# Three Essays on Game Theory and Computation

Submitted by Elham Nikram to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Economics in December 2016



I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

Signature: ……..

## Abstract

The results section of this thesis includes three chapters (Chapter 2, 3 and 4). The first two chapters are on theoretical game theory. In both chapters, by mathematical modelling and game theoretical tools, I am predicting the behaviour of the players in some real world issues.

Hoteling-Downs model plays an important role in the modern political interpretations. The first chapter of this study investigates an extension of Hoteling-Downs model to have multi-dimensional strategy space and asymmetric candidates. Chapter 3 looks into the inspection game where the inspections are not the same in the series of sequential inspections. By modelling the game as a series of recursive zero-sum games I find the optimal strategy of the players in the equilibrium.

The forth chapter investigates direct optimization methods for large scale problems. Using Matlab implementations of Genetic and Nelder-Mead algorithms, I compare the efficiency and accuracy of the most famous direct optimization methods for unconstraint optimization problems based on differing number of variables.

This thesis is dedicated to my parents, Ahmad & Nadia.

## Acknowledgments

I wish to extend my sincere gratitude to my first supervisor, **Dieter Balkenborg,** for his advice, encouragement and guidance to me while conducting this research study. It has been an honour to have his supervision. He was a great mentor for me and had a great role in my professional development and progress of this thesis. I appreciate all his contributions of time and ideas to my PhD experience.

I would also like to thank my second supervisor, **Julian Neira**, who has added a lot of insights to the fourth chapter and developing the experiments.
I am especially grateful to **Shmuel Zamir** for all his insights which helped me to precede my second chapter and also his support allowing me to present my results at prestigious international conferences.

The Business School, University of Exeter provided a great academic atmosphere which is very supportive for the junior researcher. Besides, I gratefully acknowledge the 3 years scholarship by Business school, University of Exeter throughout my PhD which made it possible for me to start and complete this research study.

My special thanks also go to my family. First of all, my parents, **Ahmad** and **Nadia**, who have supported me throughout whole of my life with tremendous love; and my younger brother, **Mohammad Ali**, who has also helped me psychologically. Without their encouragement I would never have succeeded in completing my PhD.

# List of Contents

**Chapter 3**



**Chapter 4**

# List of Tables and Figures

**Chapter 2**



**Chapter 3**

**Chapter 4**

# Chapter 1

## Introduction

The general theme of the first and second results sections of this study (Chapters 2 and 3) involve mathematical modelling analysing the behaviour of the players in different real world situations by game theoretical tools; while the third results section (Chapter 4) employs Matlab based experiments to investigate the direct numerical optimization methods.

Chapter 2 extends the famous Hoteling-Downs model to the case where the preferences of the voters do not have to be single-peaked. Where the classical results show that the equilibrium point is unique, I show that the result is robust under small perturbations. However, the structure of the model and the equilibrium change when the perturbations are not small. I provide examples and define a criteria which describe the structure of the equilibrium points when the tie situation has been resolved by perturbation.

In Chapter 3, I investigate a version of the Inspection game first introduced by Dresher (1962), where the inspector may run a mixture of partial and full inspections. I investigate the behaviour of players in the equilibrium. I show that as long as the opportunity for a full inspection exists, the inspector never starts his sequential inspections with a partial inspection. As the game has been modelled as a zero sum game, by investigating the value of the game we have an efficient tool to compare the efficiency of the full and partial inspections.

In Chapter 4, I investigate the robust optimization methods and compare Nelder-Mead and Genetic algorithms. By use of experiments to address unconstrained optimization problems I show that Nelder-Mead algorithm

can be more efficient with regarding to accuracy and required time only when the objective function has a small number of parameters. Nevertheless, Nelder-Mead algorithm is sensitive to the position of the initial guess and may stick to a non-optimizer point or local optimum.

# Chapter 2

# A Generalized Hoteling-Downs model with Asymmetric Candidates

## 2.1 Introduction

The Hoteling–Downs model Hotelling (1929) and Downs (1957) describes a two-party two-stage model of an election, where in the first stage the two candidates commit themselves to a policy platform and in the second stage voters vote for one of the two candidates. The candidates are assumed to be opportunistic, i.e. they only care about winning the election. In the classical Hoteling-Downs model voters have single-peaked preferences over policies in a one dimensional policy space. In equilibrium, voters will vote for the candidate who chooses a policy closest to their ideal point and candidates will choose a policy which gets the most votes. In an equilibrium where all voters use undominated strategies, both candidates propose the ideal policy of the median voter. By this, candidates guarantee to get at least 50 percent of the whole votes. Although Hotelling-Downs model has a significant role in the modern political interpretations, nonetheless some consider the model over simplistic and the restrictive assumptions on the preferences to guarantee existence of the equilibrium make some researchers like Kramer (1973) believe that the model cannot be a good model of reality. There may be many factors that make the model unrealistic. It is a recognized phenomena that the voters do not just care about the policy platform of a candidate. Instead charismatic behaviour, incumbency, reputation, etc. also influences the voter. It means that if we take a more realistic point of view, even with the same policy platform, one candidate can show some advantages over the other one. Cummings (1966) analysed the data of US presidential elections from 1924 to 1964. The result of his analyses confirms a higher chance of winning for the incumbents of the election. Similar analyses have been completed by R.

Chacrabarti et al. (2005) on the Lok Sabha elections in India. In contrast to the voters in US, they detect an anti-incumbency behaviour amongst the voters in India. However, they could not recognise any pattern for this behaviour which suggests that, based on the specific time and situation, the preference of the voter regarding incumbency may change. Meanwhile, they admit incumbency is always the important matter for voters in India. To address the asymmetry between the candidates theoretically in Hotelling-Downs model, Aragones and Palfrey (2002) analysed the classical Hotelling-Down model where one of the candidates can take advantages over the other one. They suppose that one of the candidates loses the election unless he chooses a platform quite close to the ideal point of the voters. The location of the median voter's ideal point has a specific distribution. The authors show that in this case the pure strategy equilibria may fail to exist. They also discuss the characteristics of the mixed strategy equilibrium point.

The other consideration about the classical Hotelling-Downs model is the number of the policy platforms for the candidates. It is often more reasonable to assume that the strategy space of the candidates is not unidimensional and single peaked. In fact, the empirical results of many studies confirm the multi-dimensional nature of real world voting procedures. Stockes (1963) provides some empirical observation of US elections which shows the electoral support of the candidates cannot be a single dimensional space. Several studies have been done to extend the strategy space of the candidates in different versions of the model. Some prominent works in this area are Plot (1967), Davis, Groot and Hinich (1972); Wendell and Thorson (1974) and McKelvey and Wendell (1976).

In this study I suppose that voters have strong preferences over a finite number of policy platforms. It has been argued by Robert Dahl (1956, pp. 37-38) that in a democratic society the only compatible rule is the majority preference. Hence, it is important that we analyse cases where there is more than one alternative and voting procedure is the majority voting. Miller (1977) shows that any majority preference of the voters over finite number of the alternatives can be represented as a direct graph called "tournament". Conversely, McGarrey (1953) and Stearns (1959) show that any tournament demonstrate at least one profile of voters' preferences.

The immediate result of extending the strategy space to such a multidimensional strategy space is a possibility of losing the Condorcet winner[1]. In fact, Miller (1977) shows that if the representing tournament of a majority preference of voters has cycles, then there would be no Condorcet winner. He considers the a case of majority voting procedure where the orders of the proposals are voted is also a matter. To analyse the undominated proposals, he introduced the Condorcet set (minimal undominated set) and discuss under which condition the sincere voting decision belongs to this set.

In another work, Miller (1980) shows that the "uncovered sets" of the game have more desirable characteristics and are more beneficial to be investigated as the solution to the game.

G. Laffond and J.F. Laslier (1992) viewed use tournament graphs to study an extension of the Hotelling-Downs model without single-peaked preferences. They look at the two-stage model, where first two candidates bindingly propose a policy play form and then an odd number of voters, whose majority preferences are described by the tournament, vote for one of the two candidates. They show that eliminating dominated strategies of the voters in the game leads to a zero-sum game between the two candidates. By employing the classical concepts of the zero-sum games they analysed the Maxmin strategies of the game. They show that the equilibrium of this so-called "Tournament Game" is unique, but will often involve mixed strategies. In fact, in equilibrium players are always mixing between an odd number of strategies[2]. A central assumption made in the paper is that if both candidates propose the same candidate platform, then they will have an equal chance of winning and hence the expected payoff is zero.

This assumption is quite restrictive. Hence, by relaxing this assumption, we can model the situation where there is an incumbent in the election. In fact, in my model I address asymmetry among the candidates in a multi-dimensional space. As I mentioned before, this is closer to real world situations. Additionally, I also analyse the case when both parties have

[1] In a majority voting procedure, a Condorcet winner is the candidate who wins in all the pairing against the other candidates.

[2] D. C. Fisher and J. Ryan (1992) show the same results about the equilibrium of the tournament game separately and differently, without mentioning its relation to Hotelling-Downs model.

incentive/disincentive to choose the same platform. The incentive/disincentive might have occurred by a third party or collusion. In general, I show how the results depend on resolving the tie situation. The classical tournament game studied by D. C. Fisher and J. Ryan (1992) is a symmetric zero-sum game, so to be more specific I refer to it as a symmetric zero-sum tournament game. In this study the first player (possibly the incumbent) is called Player 1 or Candidate 1 and the other player is Player 2 or Candidate 2. Besides, the optimal strategy for the players is the Maxmin strategy. This study has been organized in seven sections.

After the introduction, in Section 2.2 I provide a literature review on generalizing the Hotelling-Downs model which has been utilised by many researchers over the years. In Section 2.3, I present tournament game to model the Hoteling-Downs model $n$-dimensional strategy space. In Section 2.4, I briefly review the main characteristics of symmetric zero-sum tournament game which have been studied by both D. C. Fisher and J. Ryan (1992) and G. Laffond and J.F. Laslier (1992). We see that the mixed equilibrium point in the model is unique, also in equilibrium players are mixing between odd number of the strategies. Adding to the previous knowledge about the symmetric tournament game I prove that the unique equilibrium point is also regular in the sense of Harsanyi. Later, in Section 2.5 the assumption of symmetry among the candidates is relaxed. By employing the regularity characteristic of the equilibrium, I show that the uniqueness of the equilibrium will be kept if the amount of the asymmetry is negligible among the candidates.

In Section 2.5, I also analyse the behaviour of the equilibrium point under certain large perturbations of the symmetric zero-sum tournament game, namely “asymmetric zero-sum tournament game” and “symmetric non zero-sum tournament game”. We can observe that under large values of perturbation the structure of the game can be very different and the game can have several equilibria with varying support. A symmetric equilibrium does not have to exist. However, in Section 2.6 I provide sufficient conditions for the uniqueness of the equilibrium in these two classes of games. Section 2.7, investigates and analyses the equilibrium point for the regular tournament games. I show that the incumbency in any level does

not affect the equilibrium point. I also calculate the value of the game for a regular tournament game.

## 2.2 Literature Review on Generalization of Hoteling-Downs Model

Despite the significant role of the Hotelling-Downs model on modern political interpretations, over simplicity of the model is quite restrictive. In this section I review some of the studies which have generalized the Hoteling-Downs model with respect to asymmetry between the candidates and dimension of the strategy space.

The asymmetric among the voters in the classical Hoteling-Downs model has been targeted in many studies. Some of the more famous studies are Anderson and Glomm (1991), Ingberman (1992) and E. Aragones and T. R. Palfrey (2002).

Anderson and Glomm (1991) have considered two candidates election where the voters care about two different features of a candidate. Firstly, the policy they choose and secondly the non-policy factors such as charismatic behaviour, incumbency or integrity. The non-policy features are the factors that may cause asymmetry between the candidates and will make one candidate gain an advantage over the other. A candidate on the one hand likes to choose a policy platform close to the median voter's ideology; however he also does not want to compromise his ideal policy because of the non-policy features. The median's voter evaluation of the non-policy advantages of the candidate $i$, is shown by $\alpha_i + \varepsilon_i$ where $\alpha_i$ is the scale for measuring the advantages of the candidates $i$ and $\varepsilon_i$ is the standard normal distribution. Another function like $f(|x_i|)$ also measures the evaluation of the median's voter of the candidate's policy platform $(x_i)$. Hence, the utility of the median voter will be $u(x_i) = \alpha_i + \varepsilon_i - f(|x_i|),\ i = 1,2$. This form of modelling gives the chance to analyse the Nash and Stackelberg equilibrium. Hence, the authors find that there is difference in behaviour of the candidates when they have simultaneous move or the moves in order. In fact, if I suppose that the distribution and the position of the median voter is common knowledge for both candidates, the incumbent's equilibrium strategy is much closer to his ideal point.

Another work to address the asymmetric candidates in Hotelling-Downs model is E. Aragones and T. R. Palfrey (2002) where one of the candidates obtains advantages over another. The distribution of the median voter ideal policy is a common knowledge for both of candidates and they can select their strategy from the finite number of positions. In fact, they suppose the policy space as $x_i = \frac{i-1}{n-1}, i = 1, \dots, n$ on the interval $[0,1]$, where voters have Euclidean preferences. If we call the advantaged candidate by Candidate A (with committed strategy $x_A$) and the disadvantaged one by Candidate D (with committed strategy $x_D$) then a voter with strategy preference $x_i$, has the utility if Candidate A wins $U_i(x_A) = \delta - |x_i - x_A|$ and $U_i(x_D) = -|x_i - x_D|$ where $\delta > 0$. The model is equivalent to the classical Hotelling-Downs model if $\delta = 0$. It means that $\delta$ determines the advantage of Candidate A over the other Candidate. In first step, the authors show that there is no pure strategy for the candidates in this game, and they always mix between their strategies. Later, they analyse the model for a small value of $\delta$ $(0 < \delta < \frac{1}{n-1})$ and large one. They discover that the behaviour of the candidates are different for small and large value of $\delta$. In fact, for small value of $\delta$, Candidate A wins if and only if his ideal policy and the ideal point are as close as Candidate D's policy and ideal point. This is not a case for large value of $\delta$. Hence, the equilibrium is different in these cases. They also show that the solution depends on the even or odd number of the strategies and the advantaged candidates has always higher expected payoff. However for very large number of strategies his advantage shrinks to zero.

The other important issue analysed in many studies is the number of strategies. The assumption of a one dimensional strategy space is far from the reality. However, multi-dimensional strategy space gives rise to a number of new issues such as preferences of the individuals in a group or combination of strategies, ordering the preferences over the strategies, complexity of strategy space etc. I should also mention the possibility of the non-existence of a Condorcet winner, which is known as Condorcet's voting paradox.

To address the multidimensional strategy space, O. A. Davis, M. H. GoorDeGroot and M. J. Hinich (1972) considered the each alternative of the voters as a point in Euclidean space. It means that a point like

$\acute{x} = (x_1, x_2, \dots, x_n)$ represents a possible alternative. Each individual in the society has the same set of dimension of choice and locate the alternatives in the Euclidean space similarly. The space of all alternatives is called $E^n$. They suppose that each individual has a preferred point like $x$, hence his utility over all the other strategies is defined as follow

$$u_i(\|y - x\|) = \|y - x\|^2 = (x - y)'(x - y).$$

By defining above utility function, they also define the preference relation over the strategies. By their definition alternative $y$ is preferred to $z$ ($yRz$) if $\Pr(\|y - X\| \leq \|z - X\|) \geq 1/2$. In fact, $y$ is preferred to $z$ if and only if at least half of the population prefers $y$ to $z$.

This relation alone is not transitive; however the authors provide the necessary and sufficient condition to define a transitive relation over $E^n$ by $R$. This transitive relation can completely orders the point in $E^n$. Besides, the same necessary and sufficient condition for the transitive relation can be employed to show the existence of the Condorcet point.

In another work, R. E. Wendell and S. J. Thorson (1974) consider the point $\acute{x}_i = (x_i^1, x_i^2, \dots, x_i^n)$ as the position of each voter regarding the different strategies. By knowing the most preferred location of each voter they define a loss function for all the other positions. The loss function can define some indifferent contours[3] for each voter which are equivalent to the concept of Norm functions[4]. By this mean, it is proved that if all the voters use the same norm then the equilibrium point is the position of the multidimensional median.

Miller (1977) discusses the majority voting procedure among $m > 1$ number of proposals where voting procedure may depend on the order in which the proposals are voted on. First of all, he argues that the majority preference of the voters can be shown in a directed graph name "tournament". Later, by analyse the possible availability of paths and cycles between the vertices, he investigates the possibility of a Condorcet winner in his model. He shows that when there is no undominated strategy then there will be a cycle between the vertices. That means when

[3] A indifferent curve is a curve showing different bundle of strategies which the voter is indifferent.

[4] The function $\|.\| : R^n \longrightarrow R$ is the Norm function if it satisfies following conditions for each $x \in R^n$

1) $\|x\| \geq 0$. 2) $\|x\| = 0 \; iff \; x = 0$. 3) $\|\alpha x\| = \alpha\|x\|;\ \alpha > 0$. 4) $\|x + y\| \leq \|x\| + \|y\|$. 5) $\|-x\| = -\|x\|$.

there is no undominated strategy there will be also no Condorcet winner. Hence, he introduces the concept of minimal undominated set (Condorcet set). Condorcet set includes all the vertices which are not dominated by any other vertices; also any subset of this set has the same characteristics. The author shows that for his model represented by a tournament the Condorcet set exists and is unique. Hence, he also investigates the main characteristics of this set and shows that many binary majority voting processes can end up with a strategy selected in Condorcet set. However, despite being handy to analyse the majority voting systems, Condorcet set can be very large (possibly even equal to the set of the whole strategies) and it may include the Pareto-inefficient[5] points. So, later Miller in his other work in 1980 introduces another possible solution for his model named "uncovered set". The uncovered set is the set of all the vertices from which every other vertex is reachable with the path with no more than two vertices length. Miller (1980) shows that the uncovered set always exists and it is smaller than Condorcet set, however it includes all the Pareto-efficient points.

G. Laffond and J.F. Laslier (1992) are the first to use the tournaments to extend the Hotelling-Downs model to the set of the preferences which are not single peaked. Using weakly dominated strategies (and assuming that each candidate is elected with equal probability if they both propose the same policy) they reduce the model to a zero-sum game defined by a tournament graph. Hence, Maxmin strategies (see Neumann and Morgenstern (1944)) can be considered as the optimal solution of the game. They show that while the value of the game is zero, the optimal solution of the game is zero. The authors also investigate more characteristics of the optimal solution and show that in equilibrium candidates always mix between odd number of the strategies. This form of analysing seems to be more practical and inclusive rather than all the previous results. However it is still far from the reality as the main assumption of the model is symmetry among the candidates.

They are some studies like McKelvey and Ordeshook (1985) and Ansolabehere and Snyder (2000) which are capturing both

[5] Pareto-efficient is the state where it is impossible to increase the payoff of one player without decreasing the payoff another player.

multidimensional nature of the strategy space and asymmetry among the candidates, however there is no previous study to model the election with binary preferences over strategies with asymmetric candidates. In this study, I consider the model introduced first by G. Laffond and J.F. Laslier (1992) and relax the assumption of symmetry between the candidates.

## 2.3 The Model

Tournament game is introduced and analysed by D. C. Fisher and J. Ryan (1992) and G. Laffond and J.F. Laslier (1992). In this section I show and explain how these games can generalize the Hoteling-Downs model to the n-dimensional strategy space.

### 2.3.1 The Relation between Tournament Games and Multi-Dimensional Hotelling-Downs Model

In this section I describe the relationship between the Hotelling-Downs model of two-candidate elections, where voters can have arbitrary strict preferences over finitely many policy platforms, and certain matrix games that I will call them in general, tournament matrix games.

This extended Hotelling-Downs model is a game in extensive form whose players are the two candidates, Candidate 1 and Candidate 2, and an odd number of voters.

There are finite numbers of policy platforms ($n > 1$). Voters have strict preferences over the $n$ policy platforms, i.e. a voter is never indifferent between any two of them. The candidates are assumed to be opportunistic, i.e. a candidate gets utility +1 if he wins the election and -1 if he loses it. The timing is as follows: In the first stage each of the two candidates chooses simultaneously and independently a policy platform. The voters observe these choices and then each voter simultaneously and independently votes for one of the two candidates. No voter can abstain. Because the number of voters is odd, exactly one candidate will win the election by a simple majority rule. This candidate gets payoff +1, the other loses the election and gets payoff -1. The policy selected by the winner is implemented and this determines the utility of the voters.

It is easy to see that a strategy of a voter is undominated if and only if for any pair of distinct policy platforms offered by the two candidates the voter votes for the candidate who adopted his preferred among the two platforms. Suppose the two candidates select different policies and the voters use only undominated strategies. Then that candidate wins the election whose chosen policy is preferred by a majority of voters.

Which policy $i$ wins in a simple majority vote over which policy $j$ is usually summarized by a directed graph $T$ called a "tournament" in the literature (see Miller (1977)). This graph has $n$ vertices. There is an arc from node $i$ to node $j$ if a majority of voters prefer policy platform $i$ over policy platform $j$. I use $i \to j$ to show that strategy $i$ dominates strategy $j$ (similar notation to Miller (1977)). Alternatively, one can work with the associated $n \times n$ matrix $K(T) = (k_{ij})_{n\times n}$ defined as follows. The matrix has zeroes on the main diagonal. It has entry +1 in row $i$ and column $j$ if a majority of voters strictly prefer alternative $i$ over $j$ and entry -1 if it is the other way around. I call $K(T)$ the tournament matrix associated with the voter's preferences. Example 2.2 shows a simple example of a tournament and the matrix game of a symmetric zero-sum tournament game.

It is easy to see that in a perfect equilibrium of the game no voter will use a dominated strategy (see Selten (1973)). If both candidates offer the same policy platform, all voters are indifferent between the two candidates and hence every possible voting profile is optimal. Clearly, for any probability $0 < \alpha < 1$ there is a voting pattern such that the first candidate wins with probability $\alpha$. I hence have the following result.

**Theorem 2.1.** Consider a Hotelling-Downs model with arbitrary preferences of the voters over the finitely many policy platforms. Let $\boldsymbol{K}$ be the tournament matrix determined by the voters' preferences. Hence, for every perfect equilibrium of this game there exists a diagonal matrix $\boldsymbol{D}$ with entries not exceeding 1 in absolute value such that the strategies of the candidates in the equilibrium form a Nash equilibrium of the matrix game $\boldsymbol{K} + \boldsymbol{D}$. Conversely, let $\boldsymbol{D}$ be any diagonal matrix whose entries do not exceed 1 in absolute value. Then every Nash equilibrium of the matrix game $\boldsymbol{K} + \boldsymbol{D}$ can be extended to a perfect equilibrium of the Hotelling-Downs model.

I call a matrix game $M = K + D$ with $D$ as described a general tournament matrix game. The papers Fisher and Ryan (1992) and Laffond et al. (1993) consider the special tournament matrix games where the matrix $D$ is zero. I often refer to the matrix game $K$ as a symmetric zero-sum tournament game. This corresponds to the case where each voter votes for each candidate with equal probability when both candidates chose the same platform. In Section 2.4 I describe briefly the results of these papers.

### 2.3.2 Symmetric zero-sum tournament games

"Paper, Scissor and Stone" is the simplest form of a symmetric zero-sum tournament game where the player has three different strategies. The rules of the game can be shown with a graph (Figure 2.1) or a matrix game (Table 2.2). In this game two players simultaneously and independently choose a strategy among the three. If both choose the same strategy then the game has been tied otherwise scissor beats paper, paper will beat stone and stone beats scissor. The situation can be shown with a graph as follows

**Example 2. 2.**[6]

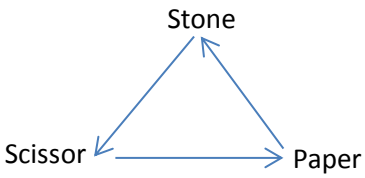



**Figure 2. 1**. The Graph representing "Scissor, Paper, Stone game"

| | Scissor | Paper | Stone |
|---|---|---|---|
| Scissor | 0 | 1 | −1 |
| Paper | −1 | 0 | 1 |
| Stone | 1 | −1 | 0 |

**Table 2. 2.** Matrix game of "Scissor, Paper, Stone game"

As we can see in this game there is a cycle between strategy preferences of the players. Hence, there is no undominated point which means that there is no Condorcet winner. However, if a player mixes between his strategies with probability of $\frac{1}{3}$, then he would assure himself the expected pay off of zero. In fact, this payoff is the maximal value he can assure

[6] In general by Miller (1977)'s results we know that there is always a profile of voters that yield the tournament graph. However, for certain graphs we may need a certain minimum number of voters (or a specific number of voters). In this example, we need at least three voters to yield the graph.

himself. Hence, $\left\{\frac{1}{3},\frac{1}{3},\frac{1}{3}\right\}$ is Maxmin strategy for the player. Maxmin strategies are also called optimal strategies for the player.

In Example 2.2 the alternatives can present possible real world alternatives for candidates. For example, scissor can represent "Lower taxation", paper "Free education" and stone "improve infrastructure". Empirical examples of the cycling preferences over alternatives and Condorcet paradox has been detected in real world situations through different studies. One example is the work by Kurrild-Klitgard (2001) about a poll of Danish voters preferred prime minister in 1994. In this study, by paired wise comparison between three candidates the author shows that voters have a cycling preference over candidates.

Example 2.3[7] provides another example of a symmetric zero-sum tournament game, which has six strategies. Similarly, to the "Paper, Scissor and Stone" in this game there is no Condorcet winner. The optimal mixed strategies for both players are $\left\{0,\frac{1}{3},\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{3}\right\}$.

**Example 2.3.**

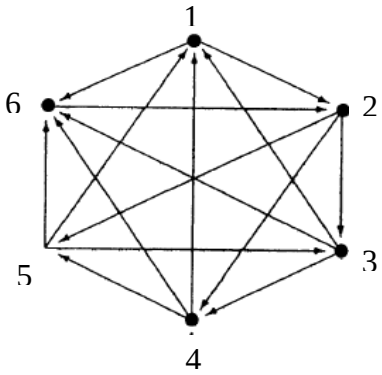



**Figure 2. 3.** A Tournament graph with six vertices representing a symmetric zero-sum tournament game with 6 strategies.

$$K=\begin{bmatrix} 0 & 1 & -1 & -1 & -1 & 1 \\ -1 & 0 & 1 & 1 & 1 & -1 \\ 1 & -1 & 0 & 1 & -1 & 1 \\ 1 & -1 & -1 & 0 & 1 & 1 \\ 1 & -1 & 1 & -1 & 0 & 1 \\ -1 & 1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

**Table 2. 4.** A matrix game of the graph in Figure 2.3, representing the same symmetric zero-sum tournament game of Figure 2.3.

We can find the cycle in this example where $6 \to 2 \to 3 \to 4 \to 5 \to 1 \to 6$. This means that there is no undominated strategy and therefore there is no Condorcet winner in the game. Besides, As we see in the "Paper, Scissor and Stone game" and Example 2.3; $K(T)$ is always a skew-symmetric matrix. It means that $-K = K^T$. To analyse the tournament games more precisely I provide following definitions.

[7] This example has been also analysed in D. C. Fisher and J. Ryan (1992).

**Definition 2. 4.** Let $P = (p_1, \dots, p_n)$ be a mixed strategy for a tournament game with $n$ strategies. Strategy $i$ is the support of $P$ if $p_i > 0$. $P$ is called a full support mixed strategy if each $p_i$ is strictly positive.

**Definition 2. 5**. Let $\boldsymbol{P = (p_1, \dots, p_n)}$ be an optimal strategy for a tournament game with $\boldsymbol{n}$ strategies. Strategy $\boldsymbol{i}$ is called a "bad quality" strategy if $\boldsymbol{p_i = 0}$, (see Laffond and Laslier (1993), pp. 187).

Hence, by Definition 2.5 Strategy 1 in Example 2.3 is bad quality strategy. We can observe easily that in Example 2.3 in the equilibrium, for a bad quality strategy the probability of winning is less than probability of losing, while for all the other strategies they are equal. This is the reason why both players play the bad quality strategy with zero probability.

In this extension of Hoteling-Down model there is no single peak preferences over the voters. The variety of voter's options makes the strategy space of each party multi-dimensional. For example, in Scissor, Paper and Stone game we see that there is no single preferred strategy but there is a cyclic preference over all the strategies. As the "tournament game" is symmetric and zero sum, to be more accurate we can refer to it as a "symmetric zero-sum tournament game".

