

STOCHASTIC DIFFUSION SEARCH AND VOTING METHODS

by

Ahmet Kutsi Nircan

B.S., Naval Architecture and Marine Engineering, İstanbul Technical University, 1989

M.S., Naval Architecture and Marine Engineering, İstanbul Technical University, 1991

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Civil Engineering

Boğaziçi University

2006

ACKNOWLEDGEMENTS

I thank my advisors Prof. Dr. Ali Rana Atılgan and Prof. Dr. A. C. Cem Say for their patience and help during this thesis. I also would like to thank jury members Prof. Dr. Levent Akin, Assoc. Prof. Dr. Osman Börekçi and Prof. Dr. M. Remzi Sanver for their valuable inputs.

It would be almost impossible to construct the thesis without the direction that Prof. Dr. Fuad Aleskerov provided. I thank him for his encouragement and time.

I also would like to thank Asst. Prof. Dr. Pınar Yolum for her time and help. Dr. Yolum kindly listened for long hours and gave valuable advice.

Without the support of my family and friends it would not be possible to go on. I thank them all, especially my mother.

Professors Dr. Miray Bekbölet and Dr. Martin Sampson were two friends who gave valuable advices. Thank you for your friendship, belief and wisdom.

ABSTRACT

STOCHASTIC DIFFUSION SEARCH AND VOTING METHODS

This study compares known social choice rules in the context of agent based search and introduces two new variants of stochastic diffusion search algorithm.

Performance comparison to match a correct ordering of 36 voting algorithms that are derived from 23 known social choice rules are made. A simple text search framework is used for the simulation where agents are given parts of a search key. Individual preferences of agents are then fed to 36 different voting algorithms. Results are compared against the known correct ordering and correct top choices. A similarity coefficient and a top choice match coefficient is used to compare the performances of the voting algorithms. Simulations are made for each length of search key part and each length of search space.

A population based search algorithm, stochastic diffusion search (SDS), is improved to include different voting methods. A shared memory and an individual memory variant are developed and performances compared against original SDS. Tests are made with text and image search frameworks. The similarity and top choice match coefficients are used for the text search framework. Image test performances are measured by calculating distance of the found location to the known correct location of the image. It is found that the new algorithms developed in this study considerably outperforms the original SDS algorithm.

ÖZET

RASTLANTISAL DAĞILMA ARAMASI VE OYLAMA YÖNTEMLERİ

Bu çalışma bilinen sosyal seçim kurallarını karşılaştırmakta ve rastlantısal dağılım aramasına iki yenilik getirmektedir.

Doğruluğu bilinen bir tercih sıralamasını bulmak için 23 sosyal seçim kuralından türetilmiş 36 oylama algoritması karşılaştırılmıştır. Basit bir karakter dizgesi sistemi kullanılarak ajanlara aranan karakterin parçaları verilmiştir. Denemeler değişken sayıda karakter parçası ile arama uzayının değişken sayıdaki kısmında denenmiştir. Buradan elde edilen tercih sıralamaları oylama algoritmalarına verilmiştir.

Sonuçlar doğru sıralama ve doğru en iyi tercihlerle karşılaştırılmıştır. Doğru sıralama ve doğru en iyi tercihler tüm aranan karakter dizinin arama uzayının tamamında taranmasıyla bulunmuştur. Bir benzerlik katsayısı ve en iyi tercih denklik katsayısı oylama algoritmalarının karşılaştırılmasında kullanılmıştır. Denemeler aranan karakter dizinin ve arama uzayının her bir uzunluğu için tekrar edilmiştir.

Rastlantısal dağılım araması yöntemi farklı oylama yöntemlerini içerebilecek şekilde değiştirilmiştir. Ortak hafıza ve bireysel hafıza olmak üzere iki yeni yöntem orijinal yöntemle karşılaştırılmıştır. Doğru sıralama ve doğru en iyi tercihler katsayıları kullanılarak karakter arama kabiliyetleri karşılaştırılmıştır. Resim araması ile de aranan resim parçasının doğru yerine olan uzaklığı ölçülmüştür.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	ix
LIST OF TABLES	xiv
LIST OF SYMBOLS/ABBREVIATIONS	xv
1. INTRODUCTION	1
2. COMPARISON OF SOCIAL CHOICE RULES	3
2.1. Simulation Framework	5
2.2. Definitions	7
2.2.1. Lower Counter Set	7
2.2.2. Upper Counter Set	8
2.3. Social Choice Rules and Simulation Results	8
2.3.1. Scoring Rules	9
2.3.1.1. Plurality Voting	9
2.3.1.2. Approval Voting	9
2.3.1.3. Run-Off Vote	10
2.3.1.4. Hare's Procedure	11
2.3.1.5. Inverse Plurality Vote	12
2.3.1.6. Borda's Rule	12
2.3.1.7. Sum Rule	14
2.3.1.8. Inverse Borda Rule	14
2.3.1.9. Nanson's Procedure	14
2.3.1.10. Coomb's Procedure	15
2.3.2. Rules Using Majority Relation	15
2.3.2.1. Condorcet and Minimal Undominated Set	15
2.3.2.2. Fishburn's Rule	19
2.3.2.3. Uncovered Set I	19
2.3.2.4. Uncovered Set II	21

