}(\ln n)$ for fixed m. As we discussed in Chapter 4 we may formulate the Dodgson rule as an Integer Linear Problem (ILP). As discussed by Chapter 5, the optimum value of the ILP is almost always the same as the optimal value of the LP. If we know that all variables in the optimum solution of the LP are integers we can clearly find an optimum solution to the ILP using an LP algorithm, such algorithms can run in polynomial time. Our numerical results seemed to suggest that the expected running time of the Dodgson rule was polynomial, and of a similar order of that required to solve an LP.

7.1 Methodological Issues

To get the numerical results in Section 3.3 and Appendix B.2.3, we had to simulate over one million elections: each simulation requiring up to a million arithmetic operations. Most of these operations were simple integer additions and subtractions. The time taken by our original 100% pure MATLAB code was too slow to achieve our aims. We found that C code was more effective for these types of calculations, running roughly a thousand times faster. An additional problem we found with MATLAB is that even if MATLAB is installed on a target machine, it may not have the power-boxes required for what we want to do. Where statistical functions are required, we are usually able to use open source and free software C libraries such as GLPK linear problem solver, or the open source statistical language R[2]. This allows us to run computationally expensive code on any machine, without worrying which MATLAB modules, if any, were licensed for use on that machine.

In our areas of interest, we found that R was more powerful than MATLAB, even if all relevant MATLAB power-boxes were installed. The downside of this approach is that R and C are less user-friendly than MATLAB. Also when writing our paper (M^{c}Cabe-Dansted and Slinko, 2006) which surveyed many different voting rules in addition to the ones we study in this thesis, we had access to preexisting MATLAB scripts implementing most of these voting rules. Rather than reimplement all these rules we used a hybrid

[2]Interestingly, R was initially written by Ross Ihaka and Robert Gentleman at Auckland University.

MATLAB and C approach. We added a MATLAB interface to the C code we use to compute the Dodgson and Kemeny winners, so that this code can be called from MATLAB.

Another difficulty we discovered is that when writing a document of this size it can be common to make the same grammatical and stylistic mistakes again and again. Although it is unlikely that this problem is limited to the field of voting theory, some of the particular mistakes are. The tool chosen to write this thesis was LyX, a front end to the LaTeX typesetting system. LyX, did not have support for any kind of grammar checker let alone one able to detect mistakes specific to the field of voting theory. For this reason we considered it worthwhile to investigate whether it would be possible to develop a grammar checker for LyX that was able to detect errors specific to voting theory as well as more general errors. It turns out that it is in fact possible to develop and add such functionality (McCabe-Dansted, 2005). This grammar checker found a number of mistakes that were not detected through proofreading.

7.2 Further Work

The McGarvey Theorem (1953) states that every tournament can be represented as a majority relation for a certain society of voters. We have proved a generalization of the McGarvey Theorem (1953) to weighted tournaments. McGarvey's work was refined by Stearns (1959) and Erdös and Moser (1964), who managed to find a tight bound on the number of voters required. It would be interesting to refine our theorem in a similar way.

We used the Chernoff theorem (1952) to prove that our approximation converged at an exponential rate to the to the Dodgson rule. Homan and Hemaspaandra (2005) instead used a variant of the Chernoff theorem (Alon and Spencer, 2000) which allowed them to also produce a specific upper bound on the frequency that their approximation differs from the Dodgson rule. It would be possible to give a similar result for our approximation using this variant of the Chernoff theorem.

The algorithm we use to compute the exact Dodgson score does not make use of the fact that we know that the Dodgson score is exactly equal to the Dodgson Quick score if the profile satisfies Tideman's criteria [T2] (page 12). Although we found that our existing algorithm typically provided good performance for reasonable profile sizes, we could make the expected running time of our algorithm $\mathcal{O}(\ln n)$ for n voters and a fixed number of alternatives (see Corollary 3.1.10). This would make the time required to compute the Dodgson winner grow very slowly as we increase the number of voters towards infinity.

We mentioned that Berg (1985) suggested that the statistical behaviour of voting rules should be studied under different assumptions on voting behaviour. Although we have done some exploratory studies of the approximability of Dodgson's rule under the Pòlya-

Eggenberger urn model in our paper (M^{c}Cabe-Dansted and Slinko, 2006), we have only done an in depth theoretical study into the approximability of Dodgson's rule under the impartial culture assumption. It would be interesting to discover whether our approximation still asymptotically converges to the Dodgson rule exponentially fast, or at all, as we increase the number of voters for other assumptions of voter behaviour, such as the Pòlya-Eggenberger urn model.

We suspect the Dodgson relaxed and rounded (R&R) winner will never differ from the Dodgson winner if the number of alternatives is small. It would be interesting to discover the smallest number of alternatives for which it is possible for the R&R winner to differ from the Dodgson winner, and to prove that for any profile with less alternatives the R&R winner will always be the Dodgson winner.

We have found that the Dodgson winner is FPT with respect to the number of alternatives m (i.e. there exists a function f and polynomial P such that finding the Dodgson winner is a $\mathcal{O}(f(m)P(n))$ time problem). It would be interesting to know whether finding the Dodgson winner is FPT with respect to the number agents n, or the Dodgson score of the winner.

The Kemeny rule can be defined in a similar way to the Dodgson rule, except that it chooses a ranking of all the candidates instead of just selecting a winner. Many papers that study the Dodgson rule also study the Kemeny, e.g. the paper by Bartholdi et al. (1989) that we cite so frequently inthis thesis. We have[3] some preliminary results, e.g. that we can compute the Kemeny ranking (if only strict preferences are allowed) from a profile in $\mathcal{O}(m!n)$ time from a profile and $\mathcal{O}[(m!)^2 \ln n]$ time from a voting situation. This is better than the similar bounds we found for the Dodgson rule. However, unlike the Dodgson rule, were unable to find an algorithm that could find the Kemeny ranking for a dozen candidates in a reasonable amount of time. This is not so surprising from a theoretical point of view, as Bartholdi et al. showed that we can compute the Dodgson winner in polynomial time for any fixed number of agents, where as Dwork et al. (2001) showed that finding the Kemeny ranking is NP-hard even if we fix the number of agents to be only 4. Also, although Dwork et al. discussed various heuristics for finding the Kemeny rule, it does not appear that these are as effective as our approximations to the Dodgson rule. It would be interesting to produce a full comparison of the results regarding the Dodgson rule, including those in this thesis, with the corresponding results for the Kemeny rule.

In the production of this thesis, a considerable amount of software was written and also reused from previous researchers in the field. This software could be of assistance to those who would continue work in this area. See `http://dansted.org/thesis06/` for files relating to the thesis that have been omitted from the main text. Also feel free

[3]These preliminary results are available at `http://dansted.org/papers/paper_kemeny.pdf`.

to contact the author at `gmatht@gmail.com` as to the use of these files and for more experimental data.

Appendix A

Preliminary Mathematics

A.1 Probability Space

A **Probability space** is a triple $(\mathcal{U}, \mathcal{F}, m)$, where $\mathcal{U}$ is the set of outcomes, $\mathcal{F}$ is the set of events and m is a probability measure that assigns probabilities to events (Durret, 2005, p1). In Durret's definition $\mathcal{F}$ can be any σ-field on $\mathcal{U}$; for simplicity will let $\mathcal{F}$ be the **Borel sets** of $\mathcal{U}$, the closure of the open sets of $\mathcal{U}$ under countable unions and intersections. As $\mathcal{F}$ is fully determined by $\mathcal{U}$, we may consider a probability space to be a pair $(\mathcal{U}, m)$. For discrete probability spaces the Borel sets of $\mathcal{U}$ are $2^{\mathcal{U}}$, the set of all subsets of $\mathcal{U}$, so $\mathcal{F} = 2^{\mathcal{U}}$. The function m is a **probability measure** if

1. $m(\mathcal{U}) = 1 \geq m(A) \geq m(\emptyset) = 0$, and

2. For a countable sequence of disjoint events $\mathcal{A}_1, \mathcal{A}_2, \ldots$

$$m(\cup_i \mathcal{A}_i) \quad = \quad \Sigma_i m(\mathcal{A}_i).$$

Where u is the **actual outcome**, and p is a statement that is true if and only if $u \in E$, we define the probability $P(p)$ as $m(E)$. We define a **random vector** X as a function $X : \mathcal{U} \to \mathbb{R}^n$ of u. A **random variable** is 1-dimensional random vector. For example, if we roll a normal six-sided dice we have

$$\begin{aligned} \mathcal{U} &= \{1, 2, 3, 4, 5, 6\}, \\ m(E) &= {}^{|E|}/_{|\mathcal{U}|}, \\ X > 4 &\iff X \in \{5, 6\}, \\ X &= u, \end{aligned}$$

so

$$\begin{aligned} P(X > 4) &= P(u \in \{5, 6\}) \\ &= m(\{5, 6\}) \\ &= {}^2\!/_6. \end{aligned}$$

The definition of a probability measure above gives us the axioms of probability. Given any two statements p and q on which $P(p)$ and $P(q)$ are defined,

1. $0 \leq P(p) \leq 1$
2. $P(\text{false}) = 0$, $P(\text{true}) = 1$.
3. $P(p \vee q) = P(p) + P(q) - P(p \wedge q)$.

We may also derive the following laws of probability,

1. $P(\neg p) = 1 - P(p)$.
2. $P(p_1 \vee p_2 \vee \cdots \vee p_i) \leq \sum_{j=1}^{i} P(p_j)$.

The **conditional probability** of p given q (see e.g. Kemeny et al., 1974, p 97) is

$$P(p|q) = \frac{P(p \wedge q)}{P(q)},$$

where $P(q) > 0$. For joint continuous distributions when it may happen that $P(q) = 0$, we use the following definition (see e.g. Anderson, 1984, p12),

$$\begin{aligned} P(x_1 \leq |X| \leq x_2 \mid Y = y) &= \lim_{h \to 0} P(x_1 \leq |X| \leq x_2 \mid y \leq Y \leq y + h) \\ &= \int_{x_1}^{x_2} f(u|y) du, \end{aligned}$$

where

$$\begin{aligned} f(u|y) &= f(u, y)/g(y), \\ g(y) &= \frac{\partial F(\infty, y)}{\partial y}, \\ f(x, y) &= \frac{\partial^2 F(x, y)}{\partial x \partial y}, \\ F(x, y) &= P(X \leq x \wedge Y \leq y). \end{aligned}$$

We define the **product space** $(\mathcal{U}_1, m_1) \times (\mathcal{U}_2, m_2)$ of two probability spaces as follows

$$(\mathcal{U}_1, m_1) \times (\mathcal{U}_2, m_2) \quad = \quad (\mathcal{U}, m),$$

where

$$\begin{aligned} \mathcal{U} &= \{(u_1, u_2) : u_1 \in \mathcal{U}_1 \wedge u_2 \in \mathcal{U}_2\}, \\ m(A_1, A_2) &= m_1(A_1) \times m_2(A_2). \end{aligned}$$

We define the n^{th} power $(\mathcal{U}, m)^n$ of a probability space similarly.

Sometimes we may wish to simplify a probability space by discarding some information about the actual outcome. For example, we may consider the outcomes $\mathcal{U}$ when rolling a six sided die to be $\{1, 2, 3, 4, 5, 6\}$. However we may only be interested in the parity of the result, in which case we may want to treat $\mathcal{U}$ as $\{\text{odds}, \text{evens}\}$.

Let $\hat{\mathcal{U}}$ be a partition of $\mathcal{U}$ such that each member of $\hat{\mathcal{U}}$ is a member of the events $\mathcal{F}$. Let $\hat{m}$ be a probability measure such that for each x in $\mathcal{U}$ we have $\hat{m}(x) = m(\{x\})$. Then we call $(\hat{\mathcal{U}}, \hat{m})$ a **simplified probability space** of $(\mathcal{U}, m)$.

For example, let $\mathcal{U} = \{1, 2, 3, 4, 5, 6\}$ and $m(S) = \frac{1}{6}|S|$ for all S in $\mathcal{F}$. Let $\hat{\mathcal{U}}$ be the set $\{\{1, 3, 5\}, \{2, 4, 6\}\}$. Let $\hat{m}$ be a probability measure where $\hat{m}(\{\{1, 3, 5\}\}) = m(\{1, 3, 5\}) = 1/2$ and $\hat{m}(\{\{2, 4, 6\}\}) = m(\{2, 4, 6\}) = 1/2$, i.e. $\hat{m}(S) = \frac{1}{2}|S|$. Then probability space $(\hat{\mathcal{U}}, m)$ is a simplified probability space of $(\mathcal{U}, m)$.