## 2.4 Main Characteristics of the Model

### 2.4.1. The Value and Optimal Strategy of Symmetric zero-sum Tournament game

D. C. Fisher and J. Ryan (1992) and G. Laffond and J.F. Laslier (1993) in two separate works, discuss the value and optimal strategies of the symmetric zero-sum tournament games. Here, I provide the brief review of the main theorems and characteristics of the symmetric zero-sum tournament game analysed in both papers.

In a symmetric zero-sum tournament, $T$, with $n$ different strategies for both players, if Player 1 plays a mixed strategy $P$ and Player 2 responds with strategy $i$ (node $i$) then the expected payoff is $((K(T)P)_i$. In a maxmin strategy Player 2 wants to select a strategy to minimize Player 1's payoff (as it is a zero-sum game this will increase her payoff). On the other hand, player 1 wants to maximize his own payoff as well. Hence, if $v$

denotes the value of the game then $P$ is the optimal strategy of the game if it satisfies the following system.

$$v = max\left(min(K(T)P_i\right),\ P \geq 0,\ 1^T P = 1; P = (p_1, p_2, \dots, p_n).$$

(2.4.1.1)

We know that in a symmetric zero sum game the value is always zero, hence in the system (2.4.1.1), $v = 0$. As a result, the system (2.4.1.1) can be simplified to

$$\begin{cases} K(T)P \geq 0, \\ P \geq 0, \\ 1^T P = 1. \end{cases} \qquad (2.4.1.2)$$

**Lemma 2. 6.** Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be optimal strategies for a symmetric zero-sum tournament game on a tournament, *T*; then $\boldsymbol{q_i > 0}$ implies $(\boldsymbol{K(T)p})_i = \boldsymbol{0}$. (See Fisher and Ryan (1993)).

By Lemma 2.6 we can conclude that in a symmetric zero-sum tournament game for an optimal strategy all the nodes will be selected from those that make the other player expected payoff zero.

**Definition 2. 7.** A subtournament game[8], $\boldsymbol{\acute{T}}$, is positive if the optimal strategy is full support.

Following theorems and corollary show that in a symmetric zero-sum tournament games the optimal strategy of the players is unique.

**Theorem 2. 8.** Let $\boldsymbol{T}$ be a symmetric zero-sum tournament on $\boldsymbol{n}$ nodes, Then

$$Rank\big(K(T)\big) = \begin{cases} n, & if\ n\ is\ even, \\ n-1, & if\ n\ is\ odd. \end{cases}$$

**Corollary 2. 9**. In a positive symmetric zero-sum tournament game, in equilibrium players are mixing between odd number of strategies.

Proof:

$K(T)P = 0$ has a nonzero solution only if $Rank\big(K(T)\big) < n$. So $Rank\big(K(T)\big)$ is equal to $n$-1, and as a result $n$ is odd.

[8] The tournament game $\acute{T}$ is the sub tournament game of the tournament game $T$, if the graph of tournament game $\acute{T}$ is the subgraph of the graph of tournament game $T$.

**Theorem 2. 10.** The symmetric zero-sum tournament game on n nodes has a unique optimal strategy, p, such that $\boldsymbol{p_i > 0}$ on a positive subtournament (which must have an odd number of nodes).

Proof:

Let $p = (p_1, p_2, \dots, p_n)^T$ and $q = (q_1, q_2, \dots, q_n)^T$ be two solutions to (1.2). Let S be the subtournament of $T$ on those nodes where either $p_i > 0$ or $q_i > 0$ (or both). Since both p and q are solutions to (1.2), by lemma 1.2.1 we can conclude that $K(S)p_S = K(S)q_S = 0$ since $p_S \neq 0$ then by theorem 1.2.3 we know that the null space of K(S) has dimension at most one, $\exists \alpha \in \mathbb{R};\ p_S = \alpha q_S$. Since $1^T p_S = 1^T q_S = 1, p_S = q_S$ and hence *p=q.*

We know that $K(S)p_S = 0$ and $p_S > 0$, hence S is a positive tournament,■.

As the game is symmetric when one strategy is a bad quality strategy for one player, it will suppose bad quality strategy for other player as well. Hence, both players can delete that strategy from their set of strategies. The tournament game can be reduced to a sub tournament game where all the strategies are playing with positive probabilities.

**2.4.2. Regular Equilibrium in the Symmetric zero-sum Tournament Game**

In this section, I show that the unique equilibrium point in symmetric zero-sum tournament game is regular in the sense of Harsanyi (1973). (For more explanation about regular equilibrium see Appendix A.4)

**Definition 2. 11.** Let $\boldsymbol{G}$ be a matrix game of a two player zero-sum game, where $\boldsymbol{G_{n\times n}} = \begin{bmatrix} \boldsymbol{a_1} \\ \vdots \\ \boldsymbol{a_n} \end{bmatrix}$; then $\boldsymbol{C_{n\times n}} = \begin{bmatrix} \boldsymbol{(1, \dots, 1)} \\ \boldsymbol{a_2 - a_1} \\ \vdots \\ \boldsymbol{a_n - a_1} \end{bmatrix}$ is called the computational matrix of the matrix game $\boldsymbol{G}$.

**Definition 2. 12.** Let $\boldsymbol{G}$ be a matrix game of a two player zero-sum game, where $\boldsymbol{C}$ is its related computational matrix; then $\boldsymbol{C_{n\times n}P} = \begin{bmatrix} \boldsymbol{1} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix}_{\boldsymbol{n\times 1}}$ , $\boldsymbol{P \geq 0}$ is called computational system.

**Theorem 2. 13.** For a positive symmetric zero-sum tournament game, *T*, with $\boldsymbol{n}$ strategies, computational matrix of the matrix game has full rank.

Proof:

We know that for a positive symmetric zero-sum tournament game , *T*, the equilibrium is unique. I denote this unique optimal solution by $P = (p_1, \dots, p_n)$; where $p_i > 0$. I also denote the matrix game of the game by $K = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_n \end{bmatrix}$ and its related computational matrix by $C$. As $P$ is the maxmin strategy, hence it should satisfy following system

$$\begin{cases} Max\ v \\ \sum_{i=1}^{n} -KP^T \geq v, \\ \sum_{i=1}^{n} p_i = 1, \\ p_i \geq 0. \end{cases} \qquad (2.4.2.1)$$

where $v$ is the value of the game. As the value of the game is equal to zero, and $P$ is the only equilibrium, hence we have

$$S_i P = 0, i = 1..n; \quad \sum_{i=1}^{n} p_i = 1;\ p_i > 0\ ;$$

(See Appendix A.3, for full description on how to find the Maxmin strategies for a zero-sum games).

Hence, $P$ satisfy following system $CP^T = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, P > 0$; which is the computational system for matrix game $K$. Hence, $C$ is invertible and has full rank.■

**Theorem 2. 14.** The equilibrium point in the positive symmetric zero-sum tournament game is regular.

Proof:

Assume a positive symmetric zero-sum tournament game with set of pure strategies for Player 1 and Player 2 as $S = \{s_1, \dots, s_n\}$; where $K =$

$(k_{ij})_{n\times n}$ is the game matrix for Player 1. Hence, $K^T$ is the game matrix for Player 2. We also know that, $K^T = -K$.

$p = (p_1, \dots, p_n)$, $q = (q_1, \dots, q_n)$ are mixed strategy vectors, and the Nash equilibrium point is $P = (p, q)$. Hence, the utility functions are

$u_1(p, q) = pKq^T = \sum_{i,j} p_i K_{ij} q_j$ and $u_2(p, q) = pK^T q^T = \sum_{i,j} - p_i K_{ij} q_j$.

We know that the set of Nash equilibrium points is not empty. We define $F_1^{(1)}, F_2^{(1)}, \dots, F_n^{(1)}$ and $F_1^{(2)}, F_2^{(2)}, \dots, F_n^{(2)}$ as follows

$F_1^{(1)} = \sum_{i=1}^n p_i - 1$, $F_1^{(2)} = \sum_{i=1}^n q_i - 1$,

$F_i^{(1)} = [u_1(s_1, q) - u_1(s_i, q)] \times p_i = [\sum_{t=1}^n (k_{1t} - k_{it}) q_t] \times p_i$;
$i = 2, \dots, n$.

$F_j^{(2)} = \left[u_2(s_1, p) - u_2\left(s_j, p\right)\right] \times q_j = \left[\sum_{t=1}^n ((-k_{1t}) - (-k_{jt}))p_t\right] \times q_j$;
$j = 2, \dots, n$.

In an equilibrium, all $F_i^{(1)}$ and $F_j^{(2)}$ are equal to zero. Because, if $s_i$ ($i \neq 1$) is another strategy in the equilibrium, then $u_1(s_1, q) - u_1(s_i, q)$ (or similarly $[u_2(p, s_1) - u_2(p, s_i)]$) is equal to zero. Otherwise, $p_i$ (or similarly $q_j$) is equal to zero.

We define $F(p, q) = \begin{bmatrix} F_1^{(1)}(p, q) \\ F_2^{(1)}(p, q) \\ \vdots \\ F_n^{(1)}(p, q) \\ F_1^{(2)}(p, q) \\ \vdots \\ F_n^{(2)}(p, q) \end{bmatrix}$; hence we have

$$\frac{dF}{dP}=\begin{bmatrix} \frac{dF_1^{(1)}}{dp_1} & \frac{dF_1^{(1)}}{dp_2} & \dots & \frac{dF_1^{(1)}}{dp_n} & \frac{dF_1^{(1)}}{dq_1} & \dots & \frac{dF_1^{(1)}}{dq_n} \\ \vdots & \vdots & & & & & \vdots \\ \frac{dF_n^{(1)}}{dp_1} & & & & & & \frac{dF_n^{(1)}}{dq_n} \\ \frac{dF_1^{(2)}}{dp_1} & & & & & & \frac{dF_1^{(2)}}{dq_n} \\ \vdots & & & & & & \vdots \\ \frac{dF_n^{(2)}}{dp_n} & & & \dots & & & \frac{dF_n^{(2)}}{dq_n} \end{bmatrix}_{(2n)\times(2n)}$$

We should show that $\frac{dF}{dP}\neq 0$.

We know that $\frac{dF_1^{(1)}}{dp_i}=1$, $\frac{dF_1^{(1)}}{dq_i}=0$. Similarly, we have $\frac{dF_1^{(2)}}{dp_i}=0$, $\frac{dF_1^{(2)}}{dq_i}=1$.

For $i>1$ we have

$$\frac{dF_i^{(1)}}{dp_j}=\begin{cases}\sum_{t=1}^{n}(k_{1t}-k_{it})q_t\,, & if\ i=j,\\ 0, & if\ i\neq j.\end{cases}$$

$\frac{dF_i^{(1)}}{dq_j}=(k_{1j}-k_{ij})\,p_i.$

$\frac{dF_i^{(2)}}{dp_j}=-(k_{1j}-k_{ij})\,q_i.$

$$\frac{dF_i^{(2)}}{dq_j}=\begin{cases}\sum_{t=1}^{n}-(k_{1t}-k_{it})p_t\,, & if\ i=j,\\ 0, & if\ i\neq j.\end{cases}$$

Hence, by some elementary operations in $\frac{dF}{dP}$, we can see that the rank of the matrix $\frac{dF}{dP}$ is equal to the rank of $\bar{\bar{F}}=\begin{bmatrix}0 & C_1\\ C_2 & 0\end{bmatrix}_{(2n)\times(2n)}$, where $C_1$ and $C_2$ are computational matrix of Player 1 and Player 2, (Calculations are shown in Appendix A.4). By Theorem 2.13 we know that $C_1$ and $C_2$ have full rank, hence $\bar{F}$ has full rank as well and the equilibrium is regular in the sense of Harsanyi,■.

**Theorem 2. 15.** In a symmetric zero-sum tournament game with the set of strategies $\{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_n}\}$ and the matrix game $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{S_1} \\ \vdots \\ \boldsymbol{S_n} \end{bmatrix}$, if ($\boldsymbol{P}$,$\boldsymbol{P}$) ($\boldsymbol{P} = (\boldsymbol{p_1}, \boldsymbol{p_2}, \dots, \boldsymbol{p_n})$) is an equilibrium point, then for all the $\boldsymbol{p_i} = \boldsymbol{0}$ we have $\boldsymbol{S_i P} < \boldsymbol{0}$.

Proof:

Assume that $\overline{K}_{m\times m}$ , $m < n$ is the largest positive subtournament of $K$. Hence, as $\overline{K}$ is a game matrix of a positive subtournament, then $m$ is odd.

If $\bar{P} = (\overline{p_1}, \overline{p_2}, \dots, \overline{p_m})$ is the optimal strategy then we have $\bar{S}_i \bar{P}^T = 0$. If we add any bad quality strategy to $\overline{K}$ (we call the new game matrix $\overline{\overline{K}}$) then optimal strategy is $\overline{\bar{P}} = (\overline{p_1}, \overline{p_2}, \dots, \overline{p_m}, \overline{p_{m+1}})$ where $\bar{p}_{m+1} = 0$. If we assume that $S_{m+1}\overline{\bar{P}} = 0$, as a result we have $\overline{\overline{K}}\overline{\bar{P}} = 0$. $\overline{\overline{K}}$ is a $(m+1)$ by $(m+1)$ matrix where $(m+1)$ is even. So, by Theorem 1.2.5 $\overline{\overline{K}}$ has full rank, as a result $\overline{\bar{P}} = 0$, which is contradiction. Hence, $S_i \overline{\bar{P}} < 0$. ■

**Corollary 2. 16.** The equilibrium point in any symmetric zero-sum tournament game is regular.

Proof:
Let $\acute{T}$ be the largest positive subtournament of $T$ where $K$ and $\acute{K}$ are the matrix game of the tournaments $T$ and $\acute{T}$, respectively. By Theorem 2.14 we know that the equilibrium in $\acute{T}$ is regular and therefore, it will remain unique under small perturbation. On the other hand, by Theorem 2.14 we know that the expected pay off of a "bad quality" strategy of $K$ is strictly negative. Hence, by small changes in the matrix game it will remain negative. So, we can conclude that the equilibrium point in the symmetric zero-sum tournament game is robust under small perturbations, ■.

**Corollary 2. 17.** The equilibrium point of the symmetric zero-sum tournament game is robust under small perturbations.

Proof:

By Corollary 2. 15, we know that a symmetric zero-sum tournament game is regular. Hence it would be robust under small perturbations, ■.

## 2.5 Generalized Hoteling-Downs model with Asymmetric Candidates

In previous sections, I reviewed the main characteristics of the symmetric zero-sum games. I also, provide some theorems and explanations that how this game can be considered as the generalization of Hotelling-Downs model, when the strategy space of the candidates is not single-peaked.

In this section, I make a perturbation on the main diagonal of the matrix game of the symmetric zero-sum tournament game. By this method, I extend the symmetric zero-sum tournament game to symmetric non zero-sum tournament game and asymmetric zero-sum tournament game (tournament game with incumbent). In this new version of the tournament games the tie situation has been resolved; and it can model different political situations.

**Definition 2. 18.** Let $\boldsymbol{T}$ be a symmetric zero-sum tournament game with matrix game $\boldsymbol{K}$. The game with matrix game $\overline{\boldsymbol{K}} = \boldsymbol{K} + \boldsymbol{\alpha I}$ for the first player and $\overline{\boldsymbol{K}} = \boldsymbol{K^T} + \boldsymbol{\alpha I}$ for second player is called symmetric non zero-sum tournament game; where $\boldsymbol{I}$ is the identity matrix.

**Definition 2. 19**. Let $\boldsymbol{T}$ be a symmetric zero-sum tournament game with matrix game $\boldsymbol{K}$. The zero-sum game with matrix game $\boldsymbol{M} = \boldsymbol{K} + \boldsymbol{\alpha I}$ is called asymmetric zero-sum tournament (tournament game with incumbent); where $\boldsymbol{I}$ is the identity matrix.

**Theorem 2. 20.** Let $\boldsymbol{T}$ be a symmetric zero-sum tournament game with matrix game $\boldsymbol{K}$. The equilibrium point of both symmetric non zero-sum tournament game and asymmetric zero-sum tournament games associated to the game matrix $\boldsymbol{K}$ are unique for all $\boldsymbol{0 < \alpha < 1}$ and $\boldsymbol{n < 7}$.

Proof:

I designed a computer algorithm with Maple which tests the uniqueness of the equilibrium point of all the symmetric non zero-sum and asymmetric zero-sum tournament games where $-1 < \alpha < 1$ and $n \leq 7$, See Appendix A.5 for the Maple code. The computer program did not find any multiple equilibria.

### 2.5.1 Symmetric non zero-sum Tournament Game

Symmetric non zero-sum tournament game, describes the situation where by selecting the same strategy both players have the same payoff which is not zero. In reality, it may happen when in a tie situation both players try to collaborate to increase/decrease their payoff. It means that this model can capture the cases when there is collusion between the parties, and they intentionally choose the same platform. The incentive for avoiding the tie situation can also be created by an external party. The results in Section 2.4.2 guarantee that if the amount of the perturbation is small enough, the equilibrium point is unique. However, by some examples I demonstrate that under large perturbations the structure of the game varies considerably and the equilibrium point is not necessary unique.

**Example 2. 21.**

Suppose the matrix game of a symmetric zero sum tournament game as follow

$K = \begin{bmatrix} 0 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ -1 & 0 & 1 & 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 0 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 0 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 1 & -1 & 1 & -1 & 0 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 0 \end{bmatrix}$. We can easily calculate that the optimal strategy is $(\frac{1}{9}, \frac{1}{9}, 0, 0, \frac{1}{3}, \frac{1}{9}, \frac{1}{3}, 0)$ for both players.

We define a symmetric non zero-sum tournament game by following matrices, $\overline{K}_1 = K + \alpha I$ and $\overline{K}_2 = K^T + \alpha I$, where $I$ is the identity matrix and $\alpha \neq 0$. $\overline{K}_1$ is the game matrix for the first player and $\overline{K}_2$ is the game matrix for second one. In this game, the equilibrium is unique when $0 < \alpha < 0.2778$[9]. In this example both players have 8 strategies. As I explained in Theorem 2.24, there is no such this example for the games with less than 8 strategies.

[9] $\alpha$ has been calculated by employing Gambit 13 to test different games.

### 2.5.2 Asymmetric zero-sum Tournament Game

In this section, we resolve the tie situation by considering a higher chance of winning for the first player in selecting the same strategy. In this case, the main diagonal of game matrix of symmetric zero-sum tournament game changes to a strictly non negative number like $\alpha$. The game is still a zero-sum game; however it is not symmetric any more. We can also call the asymmetric zero-sum tournament game, the tournament game with incumbent. As it can describe the political situations where there is an incumbent who has been in power previously. The effect of the incumbency has been analysed by many studies. M. C. Cummings (1966) analysed the data of US presidential elections from 1924 to 1964. His analyse showed that in those elections the incumbent has more chances to keep the power compared to the challenging party. In contrast, in some societies voters may like to change the party who is in power for a long time. R. Chacrabarti et al. (2005) study about the Sabha elections in India show the anti-incumbency behaviour among the voters. In both cases, one of the parties will have more chances to win when they both have chosen the same platform. The following example demonstrates a tournament game with incumbent. In this example, we have calculated the range of the possible positive amount of changes for $\alpha$ which keep the equilibrium unique.

**Example 2. 22.**

If we consider the matrix game of the symmetric zero-sum tournament game as $K = \begin{bmatrix} 0 & 1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 0 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & 0 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 0 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & 0 & 1 \\ -1 & 1 & 1 & 1 & 1 & -1 & 0 \end{bmatrix}$, then the equilibrium is unique when $0 < \alpha < 1$.

If $\alpha = 1$, the game has two equilibrium points as follow,

$P_1 = \{\left(\frac{1}{5}, \frac{1}{5}, 0, 0, 0, \frac{2}{5}, \frac{1}{5}\right), (\frac{1}{5}, \frac{1}{5}, 0, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0)\}$ and

$P_2 = \{\left(\frac{1}{5}, \frac{1}{5}, 0, \frac{1}{10}, \frac{1}{10}, \frac{3}{10}, \frac{1}{10}\right), (\frac{1}{5}, \frac{1}{5}, 0, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0)\}$.

As we see, in both equilibrium points Player 1 is mixing between even number of strategies; which is in contrast to the previous results in symmetric zero-sum tournament games. Beside, as we can observe in equilibrium point $P_1$, not necessarily both players have the same bad quality strategies.[10]

**Remark 2.23.** The fact that in symmetric zero-sum tournament game players mix between odd number of strategies (Colloraly 2.9) coming from the fact that the game matrix of a symmetric zero-sum tournament game has full rank just when number of strategies $n$ are even (Theorem 1.2.5) and otherwise rank $n-1$. In the latter case the system $Kp = 0$, may have a nontrivial solution and hence an equilibrium. In the case of an asymmetric zero-sum tournament game and symmetric non zero-sum tournament game we show (Theorem 2.25) that the game matrix always has full rank. Hence the support of the equilibrium may vary to have an odd or even number of positive probabilities.

## 2.6 Sufficient Condition for Uniqueness of the Equilibrium

As we observed in the Example 2.21 and Example 2.22 of the previous section, the equilibrium point of the symmetric non zer-sum tournament game and asymmetric zero-sum tournament game is not necessarily unique. However, in this section we provide sufficient condition for uniqueness of the equilibrium in both of the mentioned tournament games.

**Theorem 2. 23.** If $\boldsymbol{K}$ is the game matrix of symmetric zero sum tournament game then all the eigenvalues of $\boldsymbol{K}$ are purely imaginary (Hoffman (1971)).

Proof:

We know that for the symmetric zero sum tournament game $K^T = -K$. Hence, we have

$Kv = \lambda v \quad \Rightarrow \overline{Kv} = \overline{\lambda v} \quad \Rightarrow \quad K\bar{v} = \bar{\lambda}\bar{v}$ , as the elements of $K$ are real.

[10] $P_1$ and $P_2$ have been calculated by "enumerating extreme points" method in Gambit 13. As the game is a zero-sum game, $P_1$ and $P_2$ are the extreme points of a convex set of equilibrium points.

$(K\overline{v})^T v = \left(\bar{\lambda}\bar{v}\right)^T v \Rightarrow (\bar{v})^T K^T v = \bar{\lambda}(\bar{v})^T v \Rightarrow (\bar{v})^T(-K)v = \bar{\lambda}(\bar{v})^T v \Rightarrow (\bar{v})^T(-\lambda)v = \bar{\lambda}(\bar{v})^T v \Rightarrow -\lambda\|v\|^2 = \bar{\lambda}\|v\|^2 \Rightarrow$
Eigenvalues of $K$ are purely imaginary, ■.

**Theorem 2. 24.** The matrix A with eigenvalues as $\boldsymbol{\lambda_1}, \dots, \boldsymbol{\lambda_n}$ and the eigenvectors as $\boldsymbol{v_1}, \dots, \boldsymbol{v_n}$ has the same eigenvectors as the matrix

$$\alpha_m A^m + \alpha_{m-1}A^{m-1} + \cdots + \alpha_1 A + \alpha_1$$

but the eigenvalues of the latter will be

$\alpha_m\lambda^m + \alpha_{m-1}\lambda^{m-1} + \cdots + \alpha_1\lambda + \alpha_1$. (see Hoffman (1971)).

By Theorem 2. 8 we know that the matrix game of a symmetric zero-sum game has full rank just when the number of the strategies are even. Following theorem explain the rank of the matrix game of an asymmetric zero-sum tournament game.

**Theorem 2. 25.** Let $\boldsymbol{K}$ be the matrix game of a symmetric zero-sum tournament game. If $\boldsymbol{M = K + \alpha I}$ $(\boldsymbol{\alpha \neq 0})$ then $\boldsymbol{M}$ has full rank.

Proof:

By theorem 3.1.1 we know that all the eigenvalues of $K$ are in the form of $bi$, where $b$ is a real number and $i$ is the imaginary number. On the other hand, we know $M = K + \alpha I$ $(\alpha \neq 0)$ where $K$ is the game matrix of symmetric zero sum tournament game. As a result by theorems 3.1.2 we know that all the eigenvalues of $M$ are in form of $bi + 1$. $bi + 1$ cannot be zero, so all the eigenvalues of $M$ are non- zero. Hence $M$ has full rank, ■.

**Theorem 2. 26.** Let $\boldsymbol{K}$ be the game matrix of a symmetric zero-sum tournament game. If $\boldsymbol{M = K + \alpha I}$ $(\boldsymbol{\alpha \neq 0})$ the computational matrix of $\boldsymbol{M}$ has full rank.

Proof:

If we denote the row vectors of $M$ by $a_i$, we want to show that $C = \begin{bmatrix} (1,1,\dots,1) \\ a_2 - a_1 \\ \vdots \\ a_{n-}a_1 \end{bmatrix}$ has full rank. By Theorem 2.25 we know that $M$ has full

rank for $\alpha \neq 0$. Hence $a_i - a_1$ are linearly independent for $2 \leq i \leq n$; which means that

$$\sum_{i=2}^{n} \lambda_i (a_i - a_1) = 0,$$

would imply

$$-(\sum_{i=2}^{n} \lambda_i) a_1 + \sum_{i=2}^{n} \lambda_i a_i = 0,$$

By linear independence of $a_i$ we obtain $\lambda_2 = \cdots = \lambda_n = 0$.

Suppose $C$ does not have full rank. Hence, the first row of $C$, $\acute{e} = (1,1, \ldots ,1)$, is the non-trivial combination of other rows.

$$\acute{e} = \sum_{i=2}^{n} p_i (a_i - a_1),$$

with $p_{i_0} \neq 0$ for some $2 \leq i_0 \leq n$. Suppose $p_1 = -\sum_{i=2}^{n} p_i$, hence we have

$$\acute{e} = \sum_{i=2}^{n} p_i (a_i - a_1) = \sum_{i=2}^{n} p_i a_i - (\sum_{i=2}^{n} p_i) a_1 = p_1 a_1 + \sum_{i=2}^{n} p_i a_i = \sum_{i=1}^{n} p_i a_i$$

whereby

$$\sum_{i=1}^{n} p_i = 0,$$

and $p_{i_0} \neq 0$ for some $2 \leq i_0 \leq n$.

We can write this in matrix notation as

$$\acute{e} = \acute{p} M \text{ and } \acute{e} p = 0$$

where $\acute{p} = (p_1, \ldots, p_n)$ and $\acute{p} \neq 0$.

However, this leads to the following contradiction for $\alpha \neq 0$.

We know that $\acute{p} K p = \sum_{i=1}^{n} k_{ij} p_i^2 + \sum_{j>i} (k_{ij} + k_{ji}) p_{ij} = 0$, as $K$ is a skew-symmetric matrix.

$$0 = \acute{e} p = \acute{p} M p = \acute{p} (K + \alpha I) p = \acute{p} K p + \alpha \acute{p} I p = 0 + \alpha \sum_{i=1}^{n} p_i^2 \neq 0,$$

where by $I$ we denotes the identity matrix. Hence, $C$ must have full rank, ■.

**Theorem 2. 27.** Let $\boldsymbol{K}$ be a matrix game of a symmetric zero-sum game. In an asymmetric zero-sum tournament game where the matrix game is $\boldsymbol{M} = \boldsymbol{K} + \boldsymbol{\alpha I}$ and $\boldsymbol{\alpha} \neq \boldsymbol{0}$, in equilibrium, when one player plays with a full support mixed strategy the other player has a unique optimal strategy.

Proof:

By applying Theorem 2.26, we can show that in an asymmetric zero-sum tournament game the computational matrices of the game matrix of first and second player have full rank when $\alpha \neq 0$. It means that when one of the players mixes between all of his strategies with positive probability, the computational system for calculating the optimal strategies of the other player have to have a unique solution, ■.

**Theorem 2.28**. In Symmetric non zero-sum tournament games, in equilibrium, when one player plays with a full support mixed strategy the other player has a unique optimal strategy.

Proof:

The proof is the same of Theorem 2.28, ■.

In fact, Theorem 2.28 and Theorem 2.29 provide sufficient condition for uniqueness of the equilibrium.

## 2.7 Regular Tournament Game in the sense of Laffond et al.

In previous sections we saw that for a symmetric zero-sum tournament game the value is zero. However there is no theorem to provide the amount of the value for asymmetric zero-sum tournament game. For different number of strategies we may have different games with different values. In this section we first provide an example to show how the amount of the values can be different for the asymmetric zero-sum tournament games with the same number of strategies. Later, I will analyse the special case of regular tournaments in sense of Laffond et al. where not only we can predict the amount of the value also we can

guarantee the unique, uniform distribution optimal strategy for both players.