2.3.2.5.	Richelson's Rule	22
2.3.3.	Rules Using Value Function	25
2.3.3.1.	Copeland's Rule I	25
2.3.3.2.	Copeland's Rule II	25
2.3.3.3.	Copeland's Rule III	25
2.3.4.	Rules Using Tournament Matrix	28
2.3.4.1.	Simpson's Procedure (Maxmin Procedure)	29
2.3.4.2.	Minmax Procedure	29
2.3.5.	q-Paretian Rules	31
2.3.5.1.	Strong q-Paretian Simple Majority Rule	31
2.3.5.2.	Strong q-Paretian Plurality Rule	31
2.3.5.3.	Strongest q-Paretian Simple Majority Rule	34
2.4.	Conclusion	36
3.	STOCHASTIC DIFFUSION SEARCH	42
3.1.	Introduction	42
3.2.	Testing Framework	44
3.2.1.	String Search Framework	44
3.2.2.	Image Search Framework	44
3.3.	Original SDS	46
3.3.1.	Agent Properties	46
3.3.2.	Diffusion	47
3.3.3.	Halting Criteria	47
3.3.4.	Variants of SDS in Literature	48
3.3.5.	Applicability of SDS	48
3.3.6.	Comparison of SDS with Other Search Methods	49
3.3.7.	Applications of SDS	50
3.4.	SDS with Shared Memory	50
3.4.1.	SDS with Shared Count Memory	51
3.4.2.	SDS with Shared Graph Memory	51
3.4.3.	String Test Results	52
3.4.4.	Image Test Results	54
3.4.5.	Combining Results of Multiple SDS Algorithms	56

3.5. SDS With Individual Memories	56
3.5.1. String Test Results	57
3.5.2. Image Test Results	63
3.6. Conclusion	64
4. KNOWLEDGE STRUCTURE	65
4.1. Instability and Value Restriction	65
4.2. Knowledge Structures	65
4.3. Problems with Knowledge Structures	68
5. CONCLUSIONS	70
REFERENCES	72

LIST OF FIGURES

Figure 2.1.	Averages of α and β coefficients for Plurality algorithm at ‘Number of Steps’=20	9
Figure 2.2.	Averages of α and β coefficients for Approval algorithm at ‘Number of Steps’=20 and $q=0$	10
Figure 2.3.	Averages of α and β coefficients for RunOff Algorithm at ‘Number of Steps’=20	11
Figure 2.4.	Averages of α and β coefficients for Hare algorithm at ‘Number of Steps’=20	11
Figure 2.5.	Averages of α and β coefficients for InvPlrty algorithm at ‘Number of Steps’=20	12
Figure 2.6.	Averages of α and β coefficients for Borda algorithm at ‘Number of Steps’=20	13
Figure 2.7.	Number of agents checking each criterion for two criteria per agent case	13
Figure 2.8.	Number of agents checking each criterion for one criteria per agent case	13
Figure 2.9.	Averages of α and β coefficients for InvBorda algorithm at ‘Number of Steps’=20	14
Figure 2.10.	Averages of α and β coefficients for Nanson algorithm at ‘Number of Steps’=20	15

Figure 2.11. Comparing average α and β coefficients of Coomb algorithm at ‘Number of Steps’=20	16
Figure 2.12. Averages of α and β coefficients for Condorcet and CondorcetS algorithm at ‘Number of Steps’=20	18
Figure 2.13. Averages of α and β coefficients for Condorcet algorithm at ‘Number of Steps’=7	18
Figure 2.14. Comparing average similarity (α) coefficients of Condorcet and CondorcetS algorithms at ‘Number of Steps’=7	19
Figure 2.15. Averages of α and β coefficients for Fishburn algorithm at ‘Number of Steps’=20	20
Figure 2.16. Comparing average similarity (α) coefficients of Fishburn and FishburnS algorithms at ‘Number of Criteria’=10	20
Figure 2.17. Averages of α and β coefficients for Uncover1 algorithm at ‘Number of Steps’=20	21
Figure 2.18. Comparing average similarity (α) coefficients of Uncover1 and Uncover1S algorithms at ‘Number of Criteria’=10	21
Figure 2.19. Averages of α and β coefficients for Uncover2 algorithm at ‘Number of Steps’=20	22
Figure 2.20. Comparing average similarity (α) coefficients of Uncover2 and Uncover2S algorithms at ‘Number of Criteria’=10	22
Figure 2.21. Averages of α and β coefficients for Richelson algorithm at ‘Number of Steps’=20	23

