INDIVIDUAL COST, BENEFIT AND EFFICIENCY OF MANIPULATION: A COMPARATIVE STUDY OF SOCIAL CHOICE CORRESPONDENCES

BORA EVCİ
108622001

İSTANBUL BİLGİ ÜNİVERSİTESİ

SOSYAL BİLİMLER ENSTİTÜSÜ

EKONOMİ YÜKSEK LİSANS PROGRAMI

Under Supervision of

Prof. JEAN LAINÉ

2010

Individual Cost, Benefit and Efficiency of Manipulation:
A Comparative Study of Social Choice Correspondences

Manipülasyonun Bireysel Maliyet, Kazanç ve Verimi:
Sosyal Seçim Kurallarının Kıyaslamalı Bir Çalışması

Bora Evci
108622001

Tez Danışmanının Adı Soyadı : Jean LAINÉ
Jüri Üyelerinin Adı Soyadı : İpek ÖZKAL-SANVER
Jüri Üyelerinin Adı Soyadı : Ayça Ebru GİRİTLİGİL
Tezin Onaylandığı Tarih : 20.08.2010
Toplam Sayfa Sayısı : 71

Anahtar Kelimeler
1) Kemeny Mesafe Fonksiyonu
2) Genişleme Kuralı
3) Manipülasyon
4) Maliyet, Kazanç, Verimlilik

KeyWords
1) Kemeny Distance Function
2) Extension Rule
3) Manipulation
4) Cost, Gain, Efficiency

Abstract

In this study, we work on the degree of manipulabilities of some social choice correspondences (SCC) by using computational methods. We consider four SCCs; the Borda rule, the Uncovered set, the Top Set and the Copeland rule, under three extension rules; the Lexicographic extension rule, the Max-Min ordering and the Expected-Ranking. We use three different approaches to measure the manipulabilities of SCCs; the computational cost of manipulation, the gains from manipulation and the efficiency of manipulation. Since we work on full domain and SCCs under no restriction, we use computers for this huge work. We design a special software in JAVA to handle this job.

Özet

Bu çalışmada, hesaplama yöntemleri kullanılarak, bazı sosyal seçim kurallarının manipüle edilebilirlik dereceleri üzerinde çalışılmaktadır. Lexicographic, Max-Min ve Expected-Ranking olmak üzere üç tane genişleme kuralı altında, Borda kuralı, Uncovered Set, Top-Set ve Copeland kuralı olmak üzere dört tane sosyal seçim kuralı incelenmektedir. Sosyal seçim kurallarının manipüle edilebilirlik derecesi üç farklı yaklaşımla hesaplanmaktadır; manipülasyon maliyeti, manipülasyon getirisi ve manipülasyon verimliliği. Tam değer kümesi ve hiçbir şekilde sınırlandırılmamış sosyal seçim kuralları üzerine çalışıldığından, bu büyüklükte ki hesapların yapılabilmesi için bilgisayarlar kullanılmıştır. Bu amaç için de, JAVA programlama dilinde özel yazılımlar geliştirilmiştir.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Jean Lainé for his invaluable guidance and positive personal effects on me. His encouraging supervision brought my studies up to this point.

I am very thankful to Prof. Remzi Sanver for introducing this field to us. His interesting talks during the lectures which give the taste of research and his absolutely perfect guidance has increased my motivation to take part in the academic life.

I am indebted to Prof. Tarık Kara who has teached me a lot so far. Thanks to him, I am walking on the road to be an academician. He is not only one of my professors, he is also a friend and an elder brother. I feel that I am incredibly lucky to know him.

I would like to thank Eldeniz Kurbanov and Ass. Prof. Mehmet Gencer for their contributions to the software and interesting talks on programming at the beginning of this project.

I cannot underestimate the contribution of an old friend, Cemal Esner, to the execution of the programs. I am very very thankful to him for his restless effort during the project, taking responsibility for many computers and his 25-year friendship.

I would like to thank Reyhan&Cemile Tufan, Okan Kanyılmaz, Emine Karataş, Emre&Erdi Mutlu and Mahmut Mutlu for letting me use their computers to make computations.

I also would like to thank Yakup Gündoğdu, Hüseyin Gürel and Rıdvan Topçuoğlu, the teachers of Refii Terzioğlu Secondary school in Edremit/Balıkesir, for letting me use the computer lab of the school.

I am very thankful to TÜBİTAK for financial support during my master's degree.

The last, but not the least, I am appreciate to my parents for their supports. For several months, they have spent endless and sleepless nights with me.

Contents

1 INTRODUCTION

In Social Choice Theory, manipulation is a phenomenon occured in the collective decision making processes. It considers the agents' intentional behaviors of mispresenting their true preference orderings over the alternative sets to be better paid off in the vote aggregation processes.

We know from Gibbard-Satterthwaite Theorem (1973,1975) that any social choice rule is either dictatorial or manipulable. The extents of manipulabilities of social choice rules have been studied by Kelly (1993), Smith (1999) and Aleskerov and Kurbanov (1997); many criterias have been proposed.

We will contribute to this literature by proposing our criterias and corresponding results.

2 PRELIMINARY NOTES

Social Choice Theory is a theoritical approach to investigate the aggregation of individuals' interests in a collective decision making frame. Starting with Arrow(1951), one of the biggest problems of working on SCT are the negative results, one of which is known as Gibbard-Satterthwaite Theorem (1973,1975) which states that any social choice rule is either dictatorial or manipulable. Although all social choice rules are manipulable, except dictatoriality, their manipulabilities are different and many studies have been done to distinguish them. Before we give our criterias on the degree of manipulation, we will give some preliminary notes including definitions and notation.

Definition 1 *Let $A \neq \emptyset$ be any set of alternatives. A* ***preference*** *of an individual over A is a binary relation $R \subseteq A \times A$.*

Interpretation: Given any $x, y \in A$, $(x, y) \in R$ means "x is at least as good as y in view of the individual". We will write xRy instead of $(x, y) \in R$. If xRy and not yRx, then we write xR^+y which means "x is better than y in view of the individual"; namely, R^+ is the strict part of R. If xRy and yRx, then we write xIy which means "the idividual is indifferent between x and y".

Definition 2 *We say that R is* ***complete*** *if and only if xRy or yRx $\forall x, y \in A$. So, if R is complete, one of these holds: xR^+y, yR^+x, xIy $\forall x, y \in A$.*

Definition 3 *We say that R is* ***transitive*** *if and only if xRy and $yRz \Rightarrow xRz$, $\forall x, y, z \in A$.*

Definition 4 *We say that R is* ***rational*** *if and only if R is complete and transitive.*

Note: In this study, we deal with Linear Orders, $L(A)$; complete, transitive and antisymmetric binary relations, namely strict preferences. Hence, $\forall i \in N$, $\forall x, y \in A$, either xR^+y or yR^+x.

PREFERENCE AGGREGATION

When there is more than one single person, to reach a final decision, we aggregate the preferences of all individuals.

Let $A \neq \emptyset$ be the set of alternatives with $Card(A) \geq 2$. Let $N = \{1, ..., n\}$ be the society or a group of people or the set of agents with $Card(N) \geq 2$.

Let $\Re$ be the set of all strict rational preferences over A. We write $R_i^+ \in \Re$ stands for the preference of the agent $i \in N$. We write $R^+ = (R_1^+, ..., R_n^+) \in \Re^N$ for a preference profile.

Definition 5 *A Social Choice Correspondence (SCC) is a mapping* $\alpha : \Re^N \rightarrow 2^A/\emptyset$. *For any* $R^+ \in \Re^N$, *we interpret* $\alpha(R^+) \in \Re$ *as the* ***social choice****.*

In this study, we will compare the degree of manupulabilities of several social choice correspondences. Now, we will give the definitions of social choice correspondences that we will work on.

SOCIAL CHOICE RULES

Let A be the set of alternatives with $Card(A) = m$ and N be the set of agents with $Card(N) = n$. Let $r(i, x)$ be the rank of the alternative x for the agent i and $x(i, r)$ be the alternative r^{th} ranked by the agent i.

1) The Borda Rule In a profile, the Borda Score $BS(x)$ of an alternative x is $BS(x) = \sum_{i \in N} [(m + 1) - r(i, x)]$. In a voting system, the Borda Rule selects the alternative(s) with the highest Borda Score.

2) The Copeland Rule The Copeland Score $CS(x)$ of an alternative x in a tournament T; complete and asymmetric binary relation over A, is $CS(x) = \#\{y \in A : xTy\}$. The Copeland Rule picks the alternative(s) with the highest Copeland Score.

3) The Uncovered Set In a tournament, $\forall x, y \in A$, it is said that x *covers* y iff xTy and $\forall z \in A$ such that yTz, xTz. The Uncovered Set of a tournament T is $UC(T) = \{x : \nexists y \in A \text{ that covers } x\}$.

4) The Top Set The Top Set of a tournament T is the set of alternative(s) which weakly defeat any other alternative through a path, that is $TS(T) = \{x : xT^w y \text{ or } xT^w y_1 T^w y_2 ... y_k T^w y\}$.

Note: In a tournament T, $\forall x, y \in A$ if $\#\{i \in N : xR_i^+ y\} \geq \#\{j \in N : yR_j^+ x\}$, then $xT^w y$.

In this study, we are treating the outcomes of above SCCs as **Resolute Outcomes**.

Definition 6 *A Resolute Choice Function is a mapping* $C : 2^A/\emptyset \rightarrow A$ *such that* $C(X) \in X$, $\forall X \in 2^A/\emptyset$.

Roughly speaking, we say that a social choice correspondence is resolute when its set valued outcomes are interpreted as mutually compatible alternatives which are altogether chosen.

Since the outcomes of SCCs are any subsets X of $2^A/\emptyset$, we need to define a preference extension rule from the preferences over alternatives to the preferences over subsets of the alternatives.

Definition 7 *An Extension Axiom is a mapping ε which assigns to each $R \in \Re^N$ a transitive and asymmetric binary relation $\varepsilon(R)$ over $2^A/\emptyset$. Roughly speaking, given any preference over alternatives, an extension rule determines the preferences over sets.*

In this project, we used three different extension axioms.

1) The Lexicographic Extension Rule Let X and Y be any subsets of A. For $r \in \{1, ..., m\}$, $\forall i \in N$, $\forall x \in A$, if $\exists r^{'}$ such that $\forall r < r^{'}$ $x(i,r) \in X$ and $x(i,r) \in Y$ or $x(i,r) \notin X$ and $x(i,r) \notin Y$, and for $r^{'}$, $x(i,r) \in X$ and $x(i,r) \notin Y$, then $X\varepsilon(R_i)Y$ if $Card(Y) > r^{'}$ or $Y\varepsilon(R_i)X$ if $Card(Y) = r^{'}$.

2) The Max-Min Ordering Carmelo Rodríguez-Álvarez (2007) defined the following criteria to rank the sets as objects. Let $X, Y \in 2^A/\emptyset$. Let $max(X, R_i)$ be the maximum (best) alternative and $min(X, R_i)$ be the minimum (worst) alternative of the set X in view of the agent i. To decide which set is better in view of the agent i, we first compare $r(i, max(X, R_i))$ and $r(i, max(Y, R_i))$.If one of them is a higher rank than the other, then the relevant set is better for the agent i. If equal, then we check $r(i, min(X, R_i))$ and $r(i, min(Y, R_i))$. Again, if one of them is a higher rank than the other, then the corresponding set is better for the agent i. If they are equal too, then the sets X and Y are equal in view of the agent i.

3) The Expected-Rank Rule Let $V(X,i) = \sum_{x \in X} \frac{r(i,x)}{Card(X)}$ be the value of the set X in view of the agent i. Let $X, Y \in 2^A/\emptyset$. If $V(X,i) < V(Y,i)$, then $X\varepsilon(R_i)Y$. If the values are the same, then the agent i is indifferent between the sets X and Y.

MANIPULATION

In a voting system, if the preferences are private, the agents may prefer to mispresent them, which is called **manipulation**.

Definition 8 *A Social Choice Correpondence (SCC) $\alpha : \Re^N \to 2^A/\emptyset$ is **manipulable** at $R^+ \in \Re^N$ by any agent $i \in N$ iff $\exists R^{+'} \in \Re^N$ such that $R_j^{+'} = R_j^+$ $\forall j \in N \backslash \{i\}$ while $\alpha(R^{+'})\varepsilon(R_i^+)\alpha(R^+)$.*

Definition 9 *A Social Choice Correpondence (SCC) $\alpha : \Re^N \to 2^A/\emptyset$ is **strategy-proof** iff α is manipulable at no $R^+ \in \Re^N$ by no $i \in N$.*

If there is no restriction in voting, the only social choice rule which is strategy-proof is the dictatoriality.

Definition 10 *A Social Choice Correpondence (SCC) $\alpha : \Re^N \to 2^A/\emptyset$ is **dictatorial** iff $\exists d \in N$ such that $\alpha(R^+) = R_d^+$ $\forall R^+ \in \Re^N$.*

3 DEGREE OF MANIPULATION

Under those three different extension rules mentioned above, we worked on manipulation for our SCCs; the Borda Rule, the Top-Set, the Uncovered Set and the Copeland Rule. We follow three different methods to compare the manipulabilities of SCCs; the cost of manipulation, the gains from manipulation and the efficiency of manipulation.

In all three methods mentioned above, we used computations instead of theoritical approaches. If we had worked on the degree of manipulation in a theoretical way, we might have got the boundaries in which any agents can manipulate the voting system, but in this case we might have got indistinguishable manipulabilities of SCCs. To get exact and strict results, we needed to find the exact sets/profiles in which manipulations occur. By a computational method, not only we can get the exact differences in degree of manipulation, but also we can get the percentage differences between SCCs.

3.1 COMPUTATIONAL COST OF MANIPULATION

The manipulation is the behaviour of mispresenting the true preference to be better paid off in a voting system. In this method, we measured the cost of manipulation for an agent and compared the costs of manipulabilities of SCCs under a fixed extension rule.

We describe the cost of manipulation by measuring the distance between the true preference and the mispresented preference of the agent manipulating. We use the Kemeny Distance function as the distance function.

Kemeny Distance Function

Kemeny Distance Function measures the distance between two preference orderings over any set of alternatives A.

Let $R, Q \in R^+$ be linear orders defined over A. Let $r_R(i, x)$ be the rank of the alternative x in the linear order R in view of the agent i. Let $\gamma_{st} : R^2 \to \mathbb{R}$, be a function defined for all $i \in N$, for all pairs of alternatives (s, t) of A and all pairs of linear orders (R, Q) by:

$$\gamma_{st}(R, Q) = \left\{ \begin{array}{cc} 1, \text{ if } r_R(., s) < r_R(., t) \text{ and } r_Q(., s) > r_Q(., t) \\ 0, \qquad\qquad \text{otherwise} \end{array} \right\}$$

Hence, the Kemeny Distance Function, d^K, is defined as $d^K = \sum_{s \in A} \sum_{t \in A} \gamma_{st}(R, Q)$. Briefly, Kemeny function measures the distance between orderings R and Q by calculating the number of adjacent pairwise switches needed to reach Q from R.

For all profiles of linear orders, we find all the manipulation moves of all agents under three extension rules. And, then, we compute the distances between true and announced preferences of the agents by Kemeny distance function in all moves in all profiles. After then, we follow two different protocols:

Protocol 1: In the first one, we take the average distance of all moves in all profiles.

Protocol 2: In the second one, we identify the moves with the minimum distance of all moves in each profile, and, then, we take the average distance of those moves.

We do all computations in two protocols for four SCCs under three extension rules.

Note: As an example, the codes of the program for (m,n)=(7,4), the Lexicographic extension rule, the Uncovered set, protocol 2 are given in Appendix.

3.2 INDIVIDUAL GAINS FROM MANIPULATION

Campbell and Kelly (2009) defined the following criteria on gains from manipulating social choice rules:

"Let A be the set of alternatives with $Card(A) = m$ and N be the set of agents with $Card(N) = n$. For any subset X of A, $\varkappa(X)$ is the set of all strong orderings on X. A profile is a function from N into $\varkappa(X)$ and $\varkappa(X)^N$ is the set of all profiles. We denote $p(i)$ as the ordering of the agent i in the profile p. Two profiles are i-variants, where $i \in N$, if $q(j) = p(j)$ for all $j \neq i$.

For a social choice rule f and non-negative integer t, we say f allows a gain of t if there exist two profiles p and q and an individual i such that:

(1) p and q are i-variants;

(2) $f(q)$ ranks t positions higher in $p(i)$ than $f(p)$.

A rule f has a gain of k, written $G(f) = k$, if k is the maximum t such that f allows a gain of t. $G(f)$ is bounded above by $(m-1)$; with no restrictions on the rule f, the gain $G(f)$ can be as large as $(m-1)$."

They also characterizes the lower bounds of $G(f)$ in terms of m for some social choice rules.

Briefly, Campbell and Kelly determined the upper and the lower bounds of gains from manipulating the social choice rules with singleton outcomes and under unrestricted domains. They measured the increase in rank for an agent when (s)he manipulates and defined the gain intervals for some social choice rules.

In this project, we extend Campbell-Kelly's work into SCCs. Since we do not deal with any restrictions on SCCs or domains, any subset of the alternative set A could be the outcome of our SCCs. Hence, as mentioned above, we used three extension rule to rank the sets.