We may also want to relabel the elements of $\hat{\mathcal{U}}$, for example relabeling $\{1, 3, 5\}$ as odds and $\{2, 4, 6\}$ as evens. In this case we have $\hat{\mathcal{U}} = \{\text{odds}, \text{evens}\}$ and $\hat{m}(\{\text{odds}\}) = m(\{1, 3, 5\}) = \frac{1}{2} = m(\{2, 4, 6\}) = \hat{m}(\{\text{evens}\})$.

A.2 Binomial Distribution

We define a (p)-**Bernoulli trial** as a probability space $(\mathcal{U}, m)$, where $\mathcal{U}$=$\{\text{success}, \text{failure}\}$ and $m(\{\text{success}\}) = p$. For convenience we denote $q = m(\{\text{failure}\}) = 1 - p$. We define an (n, p)-**binomial probability space** as the product space of n distinct (p)-Bernoulli trials. We call the random variable X that represents the number of trials whose actual outcome is "success" an (n, p)-**binomially distributed random variable**, and we call the distribution of this random variable an (n, p)-**binomial distribution**.

It is well known that:

1. $E[X] = np$,
2. $\text{var}(X) = npq$.

The probability of x successes in n trials is

$$P(X = x) = \frac{n!}{x!(n-x)!}p^x q^{n-x},$$

where $x \in \mathbb{Z} \cap [0, n]$ (see e.g. Walpole and Myers 1993).

Lemma A.2.1 *For all positive integers n, the number of ways of choosing n objects from $2n$, is at least $\frac{4^n}{2\sqrt{n}}$, i.e.*

$$\binom{2n}{n} \geq \frac{4^n}{2\sqrt{n}}.$$

Proof. Clearly for $n = 1$ the lemma holds.

$$\binom{2}{1} = 2 \geq \frac{4}{2\sqrt{1}}.$$

If the lemma holds for some n then

$$\begin{aligned}
\binom{2(n+1)}{n+1} &= \frac{(2n+2)(2n+1)}{(n+1)(n+1)}\binom{2n}{n} \\
&= \frac{4(n+\frac{1}{2})}{(n+1)}\binom{2n}{n} \\
&\geq \frac{4(n+\frac{1}{2})}{(n+1)}\frac{4^n}{2\sqrt{n}} \\
&= \frac{(n+\frac{1}{2})}{\sqrt{n(n+1)}}\frac{4^{n+1}}{2\sqrt{n+1}} \\
&> \frac{4^{n+1}}{2\sqrt{n+1}},
\end{aligned}$$

since

$$\frac{(n+\frac{1}{2})}{\sqrt{n(n+1)}} = \sqrt{\frac{n^2+n+\frac{1}{4}}{n^2+n+0}} > 1.$$

By induction this lemma holds for all n. □

Note A.2.2 *Stirling's formula implies $\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi n}}$, however the result above provides a lower bound for $\binom{2n}{n}$. By starting the induction where $n \gg 1$ rather than $n = 1$, we may get a lower bound exceedingly close to $\frac{4^n}{\sqrt{\pi n}}$. As we are only interested in rates of convergence, the bound provided above will suffice.*

Corollary A.2.3 *A $(2n, 0.5)$-binomial random variable X has a probability of at least $\frac{1}{\sqrt{2n}}$ of equaling n where $n > 0$, i.e.*

$$P(X = n) \geq \frac{1}{2\sqrt{n}}.$$

Lemma A.2.4 *A $(2n, 0.5)$-binomial random variable X has at least a probability of 0.5 of being even.*

Proof. Let P_i be the probability that there has been an even number of successes in the first i samples. Clearly $P_0 = 1 \geq 0.5$. Assume that $P_i \geq 0.5$.

$$\begin{aligned} P_{i+1} &= (1-p)P_i + p(1-P_i) \\ &= (1-2p)P_i + p, \end{aligned}$$

so

$$\begin{aligned} P_{i+2} &= (1-2p)[(1-2p)P_i + p] + p \\ &= P_i - 4pP_i + 4p^2P_i + p - 2p^2 + p, \end{aligned}$$

by replacing P_{i+2} with $(P_{i+2} - 0.5) + 0.5$ and subtracting 0.5 from both sides we get

$$\begin{aligned} P_{i+2} - 0.5 &= (P_i - 0.5) + 0.5 - 4p(P_i - 0.5) - 2p + 4p^2(P_i - 0.5) + 2p^2 + 2p - 2p^2 - 0.5 \\ &= (P_i - 0.5) - 4p(P_i - 0.5) + 4p^2(P_i - 0.5) \\ &= (P_i - 0.5)(1-2p)^2 \geq 0. \end{aligned}$$

Hence by induction $P_i \geq 0.5$ for all even i. □

A.3 Multinomial Distribution

The **multinomial distribution** is a generalisation of the binomial distribution.

Let k be a positive integer and $\mathbf{p} = (p_1, p_2, \ldots, p_k)$ be a k-dimensional vector such $\sum_{i=1}^{k} p_i = 1$ and p_i is non-negative for all i. We define a $(k, \mathbf{p})$-**multinomial trial** as an $(\mathcal{U}, m)$-probability space where $\mathcal{U} = \{1, 2, \ldots, k\}$ and $m(\{i\}) = p_i$ for all $i \in \mathcal{U}$. For convenience, we define q_i to be $1 - p_i$, the probability that the actual outcome is not i. We define an $(n, k, \mathbf{p})$-**multinomial probability space** as the product space of n distinct $(k, \mathbf{p})$-multinomial trials.

The actual outcome $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ in the outcomes $\mathcal{U}^n$ of an $(n, k, \mathbf{p})$-multinomial probability space is an n-dimensional vector where u_i is in $\mathcal{U}$ for each i. We define a random vector X so that for each i the value of X_i is the number of values of j for which u_j equals i. Informally, this means that X_i is the number of multinomial trials that resulted in outcome i. We call X an $(n, k, \mathbf{p})$-**multinomially distributed random vector**.

It is well known that:

1. $E[X_i] = np_i$,
2. $\text{var}(X_i) = np_iq_i$,
3. $\text{cov}(X_i, X_j) = -np_ip_j$.

The probability that actual outcome i occurs x_i times for each i in $\{1, 2, \ldots, k\}$ is

$$P(X = \mathbf{x}) \quad = \quad \frac{n!}{x_1!x_2!\ldots x_k!} p_1^{x_1} p_2^{x_2} \ldots p_k^{x_k},$$

where $x_1 + x_2 + \cdots + x_k = n$.

Let $\mathbf{p} = (p_1, p_2)$ with p_1 in the range $[0, 1]$ and $p_2 = 1 - p_1$. A $(2, \mathbf{p})$-multinomial trial has two outcomes, 1 and 2. Where (1,2)=(success,failure), the trial is a $(\mathbf{p})$-Bernoulli trial. Given a random vector X distributed according to an $(n, 2, \mathbf{p})$-multinomial distribution, X_1 is distributed according to an (n, p_1)-binomial distribution.

Lemma A.3.1 *Let X be a random vector distributed according to an $(n, k, \mathbf{p})$-multinomial distribution. We partition the set of elementary outcomes $\{1, 2, \ldots, k\}$ of each $(k, \mathbf{p})$-multinomial trial into l sets $S_1, S_2, \ldots, S_l$. Let $\hat{\mathbf{p}}$ be an l-dimensional vector such that $\hat{p}_i = \sum_{j \in S_i} p_j$. Let $\mathbf{y}$ be an l-dimensional vector such that $y_i = \sum_{j \in S_i} x_j$. Then $\mathbf{y}$ is a random vector distributed according to an $(n, l, \hat{\mathbf{p}})$-multinomial distribution.*

Proof. We define a simplified probability space $(\hat{\mathcal{U}}, \hat{m})$ of the multinomial trial $(\mathcal{U}, m)$ such that the elementary outcome i of $(\hat{\mathcal{U}}, \hat{m})$ corresponds to the event S_i, i.e. the actual outcome of $(\hat{\mathcal{U}}, \hat{m})$ is i if an only if the actual outcome of $(\mathcal{U}, m)$ is in the set S_i. Clearly $\hat{m}(\{i\}) = m(S_i) = \hat{\mathbf{p}}_i$. Note that $(\hat{\mathcal{U}}, \hat{m})$ is an $(l, \hat{\mathbf{p}})$-multinomial trial. The product of the n distinct $(l, \hat{p})$-multinomial trials is an $(n, l, \hat{\mathbf{p}})$-multinomial probability space. We note that the multinomially distributed random vector on this multinomial probability space is $\mathbf{y}$. Hence the result. □

For example, let $X = (X_1, X_2, X_3, X_4, X_5, X_6)^T$ be a random vector distributed according to an $(n, 6, \mathbf{p})$-multinomial distribution. Then $\mathcal{U} = \{1, 2, 3, 4, 5, 6\}$, let $l = 3$ so $\hat{\mathcal{U}} = \{1, 2, 3\}$. Let $S_1 = \{1, 2, 3\}$, $S_2 = \{4\}$, $S_5 = \{5, 6\}$. Then the vector $\hat{X} = (X_1 + X_2 +$

$X_3,\ X_4,\ X_5 + X_6)^T$ is a random vector distributed according to an $(n, 3, \hat{\mathbf{p}})$-multinomial distribution where $\hat{\mathbf{p}} = (p_1 + p_2 + p_3,\ p_4,\ p_5 + p_6)^T$.

Corollary A.3.2 *If X is an $(n, k, \mathbf{p})$-multinomially distributed random vector, and i is an integer in the range $[0, n]$, then X_i is an (n, p_i)-binomially distributed random variable.*

Corollary A.3.3 *If X is an $(n, k, \mathbf{p})$-multinomially distributed random vector, and i is an integer in the range $[0, n]$, then X_i has at least a probability of 0.5 of being even.binomially distributed random variable.*

Obvious from the fact that a $(2n, 0.5)$-binomial random variable X_i has at least a probability of 0.5 of being even (Lemma A.2.4).

Definition A.3.4 *Let $X = (X_1, X_2, \ldots, X_k)^T$ be a k-dimensional vector. Let n be an integer less than or equal to k. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a subset of $\{1, 2, \ldots, k\}$, the possible subscripts of X. We define the subvector X_I as follows:*

$$X_I \quad = \quad (X_{i_1}, X_{i_2}, \ldots, X_{i_n})^T .$$

Definition A.3.5 *We define the sum $\sum \mathbf{x}$ of a vector $\mathbf{x}$ to be the sum of x_i over all subscripts i.*

Lemma A.3.6 *Let X be a random vector distributed according to an $(n, k, \mathbf{p})$-multinomial distribution for some $(n, k, \mathbf{p})$. Let y be a non-negative integer less than or equal to n. Let X_A be a subvector of X, and $\bar{A}$ be the complement of A.*

Then, given that $\sum X_A = y$, the conditional probability distribution of X is that of two independent multinomial distributions; X_A is distributed according to a $(y, |A|, \mathbf{p}_A / \sum \mathbf{p}_A)$-multinomial distribution and $X_{\bar{A}}$ is distributed according to an $(n - y, k - |A|, \mathbf{p}_{\bar{A}} / \sum \mathbf{p}_{\bar{A}})$-multinomial distribution.