Figure 2.22. Comparing average similarity (α) coefficients of Richelson and RichelsonS algorithms at ‘Number of Criteria’=10	24
Figure 2.23. A majority relation between a , b , c , and d	24
Figure 2.24. Averages of α and β coefficients for CopeNet algorithm at ‘Number of Steps’=20	26
Figure 2.25. Comparing average similarity (α) coefficients of CopeNet and CopeNetS algorithms at ‘Number of Criteria’=10	26
Figure 2.26. Averages of α and β coefficients for CopeOut algorithm at ‘Number of Steps’=20	27
Figure 2.27. Comparing average similarity (α) coefficients of CopeOut and CopeOutS algorithms at ‘Number of Criteria’=10	27
Figure 2.28. Averages of α and β coefficients for CopeIn algorithm at ‘Number of Steps’=20	28
Figure 2.29. Comparing average similarity (α) coefficients of CopeIn and CopeInS algorithms at ‘Number of Criteria’=10	28
Figure 2.30. Averages of α and β coefficients for Maxmin algorithm at ‘Number of Steps’=20	29
Figure 2.31. Averages of α and β coefficients for Minmax algorithm at ‘Number of Steps’=20	30
Figure 2.32. Averages of α and β coefficients for StrongQ algorithm at ‘Number of Steps’=20	32

Figure 2.33.	Comparing average similarity coefficients of StrongQ and StrQSkip algorithms at ‘Number of Criteria’=10	32
Figure 2.34.	Averages of similarity coefficients for StrongQ algorithm at ‘Number of Steps’=20	33
Figure 2.35.	Comparing average similarity coefficients of StrongQ and StrQSkip algorithms at ‘Number of Criteria’=10	33
Figure 2.36.	Averages of similarity (α) coefficients for StrgestQA and StrgestQ algorithm at ‘Number of Steps’=20	35
Figure 2.37.	Averages of similarity (α) coefficients for StrgestQA and StrgestQ algorithm at ‘Number of Steps’=20	35
Figure 2.38.	Comparing average similarity coefficients of StrgestQ , StrgestQA , and StrgestQAS algorithms at ‘Number of Criteria’=10	36
Figure 2.39.	Comparing averages of similarity coefficients for StrQSkip , Coomb , and CopeNetS at ‘Number of Criteria’=7	41
Figure 3.1.	Results of Minchinton Cells are used as addresses for RAM	45
Figure 3.2.	Similarity coefficients comparing standard and context free SDS with blackboard SDS and no diffusion algorithms	53
Figure 3.3.	Top match coefficients comparing standard and context free SDS with blackboard SDS and no diffusion algorithms	53
Figure 3.4.	Comparing performance of Context Free SDS and SDS with blackboard architecture for image search at 100 iterations	55

Figure 3.5.	Comparing performance of Context Free SDS and SDS with blackboard architecture for image search at 1000 iterations	55
Figure 3.6.	Comparing averages of time elapsed to calculate the original context free SDS and SDS with blackboard architecture for image search at 1000 iterations	56
Figure 3.7.	Performance of multiple blackboard SDS algorithms aggregated using different social choice rules	57
Figure 3.8.	Top choice match coefficient comparison for various strongest q Paretian simple majority coalitions and context free SDS iterations	62
Figure 3.9.	Top choice match coefficient comparison for various strongest q Paretian simple majority coalitions and no diffusion iterations . .	62
Figure 3.10.	Number of correct matches for right and left eye using SDS with individual memories	63
Figure 3.11.	Histogram to compare Approval algorithm with $q = 0$ and $q = 1$.	63
Figure 3.12.	Elapsed time comparison for social choice rules in SDS with individual memories	64
Figure 4.1.	Four preference orderings derived from a knowledge structure . . .	66

LIST OF TABLES

Table 2.1.	Social choice rules and the names of corresponding algorithms that are tested	4
Table 2.2.	Results according to Fishburn, uncovered set I, II and Richelson rules	24
Table 2.3.	Best performing social choice algorithms according to α for each number of criterion and number of steps.	37
Table 2.4.	Best performing social choice algorithms according to β for each number of criterion and number of steps.	38
Table 2.5.	Averages of α and β coefficients over all ‘Number of Criteria’ and all ‘Number of Steps’	39
Table 2.6.	Averages of α and β coefficients over all ‘Number of Criteria’ and at ‘Number of steps’ = 20	40
Table 3.1.	α and β values for context free SDS and no diffusion with individual memories for 10 iterations	58
Table 3.2.	α and β values for context free SDS and no diffusion with individual memories for 100 iterations	59
Table 3.3.	α and β values for context free SDS and no diffusion with individual memories for 200 iterations	60
Table 4.1.	Algorithm to find individual preference from a knowledge structure	67