We do our computations for the pairs (m, n) from $(3, 4)$ to $(8, 4)$ and from $(3, 5)$ to $(5, 5)$. The number of non-empty subsets of the alternative set A is $2^4 - 1 = 15$ for $m = 4$ and $2^5 - 1 = 31$ for $m = 5$. In two of those extension rules, some sets are indifferent to each other for a fixed agent. Hence, we have the table below which gives the total ranks for each extension rule.

	Lexicographic	Max-Min	Expected-Rank
m=4	15	10	9
m=5	31	15	15

As an example, let $A = \{a, b, c, d\}$ and $N = \{1, 2, 3, 4\}$. Let say *agent* 2's preference ordering over A is $aR_2^+bR_2^+cR_2^+d$. We shall give his preference ordering over the subsets of the alternative set A under the Max-Min ordering extension rule.

Agent 2
a
ab
ac,abc
ad,abd,acd,abcd
b
bc
bd,bcd
c
cd
d

Like Campbell-Kelly, we determine the increases in rank for the agents from manipulating the SCCs.

As in the cost of manipulation, for all profiles of linear orders, we find all the manipulation moves of all agents under three extension rules. And, then, we compute the gains in agents' rankings over the sets for all moves in all profiles. After then, we follow two different protocols:

Protocol 1: In the first one, we take the average gain of all moves in all profiles.

Protocol 2: In the second one, we identify the moves with the maximum gain of all moves in each profile, and, then, we take the average gain of those moves.

We do all computations in two protocols for four SCCs under three extension rules.

We also find the possible maximum gains on full domain of all SCCs under three extension rules.

3.3 EFFICIENCY OF MANIPULATION

In previous sections, we discussed the individual cost and benefit of manipulation. When working on the cost part, we did not care about the gain from manipulation on the same move. Vice versa, we did not consider the cost of manipulation when computing the gain. In this section, we will introduce the efficiency of manipulation.

To compute the efficiency of manipulation, for all profiles of linear orders, we determine all the manipulation moves of all agents under three extension

rules. And, then, we compute the gains in agents' rankings over the sets and the distances between true and announced preferences by Kemeny distance function for all moves in all profiles. We calculate $\frac{Gain}{Cost}$ for each manipulation move, and then, take the average of all those ratios for all moves.

We do all computations for four SCCs under three extension rules.

3.4 COMPUTATIONS

In this study, as mentioned before, we calculate the degree of manipulation of several SCCs for the pairs (n,m) from (3,4) to (8,4) and from (3,5) to (5,5). The table below shows the number of profiles for each pair (n,m):

(3,4)	13.824
(4,4)	331.776
(5,4)	7.962.624
(6,4)	191.102.976
(7,4)	4.586.471.424
(8,4)	110.075.314.176
(3,5)	1.728.000
(4,5)	207.360.000
(5,5)	24.883.200.000

While executing the computations, we considere each agent's each move in each profile; namely, we compute every move, exhaustively. As seen from the table, it is extremely difficult to do such huge calculations manually. For this reason, we use programming.

For each computation, we design different softwares in JAVA. We use several computers to execute the programs. Some computations are estimated to be so long that we do not try to execute them; for example we cannot make the computations for the pair (5,5), the Cost of Manipulation, Protocol 2, because executing each program for each SCC is estimated to take more than 700 days with a single computer.

The software used in our computations have been very carefully tested and is working correctly. The codes are totally open and any requests for the software to test new criterias are welcome.

4 RESULTS AND COMMENTS

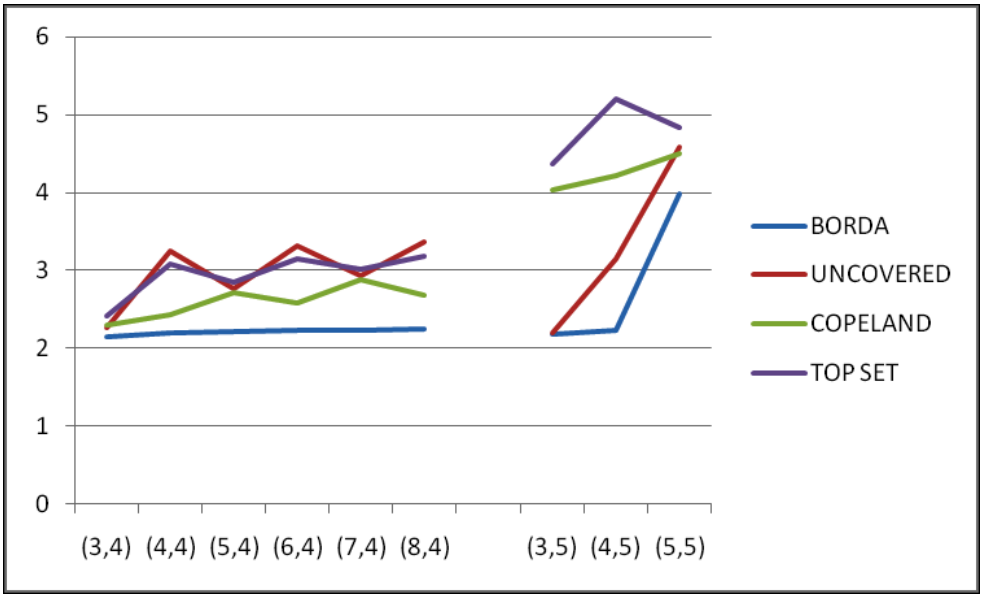
In this section, we will give the results of computations through tables and graphs.

4.1 COST OF MANIPULATION

4.1.1 PROTOCOL 1

The table and the corresponding graph shown below shows the results of the cost of manipulation of four SCCs under the Lexicographic extension rule.

COST, LEXICOGRAPHIC, PROTOCOL 1									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	2,1447	2,2022	2,2053	2,2286	2,2293	2,2436	2,1775	2,2266	3,9903
UNCOVERED	2,2608	3,2512	2,7588	3,3184	2,94	3,3601	2,2008	3,1523	4,5837
COPELAND	2,2971	2,4264	2,7139	2,5814	2,8849	2,6838	4,0368	4,2185	4,4945
TOP SET	2,4142	3,0826	2,8564	3,1417	3,023	3,1844	4,363	5,204	4,8403

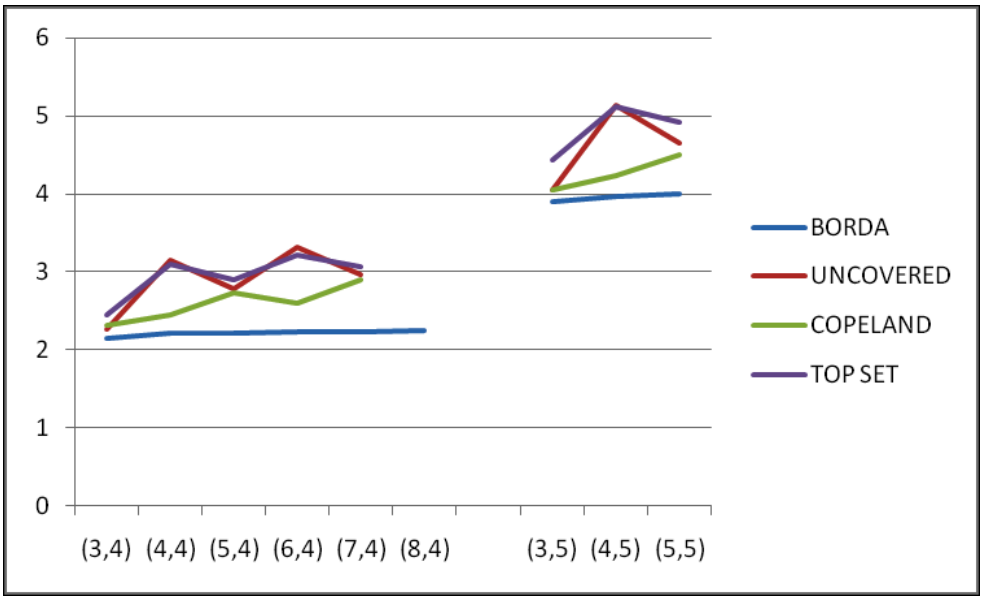


Cost of Manipulation, Protocol 1, Lexicographic Extension rule

As we see from the graph, the Borda rule is less costly to manipulate compared to the others. For the four alternative case, $m = 4$, the Uncovered set and the Top Set are almost same. The Copeland Rule is slowly increasing as the number of agents increases. And, it could be estimated that if we increase the number of alternatives m, the Top Set becomes much more costly than the other SCCs to manipulate.

Next table and the corresponding graph shows the results of the cost of manipulation of four SCCs under the Max-Min ordering extension rule.

COST, MAX-MIN ORDERING, PROTOCOL 1									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	2,1533	2,2053	2,2118	2,2313	2,2345	2,2461	3,9066	3,9692	3,9961
UNCOVERED	2,2719	3,151	2,776	3,3146	2,9581		4,0442	5,1428	4,644
COPELAND	2,3125	2,4445	2,7263	2,6038	2,894		4,059	4,2398	4,508
TOP SET	2,4538	3,1053	2,9018	3,2082	3,0651		4,4391	5,1153	4,9201

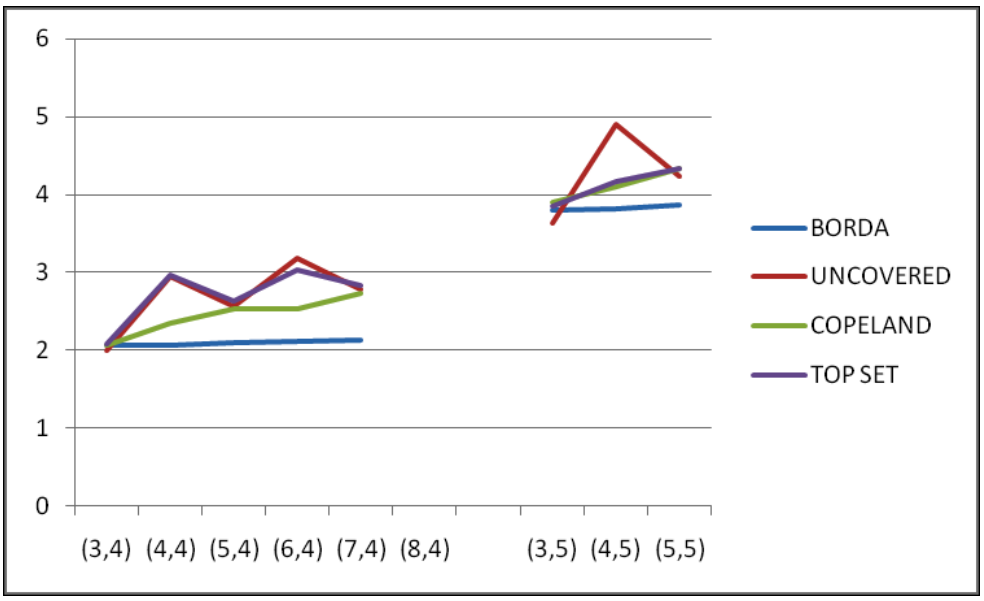


Cost of Manipulation, Protocol 1, Max-Min Ordering

Again, the Borda rule is less costly than the other rules. The Uncovered set and the the Top Set have same level costs for both $m = 4$ and $m = 5$. Again, the Copeland rule is increasing smoothly as n increases. We can estimate that the positions of the SCCs to each other do not change a lot if we increase m.

Next ones show the results of the cost of manipulation of four SCCs under the Expected-Ranking extension rule.

COST, EXPECTED-RANKING, PROTOCOL 1									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	2,0571	2,0599	2,0953	2,1138	2,1229		3,7926	3,8206	3,8654
UNCOVERED	2	2,9411	2,5579	3,179	2,7744		3,6259	4,8983	4,2347
COPELAND	2,0714	2,3488	2,5281	2,5346	2,7255		3,898	4,0997	4,343
TOP SET	2,074	2,9589	2,6318	3,0287	2,8374		3,8499	4,1629	4,334

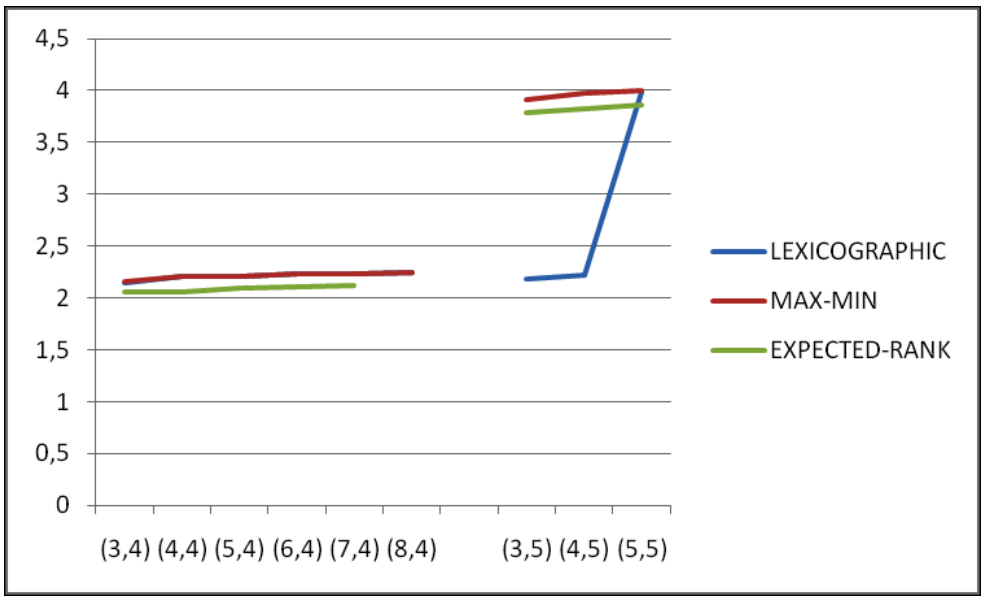


Cost of Manipulation, Protocol 1, Expected-Ranking

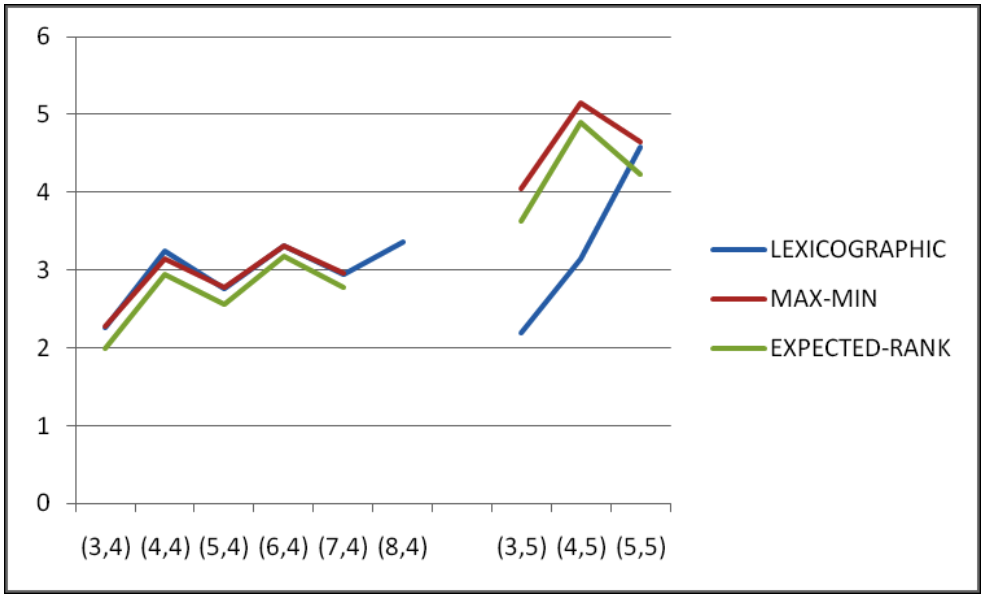
Behaviour of the Borda rule and the Copeland rule is like in the Lexicographic extension and the Max-Min ordering. Except the wavings of two SCCs, the Uncovered set and the Top Set, all rules, except the Borda rule which is the lowest one, have same level of costs.

COMPARISON OF THE EXTENSION RULES

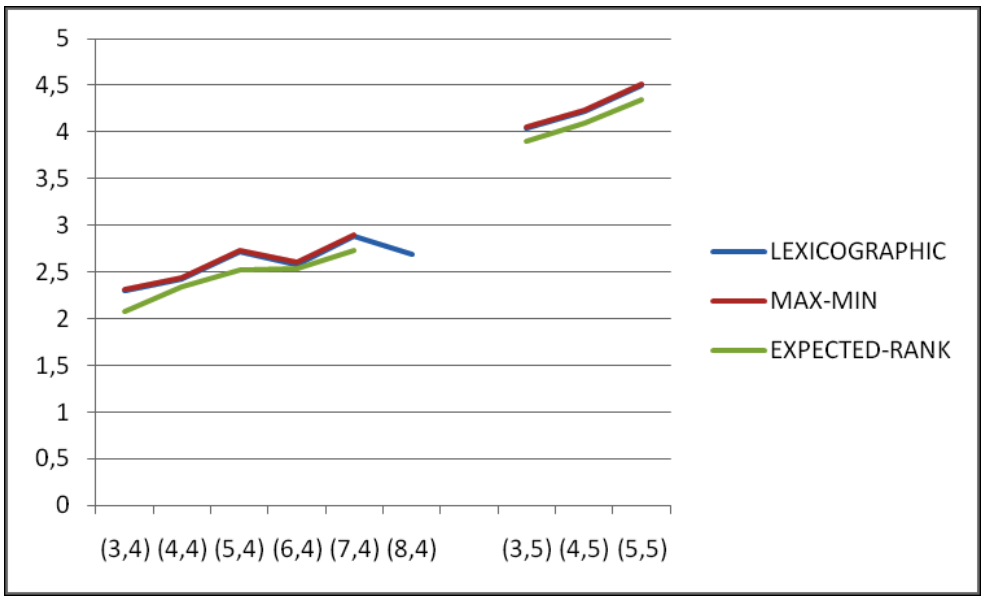
The following graphs show the behaviours of all SCCs under three extension rules.