**Example 2. 29.**

Let $K_{n\times n}$ be a matrix game of a symmetric zero-sum tournament game with $n$ strategies. Suppose the asymmetric zero-sum tournament game with matrix game $M = K + I$ and $n$ number of strategies. Following table shows the possible values of the value of the game, for different number of strategies.

| $n$ | *Value* |
|---|---|
| 3 | $\{1,1/3\}$ |
| 4 | $\{1,1/3\}$ |
| 5 | $\left\{1,\frac{1}{3},\frac{1}{4},\frac{1}{5}\right\}$ |
| 6 | $\left\{1,\frac{1}{3},\frac{1}{4},\frac{1}{5}\right\}$ |
| 7 | $\left\{1,\frac{1}{3},\frac{1}{4},\frac{1}{5},\frac{1}{6},\frac{1}{7},\frac{2}{11},\frac{3}{13},\frac{3}{17}\right\}$ |
| 8 | $\left\{1,\frac{1}{3},\frac{1}{4},\frac{1}{5},\frac{1}{6},\frac{1}{7},\frac{2}{11},\frac{3}{13},\frac{3}{17},\frac{3}{19},\frac{5}{21},\frac{5}{29},\frac{5}{31},\frac{5}{33},\frac{7}{41},\frac{7}{45}\right\}$ |

**Table 2. 5.** Possible amount of the value of the asymmetric zero-sum tournament game for $\boldsymbol{\alpha = 1}$.

**Definition 2. 29.** A symmetric zero-sum tournament is regular in the sense of Laffond et al if all its vertices have similar score.

**Theorem 2. 30.** A symmetric zero-sum tournament is regular if and only if in the equilibrium players mixes between all the strategies with uniform distribution.

Proof:

Suppose that $K$ is the matrix game of a symmetric zero-sum tournament game. If we suppose that the all the vertices have similar score, then obviously players will be indifferent between all the strategies.

On the other hand, in equilibrium players are mixing between the strategies with the uniform strategy $P = (p, p, \dots, p)$. Hence we will have $KP^T = [0]_{n\times 1}$; which means that $K[1]_{n\times 1} = [0]_{n\times 1}$. So, in each row of $K$

we should have equal number of $1$ and $-1$. This means that the score of all vertices in the tournament game is equal, ■.

Following theorem explain the optimal strategy and value of the game for the regular asymmetric zero-sum games.

**Theorem 2. 31.** Let $\boldsymbol{T}$ be a symmetric zero-sum tournament game with the game matrix $\boldsymbol{K}$, and $\overline{\boldsymbol{T}}$ an asymmetric zero-sum tournament game with game matrix $\overline{\boldsymbol{K}} = \boldsymbol{K} + \boldsymbol{\alpha I}$, $\boldsymbol{\alpha} \neq \boldsymbol{0}$. In the equilibrium, in the tournament game $\boldsymbol{T}$ players are mixing between all of the strategies uniformly if and only if the players in the tournament game $\overline{\boldsymbol{T}}$ do the same.

Besides, in this case the value for the asymmetric zero-sum tournament game $\bar{T}$ would be $\frac{\alpha}{n}$, where $n$ is the number of the strategies.

Proof:

Suppose that in the tournament game $\bar{T}$, players mix between all of the strategies uniformly.

Let $E = \begin{bmatrix} 1\ 1 \dots 1 \\ e_1 \\ \vdots \\ e_{n-1} \end{bmatrix}$ and $\bar{E} = \begin{bmatrix} 1\ 1 \dots 1 \\ \bar{e}_1 \\ \vdots \\ \bar{e}_{n-1} \end{bmatrix}$ be the computational matrices of $K$ and $\bar{K}$, respectively. In each $\bar{e}_i$ we have "-1-$\alpha$ and $\alpha$-1" or "1- $\alpha$ and $\alpha$ +1". Hence, as players are mixing between all the strategies uniformly the computational system $\bar{E}q = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$, will have the same answer as computational system $Eq = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$.

Suppose that in the tournament game $T$, players mix between all of the strategies uniformly. If $q$ is the unique optimal strategy then it would be the solution of following computational system, $E \begin{bmatrix} 1\ 1 \dots 1 \\ e_1 \\ \vdots \\ e_{n-1} \end{bmatrix} q = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$. We add a -$\alpha$ to the the first element and an $\alpha$ to the $i+1$th element of $e_i$. So,

the system will change to $\bar{E}\begin{bmatrix}1\ 1\ \dots\ 1\\ \bar{e}_1\\ \vdots\\ \bar{e}_{n-1}\end{bmatrix}q=\begin{bmatrix}1\\ \vdots\\ 0\\ 0\end{bmatrix}$, where the solution has been not changed.

Hence, tournament games $T$ and $\bar{T}$ have the same optimal solutions. As a result, the value for $\bar{T}$ would be $\frac{\alpha}{n}$ where $n$ is the number of the strategies,■.

**Example 2. 32.** For the regular symmetric zero-sum tournament game with matrix game

$K=\begin{bmatrix}0&-1&1&1&-1\\1&0&-1&1&-1\\-1&1&0&-1&1\\-1&-1&1&0&1\\1&1&-1&-1&0\end{bmatrix}$, the optimal strategy for both players is $\left\{\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5}\right\}$.

In the asymmetric zero-sum tournament game with matrix game $\bar{K}=K+\alpha I,\ \alpha\neq 0$ the optimal strategy for both players is $\left\{\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5},\frac{1}{5}\right\}$ and the value of the game is $\frac{\alpha}{5}$.

**Corollary 2. 33.** Let $\boldsymbol{T}$ be a symmetric zero-sum tournament game with matrix game $\boldsymbol{K}$ and $\bar{\boldsymbol{T}}$ be an asymmetric zero-sum tournament game with matrix game $\bar{\boldsymbol{K}}=\boldsymbol{K}+\boldsymbol{\alpha I},\ \boldsymbol{\alpha}\neq\boldsymbol{0}$.

In tournament game $\bar{T}$, in the equilibrium players cannot mix between all of their strategies uniformly when the number of the strategies are even.

Proof:

We know that in the tournament game $T$, in the equilibrium players are mixing between odd number of the strategies. Hence, they cannot have the uniform full support optimal strategies. As a result by Theorem 1, players in tournament game $\bar{T}$ cannot have uniform full support equilibrium point as well.

## 2.8 Conclusion

I analyse the tournament game (named as symmetric zero-sum tournament game in this study). Previous results have shown that it has a unique solution; I demonstrate that the uniqueness is robust under small perturbation. However, the structure of the solution changes significantly when the amount of perturbation is greater than a certain threshold dependent on each game. I analyse the tournament game with incumbent as the case of a perturbed tournament game. By providing several examples I show that the structure of the equilibrium point is fundamentally different from the standard tournament game. While there is no general criteria to describe the value and optimal strategies of the game I provide a partial criteria which is a sufficient condition for uniqueness of equilibrium. This work enables future studies on determining the threshold of perturbations and general criteria for optimal strategies in perturbed games.

## References


Anderson, S. P. and G. Glomm (1992). 'Incumbency effects in political campaigns', *Public Choice*, vol. **74**(**2**), pp. 207-219.

Ansolabehere, S. and J. M. Snyder Jr (2000). 'Valence politics and equilibrium in spatial election models', *Public Choice*, vol. **103**(**3-4**), pp. 327-336.

Aragones, E. and T. R. Palfrey (2002). 'Mixed equilibrium in a Downsian model with a favored candidate', *Journal of Economic theory*, vol. **103**(**1**), pp. 131-161.

Arrow, K. J. (1951). '1963. Social choice and individual values', *Wiley, New York.*

Chakrabarti, R., S. Gangopadhyay and S. Krishnan (2005). 'Incumbency Effects in Indian Elections–A Preliminary Exploration', *Unpublished paper. Accessed April*, vol. **15**, pp. 2014.

Cummings, M. C. (1966). *Congressmen and the Electorate*: Free Press.

Dahl, R. A. (2006). *A Preface to Democratic Theory*: University of Chicago Press.

Davis, O. A., M. H. DeGroot and M. J. Hinich (1972). 'Social preference orderings and majority rule', *Econometrica: Journal of the Econometric Society*, pp. 147-157.

Downs, A. (1957). 'An Economic Theory of Democracy Harper Collins', *New York*.

Fisher, D. C. and J. Ryan (1992). 'OPTIMAL STRATEGIES FOR A GENERALIZED SCISSORS, PAPER, AND STONE GAME', *American Mathematical Monthly*, vol. **99**(**10**), pp. 935-942.

Hans, P. (2008). 'Game theory, A multi-leveled approach', in (Editor Ed.)^Eds.), *Book Game theory, A multi-leveled approach*, City: Springer-Verlag, Berlin Heidelberg.

Harsanyi, J. C. (1973). 'Oddness of the number of equilibrium points: a new proof', *International Journal of Game Theory*, vol. **2**(**1**), pp. 235-250.

Hoffman, K. and R. Kunze 'Linear Algebra. 1971', *Englewood Cliffs, New Jersey*.

Hotelling, H. (1990). 'Stability in competition', *The Collected Economics Articles of Harold Hotelling,* pp. 50-63: Springer.

Ingberman, D. E. (1992). 'Incumbent reputations and ideological campaign contributions in spatial competition', *Mathematical and Computer Modelling*, vol. **16**(**8**), pp. 147-169.

Kramer, G. H. (1973). 'On a class of equilibrium conditions for majority rule', *Econometrica: Journal of the Econometric Society*, pp. 285-297.

Kurrild-Klitgard, P. (2001), An empirical example of the Condorcet paradox of voting for a large electorate, *Public Choice*, 107(1): 135–145.

Laffond, G., J. F. Laslier and M. Lebreton (1993). 'The Bipartisan Set of a Tournament Game', *Games and Economic Behavior*, vol. **5**(**1**), pp. 182-201.

Luce, R. D. and H. Raiffa (1957). 'Games anddecisions', *New York, JohnW iley Sons*.

Maschler, M. and E. Solan 'S, Zamir (2013)', *Game Theory*.

McGarvey, D. C. (1953). 'A theorem on the construction of voting paradoxes', *Econometrica: Journal of the Econometric Society*, pp. 608-610.

McKelvey, R. D. (1976). 'Intransitivities in multidimensional voting models and some implications for agenda control', *Journal of Economic theory*, vol. **12**(**3**), pp. 472-482.

McKelvey, R. D. and P. C. Ordeshook (1985). 'Elections with limited information: A fulfilled expectations model using contemporaneous poll and endorsement data as information sources', *Journal of Economic theory*, vol. **36**(**1**), pp. 55-85.

McKelvey, R. D. and R. E. Wendell (1976). 'Voting equilibria in multidimensional choice spaces', *Mathematics of operations research*, vol. **1**(**2**), pp. 144-158.

Miller, N. R. (1977). 'Graph-theoretical approaches to the theory of voting', *American Journal of Political Science*, pp. 769-803.
Miller, N. R. (1980). 'A new solution set for tournaments and majority voting: Further graph-theoretical approaches to the theory of voting', *American Journal of Political Science*, pp. 68-96.
Plott, C. R. (1967). 'A notion of equilibrium and its possibility under majority rule', *The American Economic Review*, pp. 787-806.
Selten, R. (1973). 'A simple model of imperfect competition, where 4 are few and 6 are many', *International Journal of Game Theory*, vol. **2**(**1**), pp. 141-201.
Stokes, D. E. (1963). 'Spatial models of party competition', *American Political Science Review*, vol. **57**(**02**), pp. 368-377.
Trudeau, R. J. (2013). *Introduction to graph theory*: Courier Corporation.
Van Damme, E. (1991). *Stability and perfection of Nash equilibria*: Springer.
Von Neumann, J. and O. Morgenstern (1945). 'Theory of games and economic behavior', *Bull. Amer. Math. Soc*, vol. **51**(**7**), pp. 498-504.
Wendell, R. E. and S. J. Thorson (1974). 'Some generalizations of social decisions under majority rule', *Econometrica: Journal of the Econometric Society*, pp. 893-912.

# A. Appendix to Chapter 2

This appendix provides mathematical background for the models and theorems in this chapter.

## A.1. Graphs and Subgraphs

Graphs are the structures that connect some set of vertices with some specific rules. Despite the simple structures graphs have wide application in mathematical modelling.

In this Appendix, the concepts and definitions of graphs and subgraphs used in the models and theorems of this study are explained, (See Trudeau (2013)).

**Definition A. 1. 1.** A graph $G$ consists of two sets. First a non-empty set $V(G)$ (set of vertices) and secondly, $E(G)$ (set of edges).

Following example shows a simple graph with four vertices and three edges.

**Example A. 1. 2.**

$V(G) = \{v_1, v_2, v_3, v_4\}, E(G) = \{v_1v_2, v_2v_3, v_3v_4\}.$

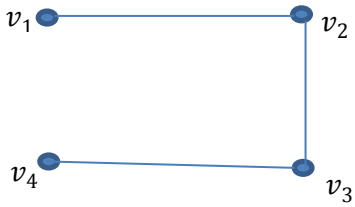



**Figure A. 1. 3.** A simple graph with four vertices and three edges

**Definition A. 1. 4.** A directed graph $D$ consists of two sets. First a non-empty set $V(D)$ (set of vertices) and secondly a finite set of ordered pair of elements of $D$ $(A(D))$, each ordered pair is called arc.

Following example shows a simple directed graph with four vertices and three arcs.

**Example A. 1. 5.**

$V(D) = \{v_1, v_2, v_3, v_4\}, A(D) = \{v_1v_2, v_2v_3, v_3v_4\}.$

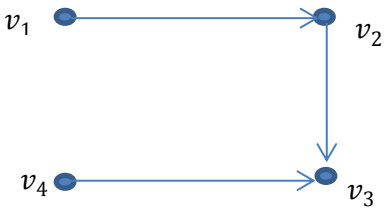



**Figure A. 1. 6.** A simple graph with four vertices and three edges

**Definition A. 1. 6.** In a directed graph like $D$, two vertices like $v_1, v_2$ are adjacent if $v_1v_2 \epsilon A(D)$.

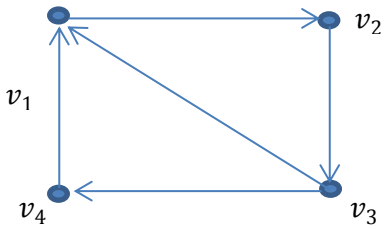
**Definition A. 1. 7.** In a directed graph like $D$, there is a path from vertex $v_1$ to $v_n$ ,if there is a finite sequence of arcs of the form $v_1v_2, v_2v_3, \dots, v_{n-1}v_n$ between $v_1$ and $v_n$.

**Definition A. 1. 8.** In a directed graph like $D$, there is a path with the length of $m$ from vertex $v_1$ to $v_n$ if the shortest path between $v_1$ to $v_n$ has exactly $m$ arcs.

**Definition A. 1. 9.** A directed graph like $D$ has a cycle, if we can find a path starting from a vertex like $v_1$ and ending to $v_1$ with the length of $|V(G)|$.

**Example A. 1. 10.**

$$V(D) = \{v_1, v_2, v_3, v_4\}, A(D) = \{v_1v_2, v_2v_3, v_3v_4, v_4v_1, v_3v_1\}.$$




- There is a path of length two between $v_1$ and $v_3$ ($v_1v_2, v_2v_3$).
- The graph has a cycle ($v_1v_2, v_2v_3, v_3v_4, v_4v_1$).

**Figure A. 1. 11.** A simple directed graph with four vertices and four edges

**Definition A. 1. 11.** A directed graph $\overline{D}$ is a subgraph of directed graph $D$, if $V(\overline{D}) \subseteq V(D)$ and $A(\overline{D}) \subseteq A(D)$.

## A.2. Zero-sum games

In this appendix, I review the definition and characteristics of the zero-sum games which is the base of the model in Section 2. In this appendix, the notations and definitions are the same of Maschler, Solan and Zamir (2013).

**Definition A. 2. 1.** A two player game is a zero-sum game if for each pair of strategies $(s_I, s_{II})$ one has

$$u_I(s_I, s_{II}) + u_{II}(s_I, s_{II}) = 0.$$

Where $u_I$ and $u_{II}$ are the utility functions of the first and second player.

Typically, the concepts of maxmin and minmax strategy are useful in analysing the zero-sum games.

Maxmin value is the value that first player guarantees that he will get, and minmax value is the value that second player guarantees he will lose no more than. For a zero-sum game the maxmin and minmax values can be found by following equations

$$\underline{v} = \max_{s_I \epsilon S_I} \min_{s_{II} \epsilon S_{II}} u(s_I, s_{II}),$$

$$\overline{v} = \min_{s_{II} \epsilon S_{II}} \max_{s_I \epsilon S_I} u(s_I, s_{II}),$$

where $u_I = -u_{II} = u$ and $S_I$ and $S_{II}$ are the set of strategies for first and second player respectively. For zero-sum games, maxmin and minmax values are equal ($\underline{v} = \overline{v} = v$), where $v$ is called value of the game. Any maxmin and minmax strategies of the game are called optimal strategy.

Maxmin strategies and Nash equilibrium are two different concepts. However, for the case of zero-sum games they are equivalent.

## A.3 Calculating Optimal Strategies in Zero-Sum Games

There are some theorems which can predict the value and optimal strategies of certain zero-sum games. However, here I briefly review two different methods for calculating the optimal strategies of a zero-sum game.

### A.3.1 Indifferent Strategies

In a mixed strategy equilibrium for a zero-sum game, we know that a player will mix between two (or more) strategies if he is indifferent between them. If we know that the player is mixing between all of his strategies, we can use this fact to find the optimal strategies.

Suppose a zero-sum game $\bar{G}$ with two players. The matrix game of the game can be denoted as $G = (g_{ij})_{n \times n} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$.

If Player 1, mixes between all of his strategies then we can find the optimal strategies for second player by solving the following system

$$C_1 q = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \quad where\ q \geq 0, q = (q_1, q_2, \dots, q_n); \qquad \text{(A.3.1.1)}$$

where $C_1 = \begin{bmatrix} (1, \dots, 1) \\ a_2 - a_1 \\ \vdots \\ a_n - a_1 \end{bmatrix}$.

Similarly, if second player mixes between all of his strategies then we can find the optimal strategies of first player by solving the following system

$$C_2 p = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \quad where\ p \geq 0,\ p = (p_1, p_2, \dots, p_n); \qquad \text{(A.3.1.2)}$$

where $-G = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ and $C_2 = \begin{bmatrix} (1, \dots, 1) \\ b_2 - b_1 \\ \vdots \\ b_n - b_1 \end{bmatrix}$.

This is not an efficient method to find all of the equilibriums of a zero-sum game. Obviously, there might be some equilibriums that a player is not mixing between all of his strategies.

### A.3.2 Linear Programing

Luce and Raiffa (1957) introduced a linear programing system to calculate the optimal strategies of a zero-sum game. In comparison with Indifferent strategies method described in A.3.1, this method is computationally more costly. However, it is more efficient as it can calculate all of the optimal strategies of a player.

For a zero-sum game with matrix game $G = (g_{ij})_{n\times n}$, the following system can give us all the possible optimal strategies (minmax strategies) for Player 1.

$$\begin{cases} Max\ v \\ (G^T x)_j \geq v \quad \forall j \\ \sum_{i=1}^{n} x_i = 1 \\ x_i \geq 0. \end{cases} \qquad \text{(A.3.2.1)}$$

To change System (A.3.2.1) to a LP program, initially we add a large enough positive number to all the entries of $G$ to make them all positive.

$$\exists a > 0; \acute{G} = \left(g_{ij} + a\right)_{n\times n} \Rightarrow \acute{G} > 0.$$

If we rewrite System (A.3.2.1) for the new matrix, $\acute{G}$, we have

$$\begin{cases} Max\ v \\ (\acute{G}^T x)_j \geq v \quad \forall j \\ \sum_{i=1}^{n} x_i = 1 \\ x_i \geq 0. \end{cases} \qquad \text{(A.3.2.2)}$$

By solving the system (A.3.2.2) we can find optimal strategies. However, the calculated value in (A.3.2.2) is the value of the game with matrix game $G$, added by $a$. we denote the new value by $v^*$. We know that as $\acute{G} > 0$ then $v^* > 0$. Hence, as $v^* > 0$, then by letting $\frac{x_i}{v^*} = u_i$ and dividing $\sum_{i=1}^{n} \acute{g_{iJ}}\, p_i \geq v^*$ to $v^*$ the system (A.3.2.2) changes to

$$\begin{cases} \min \sum_{i=1}^{n} u_i \\ (\acute{G}^T u)_j \geq 1,\ for\ all\ j \\ u_i \geq 0. \end{cases} \qquad \text{(A.3.2.3).}$$

By solving the LP system (A.3.2.3), we can find all the optimal strategies of the first player.

Similarly we can find optimal strategies for the second player by solving the following problem.

$$\begin{cases} Max\ v \\ \sum_{i=1}^{n} -Gq \geq v \\ \sum_{i=1}^{n} q_i = 1 \\ q_i \geq 0. \end{cases} \Rightarrow \begin{cases} Min\ \acute{v} \\ \sum_{i=1}^{n} Gq \leq \acute{v} \\ \sum_{i=1}^{n} q_i = 1 \\ q_i \geq 0. \end{cases} \Rightarrow \begin{cases} Min\ v^* \\ \sum_{i=1}^{n} \acute{G} q_i \leq v^* \\ \sum_{i=1}^{n} q_i = 1 \\ q_i \geq 0. \end{cases} \Rightarrow$$

(A.3.2.4) (A.3.2.5) (A.3.2.6)

$$(by\ \ \frac{q_i}{v^*} = v_i)\ \begin{cases} Max \sum_{i=1}^{n} v_i \\ \sum_{i=1}^{n} \acute{G} v_i \leq 1 \\ v_i \geq 0. \end{cases}$$

(A.3.2.7).

LP Systems (A.3.2.3) and (A.3.2.7) are dual.

## A. 4 Regular Equilibrium

In this appendix, I briefly review the concept of the regular equilibrium in the sense of Harsanyi (1973). Later, I provide the detailed calculation of the Theorem 2. 14, which investigates the regular equilibrium in perturbed asymmetric zero-sum games.

### A.4.1. Regular equilibria in normal form games

The concept of regular equilibria is defined by Harsanyi (1973) to do more refinements with the concept of Nash equilibrium. In this section I review the concept of regular equilibrium for finite normal form games. For this matter I follow the definitions and notations by Van Damme (1991), which are slightly different from the definitions given by Harsanyi (1973). However, with both definitions we can conclude the strong stability characteristics.

In this section, first I provide the definition of the $n$-person normal form game, and then I provide some notations and formulations needed to introduce the regular equilibrium.

**Definition A. 4. 1.** A finite $n$-person normal form game is a $2n$-tuple $\Gamma(\Phi_1, \Phi_2, \dots, \Phi_n, R_1, \dots, R_n)$ where $\Phi_i$ is a finite non-empty set and $R_i$ is a

mapping $R_i: \prod_{i=1}^{n} \Phi_j \to \mathcal{R}$. $\Phi_i$ is the set of pure strategies of the player $i$, and $R_i$ is the payoff function of this player.

We also define $m_i = |\Phi_i|$. The generic element of $\Phi_i$ is denoted by $\phi_i$. We also assumed that the elements of $\Phi_i$ are numbered, hence we can talk about the $k^{th}$ pure strategy of player $i$. So, the generic element of $\Phi_i$ can also be denoted by $k$.

A mixed strategy $s_i$ of player $i$ is a probability distribution on $\Phi_i$. I denote the probability which $s_i$ assigns to pure strategy $k$ of player $i$ by $s_i^k$. Hence, the set of all mixed strategies of player $i$ is

$$S_i = \left\{ s_i \epsilon f(\Phi_i, R);\ \sum_k s_i^k = 1, s_i^k \geq 0 \ \ for\ all\ k \epsilon \Phi_i \right\}.$$

If $s_i \epsilon S_i$ then we define $C(s_i)$ as follows

$$C(s_i) = \left\{ k \epsilon \Phi_i;\ s_i^k > 0 \right\}.$$

$\Phi$ and $S$ are defined as follow.

$$\Phi = \prod_{i=1}^{n} \Phi_i \ , \qquad S = \prod_{i=1}^{n} S_i.$$

A generic element of $\Phi$ is denoted by $\phi$. So, we can define $C(s)$ as

$$C(s) = \{\phi; s(\phi) > 0\} = \prod_{i=1}^{n} C(s_i).$$

In a finite normal form game, players choose their strategy independently, therefore the probability $s(\phi)$ that $\phi = (k_1, .., k_n)$ occurs if $s = (s_1, .., s_n)$ is played, is given by

$$s(\phi) = \prod_{i=1}^{n} s_i^k.$$

Hence if $s$ is played, the expected payoff of player $i$ is

$$R_i(s) = \sum_{\phi} s(\phi) R_i(\phi).$$

If $s = (s_1, .., s_n)$ then $s \backslash \bar{s}_i$ means replacing strategy $s_i$ with $\bar{s}_i$.

Let $\Gamma(\Phi_1, \Phi_2, \dots, \Phi_n, R_1, \dots, R_n)$ be an $n$-person normal form game. $X_i$ is the set of all mappings from $\Phi_i$ to $\mathcal{R}$ and we have $S_i \subset X_i$. The generic element of $X_i$ is denoted by $x_i$ and $x_i^k$ denotes value of $x_i$ at $k$. Besides, $X = \prod_{i=1}^n X_i$ and generic element of $X$ is $x$.

We define $R_i$ as follows

$R_i(x) = \sum_{\phi \epsilon \Phi} x(\phi) R_i(\phi)$; $x(\phi) = \prod_{j=1}^n x_j^{k_j}$ if $\phi = (k_1, \dots, k_n)$.

Let $F(x|\phi)$ be a mapping defined precisely as follow

$F_i^k(x|\phi) = x_i^k(R_i(x|k) - R_i(x|k_i))$ for $i \epsilon N, k \epsilon \Phi_i, k \neq k_i$,

$F_i^{k_i}(x|\phi) = \sum_k x_i^k - 1$ for $i \epsilon N$. Then Jacobian matrix is $J(s|\phi) = \frac{\partial F(x|\phi)}{\partial x}|_{x=s}$.

We can easily see that if $s$ is an equilibrium of $\Gamma$ with $\phi \epsilon C(s)$, then $F(s|\phi) = 0$. Hence, we can expect that the Jacobian matrix may have also some nice properties (such as being locally invertible). Following definition employed the Jacobian matrix to define the regular equilibrium.

**Definition A. 4. 2.** An equilibrium $s$ of $\Gamma$ is a regular equilibrium if $J(s|\varphi)$ is nonsingular for some $\varphi \epsilon C(s)$.

**Theorem A. 4. 3.** Every regular equilibrium is strongly stable.

Strongly stable equilibrium means that by small perturbation in the data of the game, the equilibrium will not change.

### A.4.2. Calculations of Theorem 2.14.

As we see in proof of the Theorem 2.14 of section 2.4.2

We define $F(p,q) = \begin{bmatrix} F_1^{(1)}(p,q) \\ F_2^{(1)}(p,q) \\ \vdots \\ F_n^{(1)}(p,q) \\ F_1^{(2)}(p,q) \\ \vdots \\ F_n^{(2)}(p,q) \end{bmatrix}$; where

$F_1^{(1)} = \sum_{i=1}^n p_i - 1$, $F_1^{(2)} = \sum_{i=1}^n q_i - 1$,

$F_i^{(1)} = [u_1(s_1, q) - u_1(s_i, q)] \times p_i = [\sum_{t=1}^{n}(k_{1t} - k_{it})q_t] \times p_i;$
$i = 2, \dots, n.$

$F_j^{(2)} = \left[u_2(s_1, p) - u_2(s_j, p)\right] \times q_j = \left[\sum_{t=1}^{n}((-k_{1t}) - (-k_{jt}))p_t\right] \times q_j;$
$j = 2, \dots, n.$

Hence, we have

$$\frac{dF}{dP} = \begin{bmatrix} \frac{dF_1^{(1)}}{dp_1} & \frac{dF_1^{(1)}}{dp_2} & \dots & \frac{dF_1^{(1)}}{dp_n} & \frac{dF_1^{(1)}}{dq_1} & \dots & \frac{dF_1^{(1)}}{dq_n} \\ \vdots & \vdots & & & & & \vdots \\ \frac{dF_n^{(1)}}{dp_1} & & & & & & \frac{dF_n^{(1)}}{dq_n} \\ \frac{dF_1^{(2)}}{dp_1} & & & & & & \frac{dF_1^{(2)}}{dq_n} \\ \vdots & & & & & & \vdots \\ \frac{dF_n^{(2)}}{dp_n} & & & \dots & & & \frac{dF_n^{(2)}}{dq_n} \end{bmatrix}_{(2n)\times(2n)}$$

We know that $\frac{dF_1^{(1)}}{dp_i} = 1$, $\frac{dF_1^{(1)}}{dq_i} = 0$. Similarly, we have $\frac{dF_1^{(2)}}{dp_i} = 0$, $\frac{dF_1^{(2)}}{dq_i} = 1$.