LIST OF SYMBOLS/ABBREVIATIONS

A	Set of alternatives
aIb	a is indifferent to b
aPb	a is preferred to b
c_i	i -th element of search key
C	Search key or criterion set
C^k	Criterion set of the k -th agent
d	Distance to the correct eye location
$D(\cdot)$	Upper counter set of the given alternative
h	Number of agents
$K(A, e)$	Knowledge stucture
l	Number of steps to be taken in the circular search space
$L(\cdot)$	Lower counter set of the given alternative
m	Number of elements in criteria
n	Number of elements in circular search space
p_k	Initial location for k -th agent
P	Ordering found by the social choice rule
P^r	Correct ordering
r_i	Score of the i -th location in search space
r_i^k	Score of k -th agent for the i -th location in search space
s_i	i -th element of circular search space
S	Circular search space
α	Similarity coefficient of two preference orderings
β	Top choice match coefficient of top alternatives of two preference orderings
$\mathfrak{S}$	Set of all possible simple majority coalitions
SDS	Stochastic Diffusion Search

1. INTRODUCTION

Most of the problems in science and engineering involve search for a solution. In engineering, solutions could be as easy as solving linear equations or as complex as combinatorial search (see Maier *et al.* (2003), Grierson and Khajepour (2002) and Dormer *et al.* (2003) for examples of combinatorial search applications in civil engineering). While some of the problems can be modeled with solvable equations some others need direct computational approaches (e.g. traveling salesmen problem, image search, etc.).

Combinatorial search with a vast search space cannot be handled with systematic search methods. Several stochastic search methods are developed for combinatorial search. Some of these methods involve multiple agents such as genetic algorithms, ant colony algorithms, memetic algorithms and particle swarm optimization.

When multiple agents are in question there needs to be a form of aggregation to obtain the result. Voting is the most important method of aggregating preferences of agents and it is already being used with non-human voters in ensembles of classifiers for pattern recognition. However only a limited number of social choice rules or voting methods had been tested to aggregate results of ensembles of classifiers and population based search algorithms (Van Erp *et al.*, 2002; Valentini and Masulli, 2002).

In this study 23 social choice rules are compared using a simple text search example that can be used as a benchmark for agent based search and optimization including ensembles of classifiers for pattern recognition. This study also looks at the use of different social choice rules for aggregation in Stochastic Diffusion Search (SDS) and its effects on performance.

The rest of the thesis is organized as follows.

Chapter 2 presents 23 social choice rules and a test framework to compare their

performances as well as the results.

Chapter 3 presents the SDS algorithmic process and introduces and compares the two variants developed in this study.

Knowledge structures were considered for possible use to improve SDS in this study. Chapter 4 explains knowledge structures and why they are not suitable for SDS.

2. COMPARISON OF SOCIAL CHOICE RULES

Social choice is the study of aggregation of individual preferences of agents in to a collective preference. Social choice rules are the rules or algorithms that define the procedure of this aggregation (Sen, 1970).

Today social choice rules are also being used by computers in computer networks (Eaves, 1999) or in some distributed algorithms.

Using ensembles of autonomous algorithms for pattern recognition is another application for social choice rules. This approach proved to be successful especially when each algorithm has a different error function. Application of voting to ensembles of pattern recognition algorithms can be done either as a final stage or as a multi-stage/hierarchical procedure. In the multi-stage/hierarchical approach, algorithms are placed in layers. The output of one layer determines the parameters of the next layer. When they are applied as a final stage each algorithm runs independently from each other and as a final stage a social choice rule is applied to find the result. Some of the applications of ensembles of autonomous algorithms may require the best alternative while others may require the complete ordering (Valentini and Masulli, 2002; Van Erp *et al.*, 2002).

In this chapter 36 algorithms derived from 23 social choice rules (see Table 2.1 for the list of social choice rules and algorithms) are compared for their accuracy in a text search simulation framework. Agents are given a partial criterion from a complete criterion set and 23 social choice rules (Aleskerov and Kurbanov, 1998) are applied to their result. Accuracy of a social choice rule is determined by comparing the known preference ordering that is found by testing the whole search space with the complete criterion set, to the result of the social choice rule.

The following simulation framework is a faster method to test the performance of social choice rules as a final stage in ensembles of autonomous algorithms since it

Table 2.1. Social choice rules and the names of corresponding algorithms that are
tested

SCORING RULES	
1 Plurality Voting	Plurality
2 Approval Voting	Approval
3 Run-Off Vote	RunOff
4 Hare's Procedure	Hare
5 Inverse Plurality Vote	InvPlrty
6 Borda's Rule	Borda
7 Sum Rule	Sum
8 Inverse Borda Rule	InvBorda
9 Coomb's Procedure	Coomb CoombS
10 Nanson's Procedure	Nanson
RULES USING MAJORITY RELATION	
11 Condorcet Vote	Condorcet CondorcetS
12 Fishburn's Rule	Fishburn FishburnS
13 Uncovered Set I	Uncovered1 Uncover1S
14 Uncovered Set II	Uncovered2 Uncover2S
15 Richelson's Rule	Richelson RichelsonS
RULES USING VALUE FUNCTION	
16 Copeland's Rule I	CopeNet CopeNetS
17 Copeland's Rule II	CopeOut CopeOutS
18 Copeland's Rule III	CopeIn CopeInS
RULES USING TOURNAMENT MATRIX	
19 Maxmin Procedure	Maxmin
20 Minmax Procedure	Minmax
Q-PARETIAN RULES	
21 Strong q-Paretian Simple Majority Rule	StrongQ StrQSkip
22 Strong q-Paretian Plurality Rule	StrongQP StrQPSkip
23 Strongest q-Paretian Simple Majority Rule	StrqestQ StrqestQA StrqestQAS