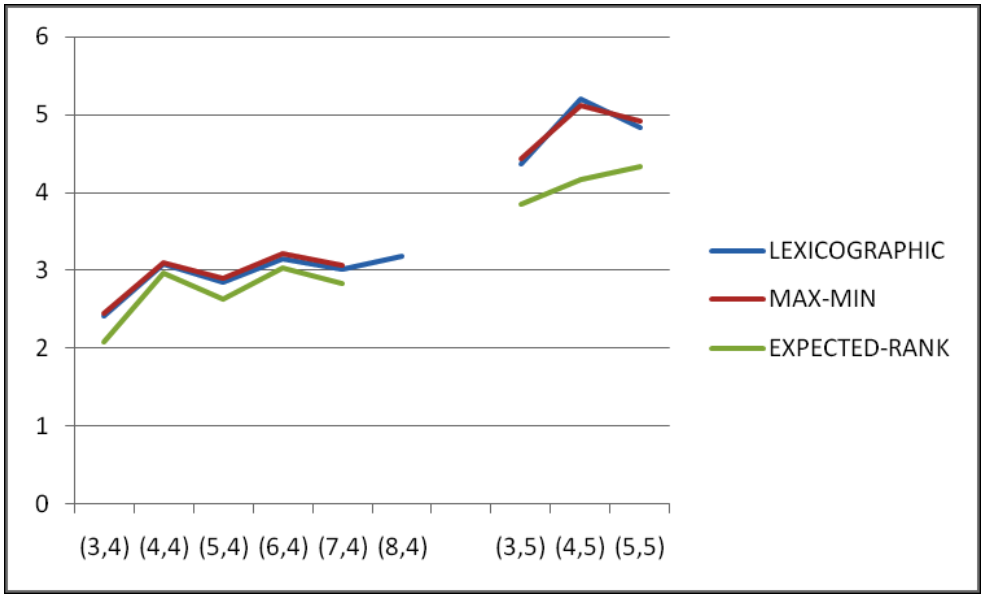
Borda, Cost of Manipulation, Protocol 1



Uncovered, Cost of Manipulation, Protocol 1



Copeland, Cost of Manipulation, Protocol 1



Top Set, Cost of Manipulation, Protocol 1

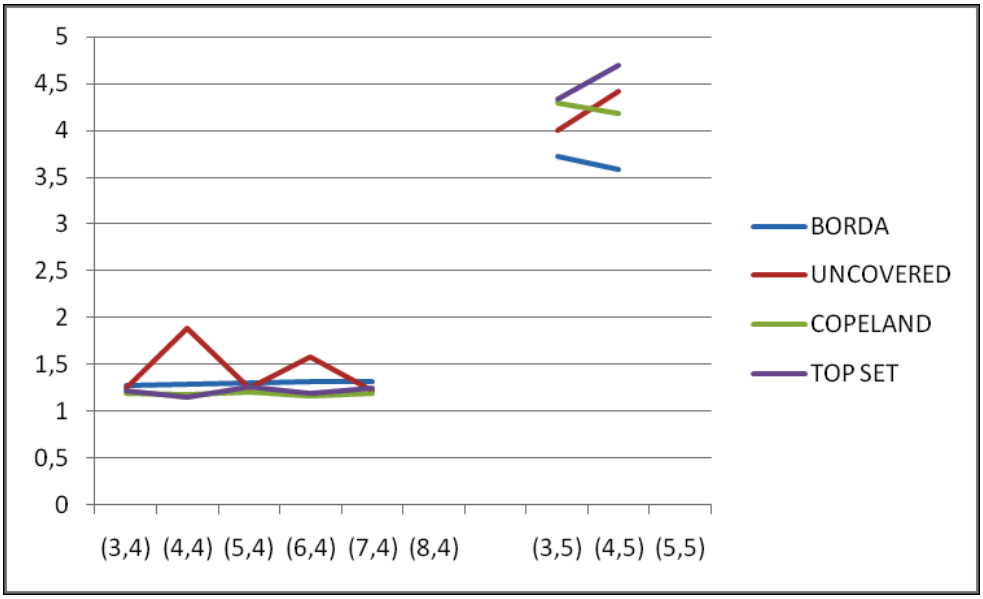
As we see from four graphs, the costs of each SCC are same for $m = 4$ under three extension rules. This is probably from the reason that all three extension

rules almost have same number of ranking for $m = 4$. For the case of $m = 5$, the Lexicographic extension rule provides much more ranking than the others. But, some properties of the Copeland rule and the Top Set must rule out this difference of the Lexicographic rule. For the Borda rule and the Uncovered set, since the Lexicographic extension rule provides middle ranks with a wider range of rankings, the cost levels are less than the Max-Min ordering and the Expected-Ranking on the average. Hence, we can say that for a fixed SCC, the cost of manipulating this SCC does not change under different extension rules for Protocol 1.

4.1.2 PROTOCOL 2

The table and the corresponding graph shown below shows the results of the cost of manipulation of four SCCs under the Lexicographic extension rule.

COST, LEXICOGRAPHIC, PROTOCOL 2									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	1,2755	1,2856	1,2946	1,3138	1,3161		3,7293	3,5795	
UNCOVERED	1,247	1,884	1,2402	1,5808	1,2159		3,9974	4,4191	
COPELAND	1,1935	1,1695	1,202	1,1591	1,1928		4,2971	4,1794	
TOP SET	1,2168	1,1476	1,2523	1,1827	1,2457		4,3329	4,6967	

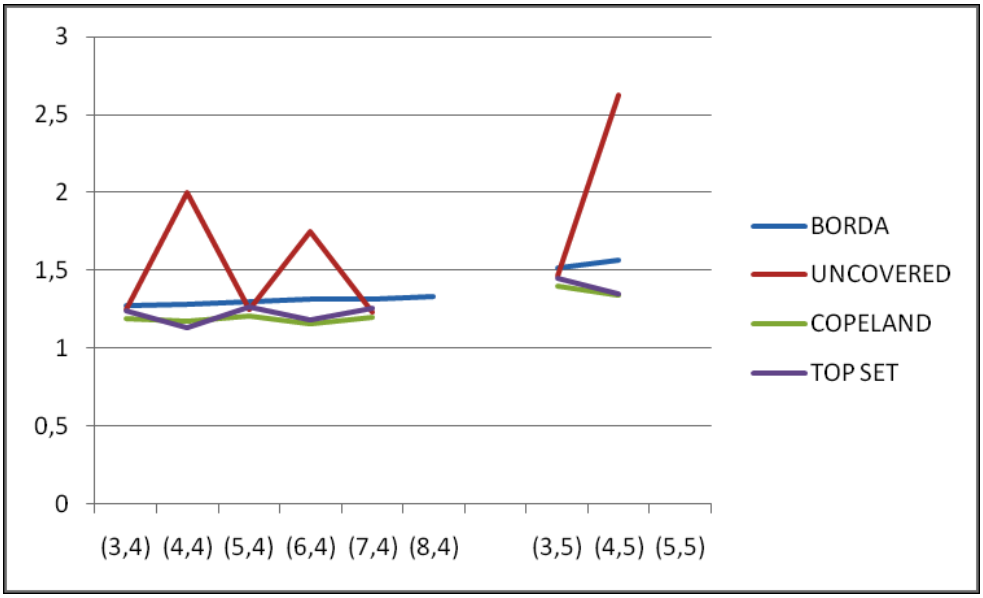


Cost of Manipulation, Protocol 2, Lexicographic Extension rule

Except the Uncovered Set's waving, which stem from outcomes being odd numbers, all rules almost have same cost levels; namely their minimal distances to manipulate are almost same.

Next table and the corresponding graph shown below shows the results of the cost of manipulation of four SCCs under the Max-Min ordering extension rule.

COST, MAX-MIN ORDERING, PROTOCOL 2									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	1,2755	1,2856	1,2946	1,3138	1,3161	1,3333	1,5181	1,564	
UNCOVERED	1,247	2,0014	1,2501	1,7457	1,2291		1,4687	2,623	
COPELAND	1,1935	1,1695	1,2058	1,1586	1,1962		1,3952	1,3388	
TOP SET	1,2368	1,1317	1,2648	1,1823	1,2565		1,4456	1,3465	

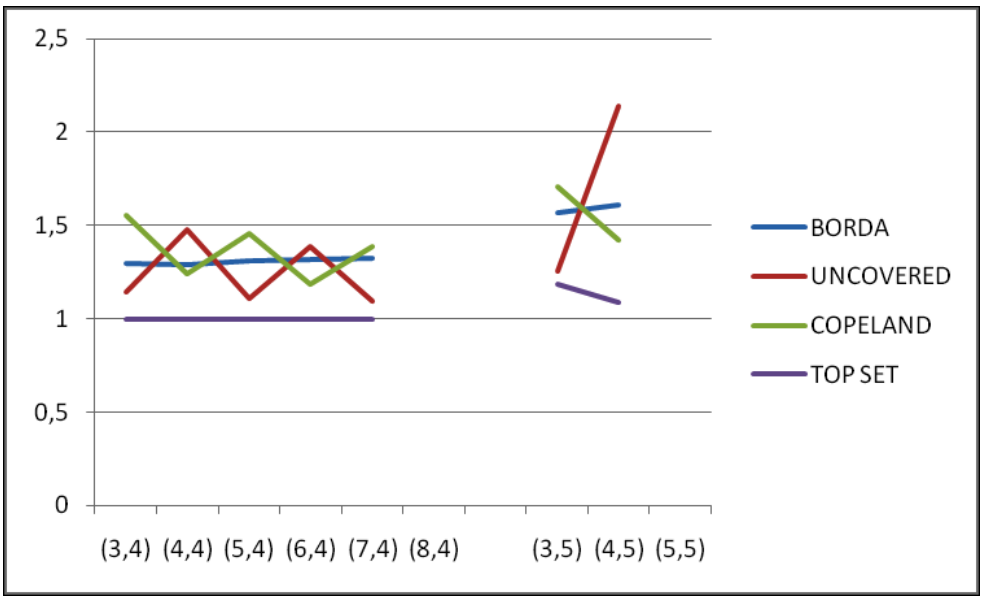


Cost of Manipulation, Protocol 2, Max-Min ordering

Again, like under the Lexicographic extension rule, except the Uncovered set's wavings when the number of agents is even, all SCCs have same level of costs.

Next ones show the results of the cost of manipulation of four SCCs under the Expected-Ranking extension rule.

COST, EXPECTED-RANKING, PROTOCOL 2									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	1,2965	1,2878	1,3094	1,3197	1,3268		1,5674	1,6075	
UNCOVERED	1,1428	1,4766	1,1048	1,3857	1,0915		1,2544	2,1415	
COPELAND	1,5555	1,2381	1,4572	1,187	1,3897		1,7085	1,4236	
TOP SET	1	1	1	1	1		1,1816	1,0875	

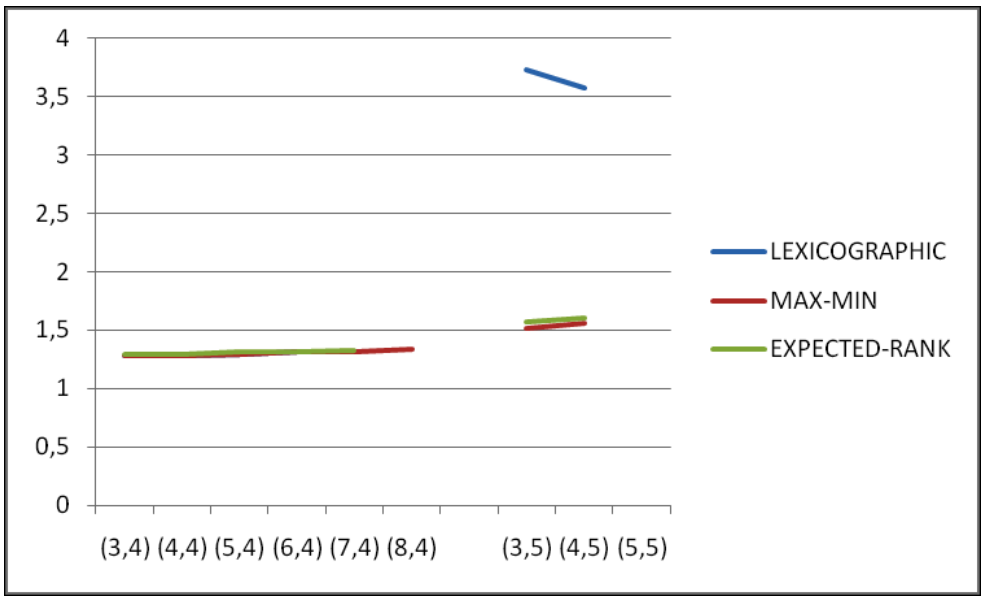


Cost of Manipulation, Protocol 2, Expected-Ranking

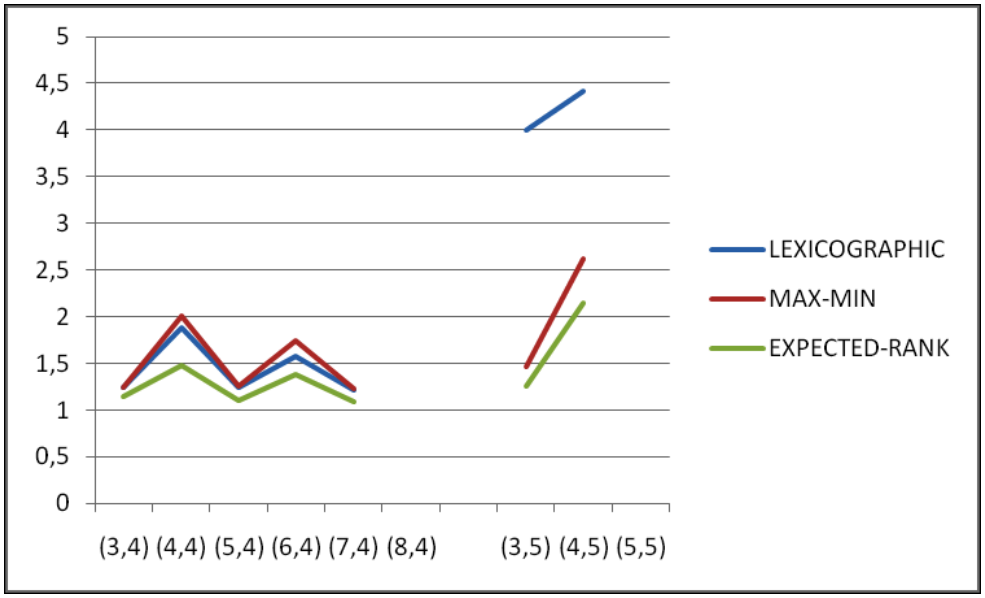
Like in the previous ones, the Borda rule and the Top Set are smooth and the Uncovered set is waving. Differently, the Copeland rule is waving compared to previous ones. This is probably because, in this situation, same thing the Uncovered set happens to the Copeland rule; there must be thresholds to manipulate the Copeland rule which depends on the number of alternatives of the outcome of voting.

COMPARISON OF THE EXTENSION RULES

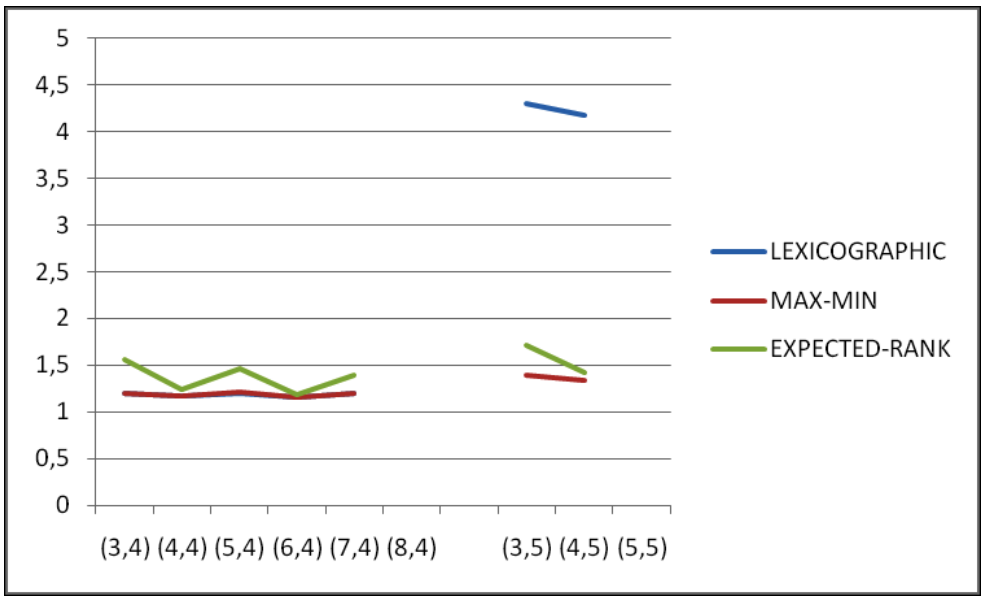
The following graphs show the behaviours of all SCCs under three extension rules.