Proof. From the standard law of probability that

$$P(A|B) \quad = \quad \frac{P(A \wedge B)}{P(B)},$$

we know that

$$P(X = \mathbf{x} \mid \sum X_A = y) \quad = \quad \frac{P(X = \mathbf{x} \wedge \sum X_A = y)}{P(\sum X_A = y)}.$$

If $\sum \mathbf{x}_A \neq y$ and $X = \mathbf{x}$ then by substitution $\sum X_A \neq y$. Clearly if $\sum \mathbf{x}_A \neq y$ we have

$$P(X \quad = \quad \mathbf{x} \wedge \sum X_A = y) = 0,$$

and so

$$P(X = \mathbf{x} \mid \sum X_A = y) = 0.$$

We consider the case where $\sum \mathbf{x}_A = y$. Now, by substitution,

$$X = \mathbf{x} \implies \sum X_A = y,$$

and we have,

$$\frac{P(X = \mathbf{x} \wedge \sum X_A = y)}{P(\sum X_A = y)} = \frac{P(X = \mathbf{x})}{P(\sum X_A = y)}.$$

From Lemma A.3.1 $(\sum X_A, \sum X_{\bar{A}})$ is multinomially distributed. Thus $\sum X_A$ is binomially distributed and hence:

$$P(\sum X_A = y) = \frac{n!}{y!(n-y)!} \left(\sum \mathbf{p}_A\right)^y \left(\sum \mathbf{p}_{\bar{A}}\right)^{n-y},$$

and $P(X = \mathbf{x})$ is determined by the standard multinomial formula:

$$\begin{aligned} P(X = \mathbf{x}) &= \frac{n!}{x_1!x_2!\ldots x_k!} p_1^{x_1} p_2^{x_2} \ldots p_k^{x_k} \\ &= \frac{n!}{\prod_i x_i!} \prod_i p_i^{x_i} \\ &= n! \frac{\prod_{i\in A} p_i^{x_i}}{\prod_{i\in A} x_i!} \frac{\prod_{i\in \bar{A}} p_i^{x_i}}{\prod_{i\in \bar{A}} x_i!} \end{aligned}$$

Combining these formulae, we get:

$$\frac{P(X = \mathbf{x})}{P(\sum X_A = y)} = \left(\frac{y!}{\prod_{i\in A} x_i!} \frac{\prod_{i\in A} p_i^{x_i}}{(\sum \mathbf{p}_A)^y}\right) \left(\frac{(n-y)!}{\prod_{i\in \bar{A}} x_i!} \frac{\prod_{x\in \bar{A}} p_i^{x_i}}{(\sum \mathbf{p}_{\bar{A}})^{n-y}}\right).$$

As $\sum_A x_i = y$ and therefore $\sum_{\bar{A}} x_i = n - y$ we may rearrange the equation above to be:

$$\frac{P(X = \mathbf{x})}{P(\sum X_A = y)} = \left(\frac{y!}{\prod_{i\in A} x_i!} \prod_{i\in A} \left(\frac{p_i}{\sum p_A}\right)^{x_i}\right) \left(\frac{(n-y)!}{\prod_{i\in \bar{A}} x_i!} \prod_{i\in \bar{A}} \left(\frac{p_i}{\sum \mathbf{p}_{\bar{A}}}\right)^{x_i}\right).$$

Note that this is the product of two multinomial distributions. Hence the result. □

Corollary A.3.7 *Let X be a random vector distributed according to an $(n, k, \mathbf{p})$-multinomial*

distribution. We partition the elementary outcomes $\mathcal{U} = \{1, 2, \ldots, k\}$ of each $(k, \mathbf{p})$-multinomial trial into l sets $S_1, S_2, \ldots, S_l$. Let $\mathbf{y} = (y_1, y_2, \ldots, y_l)$ be an l-dimensional vector such that $\sum \mathbf{y} = n$. Given that $\sum X_{S_i} = y_i$ for all i, the conditional probability distribution of X is that of l independent multinomial distributions; for each i the corresponding random subvector X_{S_i} is distributed according to a $(y, |S_i|, \mathbf{p}_{S_i} / \sum \mathbf{p}_{S_i})$-multinomial distribution.

Recall that on page xii we defined $\mathbf{1}_x$ that $\mathbf{1}_6 = (1, 1, 1, 1, 1, 1)^T$, $\mathbf{1}_3 = (1, 1, 1)^T$, and $\mathbf{1}_2 = (1, 1)^T$. Let $X = (X_1, X_2, X_3, X_4, X_5, X_6)^T$ be a random vector distributed according to a $(70, 6, \frac{1}{6}\mathbf{1}_6)$-multinomial distribution. Let $l = 3$, $S_1 = \{1, 2, 3\}$, $S_2 = \{4\}$, $S_5 = \{5, 6\}$.

Then, given $\mathbf{y} = (40, 20, 10)^T = (\sum X_{S_1}, \sum X_{S_2}, \sum X_{S_3})$ the conditional distribution of X_{S_1} is a $(40, 3, \frac{1}{3}\mathbf{1}_3)$-multinomial distribution; the conditional distribution of X_{S_2} is a trivial $(20, 1, (1))$-multinomial distribution, i.e. the product space of 20 trials that always pick the same result; the conditional distribution of X_{S_3} is a $(10, 2, \frac{1}{2}\mathbf{1}_2)$-multinomial distribution. Also, the conditional distributions of X_{S_1}, X_{S_2} and X_{S_3} are mutually independent.

Lemma A.3.8 *Let X be a $(2n, k, p)$-multinomially distributed random vector. Let ω be an integer in the range $[0, 2n]$. Let R_ω be the statement "$X_i \in 2\mathbb{Z}$ for $i = 1, 2, \ldots, \omega$". Then the probability that R_{2n} is true, is at least 2^{-k+1}.*

Proof. Let S_ω be the statement "for $k = 1, 2, \ldots, \omega$, this lemma holds".

Proof of S_1: If $k = 1$, $X_1 = 2n$ which is even. Thus the probability that all variables are even is one.

Proof of S_ω by strong induction where $k > 1$: Assume $S_{\omega-1}$. Let Q_i be the statement "$X_\omega = i$".

Let i be an integer: From Lemma A.3.6, the conditional probability distribution of the random vector $(X_1, X_2, \ldots, X_{k-1})$, given $X_\omega = 2i$, is a multinomial distribution with $k-1$ variables. Hence S_{k-1} implies $P(R_\omega | Q_{2i}) \geq 2^{-(k-1)+1}$. From the definition of a conditional probability, we know that:

$$P(R_\omega | Q_{2i}) \quad = \quad \frac{P(R_\omega \wedge Q_{2i})}{P(Q_{2i})},$$

by multiplying both sides by $P(Q_{2i})$ we get

$$P(R_\omega \wedge Q_{2i}) \quad = \quad P(R_\omega | Q_{2i}) P(Q_{2i}),$$

from substituting $P(R_\omega | Q_{2i}) \geq 2^{-(k-1)+1}$ into the above formula we find that

$$P(R_\omega \wedge Q_{2i}) \quad \geq \quad 2^{-(k-1)+1} P(Q_{2i}).$$

As Q_i implies not Q_j where $i \neq j$,

$$\begin{aligned} P(R_\omega) &= \sum_{i=0}^{2n} P(R_\omega \wedge Q_i) \\ &\geq \sum_{i=0}^{n} P(R_\omega \wedge Q_{2i}) \\ &\geq \sum_{i=0}^{n} 2^{-(k-1)+1} P(Q_{2i}) \\ &= 2^{-(k-1)+1} \sum_{i=0}^{n} P(Q_{2i}). \end{aligned}$$

From Corollary A.3.3, we know that X_ω has a probability of at least 0.5 of being even. Thus $\sum_{i=0}^{n} P(Q_{2i}) \geq 1/2$, and so

$$\begin{aligned} P(R_\omega) &\geq 2^{-(k-1)+1} \left(\frac{1}{2}\right) \\ &= 2^{-k+1}. \end{aligned}$$

□

A.4 Multivariate Normal Distribution

A **multivariate normal distribution** (see e.g. Anderson, 1984; pp. 6–50, Seber, 1977; pp. 20–41)is a generalisation of the familiar univariate normal distribution. We call a univariate normal distribution with mean μ and variance σ^2 a (μ, σ^2)-normal distribution. The distribution of a **multivariate normally distributed variable** $X = (X_1, X_2, \ldots, X_n)$ is determined by three parameters; the number of dimensions n, the vector of means $\mathbf{b}$ for each dimension, and covariance matrix Ω with $\Omega_{ij} = \text{cov}(X_i, X_j)$. We call this distribution an $(n, \mathbf{b}, \Omega)$-normal distribution.

Like a univariate normal distribution, a multivariate normal distribution can be **singular** or **non-singular**. A singular univariate distribution has a mean of 0. Likewise a singular multivariate distribution has a covariance matrix Ω with a determinant $|\Omega|$ equal to 0, i.e. Ω is singular (see e.g. Anderson, 1984; pp. 31–3). We are only interested in non-singular normal distributions, that is normal distributions that are not singular.

It is well known that a non-singular (μ, σ^2)-normal distribution has the density func-

tion:

$$\begin{aligned} f(x) &= ke^{-\frac{1}{2}\sigma^{-2}(x-\mu)^2}, \\ &= ke^{-\frac{1}{2}(x-\mu)\sigma^{-2}(x-\mu)}, \end{aligned}$$

where k is number chosen so that $\int_{-\infty}^{\infty} f(x) = 1$. Similarly, for a non-singular $(n, \mathbf{b}, \Omega)$-normal distribution, the probability density function is:

$$f(\mathbf{x}) = ke^{-\frac{1}{2}(\mathbf{x}-\mathbf{b})^T\Omega^{-1}(\mathbf{x}-\mathbf{b})},$$

where k is chosen so that $\int_{\mathbb{R}^n} f(\mathbf{x})d\mathbf{x} = 1$.

For non-singular multivariate normal distributions these distributions the inverse Ω^{-1}, of the covariance matrix, is a positive definite matrix (see e.g. Anderson, 1984; p15). Thus $(\mathbf{x}-\mathbf{b})^T\Omega^{-1}(\mathbf{x}-\mathbf{b})$ is non-negative for all values of $\mathbf{x}$. For any non-negative y, we know that e^{-y} is in the range $(0, 1]$, so f is bounded within the range $(0, k]$.

Lemma A.4.1 *Let X be an n-dimensional random vector distributed according to a non-singular multivariate normal distribution. Let S be a set of measure 0, for example, a subspace with a lower dimension than n. Then the probability that $X \in S$, is 0.*

Proof. The probability that X is in S, is the integral of the probability density function over the set S:

$$\begin{aligned} P(X \in S) &= \int_S f(\mathbf{x})d\mathbf{x} \\ &\leq \max(f(\mathbf{x})) \int_S 1d\mathbf{x}, \end{aligned}$$

this is zero because f is finite and bounded, and S has measure 0. □

A.5 Multisets

A **multiset** is like a set, but allows multiple occurrences of each element. For example the multiset {1,2,3} is equivalent to the multiset {3,2,1}, but not the multiset {1,2,2,3}. We may also denote this multiset as $\{1, 2^2, 3\}$.

Definition A.5.1 *A multiset is an ordered pair (A, m) where A is a fixed set and m is a function $M\colon A \to \{0, 1, 2, \ldots\}$ (see e.g. Stanley 1997).*

Definition A.5.2 *A multiset $N = (A, n)$ is a **submultiset** of $M = (A, m)$ if $n(a) \leq m(a)$ for all a in A. We write this as $N \subseteq M$.*

Definition A.5.3 *We define the cardinality $|M|$ of a multiset $M = (A, m)$ as*

$$M \;=\; \sum_{a \in A} m(a)$$

Lemma A.5.4 *Let $A = \{a_1, a_2, \ldots, a_{|A|}\}$ be a set and m be a variable function $m\colon A \to \{0, 1, 2, \ldots, \}$. Then we may represent the variable multiset $M = (A, m)$ with $\mathcal{O}(\ln |M|)$ bits.*

Proof. Let $V = (a_1, a_2, \ldots, a_{|A|})$ be an ordered set, containing the elements of the unordered set A. V is fixed, so we may represent V using $\mathcal{O}(1)$ bits. We may represent m by an ordered set $F = (f_1, f_2, \ldots, f_{|A|})$, where $f_i = m(a_i)$ for all i in $\{1, 2, \ldots, |A|\}$. Then $f_i \leq |M|$ for all i in $\{1, 2, \ldots |A|\}$, and so we may represent f_i using $\mathcal{O}(\ln |M|)$ bits. Thus we may represent F using $\mathcal{O}(|A| \ln |M|)$ bits which is in $\mathcal{O}(\ln |M|)$ since $|A|$ is fixed. □

Appendix B

Code and Output

B.1 Asymptotic Simpson's Rule

Using the model developed in Section 3.3.1, we generate 100,000 random elections with and infinite number of agents. We count how often the Simpson winner is the same as the Dodgson winner.