does not include the pattern recognition algorithms. In this framework, agents and their criteria represent the algorithms and their knowledge obtained from training set respectively. These agents search a space to find the best solution with their individual criteria. As in the case of a pattern recognition algorithm, agents don't have the full criteria set. The social choice rule is expected to overcome this limitation and find the exact match to the known preference ordering.

Social choice rules are compared using two coefficients. The similarity coefficient which is the ratio of the number of correct relations to the number of all possible relations between alternatives. The top choice match coefficient is the ratio of number of correct top choice matches to the maximum of the number of top choices in the two orderings.

2.1. Simulation Framework

A one dimensional circular search space S which consists of n elements $s_1, s_2, \dots, s_n$ is created where s_i could be either 0 or 1. A sequence which is given with a search key C with m elements of $c_1, c_2, \dots, c_m$ where c_j could be either 0 or 1 will be searched in S where $m < n$. Locations in S will be ordered with the number of matching elements as

$$r_i = \sum_{c_j \in C} \begin{cases} 1 & \text{if } s_{i+j} = c_j \\ 0 & \text{if } s_{i+j} \neq c_j \end{cases} \quad (2.1)$$

where i is the location in S . Due to circularity of S if $i + j \geq n$ then $i + j$ becomes $(i + j) \bmod(n)$. Locations will be ordered with their number of matching elements or scores r_i . If $r_x > r_y$ then y is preferred to x and if $r_x = r_y$ then x and y are indifferent.

Agents a_k where $k = 1, 2, \dots, h$ are also defined. Each agent tests a segment of S at locations $p_k, p_{k+1}, \dots, p_{k+l}$ where p_k is a random number between $1 \leq p_k \leq n$ and l is a predetermined integer between $1 \leq l \leq n$. Agent a_k has a portion of criteria C which is denoted as C^k . Every agent calculates the number of matching elements of

these locations with its subset of C^k and makes its ordering according to r^k .

$$r_i^k = \sum_{c_j^k \in C^k} \begin{cases} 1 & \text{if } s_{i+j} = c_j^k \\ 0 & \text{if } s_{i+j} \neq c_j^k \end{cases} \quad \text{for } i = p_k, p_{k+1}, p_{k+2}, \dots, p_{k+l} \quad (2.2)$$

The partial criteria C^k is determined by randomly removing a predetermined number of elements from C .

Different social choice rules are then used to find a collective preference ordering from the preferences of individual orderings of agents a_k where $k = 1, 2, \dots, h$. A similarity coefficient is calculated by finding the percentage of correct pair-wise preference comparisons of the collective ordering, relative to the correct ordering. The correct ordering is calculated using the complete C over the entire S .

The algorithm starts with constructing the one-dimensional search space S and the search criteria C by randomly assigning values to s_i and c_j respectively. A given number of agents are created with C^k partial criteria and they are assigned to random starting locations p_k . Once agents compare their partial criteria with the assigned search space their preference orderings are determined and 23 different social choice rules are applied. Results of 23 social choice rules are compared by calculating their similarity coefficients. The next test is done by randomly generating a new search space S .

Tests are made for C^k s having $t = 1, 2, \dots, m$ criteria and agents testing for $l = 1, 2, \dots, n$ locations on S .

Similarity coefficient α shows how similar an ordering is to the correct ordering. If P^r is the directed graph representing the correct ordering and P is the ordering found by a social choice rule then the similarity coefficient of P relative to P^r is

$$\alpha = \frac{\sum_{i=1}^n \sum_{j=i+1}^n \begin{cases} 1 & \text{if } x_j P^r x_i \equiv x_i P x_j \\ 0 & \text{otherwise} \end{cases}}{(n^2 - n)/2} \quad (2.3)$$

Therefore similarity coefficient is the ratio of total number of similar relations in P and P^r to the total number of possible relations in P^r .

Best alternative match coefficient β represents the ratio of the number of identical alternatives of best alternatives of two orderings to the maximum of the number of alternatives in these two sets. If W and V are best alternatives of two orderings then

$$\beta = \frac{\text{card}(W \cap V)}{\max(\text{card}(W), \text{card}(V))} \quad (2.4)$$

Best alternative match coefficient β is used to test the performance of finding the winner or winners for a social choice rule.