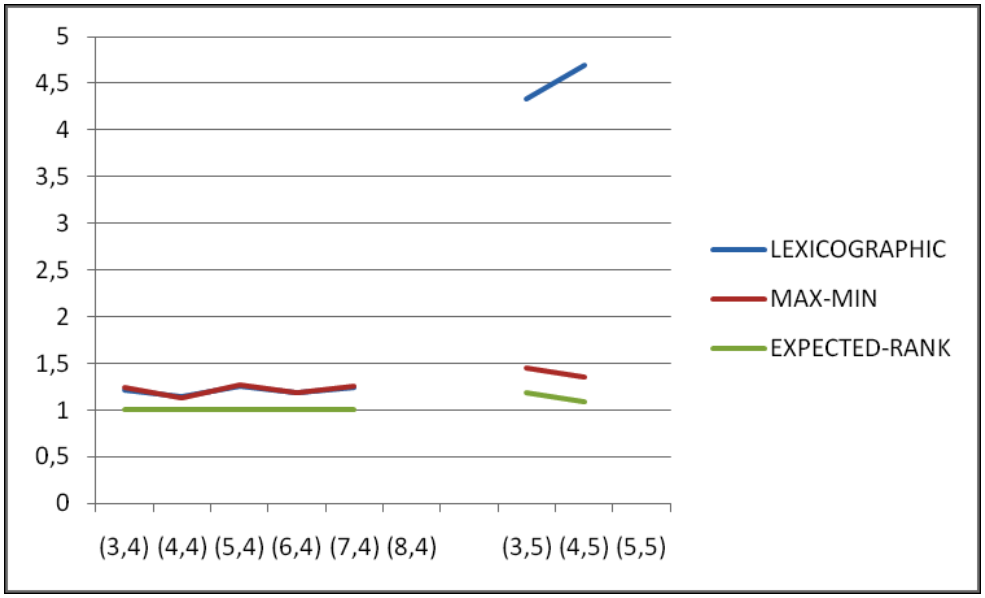
Borda, Cost of Manipulation, Protocol 2



Uncovered, Cost of Manipulation, Protocol 2



Copeland, Cost of Manipulation, Protocol 2



Top Set, Cost of Manipulation, Protocol 2

Since for $m = 5$, the lexicographic extension rule has more ranking; especially it includes middle ranks, for those number of alternatives, SCCs are more costly

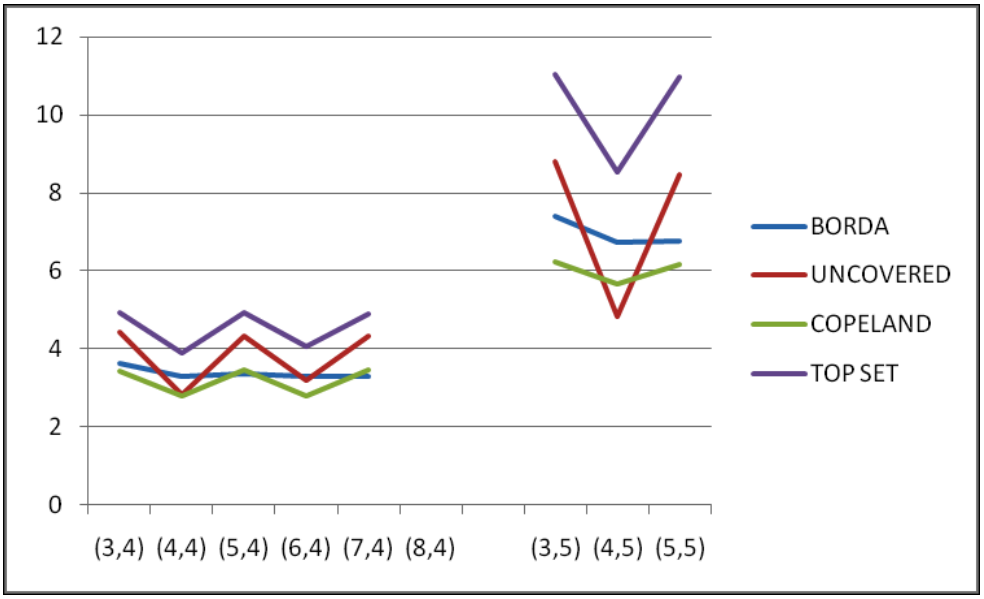
to be manipulated with the smallest Kemeny distance under the lexicographic extension rule. For $m = 4$, all extension rules give same level of manipulabilities.

4.2 GAIN FROM MANIPULATION

4.2.1 PROTOCOL 1

The table and the corresponding graph shown below shows the results of the gains from manipulation of four SCCs under the Lexicographic extension rule.

GAIN, LEXICOGRAPHIC, PROTOCOL 1									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	3,6184	3,293	3,343	3,2871	3,278		7,3844	6,7343	6,7767
UNCOVERED	4,4173	2,8223	4,336	3,1744	4,3127		8,7972	4,8341	8,4731
COPELAND	3,4202	2,7955	3,4431	2,8012	3,4492		6,2163	5,6759	6,1691
TOP SET	4,9428	3,8925	4,9279	4,0643	4,9112		11,032	8,5266	10,964

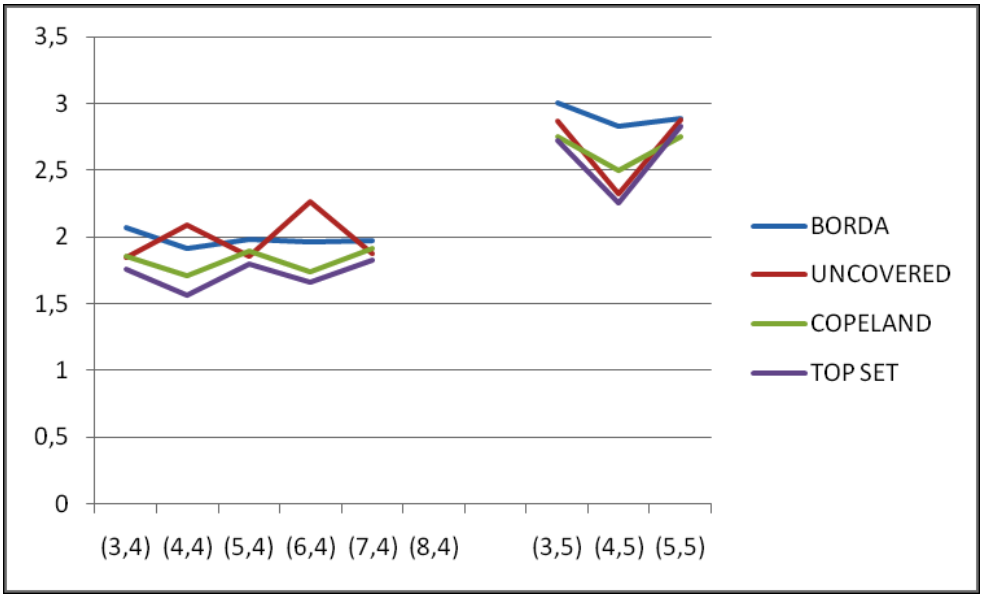


Gain from Manipulation, Lexicographic Extension rule, Protocol 1

As we see from the graph, like in cost of manipulation, the Borda rule is almost smooth. The other three SCCs are waving, but all SCCs are almost same and their gain levels of each SCCs compared to each other do not change between $m = 4$ and $m = 5$, this is because of the averaging property of the Lexicographic extension rule and Protocol 1.

Next table and the corresponding graph shown below shows the results of the gains from manipulation of four SCCs under the Max-Min ordering extension rule.

GAIN, MAX-MIN ORDERING, PROTOCOL 1									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	2,0666	1,9108	1,9815	1,9587	1,9741		3,0063	2,8269	2,8871
UNCOVERED	1,8421	2,0928	1,8554	2,2629	1,8734		2,8703	2,3239	2,8797
COPELAND	1,8593	1,7064	1,8914	1,7382	1,9144		2,7486	2,5007	2,7562
TOP SET	1,7538	1,5615	1,7941	1,665	1,8236		2,7242	2,2552	2,8301

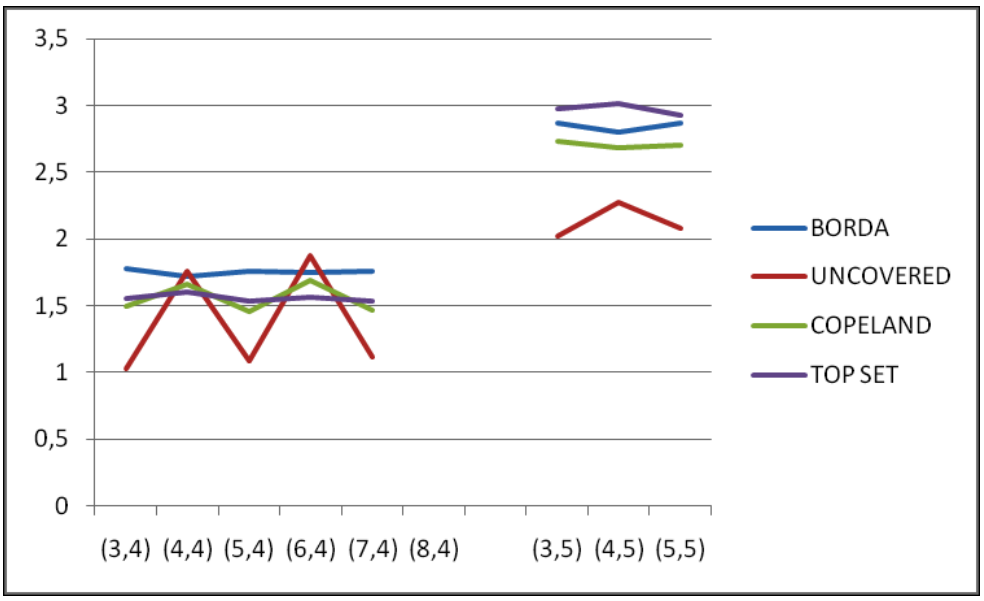


Gain from Manipulation, Max-Min ordering, Protocol 1

The Copeland rule and the Top Set wave at same degree. The Borda rule is smooth as usual. The positions of all SCCs compared to each other does not change between $m = 4$ and $m = 5$.

Next ones show the results of the gains from manipulation of four SCCs under the Expected-Ranking extension rule.

GAIN, EXPECTED-RANKING, PROTOCOL 1									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	1,7832	1,7211	1,7584	1,7516	1,7633		2,8741	2,8027	2,8731
UNCOVERED	1,0329	1,7588	1,0855	1,8795	1,1115		2,0175	2,2706	2,0811
COPELAND	1,4952	1,6648	1,4599	1,6885	1,4645		2,7311	2,6834	2,7003
TOP SET	1,5517	1,5996	1,5378	1,5689	1,5311		2,979	3,0191	2,9297

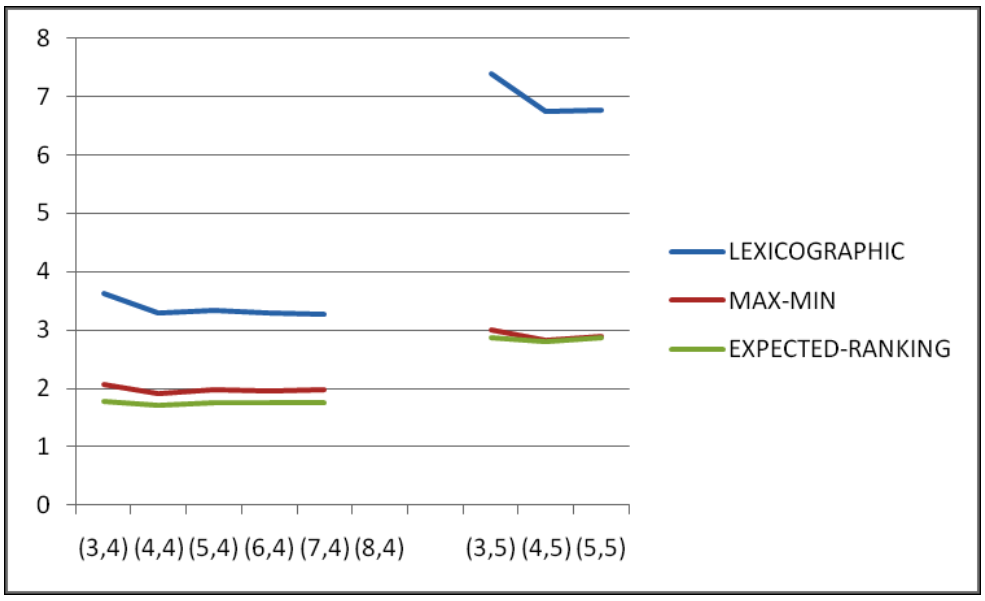


Gain from Manipulation, Expected-Ranking, Protocol 1

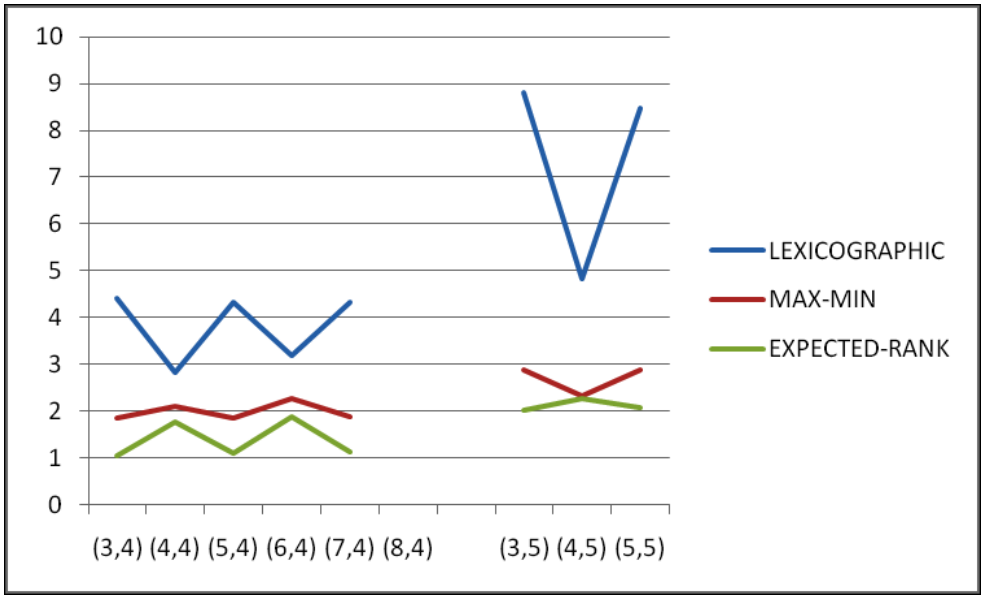
Since Expected-Ranking takes the average values, all SCCs, except the Uncovered set, are smooth. Wavings of the Uncovered set again must be due to the odd numbered outcomes.

COMPARISON OF THE EXTENSION RULES

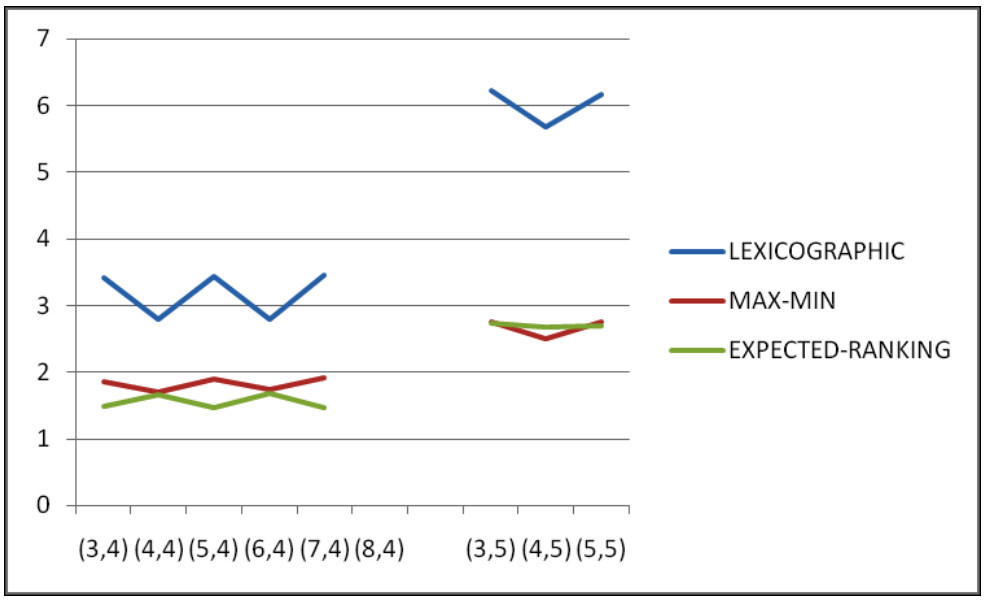
The following graphs show the behaviours of all SCCs under three extension rules.