B.1.1 `asymp.sh` — wrapper script

This shell script is used as a wrapper to ask the R statistical tool to generate randomly generated numbers and feed them into `asymp.c`.

```
#!/bin/sh

mkdir tmp
cd tmp &&
echo 'while(1){writeLines(format(rnorm(10000),digits=22))}' | nice R --vanilla
                                                                    2>&1 | (

 name=asymp
 arch=pentium4 #If you must change this if you do not have a p4

 orig_c_file=../$name.c
 mod_c_file=$name.tmp.c
 obj_file=./$name
 output_file=./$name.output

 for a in 3 4 5 6 7 8
 do

  sed "s/^#define.*NUM_ALTERS.*/#define NUM_ALTERS $a/" < $orig_c_file  >
                                                                  $mod_c_file
  gcc -Wall $mod_c_file -O3 -march=$arch -o $obj_file
  nice $obj_file | tee -a $name.output
```

```
 done

)

exit
```

B.1.2 asymp.c

```
/*****************************************************************************
 * INPUT: A list of normally distributed random numbers in stdin
 * OUTPUTS: The asymptotic frequency that the Simpson winner is the Tideman
 *      winner as $n->\infty$. Note that the Tideman winner converges to the
 *      Dodgson winner as $n->\infty$, and so this frequency is also the
                                                                    frequency
 *      that the Simpson winner is the Dodgson winner.
 */

#include <stdlib.h>
#include <assert.h>
#include <stdio.h>

#define rand_int(n) (   (int) ( ((float)n)*rand()/(RAND_MAX+1.0) )   );
#define NUM_ALTERS 4
#define MAX_VOTE_TYPES 500000
#define RAND_STATE_SIZE 256

char rand_state_array[RAND_STATE_SIZE];

typedef double FREQ;

char preflist[NUM_ALTERS];
FREQ adv_matrix[NUM_ALTERS][NUM_ALTERS];

unsigned int NUM_VOTE_TYPES;


unsigned int factorial (unsigned int x){
        unsigned int f=1;
        unsigned int i;

        for (i=2;i<=x;i++) f*=i;

        return f;
}

FREQ votefreq[MAX_VOTE_TYPES];

void init (){
        int i;
        initstate(0,rand_state_array,RAND_STATE_SIZE);
        for (i=0;i<NUM_ALTERS;i++) preflist[i]=i;
```

```
        NUM_VOTE_TYPES=factorial(NUM_ALTERS);
        assert(MAX_VOTE_TYPES>=factorial(NUM_ALTERS));
}

int pref_number;



/*Adds freq instances of the current permutation to the
 * advantages matrix */
static void add_to_adv_matrix (FREQ freq) {
        int i,j; /* positions in the preference list */
        int a,b; /* candidate b is prefered over a */

        for (i=0;i<NUM_ALTERS;i++) {
                a=preflist[i];
                for (j=0;j<i;j++) {
                        b=preflist[j];
                        adv_matrix[b][a]+=freq;
                        adv_matrix[a][b]-=freq;
                }
        }

}

int display_perms=0;

static void ForAllPerms (char varsection[], int varlen){
        int i;
        char swap1,swap2;

        if (varlen>1) {

                ForAllPerms(varsection+1,varlen-1);
                swap1=varsection[0];
                for (i=1;i<varlen;i++) {
                        swap2=varsection[i];
                        varsection[i]=swap1;
                        varsection[0]=swap2;

                        ForAllPerms(varsection+1,varlen-1);

                        varsection[i]=swap2;
                }
                varsection[0]=swap1;
        } else {

                if (display_perms && votefreq[pref_number] > 0 ) {
                        printf ("%d: ",(int)votefreq[pref_number]);
                        for (i=0;i<NUM_ALTERS;i++) printf("%c
                                                        ",'a'+(char)preflist[i]);
                        printf("\n");
                }

                add_to_adv_matrix(votefreq[pref_number]);
```

```
                pref_number++;

        }
}

static void calc_adv_matrix() {
        int i,j;
        for (i=0;i<NUM_ALTERS;i++)
                for (j=0;j<NUM_ALTERS;j++)
                        adv_matrix[i][j]=0;
        pref_number=0;
        ForAllPerms(preflist,NUM_ALTERS);
        if (display_perms) {
        for (i=0;i<NUM_ALTERS;i++) {
                for (j=0;j<NUM_ALTERS;j++) {
                        printf("%d\t",(int)adv_matrix[i][j]);
                }
                printf("\n");
        }
        }

        return;
}

/*
static void display_profile() {
        display_perms=1;
        calc_adv_matrix();
        display_perms=0;
}
*/

/*****************************************************************
 * IMPARTIAL_CULTURE
 *   randomly generates a voting culture distributed according to
 *   impartial culture hypothesis.
 * Inputs: nvoters       - number of voters in election
 *       : votefreq_size - number of possible different types of votes
 * Output: votefreq      - frequency of each type of vote.
 *
 * returns "first citizen's" vote, may be useful for tie-breaking.
 *****************************************************************/
static unsigned int impartial_culture(FREQ* votefreq, int votefreq_size, int
                                                                    nvoters){
        int i;
        double x;
        for (i=0;i<votefreq_size;i++)
                {
                        scanf("%lf",&x);
                        votefreq[i]=(FREQ)x;
                }

        return 0;
}
```

```
/* Returns the maximum defeat for each candidate */
void calc_Simpson_scores(FREQ* score_array){
        FREQ score_of_i, adv;
        int i,j=1;

        for (i=0;i<NUM_ALTERS;i++) {
                score_of_i=adv_matrix[0][i];
                for (j=1;j<NUM_ALTERS;j++) {
                        adv=adv_matrix[j][i];
                        if (adv>score_of_i) score_of_i=adv;
                }
                score_array[i]=score_of_i;
        }
}

void calc_Tideman_scores(FREQ* score_array){
        FREQ score_of_i, adv;
        int i,j;

        for (i=0;i<NUM_ALTERS;i++) {
                score_array[i]=0;
        }
        for (i=0;i<NUM_ALTERS;i++) {
                score_of_i=0;
                for (j=0;j<NUM_ALTERS;j++) {
                        adv=adv_matrix[j][i];
                        if (adv>0) score_of_i+=adv;
                }
                score_array[i]=score_of_i;
        }
}

#ifdef FREQ_INT
void calc_DQ_scores(FREQ* score_array){
        FREQ score_of_i, adv;
        int i,j;

        for (i=0;i<NUM_ALTERS;i++) {
                score_array[i]=0;
        }
        for (i=0;i<NUM_ALTERS;i++) {
                score_of_i=0;
                for (j=0;j<NUM_ALTERS;j++) {
                        adv=adv_matrix[j][i];
                        if (adv>0) score_of_i+=(adv+1)/2; /*c floors all ints*/
                }
                score_array[i]=score_of_i;
        }
}
#endif

int calc_winners(FREQ* score_array) {
        int i;
        FREQ min=score_array[0];
        int winners; /*stored as bits*/
```

```
        for (i=1;i<NUM_ALTERS;i++) {
                if (min>score_array[i]){
                        min=score_array[i];
                }
        }

        winners=0;
        for (i=0;i<NUM_ALTERS;i++) {
                winners=winners*2;
                if (min==score_array[i]){
                        if (winners!=0) printf("Warning, tied winners exist\n");
                        winners++;
                }
        }

        return winners;

}

void display_scores(FREQ *scores) {
        int i;
        for (i=0;i<NUM_ALTERS;i++){ printf("%lf\t",scores[i]); }
        printf("\n");
        printf("%d\n",calc_winners(scores));
}



int main(){
        int numVoters=2;
        int num_runs=100000;
        int run;

        int acc1,acc2; /* # of times approximation was accurate */
        int acc2b;
        int Simpson_winners;
        int Tideman_winners;

        FREQ scores[NUM_ALTERS];


        char buf[999];
        for (run=1;run<200;run++){ /* required to get skip past init info and
                                                        get into random numbers */
                scanf("%99s",buf);
                /*printf("str: %s\n",buf);*/
        }


        init();

        FREQ *vf=votefreq;

        acc1=0;
        acc2=0;
        acc2b=0;
```

```
        for (run=0;run<num_runs;run++){


                impartial_culture(vf,NUM_VOTE_TYPES,numVoters);

                calc_adv_matrix();

                calc_Tideman_scores(scores);
                Tideman_winners=calc_winners(scores);

                calc_Simpson_scores(scores);
                Simpson_winners=calc_winners(scores);
                acc1+=(Tideman_winners==Simpson_winners);

                if (Tideman_winners & Simpson_winners) {
                        acc2++;
                }

                fprintf (stderr,"%c%d/%d",(char)13,acc1,run);

        }

        fprintf (stderr,"%c",(char)13);
        fprintf (stderr,"                                                ");
        fprintf (stderr,"%c",(char)13);

        printf("%d \t%d \t%d\n",NUM_ALTERS,acc1,acc2);

        return 0;
}
```

B.1.3 Output

The first column of the output is the number of alternatives. The second column represents the asymptotic limit of the probability that the set of tied Simpson winners are the set of tied Dodgson winners, as the number of agents tends to infinity. The third column represents the asymptotic limit of the probability that the set of tied Simpson winners has a non-empty intersection with the set of tied Dodgson winners. However as the number of agents tends to infinity, the probability of a tie tends to zero. Hence both the second and third columns are simply the probability that the Simpson winner is the Dodgson winner.

```
3       100000  100000
4       99319   99319
5       98282   98282
6       97274   97274
7       96067   96067
8       94982   94982
```

B.2 Dodgson Quick vs Tideman vs Simpson

This code is designed to generate random elections, under the impartial culture hypothesis, and compare how often various rules pick the same winner as the Dodgson Rule.

B.2.1 `SiTiDQ.sh` — wrapper script

```
#!/bin/sh

mkdir tmp
cd tmp && (

 name=SiTiDQ
 arch=pentium4 #If you must change this if you do not have a p4

 orig_c_file=../$name.c
 mod_c_file=$name.tmp.c
 obj_file=./$name
 output_file=./$name.output

 #for a in 4 5 6 7 8
 for a in 3
 do

  sed "s/^#define.*NUM_ALTERS.*/#define NUM_ALTERS $a/" < $orig_c_file  >
                                                                    $mod_c_file
  gcc -Wall $mod_c_file -O3 -march=$arch -o $obj_file
  nice $obj_file | tee -a $name.output

 done

)
```