2.2. Definitions

2.2.1. Lower Counter Set

Lower counter set of an alternative x is the set of alternatives that are preferred to x .

$$L(x) = \{y \in A \mid y P x\} \quad (2.5)$$

Where x and y are alternatives belonging to A .

2.2.2. Upper Counter Set

Upper counter set of an alternative x is the set of alternatives that are preferred over x .

$$D(x) = \{y \in A \mid xPy\} \quad (2.6)$$

2.3. Social Choice Rules and Simulation Results

The social choice rules are divided into five groups.

- a. Scoring rules,
- b. Rules using majority relations,
- c. Rules using value function,
- d. Rules using tournament matrix,
- e. q-Paretian rules.

Tests are made for the case with 10 agents, a search space of 20 characters and a search key of 10 characters. Averages and standard deviations of similarity coefficients are calculated from 10,000 runs of each ‘Number of Steps’ in search space and each ‘Number of Criteria’ per agent.

During the initialization of the simulation, a search key C , agent 10 criterion sets C^k with $1 \leq m \leq 10$ and initial agent positions p_k are randomly created. For 10,000 runs a random search space is created and the correct ordering for the given search space and search key is found. In each run similarity coefficient and top choice match coefficients are calculated for each social choice rule, ‘Number of Steps’ and ‘Number of Criterion’.

2.3.1. Scoring Rules

Results are given for the case where agents search the whole space ('Number of Steps'=20) because it is the most used scenerio in ensembles of autonomous pattern recognition algorithms.

2.3.1.1. Plurality Voting. Each agent votes for its best alternative. If agent has more then one best alternative then one is chosen randomly.

It can be seen from Figure 2.1 that α drops when 'Number of Steps' is 20 and 'Number of Criteria' approaches 10. Similarity coefficient reaches its maximum when 'Number of Steps' and 'Number of Criteria' are both 7. When agents begin to check more criteria and test a larger part of search space they will begin to vote for a limited number of alternatives (i.e. the alternatives that are on the top). Since other alternatives will not be on the final ordering it will be less similar to the correct ordering even if plurality vote succeeds to find the best alternative.

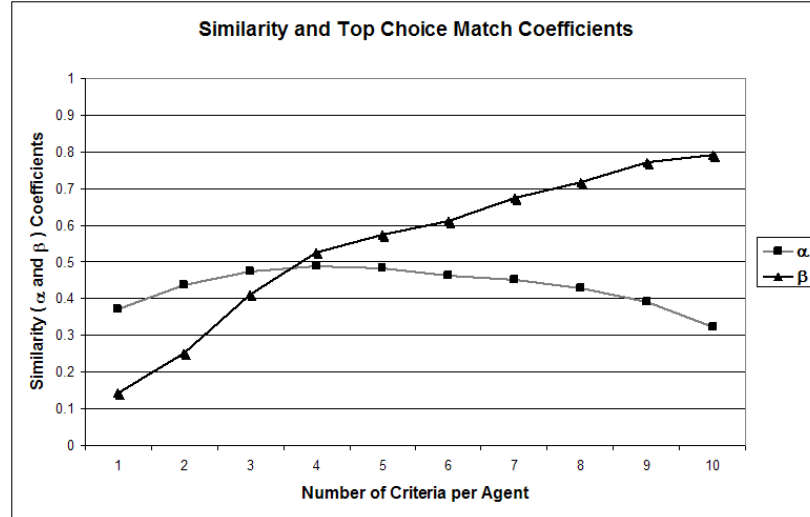


Figure 2.1. Averages of α and β coefficients for Plurality algorithm at 'Number of Steps'=20

2.3.1.2. Approval Voting. Each agent votes for its q best alternatives. When $q = 0$ only the best alternatives (even if there are more then one) are voted for. When $q = 1$

best and the second best alternatives are voted. Actually Approval Voting is defined as each agent can vote any number of alternatives from the top. When agents are humans q might be different for each agent. In this simulation q will be taken constant for all agents.

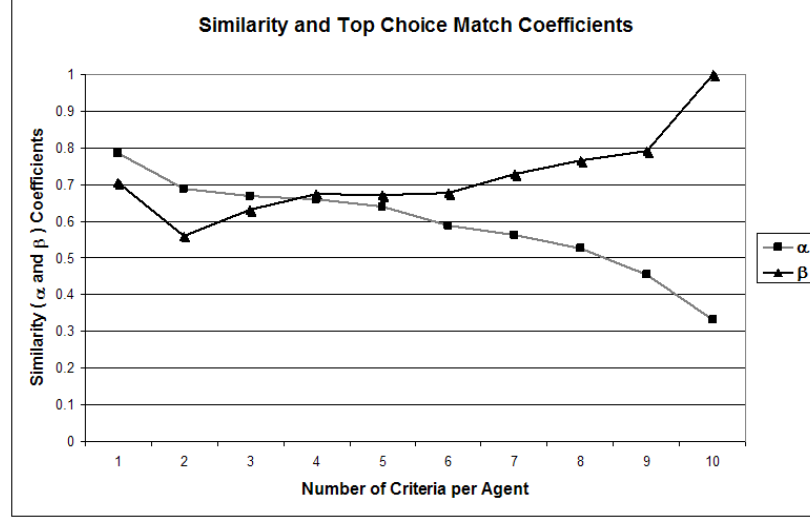


Figure 2.2. Averages of α and β coefficients for Approval algorithm at ‘Number of Steps’=20 and $q=0$