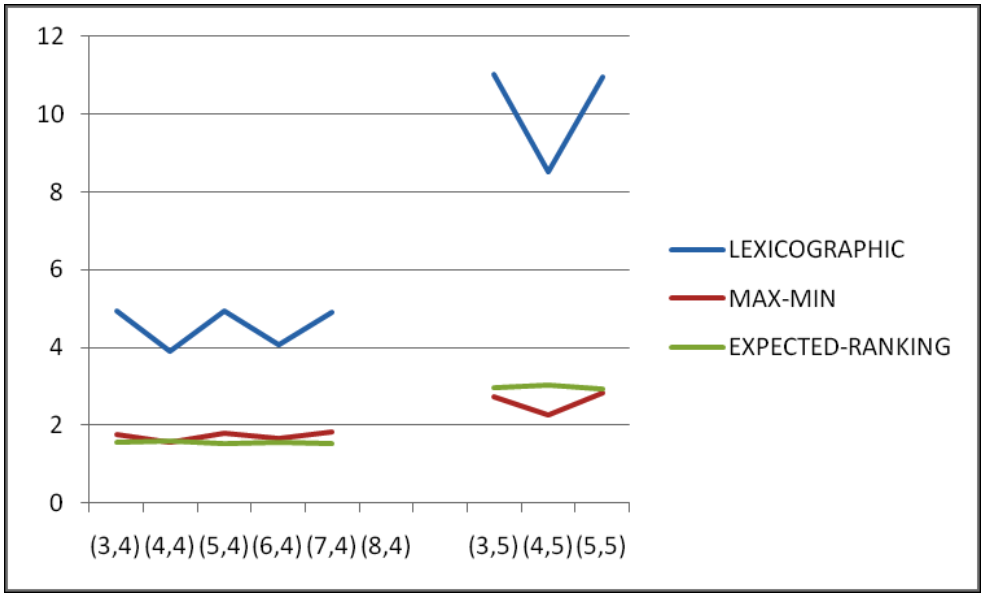
Borda, Gain from Manipulation, Protocol 1



Uncovered, Gain from Manipulation, Protocol 1



Copeland, Gain from Manipulation, Protocol 1



Top Set, Gain from Manipulation, Protocol 1

As mentioned previously, we also calculated the possible maximum gain on the full domain for all SCCs and extension rules. The following tables show the

maximum gains.

LEXICOGRAPHIC, MAXIMUM GAIN ON FULL DOMAIN									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	10/15	10/15	10/15	10/15	10/15		25/31	25/31	25/31
UNCOVERED	6/15	11/15	6/15	11/15	6/15		24/31	26/31	24/31
COPELAND	7/15	10/15	7/15	10/15	7/15		24/31	25/31	24/31
TOP SET	9/15	9/15	9/15	9/15	9/15		24/31	24/31	24/31

MAX-MIN ORDERING, MAXIMUM GAIN ON FULL DOMAIN									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	5/10	5/10	5/10	5/10	5/10		9/15	9/15	9/15
UNCOVERED	4/10	6/10	4/10	6/10	4/10		8/15	10/15	8/15
COPELAND	5/10	5/10	5/10	5/10	5/10		9/15	9/15	9/15
TOP SET	4/10	4/10	4/10	4/10	4/10		8/15	8/15	8/15

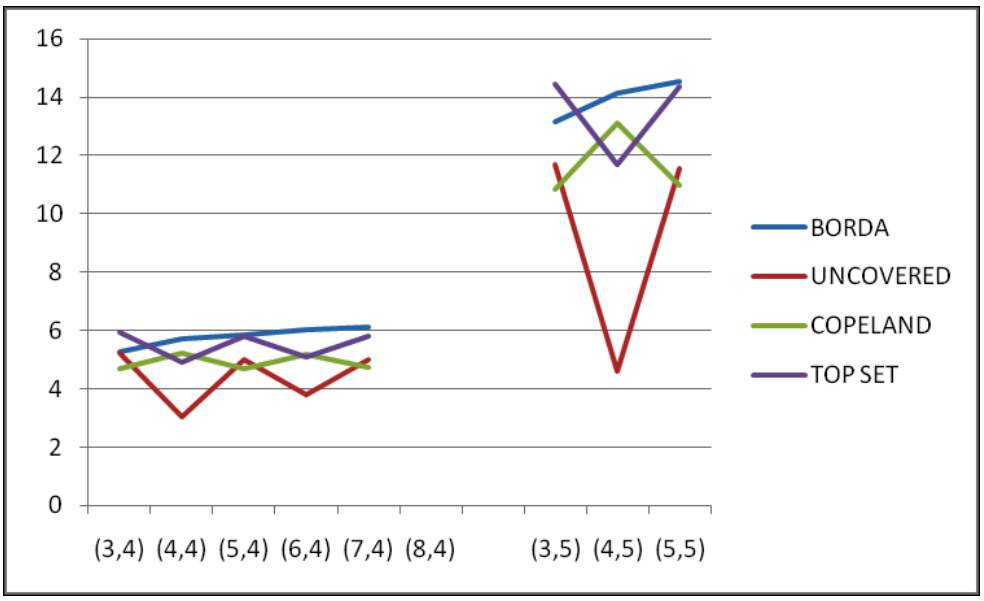
EXPECTED-RANKING, MAXIMUM GAIN ON FULL DOMAIN									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	4/9	4/9	4/9	4/9	4/9		10/15	10/15	10/15
UNCOVERED	3/9	5/9	3/9	5/9	3/9		6/15	10/15	6/15
COPELAND	4/9	4/9	4/9	4/9	4/9		10/15	10/15	10/15
TOP SET	2/9	2/9	2/9	2/9	2/9		5/15	5/15	5/15

As seen from the graphs and the tables, the Max-Min ordering and the Expected-Ranking extension rules provide the same level of gains for each SCC, but the Lexicographic extension rule provides much more gains to the SCCs. This is because the range of Lexicographic rule is almost double of the other two. Hence, whatever SCC in our list we use, the Lexicographic extension rule is almost two times beneficial than the Max-Min ordering and the Expected-Ranking extension rules for the agents manipulating.

4.2.2 PROTOCOL 2

The table and the corresponding graph shown below shows the results of the gains from manipulation of four SCCs under the Lexicographic extension rule.

GAIN, LEXICOGRAPHIC, PROTOCOL 2									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	5,2908	5,7263	5,8446	6,0196	6,1203		13,1481	14,1196	14,5542
UNCOVERED	5,2235	3,062	5,0011	3,8116	5,0179		11,6709	4,6002	11,5469
COPELAND	4,7096	5,237	4,7027	5,1964	4,7306		10,8298	13,1162	10,988
TOP SET	5,9277	4,8956	5,7991	5,0711	5,8034		14,4584	11,7018	14,3531

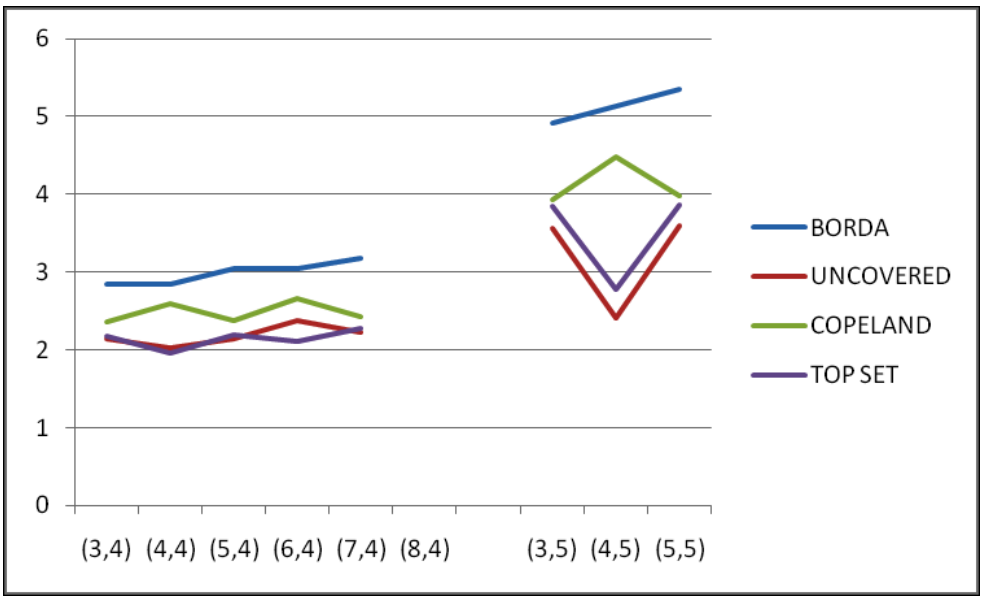


Gain from Manipulation, Lexicographic Extension rule, Protocol 2

Except the Uncovered set's wavings, all SCCs are closed to each other which means their maximum gains they can provide are averagely almost same.

Next table and the corresponding graph shown below shows the results of the gains from manipulation of four SCCs under the Max-Min ordering extension rule.

GAIN, MAX-MIN ORDERING, PROTOCOL 2									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	2,8418	2,844	3,0465	3,0467	3,1737		4,9218	5,1288	5,3489
UNCOVERED	2,1411	2,0266	2,1451	2,3701	2,2295		3,5617	2,4146	3,5891
COPELAND	2,3548	2,5984	2,3777	2,659	2,4295		3,9233	4,4884	3,9828
TOP SET	2,1842	1,958	2,1905	2,1163	2,2746		3,8399	2,7771	3,8661

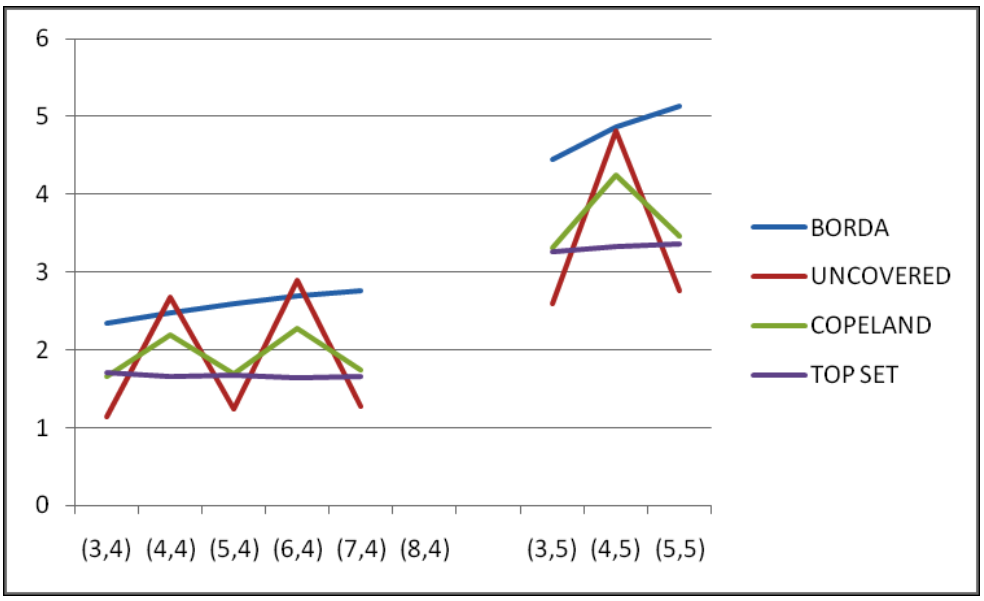


Gain from Manipulation, Max-Min ordering, Protocol 2

For $m = 4$, even there are little differences, all SCCs are smooth and closed to each other. The waves of the Uncovered set and the Top Set are probably due to the differences of number of the alternatives in outcomes of votings.

Next ones show the results of the gains from manipulation of four SCCs under the Expected-Ranking extension rule.

GAIN, EXPECTED-RANKING, PROTOCOL 2									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	2,3463	2,4791	2,5898	2,6869	2,7688		4,4524	4,8667	5,1374
UNCOVERED	1,1395	2,6845	1,2464	2,8963	1,2801		2,602	4,8221	2,7597
COPELAND	1,6575	2,192	1,6886	2,282	1,745		3,3153	4,2424	3,4702
TOP SET	1,7173	1,66	1,678	1,636	1,6594		3,2576	3,3371	3,3656

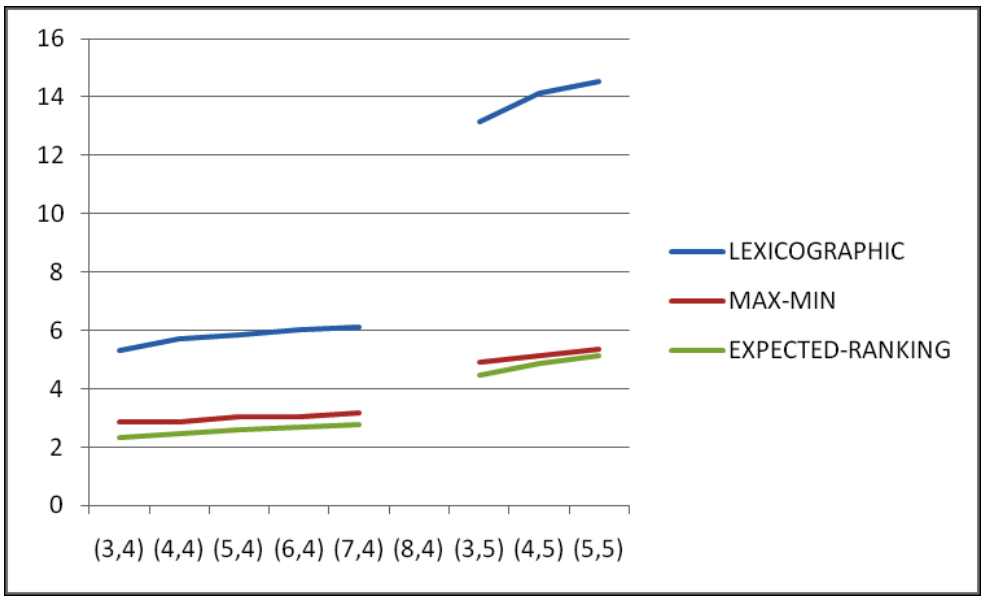


Gain from Manipulation, Expected-Ranking, Protocol 2

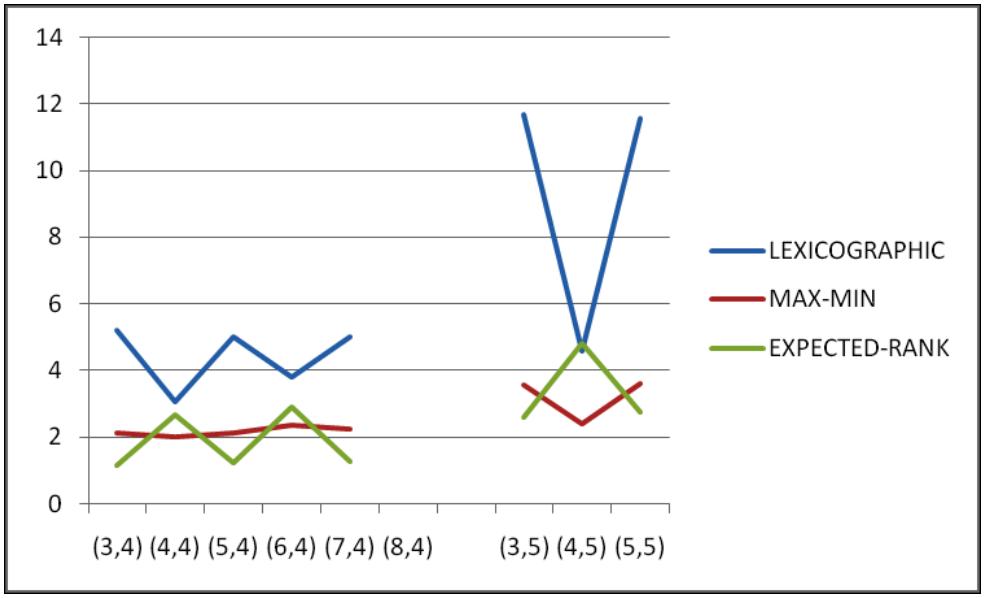
The Borda rule is smooth and the Uncovered set is waving as usual. The smoothness of the Top Set probably comes from the averageness property of the Expected-Ranking rule, as previously seen. The reason of waving of the Copeland rule must be the thresholds to manipulate it for some pairs (n, m).

COMPARISON OF THE EXTENSION RULES

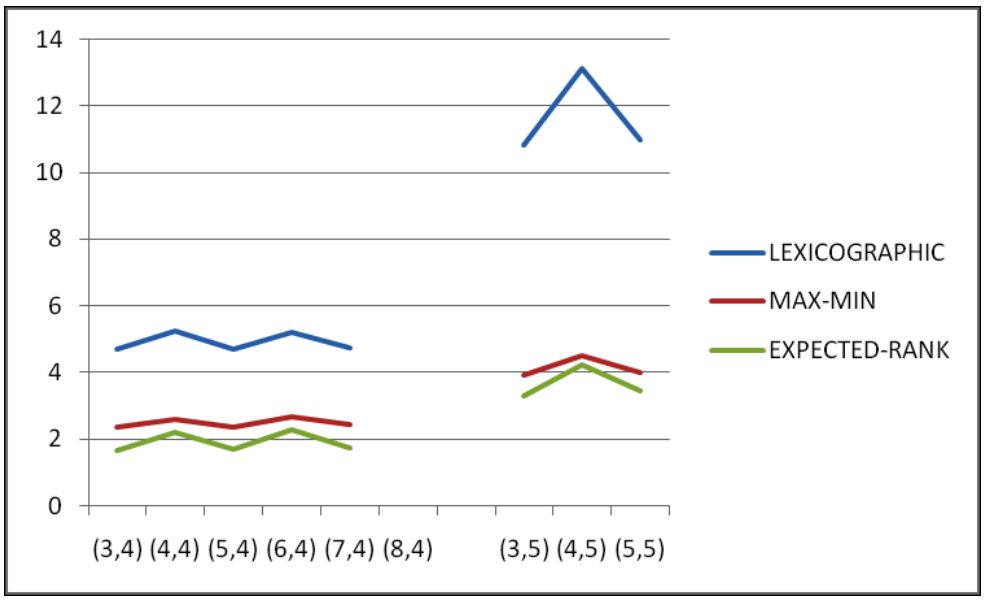
The following graphs show the behaviours of all SCCs under three extension rules.