For $i > 1$ we have

$$\frac{dF_i^{(1)}}{dp_j} = \begin{cases} \sum_{t=1}^{n}(k_{1t} - k_{it})q_t, & if\ i = j, \\ 0, & if\ i \neq j. \end{cases}$$

$\frac{dF_i^{(1)}}{dq_j} = (k_{1j} - k_{ij})\, p_i.$

$\frac{dF_i^{(2)}}{dp_j} = -(k_{1j} - k_{ij})\, q_i.$

$$\frac{dF_i^{(2)}}{dq_j} = \begin{cases} \sum_{t=1}^{n} -(k_{1t} - k_{it})p_t, & if\ i = j, \\ 0, & if\ i \neq j. \end{cases}$$

Hence, $\frac{dF}{dP} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ where

$$A = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \sum_{t=1}^{n}(k_{1t}-k_{2t})q_t & 0 & 0 & 0 \\ 0 & 0 & \sum_{t=1}^{n}(k_{1t}-k_{2t})q_t & & \vdots \\ \vdots & \vdots & \vdots & \vdots & \sum_{t=1}^{n}(k_{1t}-k_{2t})q_t \\ 0 & 0 & 0 & 0 & \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ (k_{11}-k_{21})p_2 & (k_{12}-k_{22})p_2 & (k_{13}-k_{23})p_2 & & (k_{1n}-k_{2n})p_2 \\ (k_{11}-k_{31})p_3 & (k_{12}-k_{31})p_3 & (k_{13}-k_{33})p_3 & & (k_{1n}-k_{3n})p_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (k_{11}-k_{n1})p_n & (k_{12}-k_{n2})p_n & (k_{13}-k_{n3})p_n & \cdots & (k_{1n}-k_{nn})p_n \end{bmatrix}$$

$$C = -\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ (k_{11}-k_{21})q_2 & (k_{12}-k_{22})q_2 & (k_{13}-k_{23})q_2 & & (k_{1n}-k_{2n})q_2 \\ (k_{11}-k_{31})q_3 & (k_{12}-k_{31})q_3 & (k_{13}-k_{33})q_3 & & (k_{1n}-k_{3n})q_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (k_{11}-k_{n1})q_n & (k_{12}-k_{n2})q_n & (k_{13}-k_{n3})q_n & \cdots & (k_{1n}-k_{nn})q_n \end{bmatrix}$$

$$D = -\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \sum_{t=1}^{n}(k_{1t}-k_{2t})p_t & 0 & 0 & 0 \\ 0 & 0 & \sum_{t=1}^{n}(k_{1t}-k_{3t})p_t & & \vdots \\ \vdots & \vdots & \vdots & \vdots & \sum_{t=1}^{n}(k_{1t}-k_{nt})p_t \\ 0 & 0 & 0 & 0 & \end{bmatrix}$$

In equilibrium, $\sum_{t=1}^{n}(k_{1t}-k_{it})q_t$ and $\sum_{t=1}^{n}-(k_{1t}-k_{it})p_t$ are zero.
Hence, in equilibrium

$$\frac{dF}{dP} = \begin{bmatrix} 1 & 1 \cdots & 1 & 0 & 0 \cdots & 0 \\ 0 & \cdots & 0 & (k_{11}-k_{21})p_2 & \cdots (k_{1n}-k_{2n})p_2 & \\ \vdots & & \vdots & \vdots & & \vdots \\ & \cdots & & (k_{11}-k_{n1})p_n & & (k_{1n}-k_{nn})p_n \\ 0 & 0 \cdots & 0 & 1 & 1 \cdots & 1 \\ -(k_{11}-k_{21})q_2 & \cdots -(k_{1n}-k_{2n})q_2 & & 0 & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ -(k_{11}-k_{n1})q_n & & -(k_{1n}-k_{nn})q_n & 0 & 0 \cdots & 0 \end{bmatrix}$$

If change the first row and $(n+1)$th row, then we have matrix $\bar{F}$ which has a similar rank to $\frac{dF}{dP}$. On the other hand, we can see that

$$\bar{F} = \begin{bmatrix} 0 & 0 \cdots & 0 & 1 & 1 \cdots & 1 \\ 0 & \cdots & 0 & (k_{11}-k_{21})p_2 & \cdots & (k_{1n}-k_{2n})p_2 \\ \vdots & & \vdots & \vdots & & \vdots \\ & \cdots & & (k_{11}-k_{n1})p_n & & (k_{1n}-k_{nn})p_n \\ 1 & 1 \cdots & 1 & 0 & 0 \cdots & 0 \\ -(k_{11}-k_{21})q_2 & \cdots & -(k_{1n}-k_{2n})q_2 & 0 & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ -(k_{11}-k_{n1})q_n & & -(k_{1n}-k_{nn})q_n & 0 & 0 \cdots & 0 \end{bmatrix}$$

We know that $p_i > 0$ and $q_j > 0$. If we divide the $i$th row to $p_i$ where $1 \le i \le n$ and to $q_i$ where $n+1 \le i \le 2n$, then we will have $\bar{\bar{F}}$ which has a similar rank to $\bar{F}$. On the other hand, $\bar{\bar{F}} = \begin{bmatrix} 0 & C_1 \\ C_2 & 0 \end{bmatrix}_{(2n)\times(2n)}$, where $C_1$ is the computational matrix of $K$ and $C_2$ computational matrix of -$K$. By Theorem 2.13 we know that both $C_1$ and $C_2$ have full rank. Hence, $\bar{\bar{F}}$ have full rank.

## A.5 Maple Code for Computing the Equilibrium point in Tournament Games

Following Maple code has been designed to receive the number of strategies and $0 \le \alpha \le 1$, to calculate optimal strategies of the asymmetric zero-sum tournament game and symmetric zero-sum tournament game with matrix game $M = K + I$. Optimal strategies can be calculated by linear programing method (**LP_method_maxmin** ) or making all the strategies indifferent (**Indifference_row**). There is also couple of other modules which can test different characteristic of the equilibrium point. **full_support** test if the optimal strategy has full support and **Uniform_dis** test if the optimal strategy has uniform distribution.

```
> restart;
with(LinearAlgebra):
with(Optimization):
with(simplex):
>
Matrices:=proc(a,t)
global C,B,N,N2;
local A,i,j,temp2,check,count3,s;
```

```
##for i from 1 to (2^N2) do
B[t]:=Matrix(N):
##end do:
A:=Matrix(N):
C:=Matrix(N):

##for t from 1 to (2^N2) do
#print(t);

temp2:=0;
while temp2=0 do

A:=RandomMatrix(N,N,generator=0..1,shape=triangular[upper])
:

for i from 1 to N do
 for j from 1 to N do
 B[t][i,j]:=A[i,j];
 end do:
end do:


for i from 1 to N do
 for j from 1 to N do
  if i<j and B[t][i,j]=0 then B[t][i,j]:=-1; end if:
  if i<j and B[t][i,j]=1 then B[t][i,j]:=1; end if:
  if i=j then B[t][i,j]:=a; end if;
  if i<j then B[t][j,i]:=(-1)*B[t][i,j] end if;
 end do:
end do:

check:=0;
for s from 1 to t-1 while check=0 do

count3:=0;
for i from 1 to N do
 for j from 1 to N do
  if s<>t and B[t][i,j]=B[s][i,j] then count3:=count3+1;
end if:
 end do:
end do:
###print("Comparison with",s,t,count3);
if count3=N*N then check:=1; end if;
end do: ###end of checking

if check=1 and t>1 then temp2:=0; else temp2:=1; end if;

end do; ###end of while loop
###print(t,B[t],Determinant(B[t]));

for i from 1 to N do
 for j from 1 to N do
  C[i,j]:=B[t][i,j];
```

```
 end do:
end do:


end proc:

#####################################

> Indifference_row:=proc(A)
global q,D1,Equ;
local Equ_b,i,j;

Equ:=Matrix(N):

Equ_b:=Matrix(N,1):
Equ_b[1,1]:=1;
for i from 2 to N do
 Equ_b[i,1]:=0;
end do:

##print(Equ_b,"RHS");
q:=Matrix(N,1):


for i from 1 to N do
 Equ[1,i]:=1;
end do:

for i from 2 to N do
for j from 1 to N do
 Equ[i,j]:=-(A[i,j]-A[1,j]);
end do:
end do:

###print(Equ);
##print(Rank(Equ));
D1:=Determinant(Equ);
q:=LinearSolve(Equ,Equ_b);
##print("probability",q);


end proc:

########################

> LP_method_maxmin:=proc(A)
global Y;
local i,j,q,A1,Equ,cnsts,obj,set1;

for i from 1 to N do
 for j from 1 to N do
  A1[i,j]:=A[i,j];
 end do:
```

```
end do:

for i from 1 to N do
 for j from 1 to N do
  A1[i,j]:=A1[i,j]+2;
 end do:
end do:


Equ:=Matrix(N+1);

for i from 1 to N+1 do
 for j from 1 to N+1 do
 if i<N+1 and j<N+1 then Equ[i,j]:=A1[i,j]; end if;
 if i=N+1 and j<N+1 then Equ[i,j]:=-1; end if;
 if i<N+1 and j=N+1 then Equ[i,j]:=-1; end if;
 if i=N+1 and j=N+1 then Equ[i,j]:=0; end if;

 end do:
end do:

##print(Equ);

for i from 1 to N+1 do
q[i]:=0;
end do:

for j from 1 to N+1 do
for i from 1 to N+1 do
 q[j]:=Equ[j,i]*y[i]+q[j];
end do:
end do:

for i from 1 to N do
  cnsts[i]:=q[i]<=0;
end do:
cnsts[N+1]:=q[N+1]<=-1;

set1:={seq(cnsts[i],i=1..N+1)};
##print(set1);


obj := -y[N+1]:
Y:=maximize(obj,set1,NONNEGATIVE);
##print(Y);

end proc:
> LP_method_minmax:=proc(A)
global Y;
local i,j,q,A1,Equ,cnsts,obj,set1;

for i from 1 to N do
 for j from 1 to N do
```

```
  A1[i,j]:=A[i,j];
 end do:
end do:

for i from 1 to N do
 for j from 1 to N do
  A1[i,j]:=A1[i,j]+2;
 end do:
end do:


Equ:=Matrix(N+1);

for i from 1 to N+1 do
 for j from 1 to N+1 do
 if i<N+1 and j<N+1 then Equ[i,j]:=A1[i,j]; end if;
 if i=N+1 and j<N+1 then Equ[i,j]:=-1; end if;
 if i<N+1 and j=N+1 then Equ[i,j]:=-1; end if;
 if i=N+1 and j=N+1 then Equ[i,j]:=0; end if;

 end do:
end do:

##print(Equ);

for i from 1 to N+1 do
q[i]:=0;
end do:

for j from 1 to N+1 do
for i from 1 to N+1 do
 q[j]:=Equ[i,j]*y[i]+q[j];
end do:
end do:

for i from 1 to N do
  cnsts[i]:=q[i]>=0;
end do:
cnsts[N+1]:=q[N+1]>=-1;

set1:={seq(cnsts[i],i=1..N+1)};
##print(set1);


obj := -y[N+1]:
Y:=minimize(obj,set1,NONNEGATIVE);
#print(Y);


end proc:
>
full_support:=proc(Y1)
```

```
local count1,i;
global check;

check:=0; count1:=0;

 for i from 1 to N do
 if op(2,Y1[i])>0 then count1:=count1+1; end if;
 end do:
 if count1=N then print("*************** This game is Full
Support***************"); check:=1; end if;

end proc:
>
Uniform_dis:=proc(Y1)
 local count1, I;
 global check2;
check2:=0;count1:=0;
for I from 1 to N do
 if (op(2,Y1[i]))=(1/N) then count1:=count1+1; end if;
 end do:
if count1=N then print("the Optimal strategy is Uniform");
check2:=1; end if;
```

# Chapter 3

# Inspection game with Partial Inspections

## 3.1 Introduction

An inspection game is a mathematical model for a game between two players, where one (the inspectee / potential violator) has enough potential to violate a certain legal act. The other player (the inspector) tries to verify the inspectee's adherence to those legal acts by carrying out inspections over a certain period of time.

Dresher (1962) has studied the case where the inspectee can commit at most one violation during $m$ periods of times, while the number of inspections is limited to a fixed number $n$. Dresher supposed that if the inspector inspects when the inspectee violates, the violation would be detected with probability of $P = 1$. In each stage both players know how many inspection and time periods are left. So, if $n \geq m$ then the inspectee will not violate as he knows that he will be caught for certain. Due to lack of usually $n < m$. Dresher has determined the value and optimal strategies for these cases. Various version of the Inspection game introduced by Dresher's paper, have been analysed in several studies; such as in the works by, B. von Stengel (1991), M. J. Canty et al. (2000) and Rothenstein and Zamir (2002).

In this study, I assume that the inspector may run some "partial inspections" where the probability of detection is not equal to one. The inspections with the probability of detection equal to one would be called a "full inspection".

The assumption of partial inspection is a more realistic representation of real world problems as full inspection can be too costly or time consuming. Hence, instead of a full inspection the inspector may run some partial inspections with lower cost and effort. Hence, by conducting a partial check the probability of detection would be $P$, which is not necessarily equal to one. A famous form of partial inspections a famous

form of partial inspections applies to airplane safety checks. The full safety check of the plane typically takes more than two days, which is a long and costly ground time for the airplanes, making partial inspections much more favourable. However, it is critical that effective inspections guarantee the safety of the flight. I model the situation as a non-cooperative zero sum game and describe the value and optimal strategies of the game using recursive formulae. In particular, I compare the value of the game for inspection games with full and partial inspections only and hence determine the opportunity costs for using these technologies. In a number of cases I provide closed form solutions for the values of the game and the optimal strategies.

In Section 3.2 of this study I review the literature regarding Inspection game, of which the classical form was originally proposed by Dresher (1962). In Section 3.4 I introduce the concept of partial inspections and also I analyse the inspection with just partial inspections. By providing a formula for calculating the value of this game, we can have a tool to make a comparison between full inspections and partial inspections.

In the next section, I investigate the case where the inspector can choose between full inspection, partial inspection and no control in each period of time. I show that in equilibrium the inspector always mixes between full inspections and no control. It means that as long as the opportunity for a full inspection exists, the inspector never starts his sequential inspections with a partial inspection. I also provide a way to calculate the values. This could offer a useful tool for comparison of partial and full inspection, resulting in a classical result of inspection game with full inspections.

## 3.2 Literature review

The inspection game is a mathematical model where one player verifies the commitment of the other player to the certain legal agreements. An example of this could be a customs officer and smuggler at the border control. The customs officer knows that smugglers have enough potential to illegally carry some goods across a border line. However, because of the perishable nature of the goods, the smuggler will choose one of the 1 to $m$ possible time units to carry out the violation. A further example is arm

control treaties, when the there is enough potential that the treaty is violated. Hence, the game is between an inspector and a potential violator which is called inspectee. The problem that typically arises within this situation is because of a lack of inspector's budget to run an inspection in all the events. The amount of the inspection which is left and period of time is common knowledge for both players. Hence, the inspectee knows that there is a chance to violates without being detected.

Seemingly the first genuine inspection game was introduced and analysed by Dresher (1962)[1]. In Dresher's study, the inspector has $n$ number of inspections, where there is $m$ period of time. He models the situation as some sequential zero-sum games. In his model the inspector's payoff for the detected violation is +1, and -1 for the undetected violation. If the inspector can inspect all events then the inspectee knows that he will definitely be caught if he violates. He will therefore not violate, and the payoff for both players will be 0. If we show the value of the game by $v(n,m)$, where $m$ is the number of events and $n$ is number of available inspections we have following boundary conditions.

$$v(n,0) = -1, \;\; v(n,n) = 0.$$

Dresher found the closed form of the value and optimal strategies.

Different versions of the game have been analysed so far. Maschler (1966) analysed the inspection game with different boundary conditions. He shows that the expected payoff of inspector and inspectee by $v(n,m)$ and $w(n,m)$ where $m$ is the number of events and $n$ is number of available inspections, the boundary conditions are

$v(n,n) = \alpha, w(n,n) = \beta$ where $\alpha > 1, 0 < \beta < 1$.

$v(n,n) = 0, w(n,n) = 1$.

Similar to the Dresher paper, Maschler calculates the closed form of $v(n,m)$ and $w(n,m)$. Further works by Maschler (1967) analyses the inspection game as a non-constant-sum game, where each event can produce a special signal to indicate if it is natural or violated.

[1] Dresher's game is fully explained and discussed in Section 3.2.

More recent works include Von Stengel (1991) and Von Stengel (2016), where it is suggested that the inspectee violate more than one time and collect different values of rewards by each violation. The common feature of these works is expecting a one hundred percent accurate result from an inspection. It means that when there is simultaneous inspection and violation the violation is detected for sure. However, in real world this is not always the case.

To my knowledge, the first paper to model the situation between custom officer and smuggler, where the custom officer's inspections are imperfect is Thomas and Nisgav (1976). Similarly to Dresher they assume that there are $m$ periods of time and due to lack of budget the inspector can just run $n < m$ inspections. However, the inspections can detect the violation with probability of $P$. This means that in a simultaneous inspection and violation, the payoff for the inspector will be $2P - 1$. Besides, if we show the value of the game by $g(n, m)$, where $m$ is the number of events and $n$ is number of available inspection, we will have following boundary conditions

$$g(n, 0) = -1, \quad g(n, n) = 2P - 1.$$

By all these assumptions they calculate the value of the game and optimal strategies of the players. They also introduce the case where the customs officer can use two of his inspections in a night to increase the probability of detection. Moreover, they introduce the situation where the customs officer has two different type of inspection with different probabilities of detection, and he can choose to run both in a night to achieve a higher probability of detection, or separately to cover more of the events as inspected. They model both scenarios as a recursive zero-sum game; however, they are unable to find the closed form of the value and optimal strategies as the equations are getting too complicated.

Later, Baston and Bostock (1991) investigate the problem introduced by Thomas and Nisgav (1976) one more time. In their model, the customs officer can have $k_1$ number of inspections with probability of detection $P_1$ and $k_2$ number of inspections with probability of detection $P_2$. Customs officers can run these two type of inspections separately or together. The probability of detection in a joint inspection is $p > max\{P_1, P_2\}$.

They also suppose that when there is no any violation, the customs officer's payoff is 1. It means that customs officer is indifferent between a detected violation and no violation. This is also assumed by Thomas and Nisgav (1976). This assumption is different from Dresher's model where inspectee will not violate if he is sure that he will be detected. However, this difference will not change the closed form of value found by Thomas and Nisgav (1976) for the case of one kind of inspection and $m$ number of period of times (only the boundary condition will change).

In the games analysed by Baston and Bostock (1991) the assumption plays a critical role. They modelled the situation as a zero-sum game and calculated the value of the game. However, they mention that the game with the same assumption as Dresher includes much more complicated equations even if even the probability of detection is supposed to be 1.

Further models where they include an inspection game with probability of detection less than 1 are inspection games with imperfect inspections. The concept of imperfect inspection differs from the kind of inspection (partial inspection) that I am analysing in this study. Imperfect inspection is the inspection with two different types of error. Type One Error means that the inspector may call a false alarm (with probability of $\alpha$), and Type Two Error means that the inspector may fail to detect the violation (with probability of $\beta$). In an imperfect inspection, an error has happened. However in partial inspection the inspector intentionally chooses the partial inspection, despite the fact he knows its probability of detection is not equal to 1.

In industries such as aviation, the kind of the inspection used is partial inspections, since the full inspection (inspection with probability of detection equal to 1) is too costly and time consuming. For example, a full safety check of the airplane may take more than three days which imposes a significant monetary cost to the airlines. Hence, the safety inspectors intentionally run some partial inspections. There may be some occasions which they switch to full inspections, but partial inspection is always one of the methods of inspection.

Imperfect inspection has been analysed in different version of inspection game. Some of the papers include Rothenstein and Zamir (2002) and

Avenhaus and Canty (2005). However, there are not many studies investigating the concept of partial inspection.

The kind of inspections in Thomas and Nisgav (1976) and Baston and Bostock (1991) can also be considered as "Partial inspection". However, these studies have different assumptions of the classical inspection games, which makes the equations slightly easier. Besides, the case of mixing between different types of inspections is not considered.

In this study, I investigate the concept of "partial inspection" in a model with the same assumptions of classical inspection games (similar to Dresher paper), also I analyse the case where in optimal strategy the inspector is mixing between full inspection and partial inspection.

## 3.3 Classical Inspection Game

Dresher (1962) introduced a sequential zero-sum game with two players. One of the players committed to certain legal act. However he has enough potential to violate if he is sure that he is not going to be caught. The other player runs some inspections to ensure that the inspectee does not violate. There are $m$ number of events that the violation can happen, and the number of the inspections is limited to $n$. Because of lack of budget usually $n < m$. The inspector cannot run more than one inspection in each period of time. However, he still cannot cover all the suspicious events. The inspectee may run at most one violation, and after violation the game will be ended. The Number of the inspections and events left is the common knowledge for both players in any stage of the game. The payoff of a detected violation for inspector is +1 and no detection is -1. If no violation happens in the whole the game, the payoff for both players will be 0. At the beginning of each event (period of time) the inspectee will decide to act legally or violate. Dresher (1962) models the game as a dynamical zero-sum game. He denoted the value of the game by $V(n, m)$, where $n$ is the number of inspections and m is the period of times. Hence, he describes the game by the following table.

| Inspectee / Inspector | Legal act | violation |
|---|---|---|
| Inspection | $V(n-1,m-1)$ | +1 |
| No Control | $V(n,m-1)$ | -1 |

**Table 3. 1.** Inspection game modelled by Dresher.

If $n \geq m$, then $V(n,m-1)=0$. Also, $V(n,0)=0$ and $V(0,m)=-1$ where $m>0$.

As we know that $-1 \leq V(n,m) \leq 1$, the players mix between their strategies. If $p$, is the probability of running an inspection then we have

$$p.V(n-1,m-1)+(1-p).V(n,m-1)=p.1+(1-p).(-1).$$

Besides if in equilibrium we denote the probability of violation by $q$, then we have

$$V(n-1,m-1).q+(1-q).(1)=V(n,m-1).q+(1-q).(-1).$$

Hence, we can find the value of the game by following recursive formulas

$$V(n,m)=\frac{V(n,m-1)+V(n-1,m-1)}{V(n-1,m)+2-V(n-1,m-1)}. \qquad (3.3.1)$$

Besides, in equilibrium the probability of inspection is

$$p=\frac{V(n-1,m)+1}{V(n-1,m)+2-V(n-1,m-1)}\ , \qquad (3.3.2)$$

And the probability of violation is

$$q=\frac{2}{V(n-1,m)+2-V(n-1,m-1)}\ , \qquad (3.3.4)$$

Dresher (1962) shows that explicit formula for recursive formula (3.3.1) can be given by

$$V(n,m)=-\frac{\binom{m-1}{n}}{\sum_{i=0}^{n}\binom{m}{i}}\ \ (n<m). \qquad (3.3.5)$$

As, we see by know the explicit formula for the value of the game we can find the probability of inspection and probability of violation as well.

## 3.4 Full Inspection vs. Partial Inspection

In this study, I assume that the inspector may run some partial inspections as well as full inspections. As I mentioned in the literature review, by partial inspection I mean an inspection which can detect the violation by probability $P$. That is not because of any possible mistake by the inspector and is just the nature of the inspection. The inspection in any form (full or partial) is always necessary as it is the only means to prevent the inspectee performing a violation. Similar to Dresher's paper, we have modelled the situation as a zero sum game. I denote the value of the game by $v(n_1, n_2, m, P)$, where $n_1$ is the number of full inspections (which have the probability of detection equal to one), $n_2$ is the number of the partial inspections, $m$ is the number of the period of times and $P$ is the probability of the detection in partial inspections. $n_1, n_2, m$ and $P$ are fixed through the whole game. In each stage of the game the inspector and inspectee both know how many inspections and time periods have been left. Also, in each stage of the game, the inspector can run just one inspection which can be full or partial.

The payoffs are similar to those in Dresher's paper. The only difference arises when the inspectee violates and the inspector partially inspects. In this case the inspector wins with probability $P$ while he loses with probability $1 - P$. So the expected payoff to the inspector is $2P - 1$. In any case, the game ends because the inspectee either achieves his aim or gets caught.

Obviously, when $n_2$ is equal to zero, the game is completely similar to Dresher game and we would have following conditions.

$v(n_1, 0, m, P) = 0;\ n_1 \geq m,\ 0 \leq P \leq 1.\ m \geq 1.$

$v(0,0, m, P) = -1;\ 0 \leq P \leq 1.$

And also we assume $v(n_1, n_2, 0, P) = 0$; which means when no time period has been left of course there would be no detection and not violation.

In our analysis, we exclude the cases of $P = 0$ and $P = 1$. As obviously, if we denote the value of Dresher's game by $V$ then $v(n_1, n_2, m, P) = V(n_1, m)$ and $v(n_1, n_2, m, P) = V(n_1 + n_2, m)$ where $P = 0$ and P=1; respectively.

Additional to the initial conditions discussed before, we have following initial conditions:

$v(0,m,m,P) = 2P-1, 0 < P < \frac{1}{2}, m \geq 1$; (As the probability of detection is not so high, hence in equilibrium the inspectee will choose to violate in a Nash equilibrium).

$v(0,m,m,P) = 0, \frac{1}{2} \leq P \leq 1, m \geq 1$; (As the probability of detection is high, in equilibrium the inspectee will not violate).

**Theorem 3. 1.** [2] In an inspection game, $\boldsymbol{v(0,1,m,P)}$, $\boldsymbol{m > 0}$ where $\boldsymbol{P}$ is the probability of detection in the partial inspection, the value of the game is as follows

a) If $0 < P < 1/2$ then $v(0,1,m,P) = \frac{2P}{m} - 1, m > 0$,
b) If $1/2 < P < 1$ then $(0,1,m,P) = -\frac{m-1}{m-1+2P}, m > 0$.

Proof:

For $v(0,1,m,P)$ we have following table

| Inspectee<br>Inspector | Legal act | Violation |
|---|---|---|
| Partial Inspection | $v(0,0,m-1,P)$ | $2P-1$ |
| No Control | $v(0,1,m-1,P)$ | -1 |

**Table 3. 2.** The inspection game with one partial inspections and $\boldsymbol{m}$ period of times.

We know that $v(0,0,m-1,P) = -1$. Hence, players are mixing between their strategies. By employing the formula in Appendix B.1 we have

$$v(0,1,m,P) = \frac{-1+(2P-1)v(0,1,m-1,P)}{v(0,1,m-1,P)+2P+1}. \quad (3.4.1)$$

Besides, we have following initial conditions $v(0,1,1) = 2P-1$ for $0 < P < 1/2$, and $v(0,1,1) = 0$ for $1/2 < P < 1$. As we see, the initial conditions satisfy part a and b respectively. Hence, they can be used as the start point of an induction.

[2] The formula of the part a of this theorem has been also found by Thomas and Nisgav (1976), for a model which is slightly different. In their model, in the case of no violation Inspector's payoff is 1. This change just affect the border condition and not the general formula.

We can easily see that $\frac{2P}{m+1}-1=\frac{-1+(2P-1)(\frac{2P}{m}-1)}{\frac{2P}{m}-1+2P+1}$ and $-\frac{m-1}{m-1+2P}=\frac{-1+(2P-1)(-\frac{m}{m+2P})}{-\frac{m}{m+2P}+2P+1}$. Hence, we prove the theorem inductively, ■.