B.2.2 `SiTiDQ.c`

```
#include <stdlib.h>
#include <assert.h>
#include <stdio.h>

#define rand_int(n) (   (int) ( ((float)n)*rand()/(RAND_MAX+1.0) )   );
#define NUM_ALTERS 5
#define MAX_VOTE_TYPES 100000
//#define RAND_STATE_SIZE 256
#define RAND_STATE_SIZE 8

#define BOOL int

char rand_state_array[RAND_STATE_SIZE];
```

```
typedef signed long FREQ;

char preflist[NUM_ALTERS];
FREQ adv_matrix[NUM_ALTERS][NUM_ALTERS];
FREQ direct_above_matrix[NUM_ALTERS][NUM_ALTERS];

unsigned int NUM_VOTE_TYPES;



inline static unsigned int factorial (unsigned int x){
        unsigned int f=1;
        unsigned int i;

        for (i=2;i<=x;i++) f*=i;

        return f;
}

FREQ votefreq[MAX_VOTE_TYPES];

inline static void init (){
        int i;
        initstate(0,rand_state_array,RAND_STATE_SIZE);
        for (i=0;i<NUM_ALTERS;i++) preflist[i]=i;
        NUM_VOTE_TYPES=factorial(NUM_ALTERS);
        assert(MAX_VOTE_TYPES>=factorial(NUM_ALTERS));
}

int pref_number;

/*Adds freq instances of the current permutation to the
 * advantages matrix */
inline static void add_to_adv_matrix (FREQ freq) {
        int i,j; /* positions in the preference list */
        int a,b; /* candidate b is prefered over a */

        for (i=0;i<NUM_ALTERS;i++) {
                a=preflist[i];
                for (j=0;j<i;j++) {
                        b=preflist[j];
                        adv_matrix[b][a]+=freq;
                        adv_matrix[a][b]-=freq;
                }
        }

        for (i=1;i<NUM_ALTERS;i++) {
                a=preflist[i];
                b=preflist[i-1];

                direct_above_matrix[b][a]+=freq;
        }

}

int display_perms=0;
```

```
static void ForAllPerms (char varsection[], int varlen){
        int i;
        char swap1,swap2;

        if (varlen>1) {

                ForAllPerms(varsection+1,varlen-1);
                swap1=varsection[0];
                for (i=1;i<varlen;i++) {
                        swap2=varsection[i];
                        varsection[i]=swap1;
                        varsection[0]=swap2;

                        ForAllPerms(varsection+1,varlen-1);

                        varsection[i]=swap2;
                }
                varsection[0]=swap1;
        } else {

                if (display_perms && votefreq[pref_number] > 0 ) {
                        printf ("%d: ",(int)votefreq[pref_number]);
                        for (i=0;i<NUM_ALTERS;i++) printf("%c
                                                        ",'a'+(char)preflist[i]);
                        printf("\n");
                }

                add_to_adv_matrix(votefreq[pref_number]);

                pref_number++;

        }
}

inline static void calc_adv_matrix() {
        int i,j;
        for (i=0;i<NUM_ALTERS;i++)
                for (j=0;j<NUM_ALTERS;j++) {
                        adv_matrix[i][j]=0;
                        direct_above_matrix[i][j]=0;
                }
        pref_number=0;
        ForAllPerms(preflist,NUM_ALTERS);
        if (display_perms) {
        for (i=0;i<NUM_ALTERS;i++) {
                for (j=0;j<NUM_ALTERS;j++) {
                        printf("%d\t",(int)adv_matrix[i][j]);
                }
                printf("\n");
        }
        }

        return;
}
```

```
inline static void display_profile() {
        printf("asdfasdf\n");
        display_perms=1;
        calc_adv_matrix();
        display_perms=0;
}


/*****************************************************************
 * IMPARTIAL_CULTURE
 *   randomly generates a voting culture distributed according to
 *   impartial culture hypothesis.
 * Inputs: nvoters       - number of voters in election
 *       : votefreq_size - number of possible different types of votes
 * Output: votefreq      - frequency of each type of vote.
 *
 * returns "first citizen's" vote, may be useful for tie-breaking.
 *****************************************************************/
inline static unsigned int impartial_culture(FREQ* votefreq, int votefreq_size,
                                                              int nvoters){
        int i;
        unsigned int vote=9999;
        for (i=0;i<votefreq_size;i++) votefreq[i]=0;
        for (i=0;i<nvoters;i++){
                vote=rand_int(votefreq_size);
                assert(vote<votefreq_size);
                votefreq[vote]++;
        }

        return vote;
}


/* Returns the maximum defeat for each candidate */
inline static void calc_Simpson_scores(FREQ* score_array){
        FREQ score_of_i, adv;
        int i,j=1;

        for (i=0;i<NUM_ALTERS;i++) {
                score_of_i=adv_matrix[0][i];
                for (j=1;j<NUM_ALTERS;j++) {
                        adv=adv_matrix[j][i];
                        if (adv>score_of_i) score_of_i=adv;
                }
                score_array[i]=score_of_i;
        }
}

inline static void calc_Tideman_scores(FREQ* score_array){
        FREQ score_of_i, adv;
        int i,j;

        for (i=0;i<NUM_ALTERS;i++) {
                score_array[i]=0;
        }
        for (i=0;i<NUM_ALTERS;i++) {
```

```
                score_of_i=0;
                for (j=0;j<NUM_ALTERS;j++) {
                        adv=adv_matrix[j][i];
                        if (adv>0) score_of_i+=adv;
                }
                score_array[i]=score_of_i;
        }
}


inline static BOOL calc_DQ_scores(FREQ* score_array){
        FREQ score_of_i, adv;
        int i,j;

        BOOL allexact=1;

        for (i=0;i<NUM_ALTERS;i++) {
                score_array[i]=0;
        }
        for (i=0;i<NUM_ALTERS;i++) {
                score_of_i=0;
                for (j=0;j<NUM_ALTERS;j++) {
                        adv=adv_matrix[j][i];
                        adv=(adv+1)/2; /*c floors all ints*/
                        if (adv>0) {
                                score_of_i+=adv;
                                if (adv>direct_above_matrix[j][i]) {
                                        allexact=0;
                                }
                        }
                }
                score_array[i]=score_of_i;
        }

        return allexact;
}

inline static int calc_winners(FREQ* score_array) {
        int i;
        int min=score_array[0];
        int winners; /*stored as bits*/

        for (i=1;i<NUM_ALTERS;i++) {
                if (min>score_array[i]){
                        min=score_array[i];
                }
        }

        winners=0;
        for (i=0;i<NUM_ALTERS;i++) {
                winners=winners*2;
                if (min==score_array[i]){
                        winners++;
                }
        }
```

```
        return winners;

}

static inline void display_scores(FREQ *scores) {
        int i;
        for (i=0;i<NUM_ALTERS;i++){ printf("%d\t",(int) scores[i]); }
        printf("\n");
        printf("%d\n",calc_winners(scores));
}


int main(){
        //int numVoteTypes=factorial(numAlters);
        int numVoters=2;
        //int i;
        int num_runs=100000;
        int run;

        int acc1,acc2; /* # of times approximation was accurate */
        int acc2b;
        int acc2c;
        int acc3;
        int DQ_winners;
        int Simpson_winners;
        int Tideman_winners;

        FREQ scores[NUM_ALTERS];

        init();

        FREQ *vf=votefreq;

        printf("Number of alternatives: %d\n", NUM_ALTERS);
        printf("Number of runs: %d\n",num_runs);
        printf("1st col: number of voters\n");
        printf("2nd col: # of runs where the set of Tideman winners is the set
                                                                of DQ winners\n");
        printf("3rd col: # of runs where the set of Simpson winners is the set
                                                                of DQ winners\n");
        printf("4th col: # of runs where the intersection of the Simpson winners
                                            and the DQ winners is non-empty\n");
        printf("5th col: # of runs where the intersection of the Tideman winners
                                            and the DQ winners is non-empty\n");
        printf("6th col: # of runs where we know all DQ scores equal all Dodgson
                                                                Scores.\n\n");

        printf("#voters\t#TI=DQ\t#SI=DQ\tSI^DQ>0\tTI^DQ>0\tDQexact\n");


        while (numVoters<200000) {
        //while (1) {

        acc1=0;
        acc2=0;
        acc2b=0;
```

```
        acc2c=0;
        acc3=0;


        for (run=0;run<num_runs;run++){

                impartial_culture(vf,NUM_VOTE_TYPES,numVoters);

                calc_adv_matrix();

                acc3+=calc_DQ_scores(scores);
                DQ_winners=calc_winners(scores);

                calc_Tideman_scores(scores);
                Tideman_winners=calc_winners(scores);
                acc1+=(DQ_winners==Tideman_winners);

                calc_Simpson_scores(scores);
                Simpson_winners=calc_winners(scores);
                acc2+=(DQ_winners==Simpson_winners);

        /*      display_scores(scores);*/

                if (DQ_winners & Simpson_winners) {
                        acc2b++;
                } else {
                        //printf("\n%d,%d\n",DQ_winners,Simpson_winners);
                        //display_profile();
                }

                if (Tideman_winners & DQ_winners) {
                        acc2c++;
                }

        }

        printf("%d\t%d\t%d\t%d\t%d\t%d\n",numVoters,acc1,acc2,acc2b,acc2c,acc3);
        fflush(stdout);

        numVoters=numVoters*2-1;


        }

        printf("\n\n");

        return 0;
}
```

B.2.3 Output

```
Number of runs: 100000
1st col: number of voters
```

```
2nd col: # of runs where the set of Tideman winners is the set of DQ winners
3rd col: # of runs where the set of Simpson winners is the set of DQ winners
4th col: # of runs where the set of Simpson winners has a non empty intersection
                                                    with the set of Tideman winners
5th col: # of runs where the set of Tideman winners has a non empty intersection
                                                        with the set of DQ winners
6th col: # of runs where we know all DQ scores equal all Dodgson Scores.

Number of alternatives: 3
#voters #TI=DQ  #SI=DQ  SI^DQ>0 TI^DQ>0 DQexact
2       100000  100000  100000  100000  83306
3       100000  100000  100000  100000  55721
5       100000  100000  100000  100000  78933
9       100000  100000  100000  100000  94718
17      100000  100000  100000  100000  99589
33      100000  100000  100000  100000  99997
65      100000  100000  100000  100000  100000
...     ...     ...     ...     ...     ...
131073  100000  100000  100000  100000  100000

Number of alternatives: 4
#voters #TI=DQ  #SI=DQ  SI^DQ>0 TI^DQ>0 DQexact
2       100000  100000  100000  100000  45876
3       100000  92371   100000  100000  6017
5       100000  92236   100000  100000  22977
9       99808   93787   99997   100000  54499
17      99492   95341   99980   100000  86222
33      99323   96541   99954   100000  98857
65      99361   97272   99858   100000  99994
129     99434   97859   99770   100000  100000
257     99584   98381   99697   100000  100000
513     99664   98586   99562   100000  100000
1025    99746   98796   99532   100000  100000
2049    99820   98920   99455   100000  100000
4097    99842   98956   99362   100000  100000
8193    99901   99074   99345   100000  100000
16385   99934   99195   99366   100000  100000
32769   99959   99197   99323   100000  100000
65537   99969   99265   99344   100000  100000
131073  99967   99221   99294   100000  100000

Number of alternatives: 5
#voters #TI=DQ  #SI=DQ  SI^DQ>0 TI^DQ>0 DQexact
2       100000  100000  100000  100000  19788
3       100000  86127   100000  100000  108
5       99901   84814   100000  100000  1896
9       99291   87184   99957   100000  13176
17      98646   90186   99872   100000  46905
33      98440   92658   99777   99999   86162
65      98453   94000   99575   99999   99281
129     98678   95252   99340   99998   99999
257     98993   96128   99106   99999   100000
513     99252   96764   98965   99997   100000
1025    99454   97167   98767   99999   100000
2049    99585   97530   98680   100000  100000
4097    99698   97683   98547   100000  100000
```

```
8193    99772   97826   98430   99999   100000
16385   99840   97958   98373   100000  100000
32769   99887   98073   98381   100000  100000
65537   99920   98049   98272   99999   100000
131073  99934   98156   98305   100000  100000
262145  99962   98166   98283   100000  100000
524289  99976   98236   98304   100000  100000

Number of alternatives: 6
#voters #TI=DQ  #SI=DQ  SI^DQ>0 TI^DQ>0 DQexact
2       100000  100000  100000  100000  7389
3       100000  81137   100000  100000  0
5       99761   78746   99998   100000  38
9       98709   81197   99875   100000  1177
17      97692   85351   99638   100000  13080
33      97514   88668   99530   99998   53736
65      97584   91135   99169   99993   92654
129     97927   92689   98766   99991   99851
257     98425   94009   98524   99990   100000
513     98814   94800   98128   99993   100000
1025    99078   95446   97919   99995   100000
2049    99365   96034   97795   99999   100000
4097    99528   96297   97586   99996   100000
8193    99678   96494   97394   99998   100000
16385   99777   96689   97313   100000  100000
32769   99811   96732   97181   100000  100000
65537   99877   96997   97309   100000  100000
131073  99908   97007   97224   99999   100000

Number of alternatives: 7
#voters #TI=DQ  #SI=DQ  SI^DQ>0 TI^DQ>0 DQexact
2       100000  100000  100000  100000  2229
3       100000  76941   100000  100000  0
5       99652   73983   99991   100000  0
9       98174   76108   99767   100000  31
17      96844   81372   99357   99997   1680
33      96542   85501   99123   99990   20830
65      96830   88411   98720   99985   71794
129     97407   90518   98293   99982   98147
257     97970   91947   97781   99982   99997
513     98445   93050   97348   99988   100000
1025    98798   93830   97084   99983   100000
2049    99246   94524   96762   99994   100000
4097    99399   94965   96631   99993   100000
8193    99558   95219   96417   99998   100000
16385   99670   95427   96280   99996   100000
32769   99759   95591   96228   99995   100000
65537   99810   95616   96077   99998   100000
131073  99878   95747   96079   99999   100000

Number of alternatives: 8
#voters #TI=DQ  #SI=DQ  SI^DQ>0 TI^DQ>0 DQexact
2       100000  100000  100000  100000  613
3       100000  73583   100000  100000  0
5       99487   69644   99983   100000  0
9       97678   71966   99630   100000  0
```

17	96219	78016	99031	99995	74
33	95734	82752	98746	99983	4569
65	96144	86006	98326	99962	41193
129	96781	88241	97521	99966	90701
257	97516	90120	97083	99971	99873
513	98157	91485	96707	99975	100000
1025	98621	92356	96140	99980	100000
2049	99044	93094	95775	99989	100000
4097	99293	93570	95540	99986	100000
8193	99476	93997	95422	99997	100000
16385	99618	94221	95246	99996	100000
32769	99754	94593	95235	99993	100000
65537	99823	94583	95072	99998	100000
131073	99855	94718	95097	99997	100000