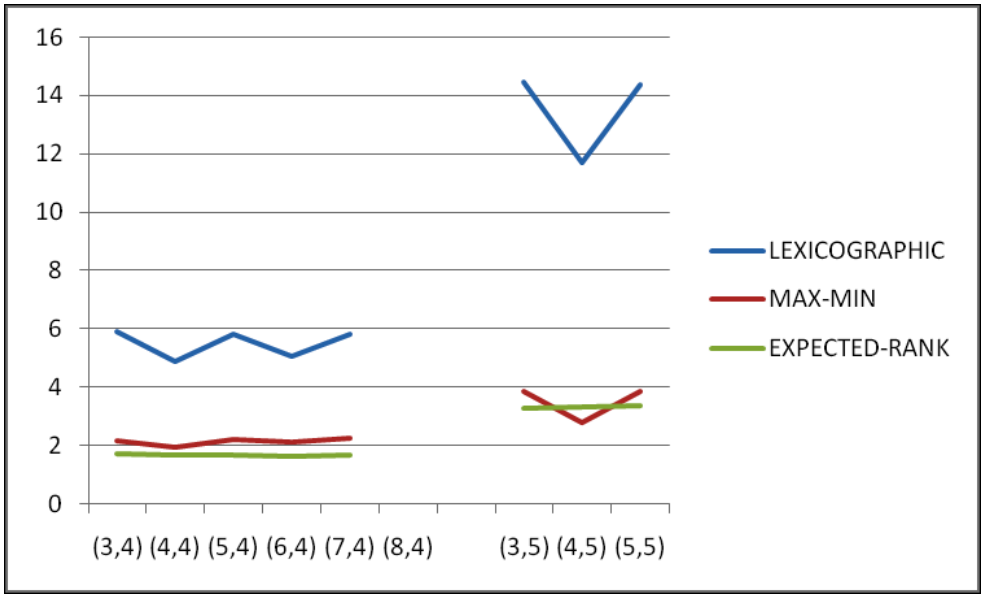
Borda, Gain from Manipulation, Protocol 2



Uncovered, Gain from Manipulation, Protocol 2



Copeland, Gain from Manipulation, Protocol 2



Top Set, Gain from Manipulation, Protocol 2

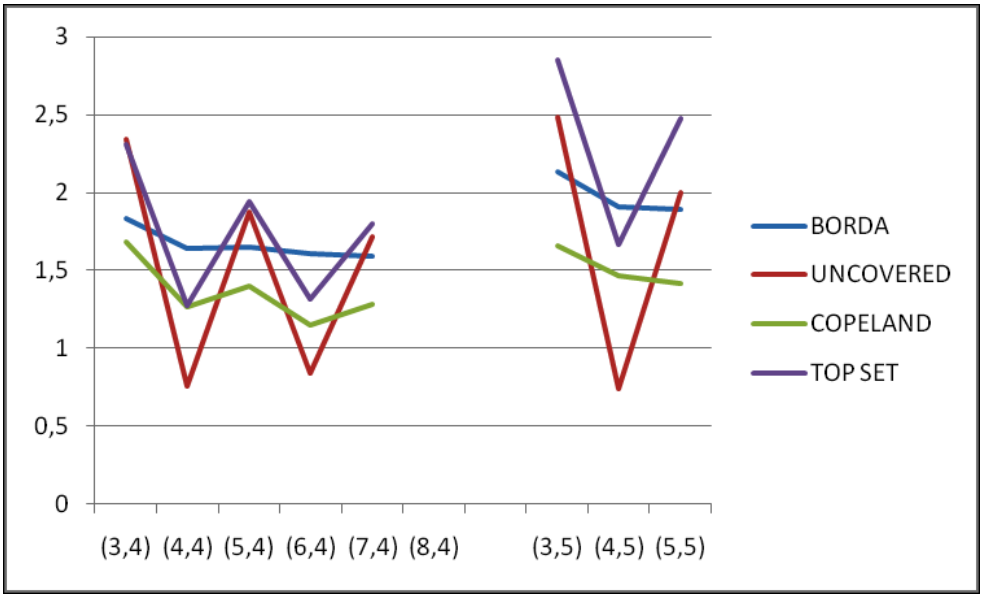
From the graphs above and the tables about the maximum gains on full domain in previous section,the Max-Min ordering and the Expected-Ranking

provide same level gains. On the other hand, the Lexicographic extension rule, like in Protocol 1, provides much more gains probably due to the wide range of the rankings.

4.3 EFFICIENCY OF MANIPULATION

The table and the corresponding graph shown below shows the results of the efficiency of manipulation of four SCCs under the Lexicographic extension rule.

EFFICIENCY (GAIN/COST), LEXICOGRAPHIC									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	1,8355	1,6445	1,6496	1,6071	1,5942		2,1311	1,9072	1,8949
UNCOVERED	2,3391	0,7533	1,8759	0,8367	1,7143		2,4866	0,7421	2,0026
COPELAND	1,6811	1,2636	1,3962	1,1458	1,2825		1,6608	1,4664	1,4151
TOP SET	2,3071	1,2752	1,9386	1,3171	1,7992		2,8521	1,6677	2,4766

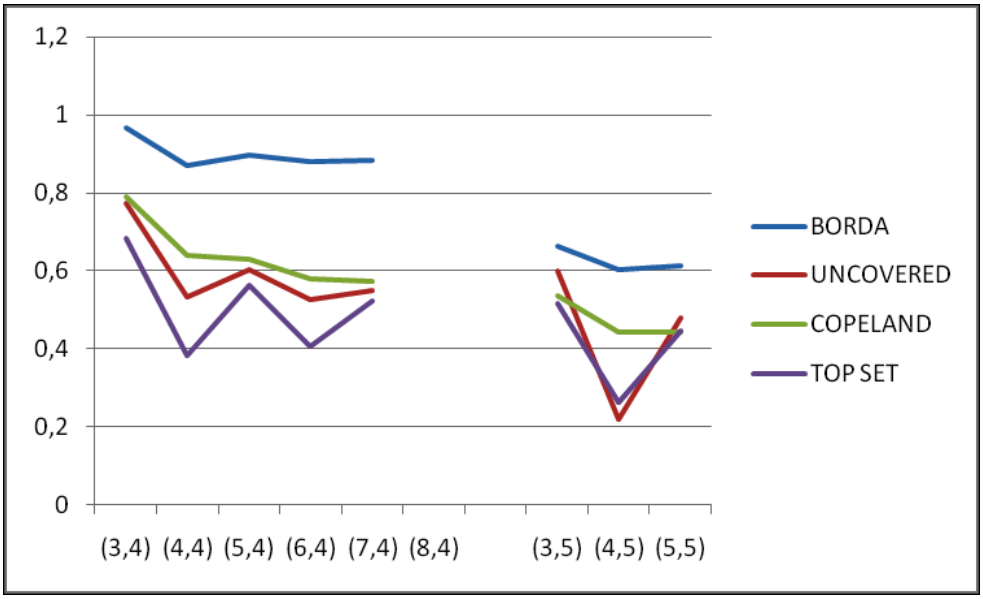


Efficiency of Manipulation, Lexicographic Extension rule

Again, the Copeland and the Borda rules are almost smooth. The wavings of the Uncovered set and the Top Set probably stem from the cardinalities of the outcome sets.

Next table and the corresponding graph shown below shows the results of the efficiency of manipulation of four SCCs under the Max-Min ordering extension rule.

EFFICIENCY (GAIN/COST), MAX-MIN ORDERING									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	0,9666	0,8702	0,8965	0,8805	0,8827		0,6623	0,604	0,6119
UNCOVERED	0,7719	0,5337	0,6046	0,5258	0,5499		0,6007	0,2192	0,4801
COPELAND	0,789	0,641	0,6296	0,5811	0,5731		0,5348	0,4411	0,4418
TOP SET	0,6846	0,3835	0,5617	0,4047	0,5217		0,5153	0,2627	0,4475

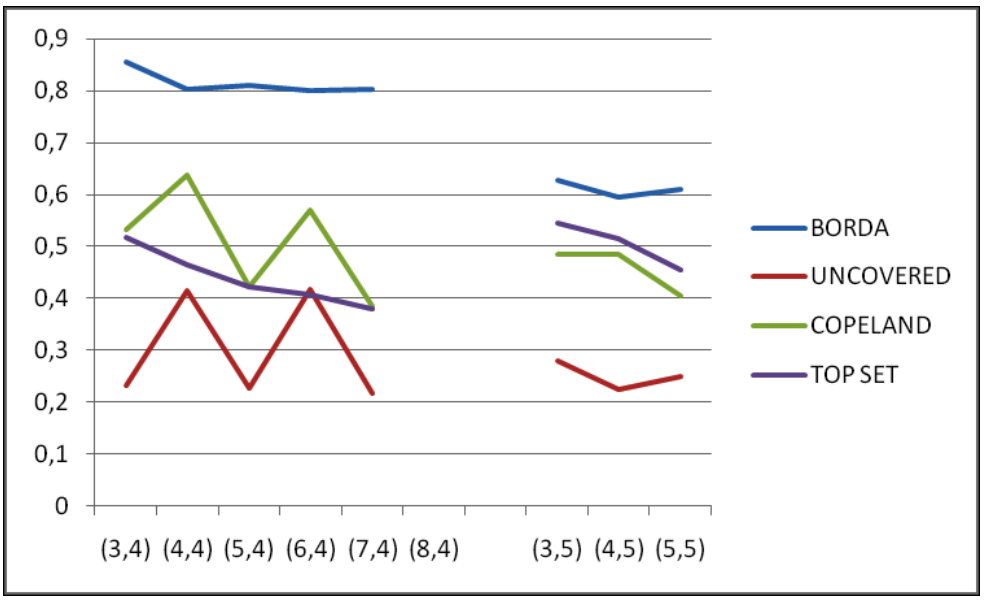


Efficiency of Manipulation, Max-Min ordering

The Uncovered set, the Top Set and the Copeland rule are almost same for both $m = 4$ and $m = 5$. The Borda rule is more smooth and higher than the other three SCCs.

Next ones show the results of the efficiency of manipulation of four SCCs under the Expected-Ranking extension rule.

EFFICIENCY (GAIN/COST), EXPECTED-RANKING									
RULES	**(3,4)**	**(4,4)**	**(5,4)**	**(6,4)**	**(7,4)**	**(8,4)**	**(3,5)**	**(4,5)**	**(5,5)**
BORDA	0,8555	0,802	0,8113	0,8013	0,8026		0,628	0,596	0,6103
UNCOVERED	0,2307	0,4156	0,2261	0,416	0,2179		0,2795	0,2247	0,2505
COPELAND	0,5333	0,6368	0,4217	0,5709	0,3849		0,4845	0,4854	0,4044
TOP SET	0,5172	0,4649	0,4227	0,4066	0,3795		0,5456	0,5137	0,4543

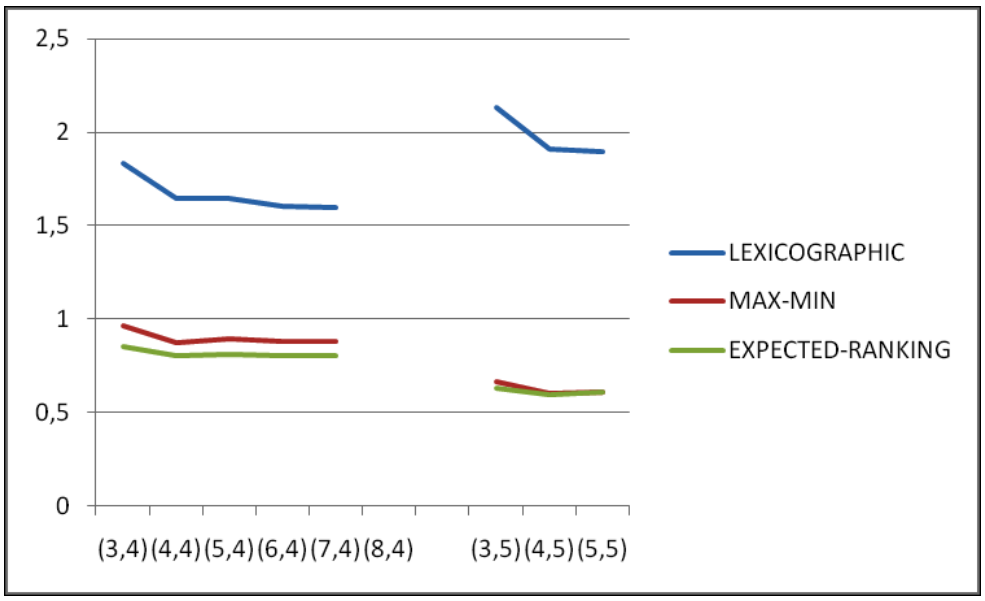


Efficiency of Manipulation, Expected-Ranking

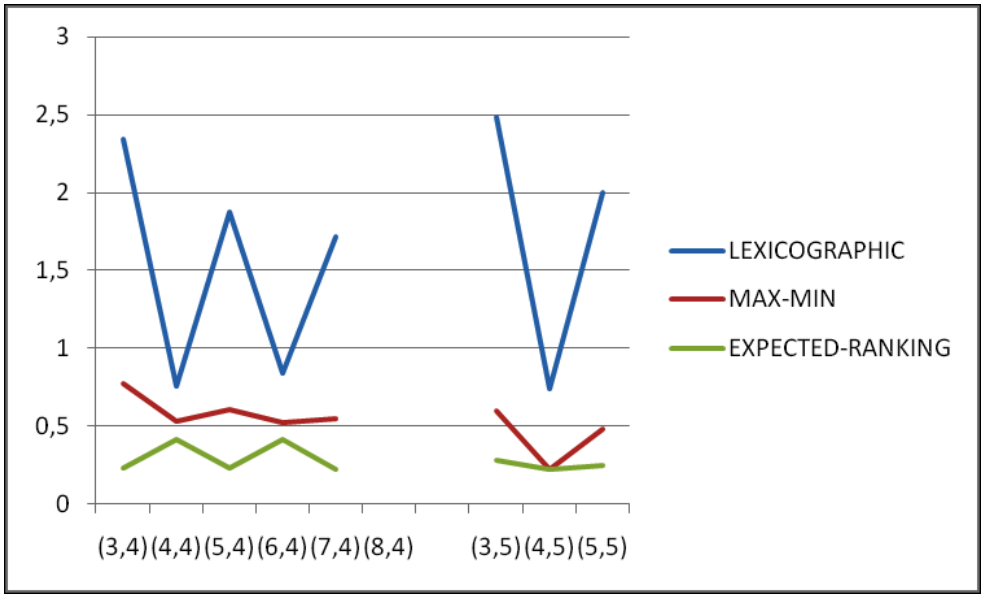
For $m = 5$, SCCs are almost smooth due to the averageness property of the Expected-Ranking rule. The Top Set is smooth for $m = 4$ as previosly seen in the Expected-Ranking rules. It can be estimated that SCCs become smoother as the number alternatives increases.

COMPARISON OF THE EXTENSION RULES

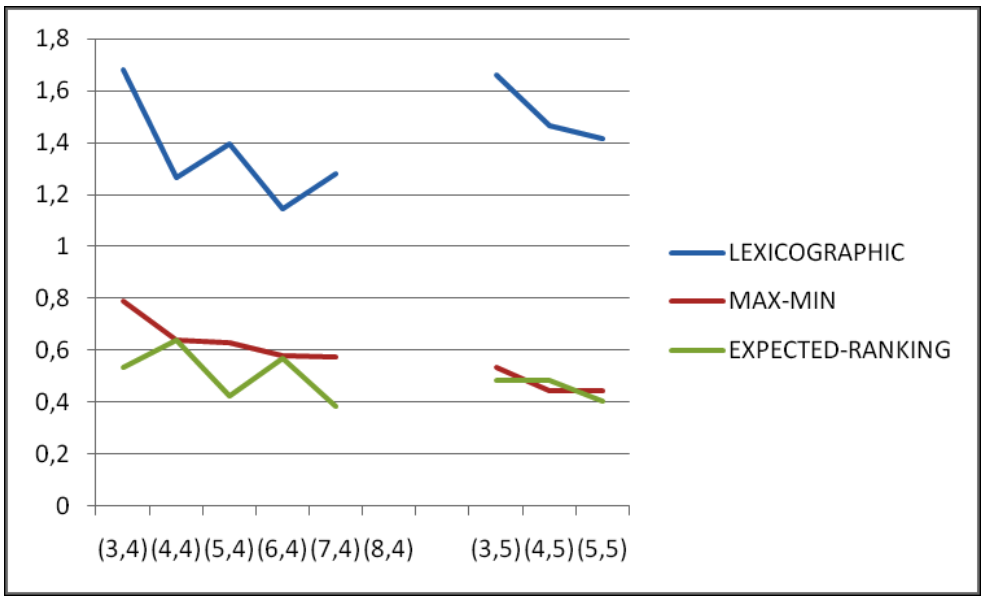
The following graphs show the behaviours of all SCCs under three extension rules.