**Theorem 3. 2.** In an inspection game, $v(0,n_2,m,P)$ where $n_2<m$, where $P$ is the probability of detection in the partial check, the value of the game is as follows

a) $v(0,n_2,m,P)=\frac{2n_2P}{m}-1, 0<P<1/2;$

b) $v(0,n_2,m,P)=-\frac{\binom{m-1}{n_2}}{\sum_{i=0}^{n_2}\binom{m}{i}(2P-1)^{n_2-i}}, 1/2<P<1.$[3]

Proof:

For $v(0,n_2,m)$ we have following table

| Inspectee / Inspector | Legal act | Violation |
|---|---|---|
| Partial Inspection | $v(0,n_2-1,m-1,P)$ | $2P-1$ |
| No Control | $v(0,n_2,m-1,P)$ | -1 |

**Table 3. 3.** The inspection game with just partial inspections.

Hence, we can calculate the value by the following recursive formulae,

$$v(0,n_2,m,P)=\frac{(2\mathrm{P}-1)v(0,n_2,m-1,P)+v(0,n_2-1,m-1,P)}{v(0,n_2,m-1,P)+2\mathrm{P}-v(0,n_2-1,m-1,P)}, \quad (3.4.2)$$

We know that $v(0,n_2,n_2,P)=2P-1$ for $0<P<1/2$ and $v(0,n_2,n_2,P)=0$ for $\frac{1}{2}<P<1$, which they both satisfy (3.4.2).

For part a, suppose that $v(0,n_2,m,P)=\frac{2n_2P}{m}-1$ we should prove that $v(0,n_2,m+1,P)=\frac{2n_2P}{m+1}-1$. We can easily see that $\frac{2n_2P}{m+1}-1=\frac{(2\mathrm{P}-1)(\frac{2n_2P}{m}-1)+\frac{2(n_2-1)P}{m}-1}{\frac{2n_2P}{m}-1+2\mathrm{P}-\frac{2(n_2-1)P}{m}+1}$. Hence, $v(0,n_2,m+1,P)=\frac{2n_2P}{m+1}-1$.

[3] While calculating $v(0,1,m,P)$ we can see in the Table 3.2 that when players are mixing, then the crucial value is $P=1/2$, as in this case $2P-1=0$. We have modelled the game as a recursive game hence $P=1/2$ will be the crucial value for $v(0,n_2,m,P)$ in Theorem 3.2 as well.

For part b[4], let $s(n_2, m) = \sum_{i=0}^{n_2} \binom{m}{i} (2p-1)^{n_2-i}$. Hence, we will have

$s(n_2, m-1) = (2P-1).s(n_2-1, m-1) + \binom{m-1}{n_2}$ and

$s(n_2, m) = s(n_2, m-1) + s(n_2-1, m-1)$. We can see that $v(0, n_2, m, P) = -\frac{\binom{m-1}{n_2}}{s(n,k)}$ satisfies the recursive formulae (3.4.2), ■.

Theorem 3.1 and Theorem 3.2 can provide the efficient tool to compare full inspections with partial inspections. Not only can we compare the different inspection games, we can also easily calculate how many partial inspections are required to achieve a certain probability of detection similar to that gained from a full inspection (classical inspection game). The following examples show different kind of useful comparisons that we can make.

**Example 3.3.** Figure 3.4 demonstrates the value of three different inspection games when the number of events is changing. The first game which is shown by red dots is the inspection game with just one full inspection. The other inspection games which are shown by green and blue dots are inspection games with just one partial with probability of detection equal to 2/3 and 1/3, respectively. As we could predict, in Figure 3.4 we can see that for any value of $\boldsymbol{m}$, $\boldsymbol{v(1, 0, m, P) \geq v(0, 1, m, 2/3) \geq v(0, 1, m, \frac{1}{3})}$. We can also observe that, as $\boldsymbol{m}$ is getting larger the value of the all the games is approaching -1.

[4] This part of the proof is similar to the proof in Rinderle (1996).

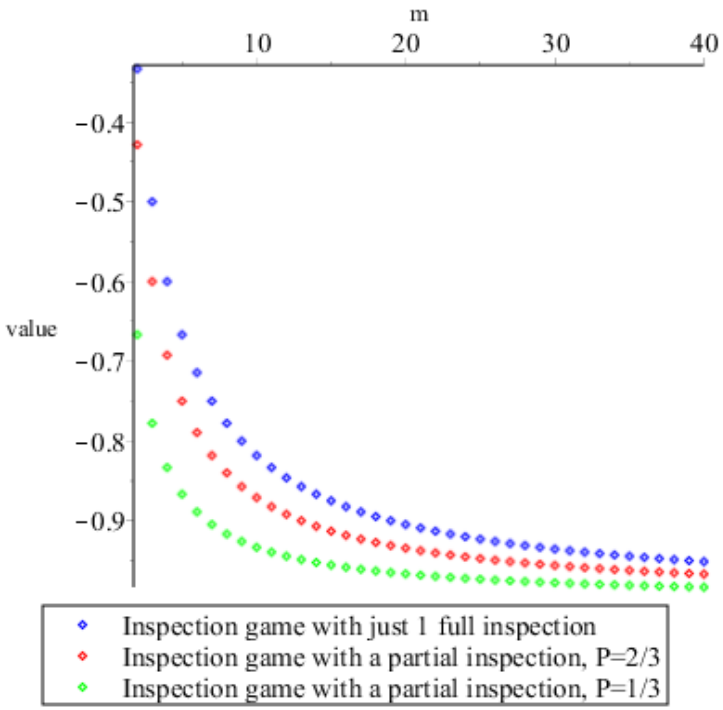



**Figure 3. 4.** Comparison between the value of the inspection game with just one full inspection, or one partial inspection where the probability of detection is 1/3 and 2/3.

**Example 3. 4.** Figure 3.5 compares the value of the inspection games with just one and two full inspections with the inspection game with one and two partial inspections, where probability of detection is 2/3. We can see that for any value of $\boldsymbol{m}$, $\boldsymbol{v(2,0,m,1) \geq v(0,2,m,2/3) \geq v(1,0,m,2/3) \geq v(0,1,m,2/3)}$.

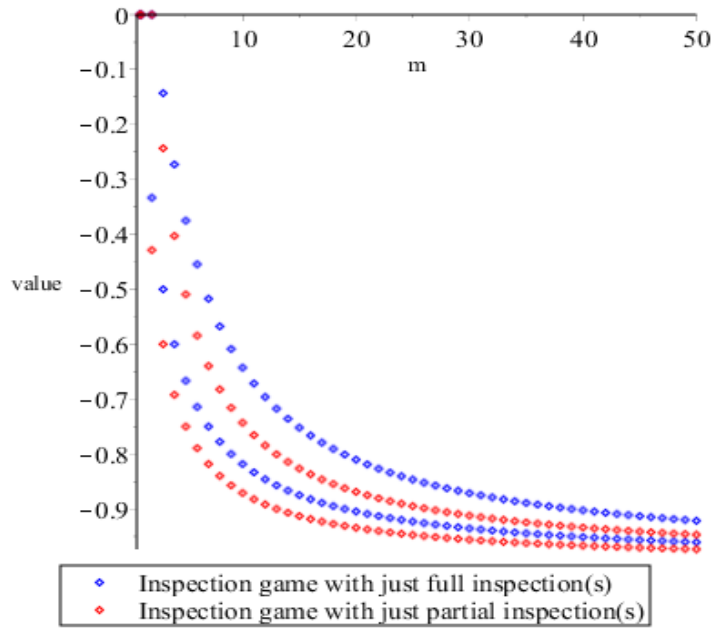



**Figure 3. 5.** The comparison between Inspection games with just full inspections, and inspection games with just partial inspections.

As I mentioned, by employing Theorem 3.1 and Theorem 3.2 we can find out how many partial inspection are required to have the same value a full inspection. The following example provides a comparison between value of the inspection game with just full inspections and just partial inspections.

**Example 3. 5.** Suppose an inspection game with 2 full inspections and 6 periods of time. Hence by the Dresher formula (formula 3.3.5) we know that $v(2,0,6,P) = V(2,7) = -0.3125$. If the inspector decides to, instead of full inspection, run partial inspections with probability of detection equal 2/5, he will need at least five partial inspections. As we know that

$$v(0,4,7,2/5) < -0.3125 < v(0,5,7,2/5).$$

**Example 3. 6.** Figure 3.6 demonstrates that if we have 100 period of times, for different numbers of full inspections ($\boldsymbol{n_1}$) how many partial inspections ($\boldsymbol{n_2}$) are required to get the same value. In this example, the probability of detection for the partial inspection is 2/3. The blue line is the relation between the number of full and partial inspections, and the red line is when $\boldsymbol{n_1 = n_2}$. Hence we can easily see that relation between the number of full and partial inspections is not linear.

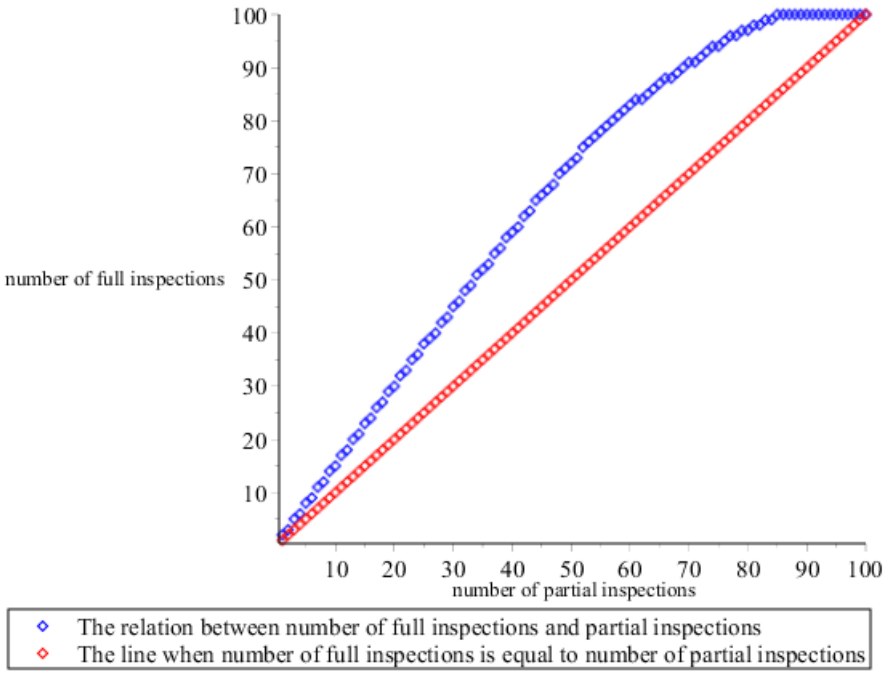



**Figure 3. 6.** The relation between number of full inspections and partial inspections to guarantee the same value.

The following theorem explains the results in Example 3.6 theoretically.

**Theorem 3. 7.** For an inspection game with $\boldsymbol{n_1}$ number of full inspections, we need at least $\left[\left(-\frac{\binom{m-1}{n_1}}{\sum_{i=0}^{n}\binom{m}{i}}+\mathbf{1}\right)\left(\frac{m}{2P}\right)\right]+\mathbf{1}$ partial inspections with a probability of detection $\boldsymbol{P}$ to achieve the same value, if $< \boldsymbol{P} < \mathbf{1/2}$ .

Proof:

By Dresher formula we know that the value of an inspection game with n full inspection is $-\frac{\binom{m-1}{n_1}}{\sum_{i=0}^{n}\binom{m}{i}}$ . Besides, by Theorem 3.2 we know that when $0 < P < 1/2$, the value of the inspection game with $n_2$ number of partial inspections is $\frac{2n_2P}{m}-1$.

$$\frac{2n_2P}{m}-1=-\frac{\binom{m-1}{n_1}}{\sum_{i=0}^{n}\binom{m}{i}} \Leftrightarrow n_2=\left(-\frac{\binom{m-1}{n_1}}{\sum_{i=0}^{n}\binom{m}{i}}+1\right)\left(\frac{m}{2P}\right),$$

Hence, we need $n_2=\left[\left(-\frac{\binom{m-1}{n_1}}{\sum_{i=0}^{n}\binom{m}{i}}+1\right)\left(\frac{m}{2P}\right)\right]+1$, ■.

We cannot have a kind of similar theorem for $1/2 < P < 1$, as it is not possible to find the explicit formula for $n_2$.

**Theorem 3. 8.** Consider an inspection game with no full inspection, $n_2$ number of partial inspections and $m$ period of time, where $P$ is the probability of detection in partial inspections. If $p$ is the probability of assigning a partial check and $q$ is the probability of selecting one of the times to violate, then for $0 < P < 1/2$ we have

$p=\frac{n_2}{m}$ (probability of assigning a partial check),

$q=1-\frac{1}{m}$ (probability of selecting one of the times to violate).

For $1/2 < P < 1$ we have

$$p=\frac{(2P)^{n_2-1}+(2P)^{n_2-2}\binom{m-n2}{1}+\cdots+\binom{m-2}{n_2-1}}{(2P)^{n_2}+(2P)^{n_2-1}\binom{m-n2}{1}+\cdots+(2P)\binom{m-2}{n_2-1}+\binom{m-1}{n_2}},$$

$$q = 1 - \frac{2P}{-\frac{\binom{m-2}{n_2}}{\sum_{i=0}^{n_2}\binom{m-1}{i}(2P-1)^{n_2-i}} + \frac{\binom{m-2}{n_2-1}}{\sum_{i=0}^{n_2-1}\binom{m-1}{i}(2P-1)^{n_2-i-1}}} \ .$$

Proof:

By employing Table 3.3 we know that in the equilibrium we have

$$pv(0, n_2 - 1, m - 1, P) + (1 - p)v(0, n_2, m - 1, P)$$
$$= p(2P - 1) + (1 - p)(-1)$$

and

$$q(2P - 1) + (1 - q)v(0, n_2 - 1, m - 1, P)$$
$$= -q + (1 - q)v(0, n_2, m - 1, P)$$

$$p = \frac{v(0,n_2,m-1,P)+1}{v(0,n_2,m-1,P)+2\mathrm{P}-v(0,n_2-1,m-1,P)} \ , \qquad (3.8.1)$$

$$1 - q = \frac{2P}{v(0,n_2,m-1,P)-v(0,n_2-1,m-1,P)+2\mathrm{P}} \ . \qquad (3.8.2)$$

For $0 < P < 1/2$, by the Theorem 3.2 we know that $v(0, n_2, m, P) = \frac{2n_2P}{m} - 1$. Hence, by formula 3.8.1 and 3.8.2 we can see that $p = \frac{n_2}{m}$ and $q = 1 - \frac{1}{m}$.

For $1/2 < P < 1$, by the Theorem 3.2 we know that $v(0, n_2, m, P) = -\frac{\binom{m-1}{n_2}}{\sum_{i=0}^{n_2}\binom{m}{i}(2P-1)^{n_2-i}}$. Hence, by employing formula 3.8.1 and 3.8.2 we can show the result, ■.

## 3.5 Inspection Game with Partial Inspection

In this section I consider a version of Inspection game, that the inspector can employ both of the inspection technologies. In other words, $n_1$ and $n_2$ are both positive. We can describe the $v(n_1, n_2, m, P)$ by following table

| Inspectee / Inspector | Legal act | violation |
|---|---|---|
| Full Inspection | $v(n_1 - 1, n_2, m - 1, P)$ | +1 |
| Partial Inspection | $v(n_1, n_2 - 1, m - 1, P)$ | $2P - 1$ |
| No Control | $v(n_1, n_2, m - 1, P)$ | -1 |

**Table 3. 7**. Inspection game with Partial Inspection

We know that

$v(n_1 - 1, n_2, m - 1, P) < v(n_1, n_2 - 1, m - 1, P) < v(n_1, n_2, m - 1, P)$.

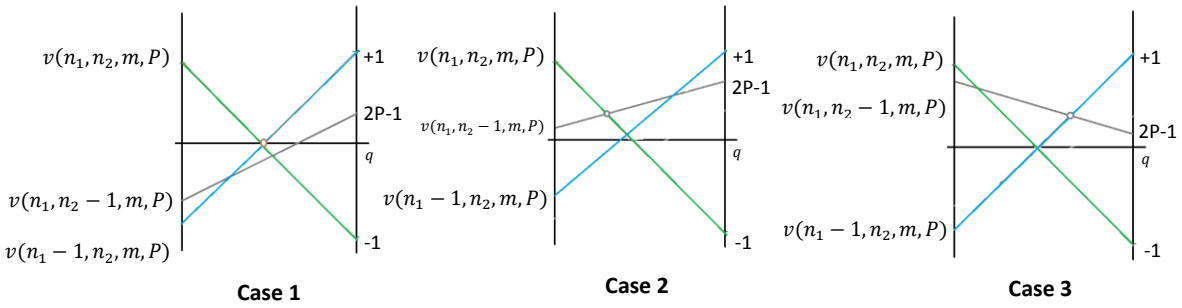
Besides, for all value of $n_1, n_2$ and $m$ $-1 \leq v(n_1, n_2, m, P) \leq 1$. On the other hand, as $0 < P < 1$ we know that $0 < 2P - 1 < 1$. Hence, we can easily observe that there is a mixed strategy equilibrium. To analyse the equilibrium three cases may happen.

**Case 1**: In equilibrium inspector mixes between No Control and Full inspection.

**Case 2**: In equilibrium inspector mixes between No Control and Partial inspection.

**Case 3:** In equilibrium inspector mixes between Full-inspection and Partial inspection.

These cases are shown in the following figures. In all the following graphs horizontal line shows the probability of the violation ($q$) by the inspectee and the vertical line shows the expected pay off for the inspector.




Green line denotes Not- check strategy, while inspectee is mixing between violate and not violate.

Blue line denotes Full- check strategy, while inspectee is mixing between violate and not violate.

Grey line denotes Partial- check strategy, while inspectee is mixing between violate and not violate.

The equilibrium point (Minmax strategy) has been denoted by a circle.

**Figure 3. 8.** Three possible cases for the equilibrium in inspection game with partial inspection

By knowing the initial condition of the value and the fact that the equilibrium is one of the cases in Figure 3.8, we can design a computer program to calculate the $v(n_1, n_2, m, P)$. The Maple code for computing the equilibrium point and its description are provided in Appendix B.1. The value of the game with one full inspection and one partial inspection is shown in the following two tables. Some other examples are provided in Appendix B.2

| $m$ | $v(1,1,m,P)$, $0 < P < 1/2$ | *Type of the Case* |
|---|---|---|
| 2 | $\frac{(2P-1)^2}{4P-3}$ | Case 3 |
| 3 | $\frac{8P^2-11P+4}{11P-8}$ | Case 1 |
| 4 | $\frac{23P^2-41P+18}{P^2+41P-30}$ | Case 1 |
| 5 | $-\frac{P^3+85P^2-194P+96}{P^3-11P^2-194P+144}$ | Case 1 |
| 6 | $-\frac{P^4-16P^3-379P^2+1114P-600}{P^4-16P^3+101P^2+1114P-840}$ | Case 1 |

**Table 3. 9.** $\boldsymbol{v(1,1,m,P)}$ for $\boldsymbol{2 \le m \le 6}$ where $\boldsymbol{0 < P < 1/2}$.

| $m$ | $v(1,1,m,P)$, $1/2 < P < 1$ | *Type of the Case* |
|---|---|---|
| 2 | 0 | Case 1 |
| 3 | $-\frac{1}{4P+3}$ | Case 1 |
| 4 | $-\frac{5P+4}{8P^2+17P+8}$ | Case 1 |
| 5 | $-\frac{17P^2+37P+18}{16P^3+65P^2+81P+30}$ | Case 1 |
| 6 | $-\frac{49P^2+103P+48}{32P^3+145P^2+191P+72}$ | Case 1 |

**Table 3. 10.** $\boldsymbol{v(1,1,m,P)}$ for $\boldsymbol{2 \le m \le 6}$ where $\boldsymbol{1/2 < P < 1}$.

Employing computer programing is computationally costly and slow. The following theorems help to find the value and optimal strategies much quicker.

**Theorem 3. 8.** For the Inspection game $\boldsymbol{v(1,1,m,P)}$, when $\boldsymbol{m>2}$, in the equilibrium the inspector is mixing between Full-inspection and Not-inspection.

Proof:

We can describe the game by the following graph.

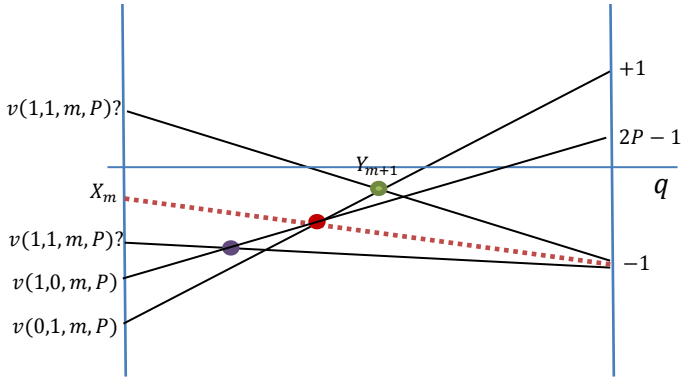



**Figure 3. 11.** Inspection game with one full inspection and one partial inspection

● is the connection of **Full check line** and **Not check line.**

● is the connection of **Partial check line** and **Not check line.**

● is the connection of **Partial check line** and **Full check line.**

..... is the line passing the intersection of Partial check line and Full check line, and (1,-1).

In the Graph 2.3 we know that $v(1,1,m,P)>v(1,0,m,P)>v(0,1,m,P)$; however we do not know if $v(1,1,m,P)\geq X_m$.

If $v(1,1,m,P)\geq X_m$ then the equilibrium point is at intersection of Full inspection and No control; on the other hand, if $v(1,1,m,P)<X_m$ then the equilibrium point is at intersection of Partial inspection and No control.

We want to show that $v(1,1,m,P)\geq X_m$, $\forall\, m>2$; and as a result the equilibrium point is the intersection of full inspection and not inspection.

We know that $v(0,1,1,P)=0, v(1,0,1,P)=0$ and $v(1,1,1,P)=0$ for $\frac{1}{2}<P<1$; as a result $v(1,1,2,P)=0$ for $\frac{1}{2}<P<1$ where in equilibrium the inspector is mixing between all of his strategies. Besides, as we know

$v(1,1,2,P) = 0$, $v(1,0,2,P) = -\frac{1}{3}$ and $v(0,1,2,P) = -\frac{1}{2P+1}$ ; as a result we can easily calculate that $v(1,1,3,P) = -\frac{1}{4P+3}$ where in equilibrium inspector is mixing between full inspection and no control.

Suppose that $v(1,1,m,P) \geq X_m$; we have to show that $v(1,1,m+1,P) \geq X_{m+1}$.

We know that $v(1,1,m,P) \geq X_m$ if and only if $v(1,1,m+1,P) \geq Y_{m+1}$; hence if we show that $Y_{m+1} \geq X_{m+1}$ then it will implies that $v(1,1,m+1,P) \geq X_{m+1}$.

Full inspection line is the line passing two following points $(0, v(0,1,m,P))$ and $(1,1)$. By Theorem 3.1 we know that $v(0,1,m,P) = -\frac{m-1}{m-1+2P}$ . So the full inspection line is $\left(-\frac{m-1}{m-1+2P} - 1\right)x + y + \frac{m-1}{m-1+2P} = 0$ .

Partial inspection line is the line passing two following points $(0, v(1,0,m,P))$ and $(1, 2P-1)$. By the Dresher formula (formula 3.3.5) we know that $v(0,1,m,P) = -\frac{m-1}{m+1}$ . So the partial inspection line is $\left(-\frac{m-1}{m+1} - 2P + 1\right)x + y + \frac{m-1}{m+1} = 0$ .

Hence, the intersection of full inspection and partial inspection line is $x_1 = \frac{m-1}{2Pm + m^2 + 2P + m - 2}$ , $y_1 = -\frac{(m-1)m}{2Pm + m^2 + 2P + m - 2}$ .

.

Hence, $Y_{m+1} = -\frac{(m-1)m}{2Pm+m^2+2P+m-2}$ .

On the other hand, the line between $(x_1, y_1)$ and *(1,-1)* is

$\frac{x-x_1}{y-y_1} = \frac{1-x_1}{-1-y_1}$. If $x = 0$, then $y = (\frac{1-y_1}{-1-x})(-x_1) + y1$.

Hence, $X_m = -\frac{(m-1)^2}{2Pm + m^2 + 2P - 1}$ .

As a result, $Y_{m+1} - X_{m+1} = \frac{4Pm}{(2Pm + m^2 + 2P + m - 2)(2Pm + m^2 + 4P + 2m)}$ , which is positive for $m > 2$ and $P > \frac{1}{2}$. Hence, if $1/2 < P < 1$ in equilibrium the inspector is mixing between full inspection and no control.

We know that for $0 < P < 1/2$, in the game $v(1,1,3,P)$ in equilibrium the inspector is mixing between full inspection and no control (Case 1). Hence, similar to the induction we made for $1/2 < P < 1$ we suppose that $v(1,1,m,P) \geq X_m$ and we have to show that $v(1,1,m+1,P) \geq X_{m+1}$, which is equivalent to show $Y_{m+1} - X_{m+1} \geq 0$.

Full inspection line is the line passing two following points

$[(0, v(0,1,m,P)), (1,1)]$. We know by Theorem 3.1 that $(0,1,m,P) = \frac{2P}{m} - 1$ , so the Full check line is $\left(\frac{2P}{m} - 2\right)x + y + 1 - \frac{2P}{m} = 0$ .

Partial inspection line is the line passing two following points

$[(0, v(1,0,m,P)), (1, 2P-1)]$. By the Dresher formula we know that $v(1,0,m,P) = -\frac{m-1}{m+1}$, so the Partial inspection line is $\left(-\frac{m-1}{m+1} - 2P + 1\right)x + y + \frac{m-1}{m+1} = 0$ .

The Intersection between the full inspection and partial inspection line is $x_1 = \frac{Pm + P - m}{Pm^2 + 2Pm - m^2 + P - 2m}$

$y_1 = \frac{2P^2 m - Pm^2 + 2P^2 - 2Pm + m^2 - P}{Pm^2 + 2Pm - m^2 + P - 2m}$

Hence, $Y_{m+1} = \frac{2P^2 m - Pm^2 + 2P^2 - 2Pm + m^2 - P}{Pm^2 + 2Pm - m^2 + P - 2m}$ .

On the other hand, the line between $(x_1, y_1)$ and *(1,-1)* is

$\frac{x - x_1}{y - y_1} = \frac{1 - x_1}{-1 - y_1}$. If $x = 0$, then $y = \left(\frac{-1-y_1}{-1-x}\right)(-x_1) + y1$.

Hence, $X_m = \frac{2P^2 m - Pm^2 + 2P^2 - Pm + m^2 - m}{m(Pm + P - m - 1)}$

As a result, $Y_{m+1} - X_{m+1} = -\frac{2P(Pm + 2P - m - 1)}{(Pm^2 + 2Pm - m^2 + P - 2m)(m+1)(Pm + 2P - m - 2)}$

We know that $Pm^2 + 2Pm - m^2 + P - 2m = m^2(P-1) + m(2P-2) + P$ which is always negative. On the other hand, $(Pm + 2P -$

$m-1)$ and $(Pm+2P-m-2)$ are both negative. Hence, $Y_{m+1}-X_{m+1}\geq 0$. Which means that for $0<P<1/2$ and $m>2$ in equilibrium the inspector mixes between full inspection and no control, ■.