B.3 Exact Dodgson Algorithm

This is the code we used to calculate the exact Dodgson scores. This module was called by a custom built library of MATLAB and R report generation code. This library is over 2000 lines long and will not be printed here. See `http://dansted.org/thesis06/` for files relating to the thesis that have been omitted from the main text. Also feel free to contact the author at `gmatht@gmail.com` as to the use of these files and for more experimental data.

```
/*************************************************************************
 * Computes Dodgson Scores (See bottom for matlab mex interface)
 * ***********************************************************************
 * This programlet calculates the Dodgson scores for a given profile
 *
 * This programlet links against the GPL'd library gplk. This means that if this
 * program is distributed, permission must be given to the recipents to
 * redistribute it under the terms of the GPL.
 *
 * compile mex file by typing:
 *      mex DodgsonScores_C.c libglpk.a
 *
 * For some reason you may need to copy libglpk.a and glp*.h into the current
 * working directory to compile.
 *
 * ***********************************************************************/

#define HAS_MAINLINE

#ifndef HAS_MAINLINE
        #define MATLAB
#endif

#ifdef MATLAB
  #include  "mex.h"
```

```
#endif

/*#ifdef MATLAB
  #include  "mex.h"
#else
  #define HAS_MAINLINE
#endif*/

#include <stdio.h>
#include <stdlib.h>
#include "glpk.h"
#include "math.h"

/* using the interior point method may be faster.
#define USE_INTERIOR_POINT
 */

typedef short alter_t;
typedef short index_t;

typedef struct node_struct {
        short count;
        alter_t a;
        index_t up;
        index_t down;
        index_t left;
} node_t;


typedef struct dodgsn_struct {
        /* Graph representation of Dodgeson Problem */
        node_t *nodearray; /* nodes in graph of dodgsn problem */

        /* Linear Problem Array */
        int    nz;  /* Number of non-zero elements */
        int    *rn; /* row of ith non-zero element */
        int    *cn; /* col of ith non-zero element */
        double *a;  /* value of ith non-zero element */

        /* glpk's representation of linear programming problem*/
        LPX    *glpk;
} dodgsn_t;



static void error(char* err) {
#ifdef MATLAB
                mexErrMsgTxt(err);
#else
                printf("%s\n",err);
                exit(0);
#endif
}


/*****************************************************************************
```

```
 * newnode - Creates a new node
 * ************************************************************************/

index_t newnode(node_t nodearray[],
                int *nextnode,
                index_t up,
                alter_t alternative){

        node_t *p;
        int    node;

        node=(*nextnode)++;
        p=&(nodearray[node]);
        p->count=0;
        p->up=up;
        p->left=0;
        p->down=0;
        p->a=alternative;

        return ((index_t)node);
}

/*****************************************************************
 * addvote: Adds a vote to the Dodgson Node Array
 * **********************************************
 *
 * INPUTS: vote - a list of alternatives, first element is voters 1st pref.
 *         d    - The alternative we are calculating the dodgsn score for.
 *         nextnode - The index of the first empty node in the node array.
 *         nodearray - see below
 *
 * OUTPUTS: nextnode  - The index for first empty node once the vote added
 *          nodearray -
 *
 * The nodearray contains a tree. The root node is at position 0, and represents
 * an empty dodgsn string.
 *
 * node.count  - the number of votes that transverse this node.
 * node.a      - choosing this node represents unswapping this alternative.
 * node.up     - represents the dodgsn node that must be chosen before this.
 * node.down   - following this link represents choosing node.alter.
 * node.left   - links to another node that shares the same node.up
 *
 * CAUTION: does not check that nodearray is large enough to store all required
 * nodes. To ensure that a buffer overrun does not occur, nodearray's size
 * should be at least ((numvoters*numalters)+1)
 *
 *****************************************************************/
static void addvote( node_t nodearray[], int *nextnode, double vote[], alter_t
                                                                        d) {
        int node;
        node_t *node_ptr;
        double* alt_ptr;
        int next_alt;

        node_ptr = nodearray;
```

```
        node = 0;

        alt_ptr=vote-1;
        while (  d != ( next_alt = (alter_t)(*(++alt_ptr)) )  ) {

                if (node_ptr->down == 0){
                        node_ptr->down=
                                newnode(nodearray,
                                        nextnode,
                                        (index_t)node,
                                        (alter_t)next_alt);
                }
                node=(int)nodearray[node].down;
                node_ptr=&nodearray[node];

                while (node_ptr->a != next_alt) {
                        if (node_ptr->left == 0) {
                                node_ptr->left=
                                        newnode(nodearray,nextnode,
                                        node_ptr->up,next_alt);
                        }
                        node=node_ptr->left;
                        node_ptr=&nodearray[node];
                }

                node_ptr->count++;
        }

}

int global_pref_number;
int global_nodearray;
int global_nextnode;

static void ForAllPerms (char varsection[], int varlen){
        int i;
        char swap1,swap2;

        if (varlen>1) {

                ForAllPerms(varsection+1,varlen-1);
                swap1=varsection[0];
                for (i=1;i<varlen;i++) {
                        swap2=varsection[i];
                        varsection[i]=swap1;
                        varsection[0]=swap2;

                        ForAllPerms(varsection+1,varlen-1);

                        varsection[i]=swap2;
                }
                varsection[0]=swap1;
        } else {

                if (display_perms && votefreq[pref_number] > 0 ) {
                        printf ("%d: ",(int)votefreq[pref_number]);
```

```
                        for (i=0;i<NUM_ALTERS;i++) printf("%c
                                                        ",'a'+(char)preflist[i]);
                        printf("\n");
                }

                add_to_adv_matrix(votefreq[pref_number]);

                pref_number++;

        }
}

/*****************************************************************************
 * CreateDodgsonGraph - creates a graph that represents the options availiable
 *      for reducing the number of swaps while still having d being a condorcet
 *      winner.
 *****************************************************************************/

static void CreateDodgsonGraph(node_t *nodearray,/*array of nodes in dodgsn
                                                                    graph*/
                        int* nextnode,    /*first empty node in array*/
                        alter_t d,        /*calculating dogson score for alt d*/
                        int numalter,     /*number of alternatives */
                        int numvoter,     /*number of voters*/
                        double* profmatrix){ /*array of votes*/
int vote;
 *nextnode=0;

 global_nextnode=nextnode;
 global_nodearray=nodearray;


 newnode(nodearray,nextnode,(index_t)0,(alter_t)0);
 for(vote=0;vote<numvoter;vote++){
        addvote(nodearray,nextnode,&profmatrix[vote*numalter],d);
 }
}

/* ***************************************************************************
 * Calculates the Dodgson score for the initial state of the linear programming
 * problem. Final Dodgson score = initial dodgsn score + f, f found from linear
 * programming
 * ***************************************************************************/
static int InitalDodgsonScore(node_t nodearray[],int nextnode){
        int i;
        int score;

        score=0;
        for (i=1;i<nextnode;i++)
                score+=nodearray[i].count;
        return(score);
}

/* ***************************************************************************
 * adds a non-zero value a to the matrix
 * OUTPUTS: Updated linear problem
```

```
 * INPUTS: rn - row number
 *         cn - col number
 *          a - value to add
 * NOTE: This cannot be used to overwrite an existing non-zero number
********************************************************************** */


static void add_nz_val(dodgsn_t *dodg, int rn,int cn, double a) {
        int nz = dodg->nz;
        dodg->rn[nz]=rn;
        dodg->cn[nz]=cn;
        dodg->a[nz]=a;
        dodg->nz=nz+1;
}

/* ************************************************************************
 * CreateInequalities - Creates the inequalities for the linear programming
 * problem used to calculate the dodgsn score for d
 *
 * REMEMBER to delete the dodg->glpk that this procedure allocates
   ************************************************************************/
static void CreateInequalities(dodgsn_t *dodg,
                        node_t nodearray[],/*array of nodes in dodgsn graph*/
                        int nextnode,    /*first empty node in array*/
                        alter_t d,        /*calculating dogson score for alt d*/
                        int numalter,     /*number of alternatives */
                        int numvoter     /*number of voters*/
                        ){
int i;
int nextrow;
int child;
node_t *n;
double max_losses; /* max times d can lose and still be condorcet winner */
int numnodes=nextnode-1;
LPX* glpk;

 glpk=lpx_create_prob();
 dodg->glpk=glpk;
 lpx_add_cols(glpk, numnodes);
 dodg->nz=0;


 max_losses=(double)(int)(numvoter/2);
 /*max_losses=(double)(int)((numvoter-1)/2);*/


/* Add constrants requiring d to be a condorcet winner, d cannot be a winner
 * if we transverse (lose to) any other alternitive more than max_losses times*/
 lpx_add_rows(glpk,numalter);

 for (i=1;i<nextnode;i++){
        n=&nodearray[i];
        add_nz_val(dodg,n->a,i,1.0);
 }
 for (i=1;i<=numalter;i++){
        lpx_set_row_bnds(glpk, i, LPX_UP, 0.0, max_losses);
```

```
 }

 nextrow=numalter+1;

/* Add constraints requiring that we do not transverse any node more times
 * than its capacity, nor lest times than zero */
 for (i=1;i<nextnode;i++){
        n=&nodearray[i];
        lpx_set_col_bnds(glpk,i, LPX_DB, 0.0, (double)(n->count));
 }


/* Add constraints requiring that we must transverse parent nodes to reach
 * child nodes
 * */
 for (i=1;i<nextnode;i++){
        n=&nodearray[i];
        if (n->down != 0) {
                lpx_add_rows(glpk,1);
                child=n->down;
                while (child!=0){
                        /* for all children */
                        add_nz_val(dodg,nextrow,child,1.0);
                        child=nodearray[child].left;
                }
                add_nz_val(dodg,nextrow,i,-1.0);
                lpx_set_row_bnds(glpk, nextrow, LPX_UP, 0.0, 0.0);

                nextrow++;

        }
 }
 lpx_set_obj_dir(glpk, LPX_MAX);
 for (i=1;i<nextnode;i++){
        lpx_set_col_coef(glpk, i, 1.0);
 }

 lpx_load_mat3(glpk, dodg->nz , dodg->rn-1, dodg->cn-1, dodg->a-1);


}

void print_dodson_str(node_t nodearray[],int node) {
        if (node!=0){
                print_dodson_str(nodearray,nodearray[node].up);
                printf("%d",nodearray[node].a);
        }
}


/* ************************************************************************
 * DisplayInequalities - Test function only
 *
 * NOTE - only works if we have solved problem with simplex algorithm.
 *   it would be a simple modification to get this to work with interior point
 *   solutions as well.
```

```
 *
 * ***************************************************************************/

#ifdef DEBUG

static void DisplayInequalities(dodgsn_t *dodg,
                        node_t nodearray[],/*array of nodes in dodgsn graph*/
                        int nextnode,    /*first empty node in array*/
                        alter_t d,        /*calculating dogson score for alt d*/
                        int numalter,     /*number of alternatives */
                        int numvoter     /*number of voters*/
                        ){
        int cols,col;
        int node;
        double num_transverse;

        cols=nextnode-1;
        for(col=1;col<=cols;col++){
                node=col;
                printf("S");
                print_dodson_str(nodearray,node);

                lpx_get_col_info(dodg->glpk, col, NULL, &num_transverse, NULL);
                printf("\t = %g/%d\n",num_transverse,nodearray[node].count);
        }
        printf("\n Z = %4g , d = %4d\n\n",lpx_get_obj_val(dodg->glpk),d);

/*      lpx_write_mps(dodg->glpk,"Dodgson.mps");
        lpx_print_sol(dodg->glpk,"Dodgson.sol");*/
}

#endif

static double solve_lp_real(dodgsn_t *dodg){
        int return_code;
        double obj_val;
        int cols;
        LPX *glpk=dodg->glpk;
        int i;
int o;

#ifdef USE_INTERIOR_POINT
        return_code = lpx_interior(dodg->glpk);
        if (return_code != LPX_E_OK)
                printf("ERROR - cannot solve linear problem(interior)");
        obj_val=lpx_get_ips_obj(dodg->glpk);
#else
/*      lpx_scale_prob(dodg->glpk);*/
/*      lpx_set_int_parm(dodg->glpk,LPX_K_DUAL,1);*/
        return_code = lpx_simplex(dodg->glpk);
        if (return_code != LPX_E_OK)
                printf("ERROR - cannot solve linear problem(simplex)");
        obj_val=lpx_get_obj_val(dodg->glpk);
#endif
        return (obj_val);
}
```

```
/* ****************************************************************
 * Returns the integer solution to the Dodgson LP.
 * Note that solve_lp_real must be called first.*/

static double solve_lp_integer(dodgsn_t *dodg){
        int return_code;
        double obj_val;
        int cols;
        LPX *glpk=dodg->glpk;
        int i;
int o;

        lpx_set_class(glpk,LPX_MIP);
        cols=lpx_get_num_cols(glpk);
        for (i=1;i<=cols;i++) {
                lpx_set_col_kind(glpk,i,LPX_IV);
        }

        lpx_set_int_parm(dodg->glpk,LPX_K_BRANCH,1);
        return_code = lpx_integer(glpk);
        if (return_code != LPX_E_OK)
                printf("ERROR - cannot solve linear problem(integer)");
        else
                obj_val=lpx_get_mip_obj(glpk);

/*      o=(int)(obj_val+0.002);


        if (( o - obj_val + 0.001) < 0 ) {
                printf("\n*************************************************\n");
                printf("******************** NON INTEGER:
                                        %d,%g,%g\n",o,o-obj_val,obj_val);
                printf("*************************************************\n");
                lpx_write_mps(dodg->glpk,"Dodgson.mps");
                lpx_print_sol(dodg->glpk,"Dodgson.sol");
        }*/
        /*printf("vals: %g,%g\n\n",obj_val,lpx_get_ips_obj(dodg->glpk));*/

        return (obj_val);
}



/* ****************************************************************
 * Makes the 4 alternative problem described by A. Slinko in his
 * "Untidy Notes on Complexity of Several Voting Procedures"
 *
 * must set num_alter to 4, and num_voter to 21
 * testing purposes only.
 *
 * ****************************************************************/

void make_slinkos_4alt_problem(double *vote) {
        double* p=vote;
        int i;
```

```
        for (i=0;i<4;i++){ *(p++)=4.0; *(p++)=1.0; *(p++)=2.0; *(p++)=3.0; }
        for (i=0;i<3;i++){ *(p++)=4.0; *(p++)=2.0; *(p++)=3.0; *(p++)=1.0; }
        for (i=0;i<3;i++){ *(p++)=4.0; *(p++)=3.0; *(p++)=1.0; *(p++)=2.0; }
        for (i=0;i<3;i++){ *(p++)=1.0; *(p++)=2.0; *(p++)=3.0; *(p++)=4.0; }
        for (i=0;i<4;i++){ *(p++)=2.0; *(p++)=3.0; *(p++)=1.0; *(p++)=4.0; }
        for (i=0;i<4;i++){ *(p++)=3.0; *(p++)=1.0; *(p++)=2.0; *(p++)=4.0; }
}

/*****************************************************************
 *  Frees the memory allocated to a dodgsn structure
 *****************************************************************/

static void dodgsn_free(dodgsn_t *dodg) {
        free(dodg->nodearray);
        free(dodg->rn);
        free(dodg->cn);
        free(dodg->a);
        free(dodg);
}

/*****************************************************************
 *  Allocates memory for a dodgsn structure
 *  INPUT: size - the size of the dodgsn problem = (numvoter*numalter)
 *  RETURNS: a Dodgson structure (filled with garbage)
 *****************************************************************/

static dodgsn_t *dodgsn_alloc(int size) {
        dodgsn_t *dodg;
        int max_nz=3*size;    /* Max number of non-zero elements in lp matrix*/

        dodg=malloc(sizeof(dodgsn_t));
        if (dodg!= NULL) {
                dodg->nodearray=malloc((size+1)*sizeof(node_t));
                dodg->rn=malloc((max_nz)*sizeof(int));
                dodg->cn=malloc((max_nz)*sizeof(int));
                dodg->a=malloc((max_nz)*sizeof(double));
                if (dodg->nodearray&&dodg->rn&&dodg->cn&&dodg->a) {
                        /*fine*/
                } else {
                        dodgsn_free(dodg);
                        dodg=NULL;
                }
        }
        if (dodg==NULL) {
                printf("********** ERROR **********\n");
                printf("**Could not allocate Memory**\n");


                error("Could not allocate Memory.");

        }
        return(dodg);
}
```

```
static int dodgsn_lowerbound(
                        int* buffer,
                        node_t *nodearray,
                        int nextnode,
                        int numalters,
                        int numvoters
) {
        int a;
        int total_min_swaps;
        int max_losses;
        node_t *n;

        max_losses=(double)((numvoters-1)/2);
        for (a=1;a<=numalters;a++) {
                buffer[a]=-max_losses;
        }
        for (n=nodearray+1;n<nodearray+nextnode;n++){
                        buffer[n->a]+=n->count;
        }

        total_min_swaps=0;
        for (a=1;a<=numalters;a++) {
                if (buffer[a] > 0){
                        total_min_swaps+=buffer[a];
                }
        }

        return (total_min_swaps);
}



/*****************************************************************
 * calc_dodgsn_scores: calculates all Dodgeson scores for a set of votes
 * ***************************************************************
 *
 * INPUTS:
 *      dodg  - a preallocated chunk of memory
 *      d     - the alternative to calculate the dodgsn score for.
 *      votes - An array of voter preferences
 *              votes[0]=1st voter's first preference
 *              votes[1]=1st voter's second preference
 *                 ...
 *              votes[numalters-1]=1st voter's last preference
 *              votes[numalters]=2nd voter's first preference
 *              votes[numalters+1]=2nd voter's second preference
 *
 *      numalters -  number of alternatives that voters can vote for
 *      numvoters -  number of voters
 *
 * RETURNS: the dodgsn socore for the alternative d.
 *
 * NOTE: does not check that array contains valid data. If it does not memory
 *   corruption may occur
 *
 *
```

```
 ************************************************************************/

static double calc_dodgsn_score(
                dodgsn_t *dodg,
                alter_t d,
                double *votes,
                int numalter,
                int numvoter,
                double *real_lbound_ptr) {

        node_t *nodearray;
        int max_swaps;
        double Z,real_Z;
        int nextnode;
        nodearray=dodg->nodearray;


        CreateDodgsonGraph(nodearray,&nextnode,d,numalter,numvoter,votes);


/*      printf ("%d ++++\n",dodgsn_lowerbound(dodg->rn,nodearray,nextnode,
                                                        numalter,numvoter));*/
/*      min_swaps=dodgsn_lowerbound(dodg->rn,nodearray,nextnode,
                        numalter,numvoter);*/

/*      for (n=nodearray;n<nodearray+nextnode;n++)
                printf("i:%d c:%d u:%d d:%d l:%d a:%d\n",
                        n-nodearray,n->count,n->up,n->down,n->left,n->a);*/
        max_swaps=InitalDodgsonScore(nodearray,nextnode);

        if (nextnode<=1){
                Z=0.0; real_Z=0.0;
        } else {

         CreateInequalities(dodg,nodearray,nextnode, d, numalter, numvoter);
         lpx_set_int_parm(dodg->glpk,LPX_K_MSGLEV,1);
         real_Z=solve_lp_real(dodg);
         Z=solve_lp_integer(dodg);

/*       printf("%d:%d-%g=%g >= %d\n",d,max_swaps,Z,max_swaps-Z,min_swaps);*/
         lpx_delete_prob(dodg->glpk);
        }

        if (real_lbound_ptr!=NULL)
                *real_lbound_ptr=(max_swaps-real_Z);
        return(max_swaps-Z);
}

/* Finds lower bounds (DQ-scores) on all the dodgson scores
static void dodgsn_lowerbounds(
                        double *lbounds,
                        double *votes,
                        int numalter,
                        int numvoter
) {
        int max_losses;
```