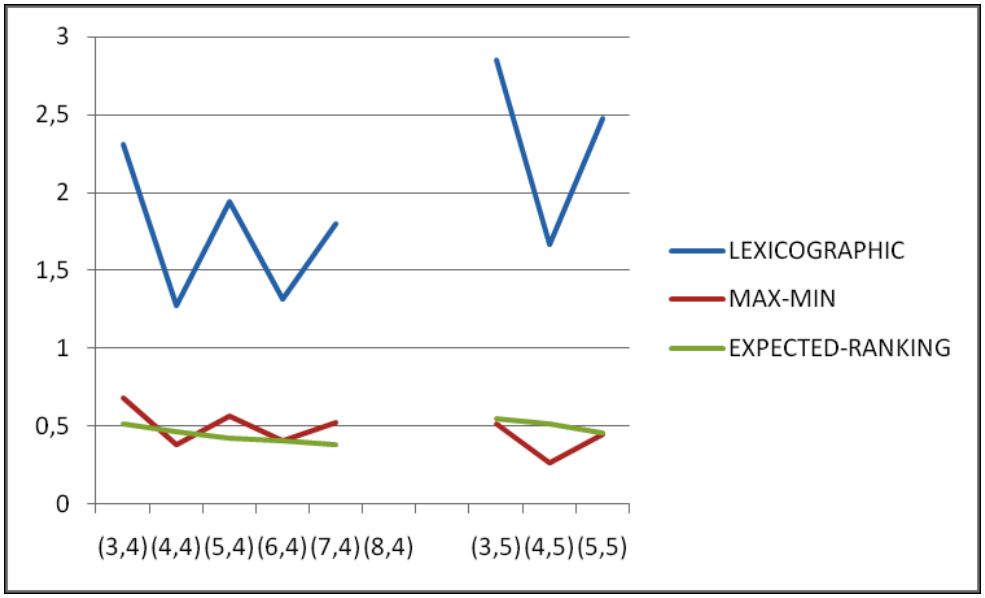
Borda, Efficiency of Manipulation



Uncovered, Efficiency of Manipulation



Copeland, Efficiency of Manipulation



Top Set, Efficiency of Manipulation

As expected, the graphs above show that the Lexicographic extension rule is much more efficient than other extension rules for an agent who is manipulating.

And, also, again the efficiencies of the Max-Min ordering and the Expected-Ranking rules are almost same.

5 GENERAL CONCLUSIONS

We worked on the degree of manipulation of four social choice correspondences; the Borda rule, the Uncovered set, the Copeland rule and the Top Set, under three extension rules; the Lexicographic, the Max-Min ordering and the Expected-Ranking extension rules. We measured the costs, the gains and the efficiency of manipulation for each SCCs under all three extension rules. The general results of our computations are shown below.

COST OF MANIPULATION

Protocol 1: For all extension rules, the Borda rule has the minimal cost. The Uncovered set, the Copeland rule and the Top Set have almost same cost levels and higher than the Borda rule.

In view of the extension rules, all SCCs have same level of costs for $m = 4$. For $m = 5$, the Top Set and the Copeland rule is also same for all extension rules, but the Uncovered set and the Borda rule have lower costs under the Lexicographic extension rule; they have same and higher cost levels in the Max-Min ordering and the Expected-Ranking extension rule compared to the Lexicographic extension rule.

Protocol 2: Excepts the Uncovered set's wavings, all SCCs have same level of costs under all extension rules.

For the extension rules, all of them are same for $m = 4$. For $m = 5$, the SCCs have higher cost levels under the Lexicographic rule, but under the Max-Min ordering and the Expected-Ranking they have same and less cost levels compared to the Lexicographic rule.

GAIN FROM MANIPULATION

Protocol 1: All SCCs provide same level of gains for both $m = 4$ and $m = 5$ under each extension rule.

For both $m = 4$ and $m = 5$, the Max-Min ordering and the Expected-Ranking provide same level of gains, but the Lexicographic extension rule gives more gains than the other two extension rules for all SCCs.

Protocol 2: Under each extension rule, for $m = 4$, all rules can give same amout of increase in rankings. For $m = 5$, there are wavings for all SCCs under all extension rules. Hence, the comparisons of the degree of manipulabilities of SCCs depend on the pairs (n, m).

For both $m = 4$ and $m = 5$, the Max-Min ordering and the Expected-Ranking provide same level of gains, but the Lexicographic extension rule gives more gains than the other two extension rules for all SCCs.

EFFICIENCY OF MANIPULATION

Under the Lexicographic extension rule, all SCCs have same efficiency. Under the Max-Min ordering and the Expected-Ranking, the Borda rule seems more efficient than other SCCs; the Uncovered set, the Copeland rule and the Top Set, which have equivalent efficiencies.

Again, for both $m = 4$ and $m = 5$, the Max-Min ordering and the Expected-Ranking provide same level of gains, but the Lexicographic extension rule gives more gains than the other two extension rules for all SCCs.

References

[1] Aleskerov, F. and Kurbanov, E. (1998) "A Degree and an Efficiency of Manipulation of known Social Choice Rules", ISS/EC-1998-01 Boğaziçi University Research Papers.

[2] Arrow, K. (1951) Social Choice and Individual Values, New York: Wiley.

[3] Barbera, S., Bossert, W., and Pattanaik, K. (2001) "Ranking sets of objects", Handbook of Utility Theory.

[4] Campbell, D. and Kelly, J. (2009) "Gains from manipulating social choice rules" Economic Theory 40: 349-371.

[5] Gibbard, A. (1973) "Manipulation of Voting Schemes: a general result" Econometrica v.41.

[6] Kelly, J. (1993) "Almost all social choice rules are highly manipulable, but few aren't" Social Choice and Welfare 10: 161-175.

[7] Moulin, H. (1986) "Choosing from a tournament" Social Choice and Welfare 3: 271-291.

[8] Peris, J. and Subiza, B. (1999) "Condorcet choice correpondences for weak tournaments" Social Choice and Welfare 16: 217-231.

[9] Rodríguez-Álvarez, C. (2007) "On the manipulation of social choice correspondences" Social Choice and Welfare 29: 175-199.

[10] Satterthwaite, M. A. (1975) "Strategy-proofness and Arrow's conditions: Existence and Correpondence Theorems for Voting Procedures and Social Welfare Functions" Journal of Economic Theory 10: 187-217.

[11] Smith, D. (1999) "Manipulability Measures of Common Social Choice Functions" Social Choice and Welfare 16: 639-661.

Appendix

Profile (main class)

```
import java.util.*;
class Profile
{
static LinkedList<String> comb(String s)
{
     LinkedList<String> retval=new LinkedList<String>();
if (s.length()==1)
{
     retval.add(s);
     return retval;
}
for (int i=0;i<s.length();i++)
{
     String prep=s.substring(i,i+1);
     String sub=s.substring(0,i)+s.substring(i+1);
     LinkedList<String> subcomb=comb(sub);
     for (int j=0;j<subcomb.size();j++)
          retval.add(prep+subcomb.get(j));
}
return retval;
}
public static void main(String[] args)
{
     LinkedList<String> combs=comb("abcd");

     Uncovered winner = new Uncovered();
     Kemeny measure= new Kemeny ();

     String [] M = new String [7];

     String x="";
     String y="";
     String newx="";
     String newy="";
     double min=0;

     double count=0;
     double distance=0;

     for( int i=0; i<= 23; i ++)
     {
```

```
for (int j=0; j<= 23; j ++)
{
      for (int k=0; k<=23; k ++)
      {
            for (int l=0; l<=23; l ++)
      {
      for (int m=0; m<=23; m ++)
      {
      for (int u=0; u<=23; u ++)
      {
      for (int v=0; v<=23; v ++)
      {
            M [0]= combs.get(i);
            M [1]= combs.get(j);
            M [2]= combs.get(k);
            M [3]= combs.get(l);
            M [4]= combs.get(m);
            M [5]= combs.get(u);
            M [6]= combs.get(v);

            x=winner.GetWinner(M);
            min=1000;

            for(int z=0; z<=23; z ++)
            {
                  M [0]= combs.get(z);
                  M [1]= combs.get(j);
                  M [2]= combs.get(k);
                  M [3]= combs.get(l);
                  M [4]= combs.get(m);
                  M [5]= combs.get(u);
                  M [6]= combs.get(v);

                  newx="";
               newy="";

                  if((combs.get(z)!=combs.get(i)))
                  {
                        y=winner.GetWinner(M);

                        for(int h=0; h<=3; h ++)
                        {
                              if((x.indexOf((combs.get(i)).charAt(h)))>=0)
                              {
                                    newx=newx.concat(((combs.get(i)).substring(h,h+1)));
                              }
```

```
        if((y.indexOf((combs.get(i)).charAt(h)))>=0)
        {
            newy=newy.concat(((combs.get(i)).substring(h,h+1)));
        }
    }

    if((newx.length())==(newy.length()))
    {
        int o=0;

        do
        {
            if((newx.charAt(o))!=(newy.charAt(o)))
            {
                    if(((combs.get(i)).indexOf(newx.charAt(o)))>((combs.get(i))
            {
                if((measure.GetDistance((combs.get(i)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(i)),(combs.get(z))));
                }
            }
            }

            o=o+1;
        }
        while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
    }

    if((newx.length())<(newy.length()))
    {
        int r=0;

        do
        {
            if((newx.charAt(r))!=(newy.charAt(r)))
            {
            if(((combs.get(i)).indexOf(newx.charAt(r)))>((combs.get(i)).indexOf(
            {
                if((measure.GetDistance((combs.get(i)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(i)),(combs.get(z))));
                }
            }
            }
```

```
                    r=r+1;
                }
                while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
            }

            if((newx.length())>(newy.length()))
            {
                int p=0;

                do
                {
                    if((newx.charAt(p))!=(newy.charAt(p)))
                    {
                    if(((combs.get(i)).indexOf(newx.charAt(p)))>((combs.get(i)).indexOf(
                    {
                        if((measure.GetDistance((combs.get(i)),(combs.get(z))))<min)
                        {
                            min=(measure.GetDistance((combs.get(i)),(combs.get(z))));
                        }
                    }
                    }

                    p=p+1;
                }
                while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                if(newy.equals(newx.substring(0,newy.length())))
                {
                    if((measure.GetDistance((combs.get(i)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(i)),(combs.get(z))));
                }
                }
            }
        }
        M [0]= combs.get(i);
        M [1]= combs.get(z);
        M [2]= combs.get(k);
        M [3]= combs.get(l);
        M [4]= combs.get(m);
        M [5]= combs.get(u);
        M [6]= combs.get(v);

        newx="";
```

```
newy="";

  if((combs.get(z)!=combs.get(j)))
  {
        y=winner.GetWinner(M);

        for(int h=0; h<=3; h ++)
        {
              if((x.indexOf((combs.get(j)).charAt(h)))>=0)
              {
                    newx=newx.concat(((combs.get(j)).substring(h,h+1)));
              }
              if((y.indexOf((combs.get(j)).charAt(h)))>=0)
              {
                    newy=newy.concat(((combs.get(j)).substring(h,h+1)));
              }
        }

        if((newx.length())==(newy.length()))
        {
              int o=0;

              do
              {
                    if((newx.charAt(o))!=(newy.charAt(o)))
                    {
                    if(((combs.get(j)).indexOf(newx.charAt(o)))>((combs.get(j)).indexOf(
                    {
                          if((measure.GetDistance((combs.get(j)),(combs.get(z))))<min)
                          {
                                min=(measure.GetDistance((combs.get(j)),(combs.get(z))));
                          }
                    }
                    }

                    o=o+1;
              }
              while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
        }

        if((newx.length())<(newy.length()))
        {
              int r=0;

              do
```

```
                {
                    if((newx.charAt(r))!=(newy.charAt(r)))
                    {
                    if(((combs.get(j)).indexOf(newx.charAt(r)))>((combs.get(j)).indexOf(
                    {
                        if((measure.GetDistance((combs.get(j)),(combs.get(z))))<min)
                        {
                            min=(measure.GetDistance((combs.get(j)),(combs.get(z))));
                        }
                    }
                    }

                    r=r+1;
                }
                while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
            }

            if((newx.length())>(newy.length()))
            {
                int p=0;

                do
                {
                    if((newx.charAt(p))!=(newy.charAt(p)))
                    {
                    if(((combs.get(j)).indexOf(newx.charAt(p)))>((combs.get(j)).indexOf(
                    {
                        if((measure.GetDistance((combs.get(j)),(combs.get(z))))<min)
                        {
                            min=(measure.GetDistance((combs.get(j)),(combs.get(z))));
                        }
                    }
                    }

                    p=p+1;
                }
                while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                if(newy.equals(newx.substring(0,newy.length())))
                {
                    if((measure.GetDistance((combs.get(j)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(j)),(combs.get(z))));
                }
```

```
            }
        }
    }
    M [0]= combs.get(i);
    M [1]= combs.get(j);
    M [2]= combs.get(z);
    M [3]= combs.get(l);
    M [4]= combs.get(m);
    M [5]= combs.get(u);
    M [6]= combs.get(v);

    newx="";
  newy="";

    if((combs.get(z)!=combs.get(k)))
    {
        y=winner.GetWinner(M);

        for(int h=0; h<=3; h ++)
        {
            if((x.indexOf((combs.get(k)).charAt(h)))>=0)
            {
                newx=newx.concat(((combs.get(k)).substring(h,h+1)));
            }
            if((y.indexOf((combs.get(k)).charAt(h)))>=0)
            {
                newy=newy.concat(((combs.get(k)).substring(h,h+1)));
            }
        }

        if((newx.length())==(newy.length()))
        {
            int o=0;

            do
            {
                if((newx.charAt(o))!=(newy.charAt(o)))
                {
                if(((combs.get(k)).indexOf(newx.charAt(o)))>((combs.get(k)).indexO
                {
                    if((measure.GetDistance((combs.get(k)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(k)),(combs.get(z))));
                    }
                }
                }
```

```
                o=o+1;
            }
            while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
        }

        if((newx.length())<(newy.length()))
        {
            int r=0;

            do
            {
                if((newx.charAt(r))!=(newy.charAt(r)))
                {
                if(((combs.get(k)).indexOf(newx.charAt(r)))>((combs.get(k)).indexOf
                {
                    if((measure.GetDistance((combs.get(k)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(k)),(combs.get(z))));
                    }
                }
                }

                r=r+1;
            }
            while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
        }

        if((newx.length())>(newy.length()))
        {
            int p=0;

            do
            {
                if((newx.charAt(p))!=(newy.charAt(p)))
                {
                if(((combs.get(k)).indexOf(newx.charAt(p)))>((combs.get(k)).indexO
                {
                    if((measure.GetDistance((combs.get(k)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(k)),(combs.get(z))));
                    }
                }
                }
```

```
                    p=p+1;
                }
                while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                if(newy.equals(newx.substring(0,newy.length())))
                {
                    if((measure.GetDistance((combs.get(k)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(k)),(combs.get(z))));
                }
                }
            }
        }
        M [0]= combs.get(i);
        M [1]= combs.get(j);
        M [2]= combs.get(k);
        M [3]= combs.get(z);
        M [4]= combs.get(m);
        M [5]= combs.get(u);
        M [6]= combs.get(v);

        newx="";
      newy="";

        if((combs.get(z)!=combs.get(l)))
        {
            y=winner.GetWinner(M);

            for(int h=0; h<=3; h ++)
            {
                if((x.indexOf((combs.get(l)).charAt(h)))>=0)
                {
                    newx=newx.concat(((combs.get(l)).substring(h,h+1)));
                }
                if((y.indexOf((combs.get(l)).charAt(h)))>=0)
                {
                    newy=newy.concat(((combs.get(l)).substring(h,h+1)));
                }
            }

            if((newx.length())==(newy.length()))
            {
                int o=0;
```

```
                    do
                    {
                        if((newx.charAt(o))!=(newy.charAt(o)))
                        {
                        if(((combs.get(l)).indexOf(newx.charAt(o)))>((combs.get(l)).indexOf(
                        {
                            if((measure.GetDistance((combs.get(l)),(combs.get(z))))<min)
                            {
                                min=(measure.GetDistance((combs.get(l)),(combs.get(z))));
                            }
                        }
                        }

                        o=o+1;
                    }
                    while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
                }

                if((newx.length())<(newy.length()))
                {
                    int r=0;

                    do
                    {
                        if((newx.charAt(r))!=(newy.charAt(r)))
                        {
                        if(((combs.get(l)).indexOf(newx.charAt(r)))>((combs.get(l)).indexOf(
                        {
                            if((measure.GetDistance((combs.get(l)),(combs.get(z))))<min)
                            {
                                min=(measure.GetDistance((combs.get(l)),(combs.get(z))));
                            }
                        }
                        }

                        r=r+1;
                    }
                    while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
                }

                if((newx.length())>(newy.length()))
                {
                    int p=0;
```

```
                    do
                    {
                        if((newx.charAt(p))!=(newy.charAt(p)))
                        {
                        if(((combs.get(l)).indexOf(newx.charAt(p)))>((combs.get(l)).indexOf(
                        {
                            if((measure.GetDistance((combs.get(l)),(combs.get(z))))<min)
                            {
                                min=(measure.GetDistance((combs.get(l)),(combs.get(z))));
                            }
                        }
                        }

                        p=p+1;
                    }
                    while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                    if(newy.equals(newx.substring(0,newy.length())))
                    {
                        if((measure.GetDistance((combs.get(l)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(l)),(combs.get(z))));
                    }
                    }
                }
            }

            M [0]= combs.get(i);
            M [1]= combs.get(j);
            M [2]= combs.get(k);
            M [3]= combs.get(l);
            M [4]= combs.get(z);
            M [5]= combs.get(u);
            M [6]= combs.get(v);

            newx="";
          newy="";

            if((combs.get(z)!=combs.get(m)))
            {
                y=winner.GetWinner(M);

                for(int h=0; h<=3; h ++)
                {
                    if((x.indexOf((combs.get(m)).charAt(h)))>=0)
```

```
        {
            newx=newx.concat(((combs.get(m)).substring(h,h+1)));
        }
        if((y.indexOf((combs.get(m)).charAt(h)))>=0)
        {
            newy=newy.concat(((combs.get(m)).substring(h,h+1)));
        }
    }

    if((newx.length())==(newy.length()))
    {
        int o=0;

        do
        {
            if((newx.charAt(o))!=(newy.charAt(o)))
            {
            if(((combs.get(m)).indexOf(newx.charAt(o)))>((combs.get(m)).indexO
            {
                if((measure.GetDistance((combs.get(m)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(m)),(combs.get(z))))
                }
            }
            }

            o=o+1;
        }
        while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
    }

    if((newx.length())<(newy.length()))
    {
        int r=0;

        do
        {
            if((newx.charAt(r))!=(newy.charAt(r)))
            {
            if(((combs.get(m)).indexOf(newx.charAt(r)))>((combs.get(m)).indexO
            {
                if((measure.GetDistance((combs.get(m)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(m)),(combs.get(z))))
                }
```

```
                    }
                    }

                    r=r+1;
                }
                while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
            }

            if((newx.length())>(newy.length()))
            {
                int p=0;

                do
                {
                    if((newx.charAt(p))!=(newy.charAt(p)))
                    {
                    if(((combs.get(m)).indexOf(newx.charAt(p)))>((combs.get(m)).indexO
                    {
                        if((measure.GetDistance((combs.get(m)),(combs.get(z))))<min)
                        {
                            min=(measure.GetDistance((combs.get(m)),(combs.get(z))))
                        }
                    }
                    }

                    p=p+1;
                }
                while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                if(newy.equals(newx.substring(0,newy.length())))
                {
                    if((measure.GetDistance((combs.get(m)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(m)),(combs.get(z))));
                }
                }
            }
        }
        M [0]= combs.get(i);
        M [1]= combs.get(j);
        M [2]= combs.get(k);
        M [3]= combs.get(l);
        M [4]= combs.get(m);
        M [5]= combs.get(z);
```

```
M [6]= combs.get(v);

newx="";
newy="";

if((combs.get(z)!=combs.get(u)))
{
    y=winner.GetWinner(M);

    for(int h=0; h<=3; h ++)
    {
        if((x.indexOf((combs.get(u)).charAt(h)))>=0)
        {
            newx=newx.concat(((combs.get(u)).substring(h,h+1)));
        }
        if((y.indexOf((combs.get(u)).charAt(h)))>=0)
        {
            newy=newy.concat(((combs.get(u)).substring(h,h+1)));
        }
    }

    if((newx.length())==(newy.length()))
    {
        int o=0;

        do
        {
            if((newx.charAt(o))!=(newy.charAt(o)))
            {
            if(((combs.get(u)).indexOf(newx.charAt(o)))>((combs.get(u)).indexO
            {
                if((measure.GetDistance((combs.get(u)),(combs.get(z))))<min)
                {
                    min=(measure.GetDistance((combs.get(u)),(combs.get(z))));
                }
            }
            }

            o=o+1;
        }
        while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
    }

    if((newx.length())<(newy.length()))
    {
```

```
                int r=0;

                do
                {
                    if((newx.charAt(r))!=(newy.charAt(r)))
                    {
                    if(((combs.get(u)).indexOf(newx.charAt(r)))>((combs.get(u)).indexOf
                    {
                        if((measure.GetDistance((combs.get(u)),(combs.get(z))))<min)
                        {
                            min=(measure.GetDistance((combs.get(u)),(combs.get(z))));
                        }
                    }
                    }

                    r=r+1;
                }
                while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
            }

            if((newx.length())>(newy.length()))
            {
                int p=0;

                do
                {
                    if((newx.charAt(p))!=(newy.charAt(p)))
                    {
                    if(((combs.get(u)).indexOf(newx.charAt(p)))>((combs.get(u)).indexO
                    {
                        if((measure.GetDistance((combs.get(u)),(combs.get(z))))<min)
                        {
                            min=(measure.GetDistance((combs.get(u)),(combs.get(z))));
                        }
                    }
                    }

                    p=p+1;
                }
                while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                if(newy.equals(newx.substring(0,newy.length())))
                {
                    if((measure.GetDistance((combs.get(u)),(combs.get(z))))<min)
```

```
                {
                    min=(measure.GetDistance((combs.get(u)),(combs.get(z))));
                }
                }
        }
    }
    M [0]= combs.get(i);
    M [1]= combs.get(j);
    M [2]= combs.get(k);
    M [3]= combs.get(l);
    M [4]= combs.get(m);
    M [5]= combs.get(u);
    M [6]= combs.get(z);

    newx="";
  newy="";

    if((combs.get(z)!=combs.get(v)))
    {
        y=winner.GetWinner(M);

        for(int h=0; h<=3; h ++)
        {
            if((x.indexOf((combs.get(v)).charAt(h)))>=0)
            {
                newx=newx.concat(((combs.get(v)).substring(h,h+1)));
            }
            if((y.indexOf((combs.get(v)).charAt(h)))>=0)
            {
                newy=newy.concat(((combs.get(v)).substring(h,h+1)));
            }
        }

        if((newx.length())==(newy.length()))
        {
            int o=0;

            do
            {
                if((newx.charAt(o))!=(newy.charAt(o)))
                {
                if(((combs.get(v)).indexOf(newx.charAt(o)))>((combs.get(v)).indexO
                {
                    if((measure.GetDistance((combs.get(v)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(v)),(combs.get(z))));
```

```
                    }
                }
                }

                o=o+1;
            }
            while((o<(newy.length()))&&((newx.charAt(o-
1))==(newy.charAt(o-1))));
        }

        if((newx.length())<(newy.length()))
        {
            int r=0;

            do
            {
                if((newx.charAt(r))!=(newy.charAt(r)))
                {
                if(((combs.get(v)).indexOf(newx.charAt(r)))>((combs.get(v)).indexOf
                {
                    if((measure.GetDistance((combs.get(v)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(v)),(combs.get(z))));
                    }
                }
                }

                r=r+1;
            }
            while((r<(newx.length()))&&((newx.charAt(r-
1))==(newy.charAt(r-1))));
        }

        if((newx.length())>(newy.length()))
        {
            int p=0;

            do
            {
                if((newx.charAt(p))!=(newy.charAt(p)))
                {
                if(((combs.get(v)).indexOf(newx.charAt(p)))>((combs.get(v)).indexO
                {
                    if((measure.GetDistance((combs.get(v)),(combs.get(z))))<min)
                    {
                        min=(measure.GetDistance((combs.get(v)),(combs.get(z))));
```

```
                                        }
                                    }
                                    }

                                    p=p+1;
                                }
                                while((p<(newy.length()))&&((newx.charAt(p-
1))==(newy.charAt(p-1))));

                                if(newy.equals(newx.substring(0,newy.length())))
                                {
                                    if((measure.GetDistance((combs.get(v)),(combs.get(z))))<min)
                                {
                                    min=(measure.GetDistance((combs.get(v)),(combs.get(z))));
                                }
                                }
                            }
                        }
                }

                if(min!=1000)
                {
                    distance=distance+min;
                    count=count+1;
                }}}}}}}}

        System.out.println("The number of total manipulation is "+ count);
        System.out.println("The number of total distance is "+ distance);
        System.out.println("The average distance is "+ (distance)/(count));
    }
    }
```