There is a drop at the similarity coefficient of Approval voting with $q = 0$ (Figure 2.2). It gives its highest similarity coefficient at ‘Number of Steps’=20 and ‘Number of Criteria’=3. This is because as the ‘Number of Criteria’ increases, more of the agents will have similar preferences and this will cause only a limited (1 in this case) number of alternatives to be in the final cast. When each agent has different criteria then there will be more alternatives in the collective preference ordering.

2.3.1.3. Run-Off Vote. If a plurality vote cannot find a winner with majority then the best two alternatives are taken and a plurality vote is applied again to this two alternatives. To find the rest of the preference ordering, winner is removed from the list and the procedure is repeated until no alternative is left.

Highest similarity coefficients were obtained when ‘Number of Criteria’=10 and ‘Number of Steps’ were between 16 and 20. Figure 2.3 shows the results for ‘Number

of Steps'=20.

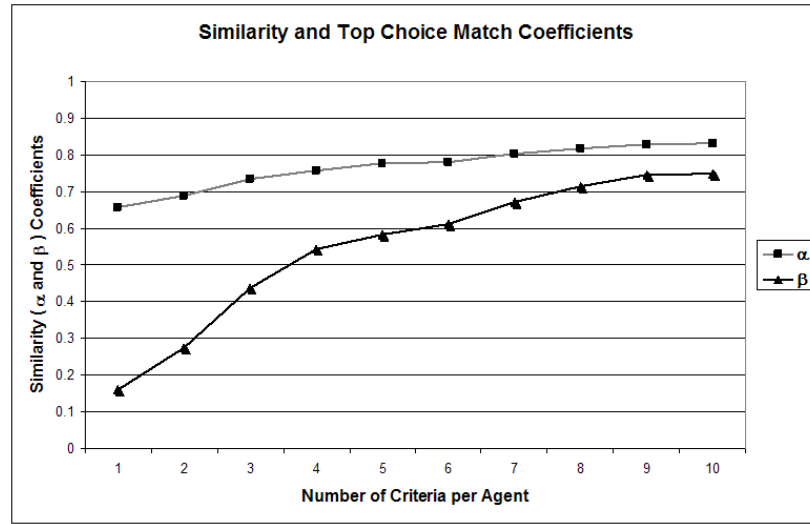


Figure 2.3. Averages of α and β coefficients for RunOff Algorithm at 'Number of Steps'=20

2.3.1.4. Hare's Procedure. If there is no simple majority then the worst alternative is omitted and the procedure is repeated until there is a winner with simple majority. To find the rest of the preference ordering, winner is removed from the list and the procedure is repeated until no alternative is left (Figure 2.4).

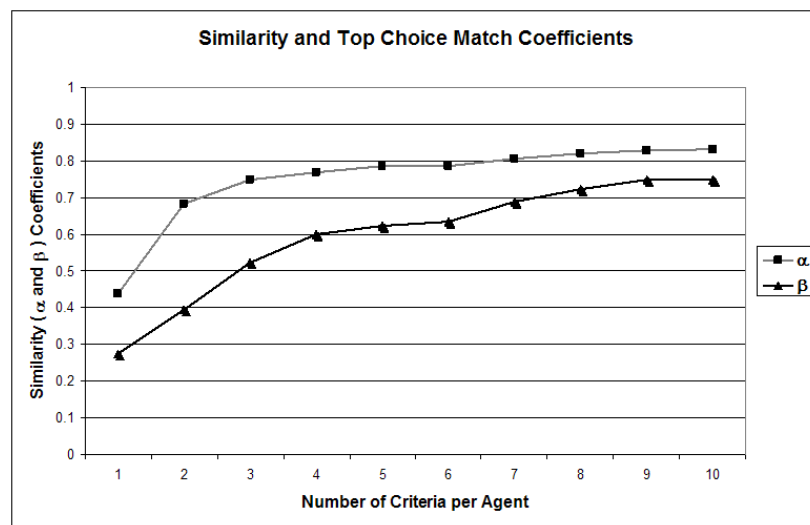


Figure 2.4. Averages of α and β coefficients for Hare algorithm at 'Number of Steps'=20

2.3.1.5. Inverse Plurality Vote. The alternative which is regarded worst by the minimum number of agents is chosen to be the winner (Figure 2.5).