**Corollary 3. 9.** For the inspection game with 1 full inspection and 1 partial inspection, if in equilibrium $\boldsymbol{p(1,1,m,P)}$ is the optimal probability of assigning the full inspection, and $\boldsymbol{q(1,1,m,P)}$ is probability of violation and $\boldsymbol{v(1,1,m,P)}$ is the value of the game then they can be calculated by following recursive formulas.

$$v(1,1,m,P)=\frac{v(1,1,m-1,P)+v(0,1,m-1,P)}{v(1,1,m-1,P)+2-v(0,1,m-1,P)} \qquad (3.5.1)$$

$$p(1,1,m,P)=\frac{v(1,1,m-1,P)+1}{v(1,1,m-1,P)+2-v(0,1,m-1,P)} \qquad (3.5.2)$$

$$q(1,1,m,P)=\frac{2}{v(1,1,m-1,P)+2-v(0,1,m-1,P)} \qquad (3.5.3)$$

Proof:
We know that we can describe the $v(1,1,m,P)$ by the following table.

| Inspectee / Inspector | Not Violate | Violate |
|---|---|---|
| Full Inspection | $v(0,1,m-1,P)$ | +1 |
| Partial Inspection | $v(1,0,m-1,P)$ | 2P-1 |
| No Control | $v(1,1,m-1,P)$ | -1 |

**Table 3. 12.** Inspection game with 1 full inspection and 1 partial inspection.

Besides, by Theorem 3.8 we know that in equilibrium the inspector is mixing between the full inspection and no control. Hence, as it has been explained in Appendix B.1 the value is equal to

$$v(1,1,m,P)=\frac{v(1,1,m-1,P)+v(0,1,m-1,P)}{v(1,1,m-1,P)+2-v(0,1,m-1,P)}.$$

$p(1,1,m,P)$ and $q(1,1,m,P)$ can also be calculated by the formulas provided in Appendix B.1, ■.

**Theorem 3. 10.** In an inspection game with $n_1$ number of full inspections, $n_2$ number of partial inspection and $m$ period of times, for $n_1 + n_2 \leq 50$ in the equilibrium we have

a) For $0 < P < 1/2$, the inspector mixes between partial inspection and full inspection (Case 3) if $n_1 + n_2 = m$; and the inspector mixes between full inspection and no control (Case 1) if $n_1 + n_2 < m$.
b) For $1/2 < P < 1$, the inspector mixes between full inspection and no control.

Proof:

By employing the Maple code provided in Appendix B.2 we have analyzed all the inspection games with $n_1$ number of full inspections and $n_2$ number of partial inspections and $m$ period of times where $n_1 + n_2 \leq m \leq 50$. The results of the computations are the same with the claim in the Theorem, ■.

Theorem 3.10 is providing a general guess for the behavior of the players in the equilibrium. It seems that except when $n_1 + n_2 = m$ and $0 < P < 1/2$ which is Case 3, in all the other situations in equilibrium the inspector is mixing between full inspection and partial inspection. It means that the inspector never starts his sequential inspections with a partial inspection. In other words, as long as any opportunity exists for running a full inspection, the inspector will not run any partial inspections. Partial inspections will be used when no remaining full inspections are available.

The other benefit, application of Theorem 3.10 is finding an efficient way to determine the value and optimal strategies of the game. The following theorem explains how Theorem 3.10 may help to find the more efficient way to calculate the value of the game and optimal strategies.

**Theorem 3. 11.** If in an inspection game with partial inspection there are a number of full inspections, a number of partial inspection and m period of times; if we know that in equilibrium the inspector is mixing between full inspection and no control then the value of the game can be calculated by the values of the previous level by following formula

$$v(n_1, n_2, m, P) = \frac{v(n_1,n_2,m-1,P)+v(n_1-1,n_2,m-1,P)}{v(n_1,n_2,m-1,P)+2-v(n_1-1,n_2,m-1,P)}. \qquad (3.5.4)$$

If we know that in equilibrium the inspector is mixing between partial inspection and no control then the value of the game can be calculated by the values of the previous level by following formula

$$v(n_1, n_2, m, P) = \frac{(2P-1)v(n_1,n_2,m-1,P)+v(n_1,n_2-1,m-1,P)}{v(n_1,n_2,m-1,P)+2P-v(n_1,n_2-1,m-1,P)}. \quad (3.5.5)$$

If we know that in equilibrium the inspector is mixing between partial inspection and full inspection then the value of the game can be calculated by the values of the previous level by following formula

$$v(n_1, n_2, m, P) = \frac{v(n_1,n_2-1,m-1,P)-(2P-1)v(n_1-1,n_2,m-1,P)}{v(n_1,n_2-1,m-1,P)+(2-2P)-v(n_1-1,n_2,m-1,P)}. \quad (3.5.6)$$

## Conclusion

I investigate the classical inspection game while not all the inspections can fully detect the violation. For some cases I show that the inspector always applies a full inspection at the beginning of the series of inspections. This characteristic leads to find a recursive formula for calculating the value of the game. The recursive formulae and its explicit solution computationally would be less costly, comparing to the computer programing method. Values of the games with a different number of full and partial inspection can provide a robust tool to compare the different type of the inspections.

## Reference


Avenhaus, R. and M. J. Canty (2005). 'Playing for time: A sequential inspection game', *European Journal of Operational Research*, vol. **167**(**2**), pp. 475-492.

Baston, V. J. and F. A. Bostock (1991). 'A GENERALIZED INSPECTION GAME', *Naval Research Logistics*, vol. **38**(**2**), pp. 171-182.

Canty, M. J., D. Rothenstein and R. Avenhaus (2001). 'Timely inspection and deterrence', *European Journal of Operational Research*, vol. **131**(**1**), pp. 208-223.

Dresher, M. (1962) *A sampling inspection problem in arms control agreements: A game-theoretic analysis*, DTIC Document.

Maschler, M. (1966). 'A price leadership method for solving the inspector's non-constant-sum game', *Naval Research Logistics Quarterly*, vol. **13**(**1**), pp. 11-33.
Maschler, M. (1967). 'The inspector's non-constant-sum game: Its dependence on a system of detectors', *Naval Research Logistics Quarterly*, vol. **14**(**3**), pp. 275-290.
Rinderle, K. (1996). *Mehrstufige sequentielle Inspektionsspiele mit statistischen Fehlern erster und zweiter Art*: Kovač.
Rothenstein, D. and S. Zamir (2002). 'Imperfect inspection games over time', *Annals of Operations Research*, vol. **109**(**1-4**), pp. 175-192.
Thomas, M. U. and Y. Nisgav (1976). 'An infiltration game with time dependent payoff', *Naval Research Logistics Quarterly*, vol. **23**(**2**), pp. 297-302.
von Stengel, B. (1991). 'Recursive inspection games, Report No. S 9106', *Computer Science Faculty, Armed Forces University Munich*.
Von Stengel, B. (2016). 'Recursive inspection games', *Mathematics of operations research*.

# B Appendix to Chapter 3

## B.1 The value and optimal strategies for a 2×2 zero-sum game

Let following table be the matrix game of a two player zero-sum game (See Von Stengel (1991)).

| Player 2<br>Player 1 | Strategy C | Strategy D |
|---|---|---|
| Strategy A | a | b |
| Strategy B | c | d |

If $a \leq b, \quad c > d, \quad a \leq c, \quad b > d$, then we can easily observe that in equilibrium players are mixing between their strategies. We denote the probability of playing Strategy A by Player 1 with $p$, the probability of playing Strategy C by Player 2 and the value of the game with $v$. As the players are indifferent between their strategies then we have

$$v = p.a + (1-p).c = p.b + (1-p).d,$$

$$v = q.a + (1-q).b = q.c + (1-q).d.$$

Hence, we have

$$p = \frac{c-d}{c-d+b-a}\ , \qquad \text{(B.1.1)}$$

$$q = \frac{b-d}{c-a+b-d}\ , \qquad \text{(B.1.2)}$$

$$v = \frac{b.c-a.d}{c-d+b-a}\ . \qquad \text{(B.1.3)}$$

## B.2 Maple code for finding the equilibrium point for the inspection game with partial inspection

The maple code provided in this section finds the equilibrium point and calculates the value of the inspection game with partial game for different value of full inspection, partial inspection and period of times. The user can set the code to work for small $(0 < P < 1/2)$ or large $(\frac{1}{2} < P < 1)$

probability of detection for the partial inspection. The program calculates the value of all the games $v(n_1, n_2, m, P)$ $n_1 \leq N_1, n_2 \leq N_2, m \leq M$ where $N_1, N_2$ and $M$ can be set by user. $v(n_1, n_2, m, P)$ where $n_1 + n_2 \leq m$ are calculated in following orders.

First $v(1,1,1,P)$ to $v(1,1,M,P)$, then $v(1,2,3,P)$ to $v(1,2,M,P)$. This continue to $v(1, N_2, N_2 + 1, P)$ to $v(1, N_2, M, P)$. Finally, $v(2, N_2, M, P)$ to $v(2, N_2, M, P)$; which will continue to $v(N_1, 2,1, P)$ to $v(N_1, 2, M, P)$.

The program runs following steps

a) Calculating $v(n_1, 0, m, P)$ by Dresher formula for $n_1 \leq N_1$ and $m \leq M$.

b) Calculating $v(0, n_2, m, P)$ $n_2 \leq N_2$ and $m \leq M$ by the formulas provided in Theorem 3.2 for small or large value of $P$ (Depends on user to select which of them)

c) Setting $v(n_1, n_2, m, P) = 0$ for $n_1 \geq m$.

d) For a given $n_1, n_2$ and $m$, the program calculates the equation of following 3 lines given the coordination of their start point and end point:

   No control line $\big[\big(0, v(n_1, n_2, m-1, P)\big), (1, -1)\big]$.

   Partial inspection line $\big[\big(0, v(n_1, n_2 - 1, m-1, P)\big), (1, 2 * P - 1)\big]$.
   Full inspection line $\big[\big(0, v(n_1 - 1, n_2, m-1, P)\big), (1, 1)\big]$.

e) The program calculates the intersection of each two different lines in part e). Hence, we will have the coordination of the following point.
   NP= intersection of no control line and Partial inspection line.
   FP= intersection of full inspection line and Partial inspection line.
   FN= intersection of full inspection line and no control line.

f) The program compares the height of the calculated points for all the values of $P$. If NP had the greatest height among the others, it means that NP is the optimal strategy (Minmax). Otherwise, the equilibrium point is FP or FN. If height of FP is smaller than height of FN then the equilibrium point is FP, otherwise it is FN.

The following maple codes calculate the value of the game $(n_1, n_2, m, P)$ $n_1 \leq 2, n_2 \leq 2, m \leq 5$. The results of the code are shown by blue.

```
> restart;
```

> $N1 := 2 : N2 := 2 : M := 5 : \quad BB := M + 3 :$

> $\#\#\#\#\# \; for\ small\ P \quad a1 := 0 : b1 := \frac{1}{2} :$

> $\#\#\#\#\# for\ large\ P\ a1 := \frac{1}{2} : b1 := 1 :$

```
>
P := proc(n, k)
RETURN(product(n-j, j=0..(k-1)))
end:

> C := proc(n, k)
RETURN(P(n,k)/k!)
end:

>
Value1:=proc(i,m)
global Value;
local j,b,A,AA;

if m<=i then Value[i,0,m]:=0; else
#print(i,m);
#print(C(m-1,i));
A:=C(m-1,i);
##print("A is",A);

AA:=0;
for j from 0 to i do
 b:=C(m,j):
AA:=AA+b:
end do:
#print(AA);
Value[i,0,m]:=-A/AA;
end if;
#print("Value is",i,m,Value[i,0,m]);
end proc:
>
for i from 1 to BB do
for m from 1 to BB do
 Value1(i,m-1);
end do:
end do:
```

```
> ###################### for large P
Vlarge:=proc(n1,n2,m1)
 global Value;
 local temp1,i,temp2;
 temp1:=0;
for i from 0 to n2 do
 temp1:=((2*P)^(i))*binomial(m1-1-i,n2-i)+temp1:
end do:
 temp2:=binomial(m1-1,n2);
 Value[0,n2,m1]:=-temp2/temp1;
 #print(Value[0,n2,m1]);

end proc:
>
> for n2 from 1 to BB do
 for m from 1 to BB do
Vlarge(0,n2,m);
 end do:
end do:
>
>
>
>
```

```
##################  for Large P
 Value[0, 1, 1] := 0 :
for i1 from 1 to 10 do
 for i2 from 1 to 10 do
  for m from 1 to 10 do
   if i1 + i2 ≥ m then Value[i1, i2, m] := 0; end if:
  end do:
 end do:
end do:
```

```
> ######### for small P, 0<P<1/2
Value[1, 1, 1] := 0;
for i1 from 1 to 10 do
 for i2 from 1 to 10 do
  for m from 1 to 10 do
 if i2>= m then Value[0, i2, m]:=2*P-1;  end if;
 if i1>= m then Value[i1, i2, m]:=0;  end if;
   end do:
  end do:
 end do:


>####### for small P
for n2 from 1 to 10 do
for m from 1 to 10 do
```

```
 Value[0,n2,m]:=((2*P*n2)/(m))-1;
 #print(0,n2,m,Value[0,n2,m]);
end do:
end do:

>
for n1 from 1 to 2 do
for n2 from 1 to N2 do
for m from 2 to M do

print("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&",n1,n2,m);

#print(Value[n1,n2,m-1]);
geometry[point](e,0,Value[n1,n2,m-1]):
geometry[point](f,1,-1):


#print(n2-1,m-1,Value[n1,n2-1,m-1]);
geometry[point](c,0,Value[n1,n2-1,m-1]):
geometry[point](d,1,2*P-1):

aa:=n2;
if n2>(m-1) then aa:=m-1; end if;
Value1(aa,m-1);
Value[0,n2,m-1]:=Value[0,aa,m-1];

#print(Value[n1-1,n2,m-1]);
geometry[point](a,0,Value[n1-1,n2,m-1]):
geometry[point](b,1,1):
geometry[line](l1,[a,b]):
#if m=8 then print("full check"): end if:
F:=geometry[Equation](l1,[x,y]);
#print("full check",F):


geometry[line](l2,[c,d]):
P1:=geometry[Equation](l2,[x,y]);
#print("partial check",P1):


geometry[line](l3,[e,f]):
N:=geometry[Equation](l3,[x,y]);
#print("Not check",N);


#print("Full & Not");
```

```
FN:=solve({F,N},{x,y});
#print("Full & Partial");
FP1:=solve({F,P1},{x,y});
#print("Partial & Not");
NP1:=solve({N,P1},{x,y});



FN:=op(2,op(2,FN));
FP1:=op(2,op(2,FP1));
NP1:=op(2,op(2,NP1));

temp1:=minimize(FN-FP1,P=a1..b1);
temp2:=minimize(FN-NP1,P=a1..b1);

check:=1;
if temp1>=0 and temp2>=0 then
print("Case1:Value is intersection of Not-Check and Full-
Check");
value1:=FN;
print("the value is",value1);
check:=1;
else
  check:=0;
  print("It is not Case 1",n1,n2,m);
  temp3:=minimize(NP1-FP1,P=a1..b1);
    if temp3<0 then
    print("Case2:Value is intersection of Not-Check and
Partial-Check");
    value1:=NP1;
    print("the value is",value1);
    else
    print("Case3:Value is intersection of Full-Check and
Partial-Check");
    value1:=FP1;
    print("the value is",value1);
    end if;
end if;

Value[n1,n2,m]:=value1:
#print(n1,n2,m);
#print(Value[n1,n2,m]);
#if check=0 then print("The plot of intersections");
plot([FN,FP1,NP1],P=0..1/2,color=[red,green,blue]); end if:
```

```
end do:
end do:
end do:
```

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 1, 1, 2

"Case1:Value is intersection of Not-Check and Full-Check"

"the value is", 0

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 1, 1, 3

"Case1:Value is intersection of Not-Check and Full-Check"

$$\text{"the value is"},\ -\frac{1}{4P+3}$$

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 1, 1, 4

"Case1:Value is intersection of Not-Check and Full-Check"

$$\text{"the value is"},\ -\frac{5P+4}{8P^2+17P+8}$$

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 1, 2, 2

"Case1:Value is intersection of Not-Check and Full-Check"

"the value is", 0

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 1, 2, 3

"Case1:Value is intersection of Not-Check and Full-Check"

"the value is", 0

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 1, 2, 4

"Case1:Value is intersection of Not-Check and Full-Check"

$$\text{"the value is"},\ -\frac{1}{8P^2+4P+3}$$

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 2, 1, 2

"Case1:Value is intersection of Not-Check and Full-Check"

"the value is", 0

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 2, 1, 3

"Case1:Value is intersection of Not-Check and Full-Check"

"the value is", 0

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 2, 1, 4

"Case1:Value is intersection of Not-Check and Full-Check"

$$\text{"the value is"},\ -\frac{1}{8P+7}$$

"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 2, 2, 2

"Case1:Value is intersection of Not-Check and Full-Check"
"the value is", 0
"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 2, 2, 3
"Case1:Value is intersection of Not-Check and Full-Check"
"the value is", 0
"&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&", 2, 2, 4
"Case1:Value is intersection of Not-Check and Full-Check"
"the value is", 0

## B.3 The value of game for $v(1,2,m,P)$ and $v(2,1,m,P)$ for all the values of $P$

| $m$ | $v(1,2,m,P),\ 0 < P < 1/2$ | *Type of the Case* |
|---|---|---|
| 2 | $\frac{(2P-1)^2}{4P-3}$ | Case 3 |
| 3 | $\frac{4P^2-4P+1}{3P-2}$ | Case 3 |
| 4 | $\frac{24P^2-29P+9}{23P-15}$ | Case 1 |
| 5 | $\frac{47P^2-67P+24}{P^2+55P-36}$ | Case 1 |
| 6 | $-\frac{2P^3+225P^2-377P+150}{2P^3-15P^2-317P+210}$ | Case 1 |

**Table B.2.1.** $v(1,2,m,P),\ 0 < P < 1/2.$

| $m$ | $v(1,2,m,P),\ 1/2 < P < 1$ | *Type of the Case* |
|---|---|---|

| | | |
|---|---|---|
| 2 | 0 | Case 1, 2 & 3 |
| 3 | 0 | Case 1, 2 & 3 |
| 4 | $-\frac{1}{8P^2+4P+3}$ | Case 1 |
| 5 | $-\frac{7P^2+4P+3}{16P^4+24P^3+31P^2+14P+6}$ | Case 1 |
| 6 | $-\frac{62P^4+101P^3+132P^2+63P+27}{64P^6+192P^5+398P^4+429P^3+348P^2+141P+45}$ | Case 1 |

**Table B.2.2.** $v(1,2,m,P)$, $1/2 < P < 1$.

| $m$ | $v(2,1,m,P)$, $0 < P < 1/2$ | *Type of the Case* |
|---|---|---|
| 2 | 0 | Case 1, 2 & 3 |
| 3 | $\frac{(2P-1)(4P^2-4P+1)}{12P^2-18P+7}$ | Case 3 |
| 4 | $-\frac{46P^4-118P^3+116P^2-52P+9}{2P^4-86P^3+182P^2-133P+33}$ | Case 1 |
| 5 | $\frac{348P^4-1006P^3+1115P^2-561P+108}{8P^5-20P^4+754P^3-1601P^2+1167P-288}$ | Case 1 |
| 6 | $-\frac{2P^6-4P^5+808P^4-2631P^3+3220P^2-1755P+360}{6P^6-48P^5+26P^4-1986P^3+4351P^2-3201P+792}$ | Case 1 |

**Table B.2.3.** $v(1,2,m,P)$, $0 < P < 1/2$.

| $m$ | $v(2,1,m,P)$, $1/2 < P < 1$ | *Type of the Case* |
|---|---|---|

| | | |
|---|---|---|
| 2 | 0 | Case 1, 2 & 3 |
| 3 | 0 | Case 1, 2 & 3 |
| 4 | $-\frac{1}{8P+7}$ | Case 1 |
| 5 | $-\frac{3(1+P)}{8P^2+20P+11}$ | Case 1 |
| 6 | $-\frac{23P^2+61P+36}{32P^3+153P^2+223P+96}$ | Case 1 |

**Table B.2.4.** $v(1,2,m,P)$, $0 < P < 1/2$.

# Chapter 4

# A Comparison between Nelder-Mead and Genetic Algorithm for Large Scale Optimization Problems

## 4.1 Introduction

Optimization problems can appear in a wide range of sciences where we are interested in finding the best solution to a problem after exploring all feasible solutions. In this study we aimed to compare the most popular direct numerical optimisation methods: the Nelder-Mead algorithm and the Genetic algorithm.

Numerical optimization methods have wide application in many optimization problems arisen in different field of science. Instead of focusing on exact solution of the problem, they provide an approximation of the solution with reasonable accuracy. Hence, they can cover a wider range of the complex problems. There are many different numerical methods which are compatible with different types of problems and their associated characteristics. However, numerical methods can broadly be divided into two main categories: gradient methods and direct methods.

Gradient methods are a form of numerical method which use the special features of the objective function such as continuity or gradient to establish an iterative method to approximate the solution. These methods are usually fast to converge. However, they are not compatible with irregular objective functions[1] and they may just provide the local optimum of the problem. One of the most famous examples of gradient method is Newton method which needs differentiation in each iteration. As a result, it can be computationally costly and also it is only applicable for continuous differentiable functions. As other example of a gradient

[1] A function is irregular, if its derivative is not well-defined or does not exist.

method we can mention Gauss-Newton method introduced by Harley (1961) and another algorithm introduced by Marquardt (1977).

In contrast, direct methods of optimization can be applied with a wider range of objective functions as they just require the value of the objective function. Convergence is almost surely slower compared to gradient methods in any form of the application, but overall there is no general criterion for the objective function. The Nelder-Mead simplex algorithm[2] (See Nelder and Mead (1965)) and the Genetic algorithm (see Holland (1975)) are two of the most widely used direct methods to deal with optimization problems.

The Nelder-Mead simplex method has been widely used in optimization problems. It is also an efficient tool for the optimization problems arising in Economics. Hugget, Ventura, Yaron (2011) employ the Nelder-Mead algorithm in a Macroeconomics problem when they are investigating sources of lifetime inequality.

Because of the important role of the algorithm in optimization problems it appears in many numerical methods handbooks like the one by Press, Flannery, Teukolsky and Vettering (1992), beside it is part of the MATLAB's optimization package. However, the method still does not have a satisfactory convergence theory. Not only is there a chance of approaching and sticking to a local optimum point instead of the global one, but also the algorithm may converge to a non-stationary point[3]. McKinnon (1998) provides a number of the examples which cause the algorithm to converge a non-stationary point.

The inefficiency of the method for higher dimension problems has been also observed and analysed in some studies. Byatt (2000) and Torczon (1989) provide some results for minimization of the function $f(x) = \sum_{i=1}^{n} x_i^2$ for different value of $n$. They observe in their numerical implementation of the method, that it is working inefficiently when $n$ is moderately large (approximately more than 32). Hans and Neumann (2006) provide some theoretical aspects for employing Nelder-Mead simplex algorithm for the mentioned function and they show that the

[2] Nelder-Mead simplex algorithm should not be mix with Dantzing simplex method (see Dantzing (1947)) which is an optimization method in linear programming.
[3] A point with non-zero gradient, which is obviously is not candidate to be a optimum point.

algorithm is inherently become inefficient by increasing the number of parameters. The mentioned works provide an example for the effect of the dimensionality[4] on the Nelder-Mead simplex algorithm, which means that we can expect the same problem for some other unconstraint optimization problems as well.

Genetic Algorithm is another direct optimization method which also has been widely used for optimization in many different fields like electromagnetics (see Weile and Michielssen (1997)), water distribution systems (van Zyl and Savic(2004)) and economic predictions (see Shin and Lee (2002)). It is also employed in economics and finance optimization problems. Pereira (2000) presents the Genetic algorithm as a strong tool in finance problems like choosing the optimized parameters for a specific trading rule. Another example is Chen and Chang (1995) which employ the Genetic algorithm to solve the economic dispatch problem in large scale systems.

This method starts from the set of initial solutions and iterates toward more optimized set of solutions using techniques inspired by natural evolution and Darwinian process[5]. Beside the flexibility to deal with many ranges of the optimization problem, Genetic algorithm can also be easily coded to solve the unconstraint optimization problems where the objective function is not necessarily differentiable.

Many studies show that Genetic algorithm is working well for big and complex problems; however it might be slow to achieve a very precise answer (see Yugeng, Tianyou and Weimin (1996)).

In this study, we briefly compare the Nelder-Mead and Genetic algorithms with respect to speed, accuracy and the resilience as the problem increases in size. For this matter, by Matlab implementation of both algorithms we perform experiments.

We observe that Nelder-Mead algorithm with regards to the number of iterations required and processing time is efficient only when the number of parameters is small. We provide an example which shows that, regardless of the number of parameters, Nelder-Mead algorithm can stick

[4] The effect of large number of parameters on efficiency of the method is called dimensionality.
[5] Full process of Genetic algorithm and its operators are provided in Section 4.3.

to a local optimum point based on the position of the starting point. Genetic algorithm shows faster approaching to the global optimum point and also can skip the local optimum points by mutation functions.

It is obvious that changing parameters or employing hybrid methods may improve the final results for some specific problems; however we choose to use the default parameters within Matlab for a pure comparison.

This study has been organized in four sections. In Section 4.2 and 4.3 we briefly review the Nelder-Mead and Genetic algorithms and their special characteristics. In Section 4.4, we introduce some experiments to compare both methods with regarding to accuracy and required time. In Section 4.4, we provide an experiment to see dependency of Nelder-Mead algorithm on the place of the initial guess. We also show this problem can be skipped by Genetic Algorithm.

The Matlab version used to perform the experiments in this study is R2013b, and as mentioned, implements the default parameters for Genetic and Nelder-Mead algorithm otherwise specified[6].

## 4.2 Nelder-Mead Simplex Algorithm[7]

In this section we explain the Nelder-Mead simplex algorithm and its main characteristics.

The Nelder-Mead simplex algorithm is based on iteratively constructing sequence of simplices, where the amount of the objective function for the vertices of the simplex evaluated and sorted in each iteration. The algorithm applies four possible operators of reflection, expansion, contraction and shrinks to construct a new simplex which has improved evaluated functions on its vertices in general. The algorithm terminates when the vertices of the simplex meet the stopping criteria.

For minimizing a real valued function like $f: \mathcal{R}^N \rightarrow \mathcal{R}; \quad f(X) = y$, Nelder-Mead algorithm is starting with a simplex defined in $N + 1$ dimension. The algorithm also can start with just one initial point, where we define a

[6] Matlab's default settings for Nelder-Mead algorithm and Genetic algorithm have fully provided and discussed in section 4.2.2 and 4.3.2.

[7] The algorithm is also called Nelder-Mead algorithm.

function which can construct a simplex in N+1 dimension based on that point. Each vertex of the simplex is defined by function values at a different point. Having ranked the vertices of the simplex, we aim to move the worst performing vertex of the simplex around the centroid to update the simplex. In updating procedure four operators of Reflection, Expansion, Contraction, and Shrink are used. If we define the centroid of the simplex as average of all the vertices of the simplex to expect the worst one, then Nelder-Mead operators are as follow.

a) Reflection: reflect the worst vertex around the centroid. The function for finding the reflected point ($x_r$) is $x_r = c + \alpha(c - x_h)$ where $\alpha > 0$. Figure 4.1 shows the reflection function for a 2-dimensional simplex.

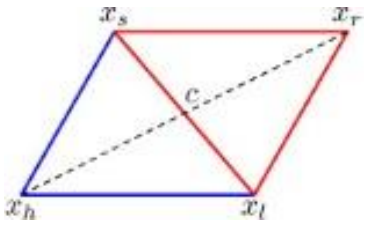



**Figure 4. 1.** Reflection operator in Nelder-Mead algorithm, where $\boldsymbol{f(x_s) < f(x_h) < f(x_l)}$ and $\boldsymbol{c}$ is the centroid.

b) The contraction operator makes the simplex smaller by making the worst vertex closer to the centroid. There can be two forms of contraction, contraction inside and contraction outside. The contracted point can be calculated by $x_c = c + \beta(x_r - c)$ for outside contraction or $x_c = c + \beta(x_h - c)$ for inside contraction where $x_r$ is the reflection of $x_h$ and $0 < \beta < 1$. Figure 4.2 shows the contraction function in a 2-dimentional simplex.


**Figure 4. 2.** Contraction operator in Nelder-Mead algorithm, where $\boldsymbol{f(x_s) < f(x_h) < f(x_l)}$ and $\boldsymbol{c}$ is the centroid. The left image shows outside contraction and right image shows inside contraction.

c) Expansion operator, expand the simplex by multiply the distance of the worst vertex and the centroid by a coefficient greater than $\alpha$. The function for finding the expanded point ($x_e$) is $x_e = c + \gamma(x_r -$

$c$) where $x_r$ is the reflected point of the worst vertex and $\gamma > 1$, $\gamma > \alpha$. Figure 4.3 shows the expansion function for a 2-dimensional simplex.