Kemeny (class)

```
public class Kemeny
{
    String x="";
    String y="";

    public int GetDistance(String x, String y)
    {
        int A= x.indexOf("a");
        int B= x.indexOf("b");
        int C= x.indexOf("c");
        int D= x.indexOf("d");
        int E= x.indexOf("e");

        int AA= y.indexOf("a");
        int BB= y.indexOf("b");
        int CC= y.indexOf("c");
        int DD= y.indexOf("d");
        int EE= y.indexOf("e");

        int distance=0;

        if((A<B&&AA>BB)||(A>B&&AA<BB))

            distance=distance+1;
            else
            distance=distance;

        if((A<C&&AA>CC)||(A>C&&AA<CC))

            distance=distance+1;
            else
            distance=distance;

        if((A<D&&AA>DD)||(A>D&&AA<DD))

            distance=distance+1;
            else
                distance=distance;

        if((A<E&&AA>EE)||(A>E&&AA<EE))

            distance=distance+1;
            else
                distance=distance;
```

```
    if((B<C&&BB>CC)||(B>C&&BB<CC))

        distance=distance+1;
        else
            distance=distance;

    if((B<D&&BB>DD)||(B>D&&BB<DD))

        distance=distance+1;
        else
            distance=distance;

    if((B<E&&BB>EE)||(B>E&&BB<EE))

        distance=distance+1;
        else
            distance=distance;

    if((C<D&&CC>DD)||(C>D&&CC<DD))

        distance=distance+1;
        else
            distance=distance;

    if((C<E&&CC>EE)||(C>E&&CC<EE))

        distance=distance+1;
        else
            distance=distance;

    if((D<E&&DD>EE)||(D>E&&DD<EE))

        distance=distance+1;
        else
            distance=distance;

    return distance;
}}
```

Uncovered (class)

```
public class Uncovered
{
    String [] N=new String [7];

    public String GetWinner(String [] N)
    {
        int AB=0;
        int AC=0;
        int AD=0;
        int BC=0;
        int BD=0;
        int CD=0;

        for(int i=0; i<=6; i ++)
        {
            if((N[i]).indexOf("a")<(N[i]).indexOf("b"))
            {AB=AB+1;}

            if((N[i]).indexOf("a")>(N[i]).indexOf("b"))
            {AB=AB-1;}

            if((N[i]).indexOf("a")<(N[i]).indexOf("c"))
            {AC=AC+1;}

            if((N[i]).indexOf("a")>(N[i]).indexOf("c"))
            {AC=AC-1;}

            if((N[i]).indexOf("a")<(N[i]).indexOf("d"))
            {AD=AD+1;}

            if((N[i]).indexOf("a")>(N[i]).indexOf("d"))
            {AD=AD-1;}

            if((N[i]).indexOf("b")<(N[i]).indexOf("c"))
            {BC=BC+1;}

            if((N[i]).indexOf("b")>(N[i]).indexOf("c"))
            {BC=BC-1;}

            if((N[i]).indexOf("b")<(N[i]).indexOf("d"))
            {BD=BD+1;}

            if((N[i]).indexOf("b")>(N[i]).indexOf("d"))
            {BD=BD-1;}
```

```
        if((N[i]).indexOf("c")<(N[i]).indexOf("d"))
        {CD=CD+1;}

        if((N[i]).indexOf("c")>(N[i]).indexOf("d"))
        {CD=CD-1;}
}
String Winner="";
String Covered="";

if(AB>0)
{
        if(AC>0&&BC>0&&BD<0)
        {
                Covered=Covered.concat("b");
        }
        if(AD>0&&BD>0&&BC<0)
        {
                Covered=Covered.concat("b");
        }
        if(BD<0&&BC<0)
        {
                Covered=Covered.concat("b");
        }
        if(AC>0&&AD>0&&BC>0&&BD>0)
        {
                Covered=Covered.concat("b");
        }
}
if(AC>0)
{
        if(AB>0&&BC<0&&CD<0)
        {
                Covered=Covered.concat("c");
        }
        if(AD>0&&CD>0&&BC>0)
        {
                Covered=Covered.concat("c");
        }
        if(CD<0&&BC>0)
        {
                Covered=Covered.concat("c");
        }
        if(AB>0&&AD>0&&BC<0&&CD>0)
        {
                Covered=Covered.concat("c");
```

```
        }
    }
    if(AD>0)
    {
        if(AB>0&&BD<0&&CD>0)
        {
            Covered=Covered.concat("d");
        }
        if(AC>0&&CD<0&&BD>0)
        {
            Covered=Covered.concat("d");
        }
        if(CD>0&&BD>0)
        {
            Covered=Covered.concat("d");
        }
        if(AB>0&&AC>0&&BD<0&&CD<0)
        {
            Covered=Covered.concat("d");
        }
    }
    if(AB<0)
    {
        if(AC>0&&BC>0&&AD<0)
        {
            Covered=Covered.concat("a");
        }
        if(AD>0&&BD>0&&AC<0)
        {
            Covered=Covered.concat("a");
        }
        if(AD<0&&AC<0)
        {
            Covered=Covered.concat("a");
        }
        if(AC>0&&AD>0&&BC>0&&BD>0)
        {
            Covered=Covered.concat("a");
        }
    }
    if(BC>0)
    {
        if(AB<0&&AC<0&&CD<0)
        {
            Covered=Covered.concat("c");
        }
```

```
        if(BD>0&&CD>0&&AC>0)
        {
            Covered=Covered.concat("c");
        }
        if(CD<0&&AC>0)
        {
            Covered=Covered.concat("c");
        }
        if(AB<0&&BD>0&&AC<0&&CD>0)
        {
            Covered=Covered.concat("c");
        }
    }
    if(BD>0)
    {
        if(AB<0&&AD<0&&CD>0)
        {
            Covered=Covered.concat("d");
        }
        if(BC>0&&CD<0&&AD>0)
        {
            Covered=Covered.concat("d");
        }
        if(CD>0&&AD>0)
        {
            Covered=Covered.concat("d");
        }
        if(AB<0&&BC>0&&AD<0&&CD<0)
        {
            Covered=Covered.concat("d");
        }
    }
    if(AC<0)
    {
        if(AB>0&&BC<0&&AD<0)
        {
            Covered=Covered.concat("a");
        }
        if(AD>0&&CD>0&&AB<0)
        {
            Covered=Covered.concat("a");
        }
        if(AD<0&&AB<0)
        {
            Covered=Covered.concat("a");
        }
```

```
        if(AB>0&&AD>0&&BC<0&&CD>0)
        {
            Covered=Covered.concat("a");
        }
    }
    if(BC<0)
    {
        if(AB<0&&AC<0&&BD<0)
        {
            Covered=Covered.concat("b");
        }
        if(BD>0&&CD>0&&AB>0)
        {
            Covered=Covered.concat("b");
        }
        if(BD<0&&AB>0)
        {
            Covered=Covered.concat("b");
        }
        if(AB<0&&BD>0&&AC<0&&CD>0)
        {
            Covered=Covered.concat("b");
        }
    }
    if(CD>0)
    {
        if(AC<0&&AD<0&&BD>0)
        {
            Covered=Covered.concat("d");
        }
        if(BC<0&&BD<0&&AD>0)
        {
            Covered=Covered.concat("d");
        }
        if(BD>0&&AD>0)
        {
            Covered=Covered.concat("d");
        }
        if(AC<0&&BC<0&&AD<0&&BD<0)
        {
            Covered=Covered.concat("d");
        }
    }
    if(AD<0)
    {
        if(AC>0&&CD<0&&AB<0)
```

```
        {
            Covered=Covered.concat("a");
        }
        if(AB>0&&BD<0&&AC<0)
        {
            Covered=Covered.concat("a");
        }
        if(AB<0&&AC<0)
        {
            Covered=Covered.concat("a");
        }
        if(AC>0&&AB>0&&CD<0&&BD<0)
        {
            Covered=Covered.concat("a");
        }
    }
    if(BD<0)
    {
        if(AB<0&&AD<0&&BC<0)
        {
            Covered=Covered.concat("b");
        }
        if(BC>0&&CD<0&&AB>0)
        {
            Covered=Covered.concat("b");
        }
        if(BC<0&&AB>0)
        {
            Covered=Covered.concat("b");
        }
        if(AB<0&&BC>0&&AD<0&&CD<0)
        {
            Covered=Covered.concat("b");
        }
    }
    if(CD<0)
    {
        if(BD<0&&BC<0&&AC>0)
        {
            Covered=Covered.concat("c");
        }
        if(AD<0&&AC<0&&BC>0)
        {
            Covered=Covered.concat("c");
        }
        if(AC>0&&BC>0)
```

```
            {
                Covered=Covered.concat("c");
            }
            if(BD<0&&AD<0&&BC<0&&AC<0)
            {
                Covered=Covered.concat("c");
            }
        }

        if(Covered.indexOf("a")==(-1))
        {Winner=Winner.concat("a");}
        if(Covered.indexOf("b")==(-1))
        {Winner=Winner.concat("b");}
        if(Covered.indexOf("c")==(-1))
        {Winner=Winner.concat("c");}
        if(Covered.indexOf("d")==(-1))
        {Winner=Winner.concat("d");}

        return Winner;
}}
```