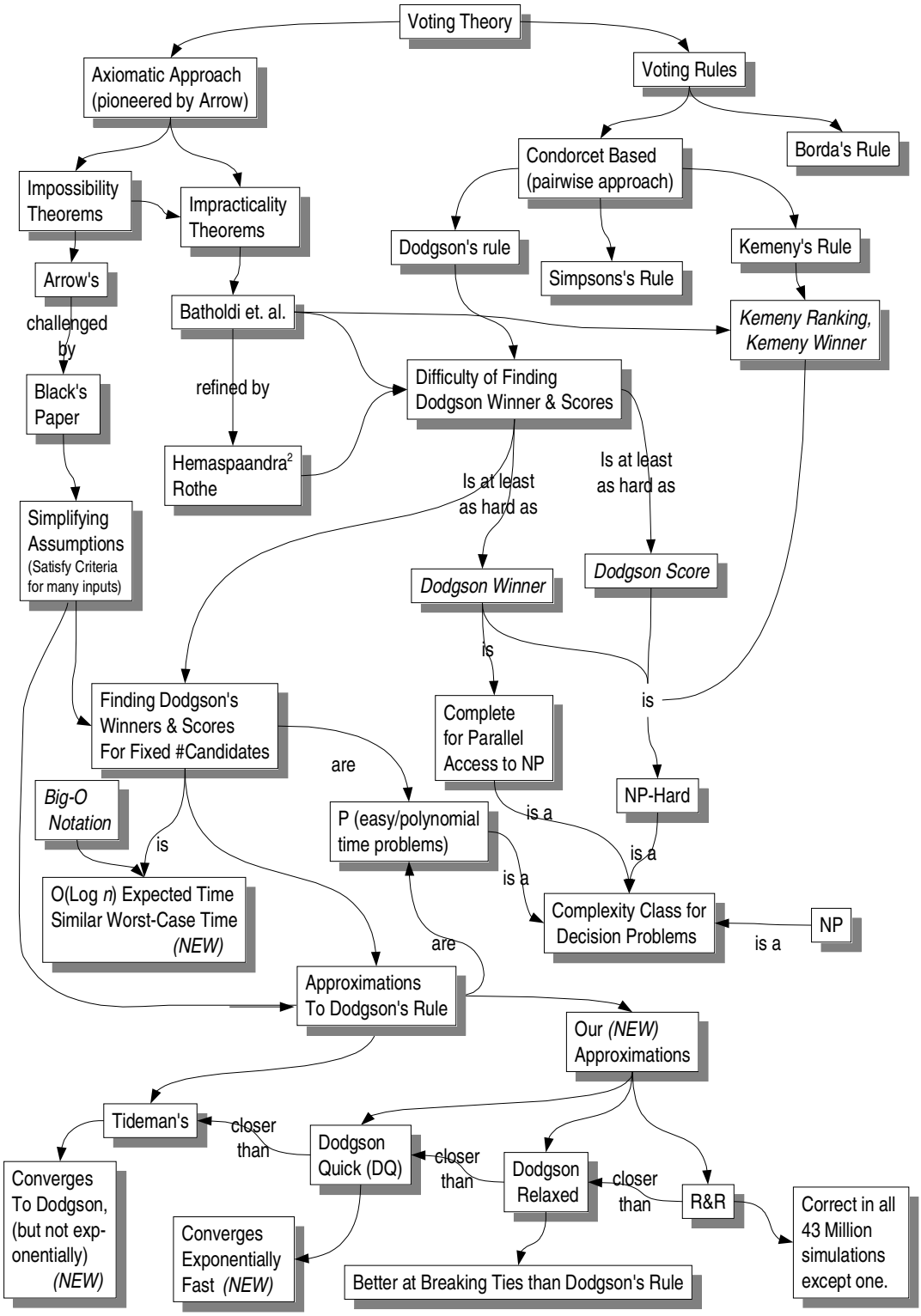
```
        int *lostto;
        int lost;
        int min_swaps;
        int above,below;
        double* vote;
        int v,i,j;

        max_losses=numvoter/2;
        /*max_losses=(numvoter-1)/2;*/


        lostto = calloc (sizeof(int),numalter*numalter);
        if (lostto==NULL)
                error("Could not allocate Memory for lostto.");
        for(i=0;i<numalter*numalter;i++) {
                lostto[i]=0;
        }

        vote=votes;
        for (v=0;v<numvoter;v++) {
                for (i=1;i<numalter;i++) {
                        for (j=0;j<i;j++) {
                                above=(int)vote[j];
                                below=(int)vote[i];
                                lostto[(below-1)*numalter+(above-1)]++;
                        }
                }
                vote=vote+numalter; /* goto next vote */
        }

        for (i=0;i<numalter;i++) {
                        min_swaps=0;
                        for (j=0;j<numalter;j++) {
                                lost=lostto[(i)*numalter+(j)]++;
/*                              printf("        lost %d\n",lost);*/
                                if (lost > max_losses) {
                                        min_swaps+=lost-max_losses;
                                }
                        }
                        lbounds[i]=(double)(min_swaps);
/*                      printf("%d\n",min_swaps);*/
        }
        free(lostto);
}

/*********************************************************************
 * calc_dodgsn_scores: calculates all Dodgeson scores for a set of votes
 * *******************************************************************
 *
 * INPUTS: votes - An array of voter preferences
 *              votes[0]=1st voter's first preference
 *              votes[1]=1st voter's second preference
 *                 ...
 *              votes[numalters-1]=1st voter's last preference
 *              votes[numalters]=2nd voter's first preference
 *              votes[numalters+1]=2nd voter's second preference
```

```
 *
 *         numalters -  number of alternatives that voters can vote for
 *         numvoters -  number of voters
 *
 * OUTPUTS: scores - An array of the Dodgson Scores for each alternative
 *
 * NOTE:
 *   - does not check that array contains valid data. If it does not memory
 *   corruption may occur
 *
 *   - Does not calculate exact dogdson scores for alternatives which we
 *   know are not winners, I.e alternatives for which even the
 *   DQ-score (lower-bound) is greater than the Dodgson score for some other
 *   alternative. This does not affect the set of tied winners calculated
 *   from these scores.
 *
 ************************************************************************/

static void calc_dodgsn_scores(
                double *lbounds,
                double *real_lbounds,
                double *scores,
                double *votes,
                int numalter,
                int numvoter)
{

 alter_t d;
 dodgsn_t *dodg;
 double large_number=numalter*numvoter; /* > max score */
 double min_score;
 double lowest_known;
 double score;
 int min_score_index;
 int i;

 dodgsn_lowerbounds(lbounds,votes,numalter,numvoter);
 memcpy(scores,lbounds,sizeof(double)*numalter);
 memcpy(real_lbounds,lbounds,sizeof(double)*numalter);

 dodg = dodgsn_alloc(numalter*numvoter);

 lowest_known=HUGE_VAL;
 min_score=large_number;
 while (lowest_known > min_score) {
         min_score=scores[0];
         min_score_index=0;
         for(i=1;i<numalter;i++) {
                 if (scores[i]<min_score) {
                         min_score_index=i;
                         min_score=scores[i];
                 }
         }
         if (min_score <= lowest_known) {
                 /* as min_score is currently an upper bound, we do not
                  * know if min_score >= lowest_known */
```

```
                d=min_score_index+1;

/*              printf("LB: %g = ",min_score);*/
                score=calc_dodgsn_score(dodg,d,votes,numalter,numvoter,
                                &(real_lbounds[min_score_index]) );

                /* Add a large number to score so we don't pick it again */
                scores[min_score_index]=score+large_number;

                if (score<lowest_known) {
                        lowest_known=score;
                }
         }

 }
 for(i=0;i<numalter;i++) {
        if (scores[i] >= large_number) {
                scores[i]-=large_number;
        }
 }
 dodgsn_free(dodg);

}



#ifdef HAS_MAINLINE

int main() {
        double vote[255];
        double scores[255];
        double lbounds[255];
        double real_lbounds[255];

        alter_t d;
        int numalter=4;
        int numvoter=21;
        make_slinkos_4alt_problem(vote);

        calc_dodgsn_scores(lbounds,real_lbounds,scores,vote,numalter,numvoter);
        for (d=1;d<=numalter;d++){
                printf ("Ds[%d]=%g\n",d,scores[d-1]);
        }


        return(0);
}

#endif

/* INPUTS
 * votes is a MxN array of votes.
 * In C it appears as a 0..MxN-1 array of doubles
 * votes[j*numalters+i] is voter j's ith preference
 *
 * Returns:
```

```
 * 0 - An array of lower bounds on the Dodgson scores
 *              These lower bounds are O(m(n+m)) to compute
 *              Are based on the Case where there are no "wasted swaps"
 * 1 - As above, but tightens the lower bounds by using a real approximation
 *      to the Dodgson LP.
 *              Note only tightens the bounds of alternatives which might be
 *              a dodgson winner
 * 2 - As above, but uses integer linear programming to find exact values
 *              for the dodgson scores of alternatives which might be
 *              a dodgson winner.
 *
 *              If you use these scores you will always find the correct dodgson
 *              winner.
 * an array Scores where Score(i) is the Dodgeson Score of the ith alternative*/
#ifdef MATLAB
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
                 const mxArray *prhs[])
{
  double *scores;
  double *real_lbounds;
  double *lbounds;
  int mrows,ncols;
  int numvoters,numalters;
  double *votes;

  /* Check for proper number of arguments. */
  if (nrhs != 1) {
    mexErrMsgTxt("One input required.");
  } else if (nlhs > 3) {
    mexErrMsgTxt("Too many output arguments");
  }

  mrows = mxGetM(prhs[0]);
  ncols = mxGetN(prhs[0]);

  numvoters=ncols;
  numalters=mrows;

  /* The input matrix must contain scalar doubles.*/
  if (!mxIsDouble(prhs[0])) {
    mexErrMsgTxt("Matrix elements must be scalar doubles.");
  }

  /* Create matrix for the return argument. */
  plhs[0] = mxCreateDoubleMatrix(1,numalters, mxREAL);
  plhs[1] = mxCreateDoubleMatrix(1,numalters, mxREAL);
  plhs[2] = mxCreateDoubleMatrix(1,numalters, mxREAL);
  lbounds = mxGetPr(plhs[0]);
  real_lbounds = mxGetPr(plhs[1]);
  scores = mxGetPr(plhs[2]);

  votes=mxGetPr(prhs[0]);

  calc_dodgsn_scores(lbounds,real_lbounds,scores,votes,numalters,numvoters);
}
#endif
```