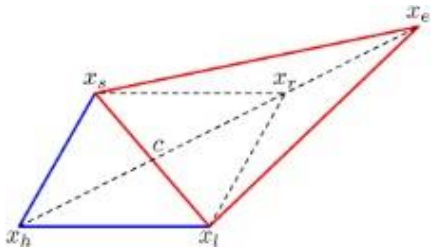



**Figure 4. 3.** Expansion operator in Nelder-Mead algorithm, where $\boldsymbol{f(x_s) < f(x_h) < f(x_l)}$ and $\boldsymbol{c}$ is the centroid.

d) Shrink operator, for a N+1 dimensional simplex calculates the N new vertices where all of them are closer to the best vertex. The location of the best vertex is kept fixed. The new vertices are calculated by $x_j = x_l + \delta(x_j - x_l)$. Figure 4.4 shows the Shrink function for a 2-dimensional simplex.

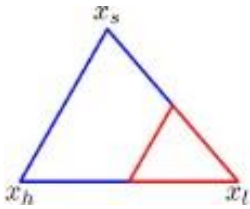



**Figure 4. 4.** Shrink operator in Nelder-Mead algorithm, where $\boldsymbol{f(x_s) < f(x_h) < f(x_l)}$.

$\alpha, \beta, \gamma$ and $\delta$ can have different values, however in standard Nelder-Mead algorithm it always suppose that $\alpha = 1, \beta = \frac{1}{2}$ , $\gamma = 2$ and $= \frac{1}{2}$.

The steps of the Nelder-Mead algorithm for minimization of a real value function with N variables are as follows:

**Step 1**: The algorithm starts with the first simplex which has N+1 vertex. Nelder-Mead algorithm can also start with just one start point. We can define a rule that after receiving the initial guess it will make the other vertices of the algorithm. Suppose that $X_0, X_1, \dots, X_n$ are the vertices of the first simplex.

**Step 2:** The vertices of the simplex are sorted in increasing order. Hence we have $f(X_0) < f(X_1) < \cdots < f(X_n)$. We suppose that $X_u$ is the best vertex, $X_v$ is the second worst vertex and $X_w$ is the worst vertex. By employing all the vertices except the worst one, we calculate the Centroid point $(C)$.

**Step 3:** In general, this step updates the worst vertex of the simplex or it will shrink the whole simplex. The algorithm first tries the Reflection operator. The worst vertex is reflected around the centroid, and value of the reflected vertex is evaluated. If we show the reflected point by $X_R$, then

a) If $X_R$ is better than the best vertex ($f(X_R) < f(X_u)$), then the simplex experience expansion. If the expanded point ($X_s$) was better than reflected point, then we accept $X_s$ and the algorithm goes to Step 2. Otherwise, we accept $X_R$.
b) If $X_R$ is just better than the second worst vertex ($f(X_u) \leq f(X_R) < f(X_v)$), then the worst vertex replaced with $X_R$, and the algorithm goes to Step 2;

Otherwise, the algorithm runs Contraction in following way. We show the inside contracted point by $X_{C1}$, and outside contracted point by $X_{C2}$.

a) If $f(X_R) < f(X_w)$ then algorithm calculated outside contracted point ($X_{C2}$), if $f(X_{C2}) < f(X_R)$ then algorithm accept $X_{C2}$ and goes to Step 2. Otherwise, the simplex shrinks and the algorithm goes to Step 2.
b) If $f(X_R) \geq f(X_w)$ then algorithm calculates inside contracted point ($X_{C1}$), if $f(X_{C1}) < f(X_R)$ then algorithm accept $X_{C1}$ and goes to Step 2. Otherwise, the simplex shrinks and the algorithm goes to Step 2.

The algorithm is repeated until the time, it gets to one of the stopping criteria. The most common stopping criteria are number of Iterations, number of function evaluations, X tolerance or function tolerance.

Example 4.1 briefly explains the steps of the Nelder-Mead algorithm for minimization of $f(x) = |x_1| + |x_2|$, where the starting point is $[1,1]$.
**Example 4. 1.** In this example we follow few iterations of the Nelder-Mead algorithm to find minimizer of $f(x) = |x_1| + |x_2|$, where the starting point is $[1,1]$. The coefficients of the Nelder-Mead functions supposed to be the standard one.

As the initial point is $[1,1]$, by assigning a function to $[1,1]$ we make two other vertices. In this example we suppose that the other vertices are

$[3,1]$ and $[1,2]$. Hence, we can sort the vertices as $X_3^{(0)} = [1,3], X_2^{(0)} = [2,1]$ and $X_1^{(0)} = [1,1]$ where $f\left(X_1^{(0)}\right) < f\left(X_2^{(0)}\right) < f(X_3^{(0)})$. The centroid of the simplex is $C = [1,1.5]$. The reflection of the worst vertex is $X_R^{(0)} = [-1,3]$, which is worse than the worst vertex so algorithm runs contract inside witch is [2, 1.5] and better than the worst vertex. Hence new simplex is $X_3^{(1)} = [2,1.5], X_2^{(1)} = [2, 1]$ and $X_1^{(1)} = [1,1]$.

Figure 4.3 shows the some of the first sequential simplices that Nelder-Mead algorithm makes to find the minimizer of the objective function.

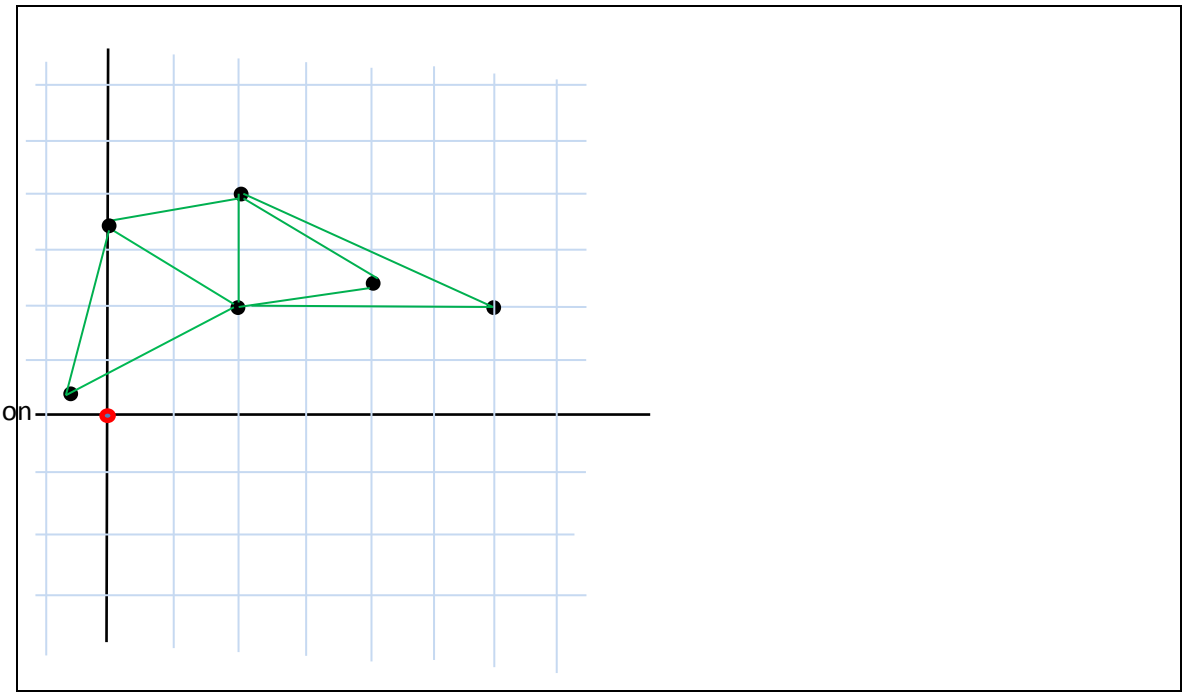



**Figure 4. 1.** Three first sequential simplices that Nelder-Mead algorithm makes to find the minimizer of $\boldsymbol{f(x) = |x_1| + |x_2|}$.

### 4.2.1 Convergence of the Nelder-Mead algorithm

Despite the wide application of the Nelder-Mead algorithm in optimization problems, still there is lack of a global convergent theorem. The method is dependent on position of the initial guess. It means that based on the position of the initial guess the algorithm may approaches a local optimum. There is no way to be sure that the method will show the global optimizer of the problem. Adding to the problem of the position of the initial guess, in general there is no guarantee the method converge.

Lagarias, Reeds, Wright and Wright (1998) investigate the convergent properties of Nelder-Mead algorithm for the one dimensional functions. They show that the method is convergent for the one dimensional strictly

convex functions as long as the expansion function in the Nelder-Mead method is genuine. McKinnon (1998) also proves that the method is convergent for functions with more than three continuous derivations.

However, he also provide couple of examples where the Nelder-Mead algorithm produce simplices elongate of each other where the best vertex is not changing. In fact, he shows that for functions like

$$f(x,y) = \begin{cases} \theta\phi|x|^{\tau} + y + y^2; x \leq 0 \\ \theta|x|^{\tau} + y + y^2; x \geq 0 \end{cases} \qquad (4.1.1)$$

Nelder-Mead algorithm repeatedly runs inside contraction while the best vertex is not changing. This situation is shown in Figure 4.2.

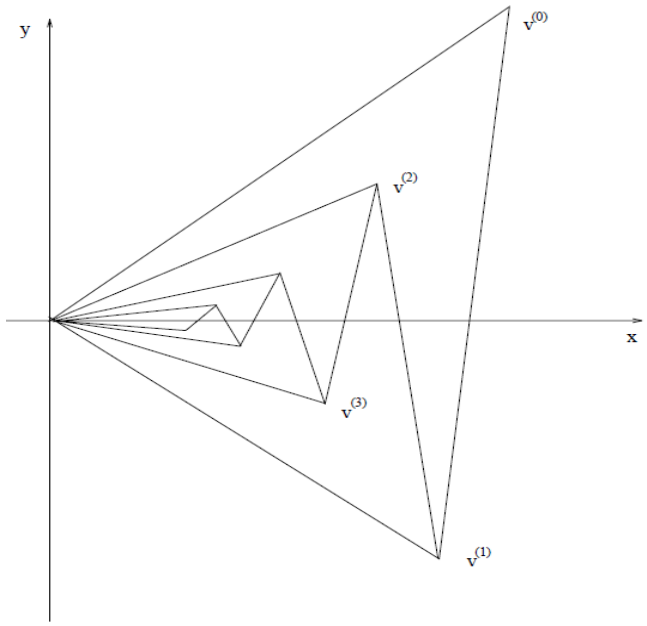



**Figure 4. 2.** Sequential simplices made by Nelder-Mead algorithm to solve function 4.1.1, where $\boldsymbol{\tau = 2, \theta = 6}$ and $\boldsymbol{\phi = 60}$ (see McKinnon (1998)).

### 4.2.2. Matlab implantation of Nelder-Mead algorithm[8]

The Nelder-Mead algorithm is widely used in optimization problems, so it is part of the optimization package of many mathematical programing softwares like Matlab and Mathematica.

In Matlab, the method accessible via optimization app and also can be run by `FMINSEARCH` command, on the command line. The initial guess should be

[8] The Matlab version used in this study is R2013b. For reviewing the optimization settings, Matlab's users guide is employed (see Ljung (1995)).

input by the user, then the N+1 vertices for producing the initial simplex is made by adding 0.05 to each component of the initial guess. The coefficients of the Nelder-Mead algorithm are equivalent to the standard Nelder-Mead algorithm. The default stopping criteria are maximum iterations of less than 200×number of the variables, maximum function evaluations to be less than 200×number of the variables, and also both X tolerance and function tolerance to be less than $10^{-4}$. X tolerance specifies the termination tolerance for X. Function tolerance specifies the termination tolerance for the objective function value. As an example, if the best function value of the previous iteration is 1.10001 and current best function value is 1.00000 the algorithm will stop as $|1.10001 - 1| < 0.0001$.

Table 4.1, shows the Matlab implementation of Nelder-Mead algorithm for finding the minimum value of $f(x) = |x_1| + |x_2|$. The starting point is set to be $x = [1,1]$, and stopping criteria is Max Iterations less than 10. For the rest of the settings, Matlab's default parameters were employed.

| Iteration | Func-count | min f(x) | Procedure |
|---|---|---|---|
| 0 | 1 | 2 | |
| 1 | 3 | 2 | initial simplex |
| 2 | 5 | 1.975 | expand |
| 3 | 7 | 1.8625 | expand |
| 4 | 9 | 1.75625 | expand |
| 5 | 11 | 1.47812 | expand |
| 6 | 13 | 1.37187 | reflect |
| 7 | 15 | 1.14375 | reflect |
| 8 | 17 | 1.14375 | contract inside |
| 9 | 19 | 1.01445 | expand |
| 10 | 21 | 1.01445 | contract outside |

Exiting: Maximum number of iterations has been exceeded
- increase MaxIter option.
Current function value: 1.014453

**Table 4. 3.** Matlab implementation of the Nelder-Mead algorithm to find the minimum value of $f(X) = |x_1| + |x_2|$. The start point is $X = (1,1)$, the stopping is Maximum number of Iterations less than 10, with the remaining settings being Matlab's default for Nelder-Mead algorithm. The minimum value of the objective function is shown in each iteration of the algorithm.

## 4.3 Genetic Algorithm and Mathematical Optimization

### 4.3.1 History of the Genetic algorithm

The Genetic algorithm is an optimization method which has been inspired by evolution in nature and Darwinian process. The initial idea of

employing evolution to find optimised solutions appears in the works of Rechenberg (1965) and Schwefel (1975, 1977). By employing evolution they optimize some real valued parameters which had applications in designing devices such as airplane wings. Later, some others scientists like Fogel, Owens and Walsh (1966) developed evolutionary programming for optimizing a way of allocating candidates to some tasks, by repeatedly changing in the allocation.

Genetic algorithm, as the famous optimization method which nowadays we know, was first invented by Holland (1960). He initially was looking for a method not to find the optimized solution of a specific problem, but a method compatible to the wide range of problems. In his method, the set of initial solutions (initial population) is evolving to the next set of solutions (next generation) by some functions similar to the "natural selection" in Darwinian theory. The algorithm repeatedly produces solutions (one generation after another) and will finish when it gets to some stopping criteria.

### 4.3.2 Genetic algorithm for optimization of the real valued functions

In general, Genetic algorithm first designates a fitness function to a problem. A fitness function estimates how much a solution is close to the optimum one. In the next step, the algorithm starts with a set of feasible solutions called "first generation". Then, three operators of Selection of the fittest, Crossover and Mutation are employed to make another set of solutions. Selection chooses the most fitted solutions in the first generation and by duplicating them gives them greater chance to produce offspring. Crossover produces the next set of solution by fitted solutions provided in the previous step. Later, the Mutation function randomly chooses one or more of the elements of the current generation and makes a random change in which is then present in the next generation. The algorithm repeatedly makes one generation after another. Each generation is in general more fitted.

Genetic algorithm's operators (Selection, Crossover, Mutation) can be applied on a population in different ways. The famous methods of Selection are Roulette wheel, tournament and uniform. The famous crossover methods are single point crossover, two point crossover and

scattered crossover. The famous way of mutation is Uniform. All these operators have been fully explained and discussed in Appendix C.

The Genetic algorithm can be employed for a wide range of the problems with different types of solution sets. In this study we focus on the application of Genetic algorithm to find the optimized solution of real valued functions. For minimizing a real valued function, first we set the suitable fitness function for a program. Besides, following constants are set to be fixed through the all the iterations:

Number of individuals in each chromosome, Length of each chromosome, number (or percentage) of best fitted individuals who are going to survive in each iteration, selection and crossover methods and mutation rate.

To optimize a real valued function like $f(X)$, Genetic algorithm runs following steps.

**Step 1**: Randomly generate a set of $N$ feasible solutions which will be the first population. Each number in the set is called an individual (Chromosome). To run the operators on these numbers, Genetic algorithm changes them to the binary form. It means that each individual in the population will be a sequence of 0 and 1. Each 0 or 1 is called a gene.

**Step 2:** Calculate the fitness of each individual in the population, based on the fitness function.

**Step 3:** Based on the fitness scaling, the fittest individuals survive and the worst ones are deleted from the population. Best individuals reproduce and have a greater chance to produce offspring.

**Step 4:** Crossover operator makes a new generation, which should be more fitted in general.

**Step 5:** Based on the probability of mutation, some percentage of the genes are randomly mutated from 1 to 0, or 0 to 1.

**Step 6:** Replace the new population with the previous one.

**Step 7:** If the stopping criteria has not been met, go to Step 2.

To illustrate, Example 4.2 applies genetic algorithm to find the minimum of $f(x) = x^2$. The first iteration of the method is fully explained, where

the number of individuals is 5, length of each chromosome is 9, individuals have 80 percent chance of survival, selection method is Roulette wheel, crossover method is single point and the probability of mutation is 0.03.

**Example 4. 2.**

**Step 1:** To find the minimum of $\boldsymbol{f(x) = x^2}$, $\boldsymbol{ft(x) = \left(\frac{1}{1+x^2}\right)^2}$ can be a suitable scaling function. Genetic algorithm first produces 5 randomly numbers which can be -14.5, -10.7, -3.13, 3.5, 4.7. We consider following binary format for the individuals in each generation.

-14.5 → [011101000],

-10.7→[010100111],

-3.13→ $[000111101]$,

3.5→[101000111],

4.7→[101000111].

In this form of binary format, the first digit shows the sign of the number; the next four digits show the integer part of the number, and last four digit show the fraction part of the number.

**Step 2:** Fitness of each individual is calculated. Hence, we will have

$ft$ (-14.5)=$0.000022$, $ft$ (-10.7)=$0.000074$, $ft$ (-3.13)=$0.008578$, $ft$ (3.5)=0.005695, $ft$ (4.7)=$0.001875$.

**Step 3**: As there is 80 percent chance of survival, all the individuals pass to the next generation. Roulette wheel operator chooses 5 random numbers between 0 and 5 ($R_i$), also it calculates $P_i = \frac{ft(x_i)}{\sum_{i=1}^{n} ft(x_i)}$ , $i = 1..5$. Table 4.5 shows all the $P_i$ and cumulative probability $C_i$ ($C_i = \sum_{j=1}^{i} P_j$) and $R_i$.

| $i$ | $P_i$ | $C_i$ | $R_i$ |
|---|---|---|---|
| 1 | 0.001293108453 | 0.001293108453 | 0.041 |
| 2 | 0.004326529042 | 0.005619637495 | 0.003 |
| 3 | 0.4950285284 | 0.5006481659 | 0.014 |
| 4 | 0.3286976729 | 0.8293458388 | 0.688 |
| 5 | 0.1706541617 | 1 | 0.328 |

**Table 4. 4.** Roulette wheel method and probability of selection of individuals.

For $j\epsilon N;\ 0 \leq j \leq 4$ and $C_0 = 0$; if $C_j < R_i < C_{j+1}$ then $x_{i+1}$ will be kept in the population. Hence, the population will be updates to -3.13, -10.7, -3.13, 3.5, -3.13. As we can see, the algorithm gives more chances to -3.13 to produce offspring.

**Step 4:** Crossover operator produces a new population by crossover between -3.13 and -10.7; -10.7 and -3.13; -3.13 and 3.5; 3.5 and -3.13. For single point crossover, first a number between 1 to 8 is randomly selected (crossover point). Secondly, the digits of the two numbers are swapped based on the crossover point. Figure 4.7 illustrates the single crossover between -3.13 and -10.7, if the cross over point is 4.

$$\begin{matrix}[0001|11101] \\ [0101|00111]\end{matrix} \Longrightarrow [000100111] = \text{-}4.7$$

**Figure 4. 5.** Single crossover between -3.13 and -10.7, when the crossover point is 4.

Hence, if we calculate all the other crossovers then the new generation is

-4.7, 11.5, 2.8, -3.5, 1.6.

**Step 5:** There are 5 chromosomes in the population each have 9 genes, as the probability of mutation is 0.03, 1 gene ($[3 \times 5 \times 9/100]$[9]) from one of the individuals is randomly selected to mutate. Figure 4.5 illustrates a possible mutation on -4.7.

$$-4.7 = [000100111] \Longrightarrow [100100111] = 4.7$$

**Figure 4. 6.** A mutation on -4.7.

**Step 6**: The new generation is 4.7, 11.5, 2.8, -3.5, and 1.6. This new population is replaced with previous one.

Step 2 to 6 will be repeated until the time we get to one of the stopping criteria. The most common stopping criteria are number of generations, number of function evaluations, function tolerance[10] and time limit.

[9] The [ ] shows the integer part of a number.
[10] Function tolerance has a same definition as the one in Nelder-Mead algorithm, however here the average fitness value of the previous generation is compared to the current average fitness value.

In each iteration of the Genetic algorithm a new set of individuals is produced, the new set has a better fitness in general. This does not mean that all the new individuals are better than the previous generation. There might be some individuals with less fitness compared to individuals in previous generations. This is mostly because of the mutation operator which may produce a not fitted solution. We show in Section 4.3.1 that how much mutation operator is important part for Genetic algorithm. In fact, the efficiency of this method to explore the wider area of feasible solution is coming from mutation.

### 4.3.3 Mutation function in Genetic algorithm

In this section we investigate some special features of the Genetic algorithm, which make it different from the other algorithms. The most different and important operator in Genetic algorithm is Mutation. By Mutation we intentionally may produce some less fitted individuals, however in general this operator has important role in efficiency and accuracy of the Genetic algorithm. We run the following experiment to show how mutation function increases the efficiency of the Genetic algorithm.

In this example, the objective function is $f(x) = |x_1| + |x_2|$, where we want to estimate its minimum. The `"ga"` command in Matlab[11] is applied where the stopping criteria is the number of generations (to be at most 10). The defaults within Matlab are accepted for other settings. Table 2.2 shows the results where the mutation function is Uniform with the rate of 0.01 in each generation. In contrast, Table 4.6 shows the result when there is no mutation. Both tables show the minimum value of the function in each generation. In Table 4.6 (Genetic algorithm with mutation rate of 0.01), the Genetic algorithm is approaching the optimum value (which is zero). For some generations the best value of the objective function is not changing but the average best value from each generation is improving, which means that the algorithm has not stuck and is in progress.

As we see in Table 4.7 (Genetic algorithm without Mutation), after a few Iterations of the algorithm, none of the best value of the objective function and average value are changing. This means that the algorithm has stuck on a non-optimizer point, and cannot skip it.

[11] In Section 4.3.2 Matlab implantation of Genetic algorithm is fully explained.

| Generations | f-count | Best f(x) | Mean f(x) | Stall Generations |
|---|---|---|---|---|
| 1 | 40 | 0.1082 | 1.92 | 0 |
| 2 | 60 | 0.1082 | 1.841 | 1 |
| 3 | 80 | 0.1082 | 1.484 | 2 |
| 4 | 100 | 0.1082 | 1.25 | 3 |
| 5 | 120 | 0.1082 | 0.6499 | 4 |
| 6 | 140 | 0.06233 | 0.3478 | 0 |
| 7 | 160 | 0.06233 | 0.2185 | 1 |
| 8 | 180 | 0.06233 | 0.2496 | 2 |
| 9 | 200 | 0.06233 | 0.1954 | 3 |
| 10 | 220 | 0.06233 | 0.1746 | 4 |

Optimization terminated: maximum number of generations exceeded.

**Table 4. 7.** Genetic algorithm implementation to find the minimum value of $\boldsymbol{f(x) = |x_1| + |x_2|}$. The mutation function has been set to be Uniform with 0.01 rate and the stopping criteria is Max Iteration less than 10. For the rest of the settings, Matlab default settings are accepted.

| Generation | f-count | Best f(x) | Mean f(x) | Stall Generations |
|---|---|---|---|---|
| 1 | 40 | 0.08831 | 0.7234 | 0 |
| 2 | 60 | 0.08831 | 0.4706 | 1 |
| 3 | 80 | 0.08831 | 0.2716 | 2 |
| 4 | 100 | 0.08831 | 0.1759 | 3 |
| 5 | 120 | 0.08831 | 0.1016 | 4 |
| 6 | 140 | 0.08831 | 0.09862 | 5 |
| 7 | 160 | 0.08831 | 0.09862 | 6 |
| 8 | 180 | 0.08831 | 0.09862 | 7 |
| 9 | 200 | 0.08831 | 0.09862 | 8 |
| 10 | 220 | 0.08831 | 0.09862 | 9 |

Optimization terminated: maximum number of generations exceeded.

**Table 4. 8.** Genetic algorithm implementation to find the minimum value of $f(x) = |x_1| + |x_2|$, where mutation function is not applicable. The stopping criteria is Maximum number of Iterations to be less than 10. For the rest of settings, Matlab default settings are accepted.

### 4.3.4 Matlab Implementation of Genetic algorithm[12]

In this section we explain how Genetic algorithm can be run via Matlab's command line and packages.

Genetic algorithm is part of the optimization package of Matlab, besides it can be run via `"ga"` command in command line.

These are Matlab's defaults regarding the population, crossover, mutation, stopping criteria and etc. for an unconstraint optimization problem.

[12] The Matlab version used in this study is R2013b. For reviewing the optimization settings, Matlab's users guide is employed (see Ljung (1995)).

The populations have 20 individuals; besides the initial population are individuals like $(x_1, x_2, \dots, x_n)$ which are selected uniformly and each $x_i$ is a number between 0 and 1.

2 is the number of the individuals which is guaranteed will survive and the reproduction operator is Roulette wheel. The crossover is scattered. Mutation rate is 0.01. Finally, the stopping criteria are as follows: Number of generations to be at most 100, infinity time limit, unbounded fitness limit, stall generations[13] to be at most 50, stall time limit infinity and function tolerance to be at most $10^{-6}$.

## 4.4 Comparison of Nelder-Mead algorithm and Genetic Algorithm

Both Genetic algorithm and Nelder-Mead algorithm are considered as direct optimization methods. However, as we briefly explained the nature of both algorithms is quite different. In this section we aim to provide some experiments by Matlab to show how this different nature can affect the efficiency of the method in different ways, especially when the size of the problem is growing. Figure 3.1 and 3.2 provides a general overview of how both algorithms approach the optimized solution. They show the results of the first 20 iterations of both algorithms for minimization of $f(X) = |x_1| + |x_2|$. For Nelder-Mead algorithm the initial point is $X = (1,1)$ and for Genetic algorithm all the first generation elements are considered to be $1$. As we see, Nelder-Mead algorithm gradually approaches to the optimized solution, but the Genetic algorithm quickly gets close to the solution. However, later it fluctuates around the solution (point [0,0]).

[13] When the weighted average change in the fitness function value over all the Stall generations is less than Function tolerance, the algorithm terminates.

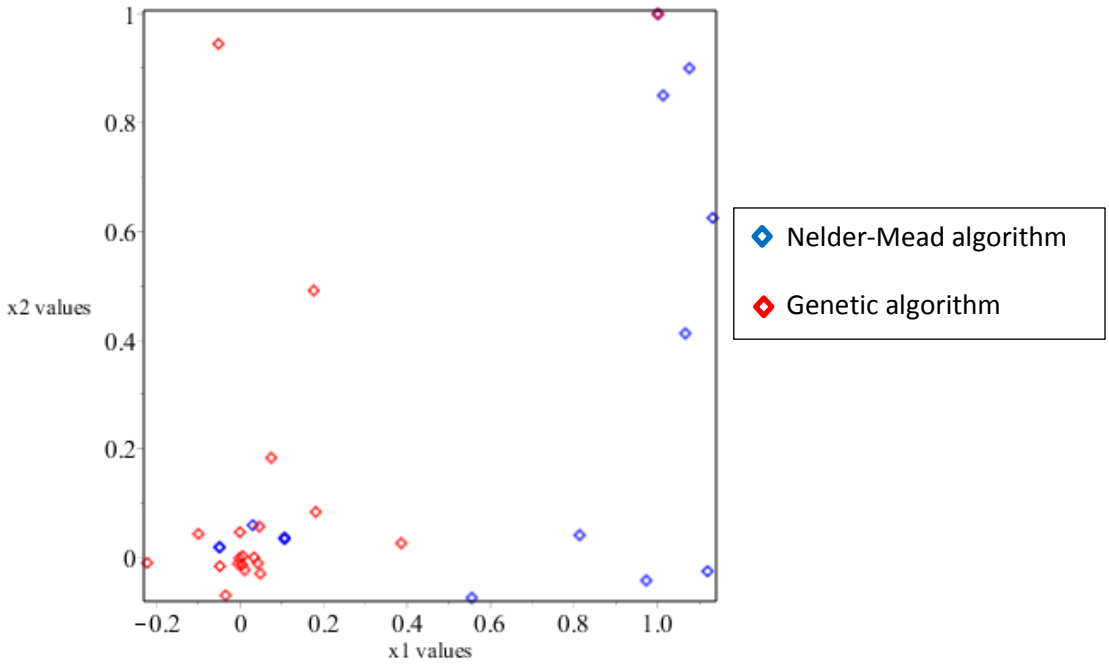



**Figure 3.1.** The results of 20 first iterations for Nelder-Mead and Genetic algorithm for minimization of $f(X) = |x_1| + |x_2|$. The initial point for NM is $X = (1,1)$ and all the elements of the first generation for GA are $1$. Rest of the settings for both algorithms are Matlab's default.

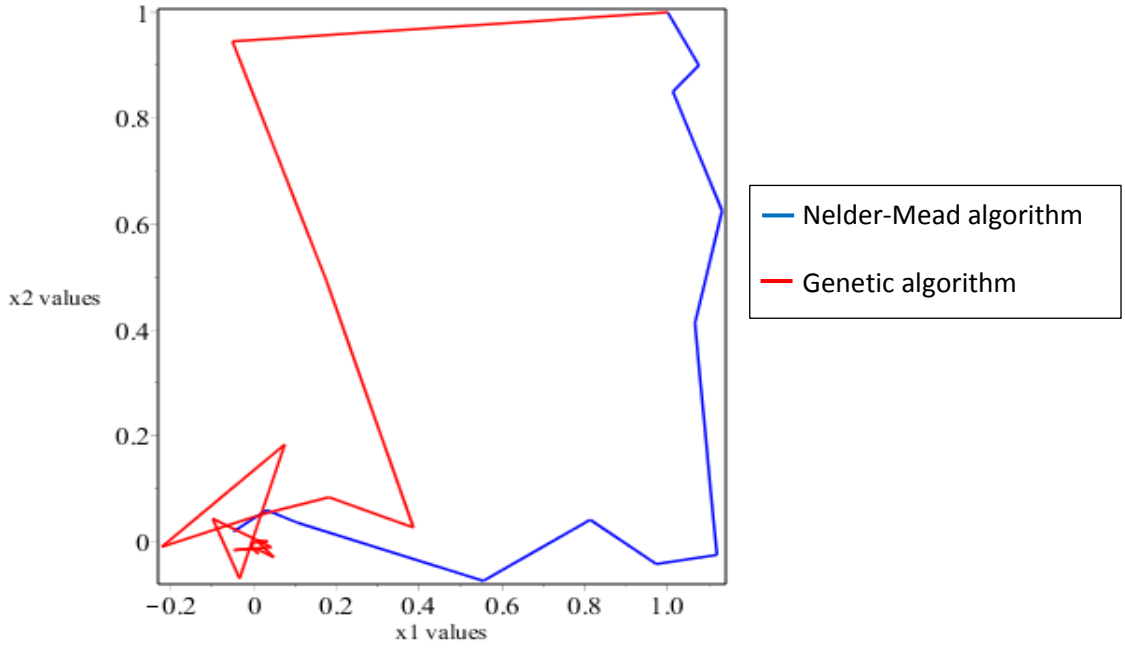



**Figure 3.2.** The results of 20 first iterations for Nelder-Mead and Genetic algorithm for minimization of $f(X) = |x_1| + |x_2|$. The initial point for NM is $X = (1,1)$ and all the elements of the first generation for GA are $[1]_{n\times n}$. Rest of the settings for both algorithms are Matlab's default.

In the next sections we design some Matlab experiments to compare the algorithm regarding the speed and accuracy when the size of algorithm is growing, and resilience to the initial guess.

### 4.4.1 Speed and Accuracy

In this section, we run an experiment on some test functions to compare the speed and accuracy of Nelder-Mead algorithm and Genetic algorithm when the size of the problem is growing. The algorithms are also compared when we aim to catch a very precise solution. The experiment will be as follows.

**Experiment A**

a) For Nelder-Mead algorithm the initial point is set to be $X = [1]_{n\times n}$ , where $n$ is the size of the problem. For rest of the settings, Matlab's default are accepted.
b) For Genetic algorithm all the elements of the first generation are set to be $[1]_{n\times n}$, for rest of the settings Matlab's default are accepted.
c) Both algorithms employed to minimize the real value function $f(X) = y, R^n \to R$.
d) Both algorithms will stop when the best function value gets to $10^{-2}$ and $10^{-4}$.
e) The results regarding to the required time and accuracy will be observed.

In Example 4.3 and Example 4.4, the experiment is employed on the functions $\sum_{i=1}^{n} x_i^2$ and $\sum_{i=1}^{n} x_i^4$ when $n$ is increasing. We can observe that Nelder-Mead algorithm shows much more sensitivity to the number of the parameters. For more than around 200 variables, the required time for Nelder-Mead algorithm is growing exponentially and it is much more than the Genetic algorithm.

In Example 4.5, the experiment is employed on the function $\sum_{i=1}^{n}|x_i|$. We can observe that by increasing $n$, the accuracy of the approximations by Nelder-Mead algorithm is questionable as it may stick to a non-stationary point. While, for any number of the parameters Genetic algorithm approaches to the optimized solution in reasonable amount of time.

In Example 4.6, the experiment is employed on $\sum_{i=1}^{n} x_i^2$ , $\sum_{i=1}^{n} x_i^4$ and $\sum_{i=1}^{n} |x_i|$ while we want to get a precise answer like $10^{-6}$. We observe that as the size of the problem is growing, both algorithms may face different difficulties. While Genetic algorithm is getting too slow, Nelder-Mead algorithm may stick on a non-stationary point. When the algorithm needs more than 12 hours to get to the answer, we stop it. We mention this situation with "more than 12 hours" in the table.

**Example 4. 3.** In this experiment the objective function is $\sum_{i=1}^{n} x_i^2$.

| $n$ | The ratio of required time for NM and GA to get $10^{-2}$ (NM/GA) | The ratio of required time for NM and GA to get $10^{-4}$ (NM/GA) |
|---|---|---|
| 2 | 0.26 | 0.1104 |
| 5 | 0.12 | 0.0791 |
| 10 | 0.14 | 0.1696 |
| 20 | 0.6071 | 0.1016 |
| 40 | 0.3606 | 0.0278 |
| 60 | 0.2083 | 0.1490 |
| 80 | 0.1539 | 0.1662 |
| 100 | 0.1969 | 0.2649 |
| **200** | **31.3008** | **12.3427** |
| 300 | 1.5214 | 2.4187 |
| 400 | 1.8668 | 17.44 |

**Table 4. 9.** Experiment A is applied on $\sum_{i=1}^{n} x_i^2$. As we see when the number of parameter in objective function is more than 200 the required time for Nelder-Mead algorithm is much more than Genetic algorithm.

**Example 4. 4.** In this experiment the objective function is $\sum_{i=1}^{n} x_i^4$.

| $n$ | The ratio of required time for NM and GA to get $10^{-2}$ (NM/GA) | The ratio of required time for NM and GA to get $10^{-4}$ (NM/GA) |
|---|---|---|
| 2 | 0.2661 | 0.05449 |
| 5 | 0.2051 | 0.0215 |
| 10 | 0.1636 | 0.0758 |
| 20 | 2.3894 | 2.8241 |
| 40 | 0.8215 | 32.3044 |
| 60 | 0.3607 | 0.3835 |
| 80 | 0.7691 | 0.4408 |
| 100 | 0.0910 | 0.2917 |
| **200** | **31.9097** | **10.8511** |
| 300 | 45.6721 | 15.7904 |
| 400 | 70.4575 | 53.8360 |

**Table 4. 10.** Experiment A is applied on $\sum_{i=1}^{n} x_i^4$. As we see when the number of parameter in objective function is more than 200 the required time for Nelder-Mead algorithm is much more than Genetic algorithm.

**Example 4. 5.** In this experiment the objective function is $\sum_{i=1}^{n}|x_i|$.

| | Required time to get $10^{-2}$ | | Required time to get $10^{-4}$ | |
|---|---|---|---|---|
| $n$ | NM | GA | NM | GA |
| 2 | 0.019254 | 0.326261 | 0.039635 | 0.259384 |
| **5** | **Sticks on 0.047** | 3.686796 | **Sticks on 0.047** | 36.998387 |
| 10 | Sticks on 5.521 | 14.558425 | Sticks on 5.521 | 91.690288 |
| 20 | Sticks on 14.2314 | 50.627152 | Sticks on 14.2314 | 433.756074 |
| 40 | Sticks on 35.6911 | 1058.645011 | Sticks on 35.6911 | 8788.1212 |
| 60 | Sticks on 53.3124 | 3656.672690 | Sticks on 53.3124 | 11393.1479 |

**Table 4. 11.** Experiment A is applied on $\sum_{i=1}^{n}|x_i|$. As we see when number of parameters are more than 5 the Nelder-Mead algorithm sticks at some non-optimized point and it cannot go further, while Genetic algorithm approaches to the optimized solution in a reasonable amount of time.

**Example 4. 6.**

| $n$ | Required time to get $10^{-6}$ for $\sum_{i=1}^{n}x_i^2$ | | Required time to get $10^{-6}$ for $\sum_{i=1}^{n}x_i^4$ | | Required time to get $10^{-6}$ for $\sum_{i=1}^{n}\|x_i\|$ | |
|---|---|---|---|---|---|---|
| | NM | GA | NM | GA | NM | GA |
| 2 | 0.0257 | 2.5617 | 0.0356 | 0.8056 | Sticks on 0.047 | 4863.2788 |
| 5 | 0.0682 | 14.7180 | 0.0788 | 6.7447 | Sticks on 5.521 | More than 12 hours |
| 10 | 0.2794 | 38.8972 | 0.2932 | 81.3114 | Sticks on 14.2314 | More than 12 hours |
| 20 | 2.0646 | 473.2614 | 26.0608 | 11.8491 | Sticks on 35.6911 | More than 12 hours |
| 40 | Sticks on 1.284e-04 | 1943.6764 | 807.5236 | 28.7380 | Sticks on 53.3124 | More than 12 hours |
| 60 | 3190.1431 | 7021.4342 | 12.2927 | 52.9163 | Sticks on 72.5001 | More than 12 hours |
| 80 | 5242.4003 | 16169.5863 | 7.7829 | 45.4004 | Sticks on 105.20 | More than 12 Hours |
| 100 | 5764.9515 | 25086.7694 | 5448.6296 | 28009.0235 | Sticks on 271.05 | More than 12 Hours |
| 200 | 3693.0811 | 2048.2406 | 3358.2581 | 2254.4718 | Sticks on 541.07 | More than 12 Hours |
| 300 | Sticks on 1.236e-04 | More than 12 Hours | 14756.7279 | 6318.6862 | Sticks on 1155.22 | More than 12 Hours |
| 400 | Sticks on 1.560e-04 | More than 12 Hours | Sticks on 1.421e-04 | More than 12 Hours | Sticks on 7802.43 | More than 12 Hours |

**Table 4. 12.** Experiment A is ran for functions $\sum_{i=1}^{n}x_i^2$, $\sum_{i=1}^{n}x_i^4$ and $\sum_{i=1}^{n}|x_i|$ to get $10^{-6}$ for the objective function. As we see, for large value of $n$ Nelder-Mead has more chance to stick at a non-optimized point. Meanwhile, the Genetic algorithm slows down.

### 4.4.2 Resilience

In this section we run an experiment to show the dependency of the Nelder-Mead algorithm on the initial point and it can provide an estimation for a local optimum not a global one. However, regardless of

the position of the initial point the Genetic algorithm can estimate the global optimum.

The experiment is as follow.

**Experiment B**

a) Consider the functions
   $f_0(x) = \min(|x|, |0.8 - x| + 0.5)$,
   $f_1(x) = \min(|x|, |0.4 - x| + 0.2)$ and
   $f_2(x) = \min(|x|, |0.2 - x| + 0.1)$.
b) Run the Nelder-Mead algorithm to minimize $f_0, f_1$ and $f_2$ several times with different initial values. The initial point are $x = 1.5, 1.3, 1.1, 0.9, 0.7, 0.5, 0.3$ and $0.1$. For rest of the settings Matlab's default settings are accepted.
c) Run the Genetic algorithm to minimize $f_0, f_1$ and $f_2$ several times with different initial populations. We suppose all the member of the first population are equal $x$ where
   $x = 1.5, 1.3, 1.1, 0.9, 0.7, 0.5, 0.3$ and $0.1$. For rest of the settings Matlab's default settings are accepted.

Figure 4.12 shows the graph of the $f_0, f_1$ and $f_2$ when $-1 \leq x \leq 1$. As we can see, the global minimum of all the functions are at $x = 0$. Where they have a local minimum at $(0.8, 0.5), (0.4, 0.2)$ and $(0.2, 0.1)$, respectively. Table 4.13 shows the results of the Experiment B on these three functions. As we can see in Table 4.13 based on the place of the initial guess, the Nelder-Mead algorithm may approximate a local minimum instead of global minimum, while Genetic algorithm always provides a reasonable estimation for the global minimizer.

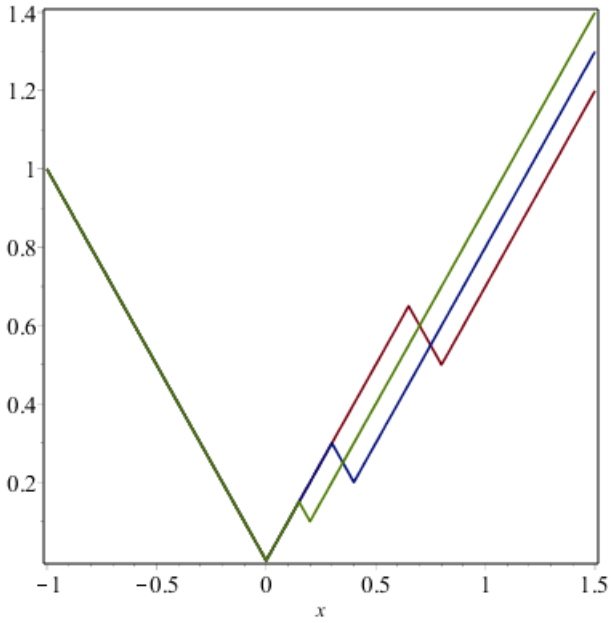



**Figure 4. 13.** $\boldsymbol{f_0, f_1}$ and $\boldsymbol{f_2}$ are shown with red, blue and green lines respectively, when $\boldsymbol{-1 \le x \le 1}$. The global minimum value of all the functions is 0, while they have local minimum at $(\boldsymbol{0.8, 0.5})$, $(\boldsymbol{0.4, 0.2})$ and $(\boldsymbol{0.2, 0.1})$ respectively.

| Initial point | Best function value for $f_0(x)$ | | Best function value for $f_1(x)$ | | Best function value for $f_2(x)$ | |
|---|---|---|---|---|---|---|
| | NM | GA | NM | GA | NM | GA |
| 1.5 | **0.5000** | 0.001824477 | $3.5527\times 10^{-15}$ | 0.00798839 | $3.5527\times 10^{-15}$ | 0.0238 |
| 1.3 | **0.5000** | 0.004202643 | $1.1102\times 10^{-15}$ | 0.00795729 | $1.1102\times 10^{-15}$ | 0.0051 |
| 1.1 | **0.5000** | 0.00184088 | $3.1086\times 10^{-15}$ | 0.0051631 | $3.1086\times 10^{-15}$ | 0.0124 |
| 0.9 | **0.5000** | $8.4947\times 10^{-4}$ | **0.20000** | 0.001339320 | $3.3306\times 10^{-16}$ | 0.0033 |
| 0.7 | **0.5000** | 0.0047894 | $2.8865\times 10^{-15}$ | 0.0029093 | $2.8865\times 10^{-15}$ | 0.0050 |
| 0.5 | $4.4408\times 10^{-16}$ | 0.0055141 | **0.2000** | $6.2638\times 10^{-4}$ | $4.4408\times 10^{-16}$ | 0.0084 |
| 0.3 | $8.3266\times 10^{-16}$ | 0.003045794 | **0.2000** | 0.0068190 | **0.1000** | 0.0089 |
| 0.1 | $1.9428\times 10^{-16}$ | 0.0092903 | $1.9428\times 10^{-16}$ | 0.001012 | $1.9428\times 10^{-16}$ | $2.0888\times 10^{-4}$ |

**Table 4. 14.** Nelder-Mead Algorithm and Genetic algorithm are applied for $\boldsymbol{f_0, f_1}$ and $\boldsymbol{f_2}$ when the initial point for Nelder-Mead algorithm is changing and the element of the first generation in Genetic algorithm are all equal to the initial point of Nelder-Mead algorithm. As we can see, where Genetic algorithm always provides a reasonable estimation of the global minimum, Nelder-Mead algorithm may approach to the local optimum.

For more illustrations, Table 4.14 and Table 4.15 show the iterations of the Nelder-Mead algorithm and Genetic algorithm for $f_0(x) = \min(|x|, |0.8 - x| + 0.5)$. The initial guess for Nelder-Mead algorithm is $x = 1$. All the

individuals in the first generation of the Genetic algorithm is also 1. For the rest of the settings, Matlab's defaults are accepted.

As we see in Table 4.14, Nelder-Mead algorithm approaches the local minimum (0.5). In fact, after some iterations the algorithm approaches to the local minimum and then by repeatedly inside contracting it makes a smaller simplex were the best vertex is always 0.5. There is no operation embedded in Nelder-Mead algorithm which makes it able to skip the local minimum.

In Table 4.15, we see that initially the algorithm shows the amount of the local optimum. However, quickly it gets closer to the global optimum. Both the best value of and average of values of a generation can increase in one specific generation (for example iteration 40), but in general the algorithm is proving an approximation for the global optimum. In fact, mutation operator embedded in Genetic algorithm makes it able to jump another area of feasible solutions; as a result it will explore a wider area.

| Iteration | Func-count | min f(x) | Procedure |
|---|---|---|---|
| 0 | 1 | 0.7 | |
| 1 | 2 | 0.7 | initial simplex |
| 2 | 4 | 0.6 | expand |
| 3 | 6 | 0.5 | reflect |
| 4 | 8 | 0.5 | contract inside |
| 5 | 10 | 0.5 | contract inside |
| 6 | 12 | 0.5 | contract inside |
| 7 | 14 | 0.5 | contract inside |
| 8 | 16 | 0.5 | contract inside |
| 9 | 18 | 0.5 | contract inside |
| 10 | 20 | 0.5 | contract inside |
| 11 | 22 | 0.5 | contract inside |
| 12 | 24 | 0.5 | contract inside |
| 13 | 26 | 0.5 | contract inside |

Optimization terminated:
the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04

**Table 4. 15.** Nelder-Mead algorithm is applied for the objective function $\boldsymbol{f_0(x) = \min(|x|, |0.8 - x| + 0.5)}$. The start point is $\boldsymbol{x = 1}$ and the stopping criterion is $\boldsymbol{10^{-4}}$ Function-tolerance. As we see the algorithm sticks on the local minimum $\boldsymbol{f(x) = 0.5}$.

| Generation | f-count | Best f(x) | Mean f(x) |
|---|---|---|---|
| 1 | 40 | 0.5 | 0.5587 |
| 2 | 60 | 0.5 | 0.6039 |
| 5 | 120 | 0.2795 | 0.75 |
| 10 | 220 | 0.06749 | 0.3312 |
| 15 | 320 | 0.0286 | 0.2483 |
| 20 | 420 | 0.0286 | 0.2859 |
| 30 | 620 | 0.009367 | 0.1583 |
| 40 | 820 | 0.00797 | 0.2894 |
| 50 | 1020 | 0.00797 | 0.1568 |
| 51 | 1040 | 0.00797 | 0.2081 |

Optimization terminated: average change in the fitness value less than options.TolFun.

**Table 4. 16.** Genetic algorithm is applied for the objective function $\boldsymbol{y = \min(|x|, |1 - x| + 0.5)}$. The stopping criteria is $\boldsymbol{10^{-4}}$ Function-tolerance, the first generation is $\boldsymbol{x = 1}$. The algorithm provides the reasonable approximation of the global minimum point $\boldsymbol{x = 0}$.

## Conclusion

Nelder-Mead and Genetic algorithms are widely used in different fields of science as robust optimization methods. In this study, we compare these two methods in the sense of required time, accuracy and dependency to the initial guess for the problem with large scale. All the experiments show that Nelder-Mead algorithm can be efficient just when the objective function has a small number of parameters. Mutation operator within Genetic algorithm is a strong tool for the efficient progress of the algorithm, and its approach to the global optimum point.

## References


Chen, P.-H. and H.-C. Chang (1995). 'Large-scale economic dispatch by genetic algorithm', *IEEE transactions on power systems*, vol. **10**(**4**), pp. 1919-1926.

Dorsey, R. E. and W. J. Mayer (1995). 'Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features', *Journal of Business & Economic Statistics*, vol. **13**(**1**), pp. 53-66.

Fogel, L., A. Owens and M. Walsh (1966). 'Artificial Intelligence Through Simulated Adaptation', in (Editor Ed.)^Eds.), *Book Artificial Intelligence Through Simulated Adaptation*, City: Wiley, New York.

Hartley, H. O. (1961). 'The modified Gauss-Newton method for the fitting of non-linear regression functions by least squares', *Technometrics*, vol. **3**(**2**), pp. 269-280.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press.

Huggett, M., G. Ventura and A. Yaron (2006). 'Human capital and earnings distribution dynamics', *Journal of Monetary Economics*, vol. **53**(**2**), pp. 265-290.

Huggett, M., G. Ventura and A. Yaron (2011). 'Sources of lifetime inequality', *The American Economic Review*, vol. **101**(**7**), pp. 2923-2954.

Klein, K. and J. Neira (2014). 'Nelder-Mead Simplex Optimization Routine for Large-Scale Problems: A Distributed Memory Implementation', *Computational Economics*, vol. **43**(**4**), pp. 447-461.

Lagarias, J. C., J. A. Reeds, M. H. Wright and P. E. Wright (1998). 'Convergence properties of the Nelder--Mead simplex method in low dimensions', *SIAM Journal on optimization*, vol. **9**(**1**), pp. 112-147.

Ljung, L. (1995). *System identification toolbox: user's guide*: Citeseer.

Marquardt, D. W. (1963). 'An algorithm for least-squares estimation of nonlinear parameters', *Journal of the society for Industrial and Applied Mathematics*, vol. **11**(**2**), pp. 431-441.

McKinnon, K. I. (1998). 'Convergence of the Nelder--Mead Simplex Method to a Nonstationary Point', *SIAM Journal on optimization*, vol. **9**(**1**), pp. 148-158.

Mitchell, M. (1998). *An introduction to genetic algorithms*: MIT press.

Nelder, J. A. and R. Mead (1965). 'A simplex method for function minimization', *The computer journal*, vol. **7**(**4**), pp. 308-313.

Pereira, R. (2000). 'Genetic Algorithm Optimisation for Finance and Investments'.

Press, W., B. Flannery, S. Teukolsky and W. Vetterling (1992). 'Numerical recipes in C (II) Cambridge University Press', in (Editor Ed.)^Eds.), *Book Numerical recipes in C (II) Cambridge University Press*, City: Cambridge.

Rechenberg, I. (1965). 'Cybernetic solution path of an experimental problem'.

Schwefel, H.-P. (1975). *Evolutionsstrategie und numerische Optimierung*: Technische Universität Berlin.

Schwefel, H.-P. (1977). *Numerische optimierung von computer-modellen mittels der evolutionsstrategie*: Birkhäuser, Basel Switzerland.

Yugeng, X. (1996). 'CHAI Tianyou and YUN Weimin (Research Center of Automation, Northeastern University. Shenyang, 110006, PRC); Survey on Genetic Algorithm [J]', *CONTROL THEORY & APPLICATIONS*, vol. **6**.

## Appendix C

In this appendix, we explain more about the operators in Genetic algorithm. In general, the algorithm has three operators which are: Selection and reproduction, Crossover and Mutation. While the general format of the algorithm is the same, the operator may vary in different problems (see Mitchell (1998)).

### C.1 Selection

Selection can have different forms like Roulette wheel, Tournament or Elite count (Elitism). Each method has its own characteristics and it is suitable for specific type of problems. It may happen also that first Elitism operator is run and then another selection method also is employed.

#### a) Roulette wheel Selection

In this operator, first based on the fitness function $(ft(x))$ all the individual's fitness are evaluated. Then $P_i = \frac{ft(x_i)}{\sum_{i=1}^{n} ft(x_i)}$ , $i = 1..n$ and $C_i = \sum_{j=1}^{i} P_i$ $(C_0 = 0)$ are calculated where $n$ is the number of individuals in the population. Obviously the more fitted individual will have higher probability $(P_i)$. We can demonstrate the situation in following graph, where area 3 is for the most fitted area 1 is for less fitted individuals.

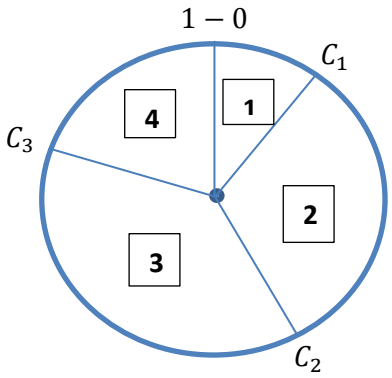



**Figure C. 1.** The Roulette wheel Selection operator.

$n$ numbers between 0 and 1 are randomly selected. For $j \epsilon N$; $0 \leq j \leq n$ and $C_0 = 0$; if $C_j < R_i < C_{j+1}$ then $x_{i+1}$ will be kept in the population. Obviously by this method the more fitted individuals has more chance to reproduce.

#### b) Tournament Selection

In this operator, first based on the fitness function $(ft(x))$ all the individual's fitness are evaluated. If you want to pass $l < n$ ($l$ is called

tournament size) number of individuals to the next generation; first, $j < l$ number of individuals are randomly (uniform distribution) selected and the best individual is kept in the population. This procedure will be repeated until the time that there is $l$ number of individuals in the population.

### c) Elitism Selection

Selection operators determines in each generation how many of the less fitted individuals will die (will be deleted from the population). It can be in the form of exact number of individuals or percentage.

## C.2 Crossover

Crossover produces a new generation by parents being selected by Selection operator. Single point cross over, two point cross over and scattered crossover are the most methods of crossover.

### a) Single point crossover

For single point cross over between chromosome A and B, first a number like $j$ between 1 and $m$ (length of the chromosome) is randomly selected. Then for the offspring the first $j$ genes are copied from chromosome A and rest of the genes from chromosome B.

If we show the parents chromosomes by [a,b,c,d,e,f,g] and [1,2,3,4,5,6]. Then single crossover is as follow if the crossover point is 4.

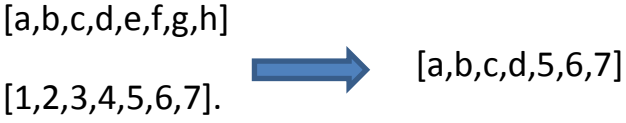



**Figure C. 2.** A single point Crossover.

### b) Two point Crossover

It is similar to single point crossover, however two numbers are selected randomly between 1 and $m$ (length of the chromosome).

If we show the parents chromosomes by [a,b,c,d,e,f,g] and [1,2,3,4,5,6]. Then two point crossover is as follow if the crossover point is 4 and 6.

[a,b,c,d,e,f,g,h]

[1,2,3,4,5,6,7]. → [a,b,c,d,5,6,h]

**Figure C. 3.** A two point Crossover.

### c) Scattered Crossover

It first generates a random binary vector with the same length of the chromosomes. Then wherever in the vector is 1, the gene is selected from parent chromosome A and wherever in the vector is 0, the genes is selected from parent chromosome B.

If we show the parents chromosomes by [a,b,c,d,e,f,g] and [1,2,3,4,5,6]. Then scattered crossover is as follow if the crossover vector is

[1 0 1 1 0 0 1].

[a,b,c,d,e,f,g,h]

[1,2,3,4,5,6,7]. —[1 0 1 1 0 0 1]→ [a,2,c,d,5,6,h]

**Figure C. 4.** A Scattered Crossover.

### C.3 Mutation

The Mutation operator randomly changes some genes/ chromosomes in the population. The typical way of mutation is first calculating A=[number of individuals in the population×length of each chromosome×mutation rate][14]. Then randomly A number of the genes in a population are selected to be changed.

[14] [ ] calculates the integer part of a number.