Figure B.1: Overview of Concepts Relevant to this Thesis



Legend: $a \rightarrow b$ indicates that the concept b was derived from a.

This figure is included solely to assist those who find visual summaries easier to understand. It does not introduce any new material, nor is it otherwise required to understand this thesis.

Reference List

Alon, N. and Spencer, J. H. *The probabilistic method*. Wiley, 2nd edition, 2000.

Anderson, T. W. *An Introduction to Multivariate Statistical Analysis*. John Wileys and Sons, Brisbane, 2nd edition, 1984.

Arrow, J. K. *Social choice and individual values*. Wiley, New York, 1951,1963.

Barry, P. L. Radiation resistant computers. Feature, Science@NASA, 2005. `http://science.nasa.gov/headlines/y2005/18nov\_eaftc.htm`.

Bartholdi, III., Tovey, C. A., and Trick, M. A. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare: Springer-Verlag*, 6:157–165, 1989. Industrial and Systems Engineering, Georgia Institute of Technology.

Bazgan, C. . Ph.D. thesis, INRIA, Orsay, France, 1995.

Berg, S. Paradox of voting under an urn model: The effect of homogeneity. *Public Choice*, 47:377–387, 1985.

Berg, S. and Lepelly, D. On probability models in voting theory. *Statist. Neerlandica*, 48:133–146, 1994.

Black, D. On the rationale of group decision-making. *Journal of Political Economy*, 56:23–34, 1948.

Black, D. *Theory of commitees and elections*. Cambridge University Press, Cambridge, 1958.

Black, D. On arrow's impossibility theorem. *Journal of Law and Economics*, 12(2):227–248, 1969.

Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

Dantzig, G. B. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.

Debord, B. *Axiomatisation de procédures d'agrégation de préférences*. Ph.D. thesis, Université Scientifique, Technologique et Médicale de Grenoble, 1987.

Dembo, A. and Zeitouni, O. *Large deviations techniques*. Johns and Barlett, 1993.

Dodgson, C. L. *A method for taking votes on more than two issues*. Clarendon Press, Oxford, 1876. Reprinted in (Black, 1958) with discussion.

Downey, R. G. Parameterized complexity for the skeptic. 2003. (invited paper). In Proc. of 18th IEEE Conference on Computational Complexity, July 2003. `citeseer.nj.nec.com/downey03parameterized.html`.

Downey, R. G. and Fellows, M. R. Fixed parameter tractability and completeness i: Basic theory. *SIAM Journal of Computing*, 24:219–44, 1995.

Dummett, M. The borda count and agenda manipulation. *Social Choice and Welfare*, 15(2):289–96, 1998.

Durret, R. *Probability: Theory and Examples*. Thomson Brooks/Cole, Australia, 3rd edition, 2005.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the web. In *Tenth International World Wide Web Conference*, pages 613–22. 2001. `http://www10.org/cdrom/papers/577/`.

Eisenbrand, F. Fast integer programming in fixed dimension. In Di Battista, G. and Zwick, U., editors, *Algorithms - ESA 2003 : 11th Annual European Symposium*, volume 2832 of *Lecture Notes in Computer Science*, pages 196–207. Springer, Budapest, 2003. ISBN 3-540-20064-9. `http://citeseer.ist.psu.edu/eisenbrand03fast.html`.

Erdös, P. and Moser, L. On the representation of directed graphs as unions of orderings (in english). *Publ. Math. Inst. Hung. Acad. Sci.*, 9:125–132, 1964.

Fishburn, P. C. Condorcet social choice functions. *SIAM Journal on Applied Mathematics*, 33:3:469–489, 1977.

Gibbard, A. Manipulation of voting schemes: A general result. *Econometrica*, 41, 1973.

Gonzaga, C. C. An algorithm for solving linear programming problems in $o(n^3l)$ operations. *Progress in mathematical programming (Pacific Grove, CA, 1987)*, pages 1–28, 1989.

Hemaspaandra, E., Hemaspaandra, L., and Rothe, J. Exact analysis of dodgson elections: Lewis carroll's 1876 voting system is complete for parallel access to np. *Journal of the ACM*, 44(6):806–825, 1997.

Homan, C. M. and Hemaspaandra, L. A. Guarantees for the success frequency of an algorithm for finding dodgson-election winners. 2005.

Karatsuba, A. and Ofman, Y. Multiplication of many-digital numbers by automatic computers. *Doklady Akad. Nauk SSSR*, 145:293–94, 1962. Translation in Physics-Doklady 7, 595-596, 1963.

Karmarkar, N. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

Kemeny, J. Mathematics without numbers. *Daedalus*, 88:577–91, 1959.

Kemeny, J. G., Snell, J. L., and Thompson, G. L. *Introduction to Finite Mathematics*. Prentice-Hall, Inc., New Jersey, 3rd edition, 1974.

Khachian, L. G. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 147:191–94, 1979.

Laslier, J.-F. *Tournament solutions and majority voting*. Springer, Berlin - New York, 1997.

Lenstra, Jr., H. W. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

McCabe-Dansted, J. C. LyX grammar checker. 2005. `http://wiki.lyx.org/Tools/LyX-GrammarChecker`.

M^{c}Cabe-Dansted, J. C. and Slinko, A. Exploratory analysis of similarities between social choice rules. *Group Decision and Negotiation*, 15:1–31, 2006. `http://dx.doi.org/10.1007/s00355-005-0052-4`.

M^{c}Connell, S. *Rapid Development*. Microsoft Press, Washington, USA, 1996. ISBN 1-55615-900-5.

M^{c}Garvey, D. C. A theorem on the construction of voting paradoxes. *Econometrica*, 21:608–610, 1953.

Norman, L. J. and Samuel, K. *Discrete Distributions*. Houghton Mifflin, Boston, 1969.

Nurmi, H. Voting procedures: A summary analysis. *British Journal of Political Science*, 13(2):181–208, 1983.

Risse, M. Arrow's theorem, indeterminacy, and multiplicity reconsidered. *Ethics*, 111(4):706–34, 2001.

Risse, M. Why Count de Borda cannot beat the Marquis de Condorcet. *Social Choice and Welfare*, 25(1):95–114, 2005.

Saari, D. G. Capturing the "will of the people". *Ethics*, 113:333–349, 2003.

Saari, D. G. and Merlin, V. R. A geometric examination of kemeny's rule. *Social Choice and Welfare*, 17(3):403–38, 2000.

Satterthwaite, M. A. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Discussion Papers 122, Northwestern University, Center for Mathematical Studies in Economics and Management Science, 1974. `http://ideas.repec.org/p/nwu/cmsems/122.html`.

Schönhage, A. and Strassen, V. Schnelle Multiplikation großer Zahlen. (German) [Fast multiplication of large numbers]. 7(3–4):281–292, 1971. ISSN 0010-485X (printed version), 1436-5057 (electronic version).

Seber, G. A. F. *Linear Regression Analysis*. Wiley, Sydney, 1977.

Shah, R. *Statistical Mappings of Social Choice Rules*. Master's thesis, Stanford University, 2003.

Simpson, P. B. On defining areas of voter choice: Professor tullock on stable voting. *The Quarterly Journal of Economics*, 83(3):478–90, 1969.

Stanley, R. P. *Enumerative combinatorics. Vol. 1.* Cambridge University Press, 1997.

Stearns, R. The voting problem. *Am. Math. Mon.*, 66:761–3, 1959.

Tideman, T. N. *Social Choice and Welfare*, 4:185–206, 1987.

Vaidya, P. M. An algorithm for linear programming which requires $o(((m+n)n^2+(m+n)^{1.5}n)l)$ arithmetic operations. *Math. Programming*, 47(2):175–201, 1990.

Vidu, L. An extension of a theorem on the aggregation of separable preferences. *Social Choice and Welfare*, 16(1):159–67, 1999.

Walpole, R. E. and Myers, R. H. *Probability and Statistics for Engineers and Scientists*. Maxwell Macmillian International, Sydney, 1993.

Young, H. P. and Levenglick, A. A consistent extension of condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300, 1978.

Index