The reason for the drop in similarity coefficient is same as plurality vote (see 2.3.1.1).

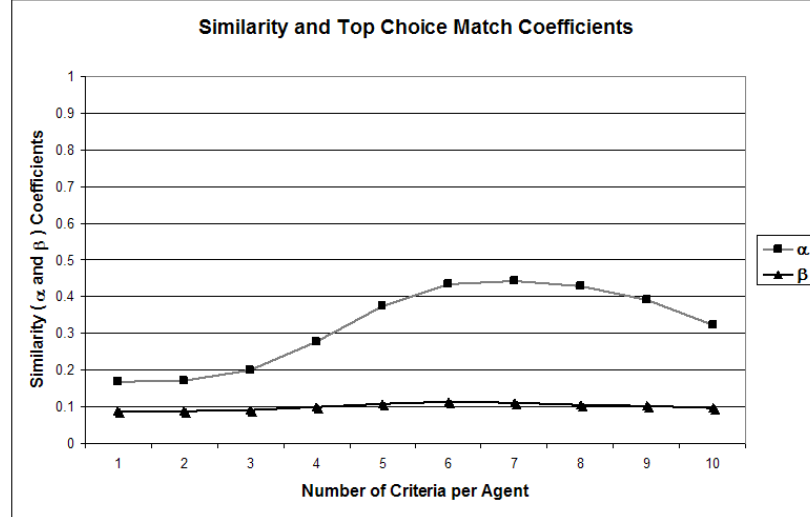


Figure 2.5. Averages of α and β coefficients for InvPlrty algorithm at ‘Number of Steps’=20

2.3.1.6. Borda’s Rule. Agents vote by scoring $m - 1$ to the best alternative(s) and lesser amounts to the following alternatives. These values are added to find the winner and the ordering.

There is a drop at ‘Number of Criteria’=2 relative to 1 and 3 in Borda’s rule (Figure 2.6) and most of the rules in this study. The search key and random criteria of agents are kept the same and only the search space were randomly created during the runs. It can be seen from Figure 2.7 that criterion number 9 was being checked by four agents and three other criteria were being checked by three agents for the case of two criteria. On the other hand two criteria were being checked by only one agent and rest of the agents was checking one criterion each. Results can be worse compared to one criterion per agent case in some of the runs because of this bias. The case for one criterion per agent has a more even distribution (Figure 2.8). Some search space

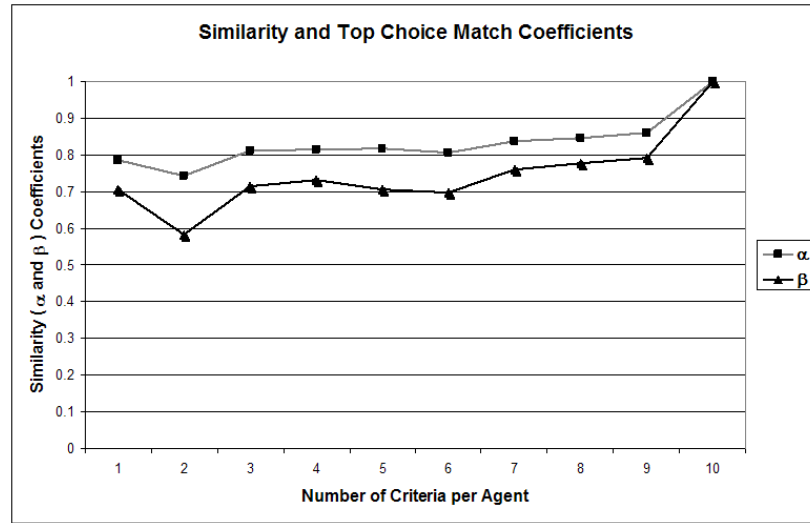


Figure 2.6. Averages of α and β coefficients for Borda algorithm at ‘Number of Steps’=20

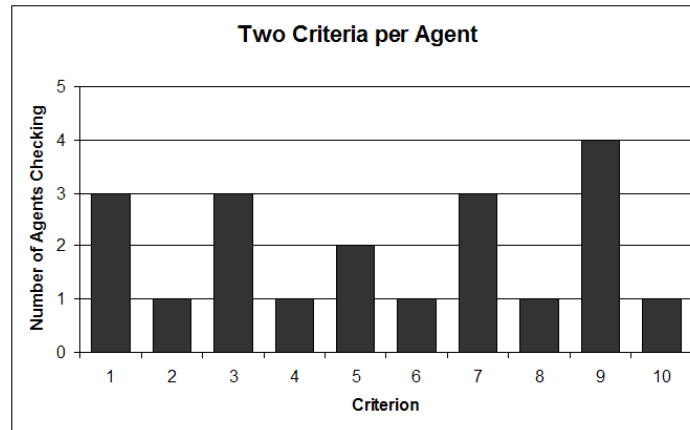


Figure 2.7. Number of agents checking each criterion for two criteria per agent case

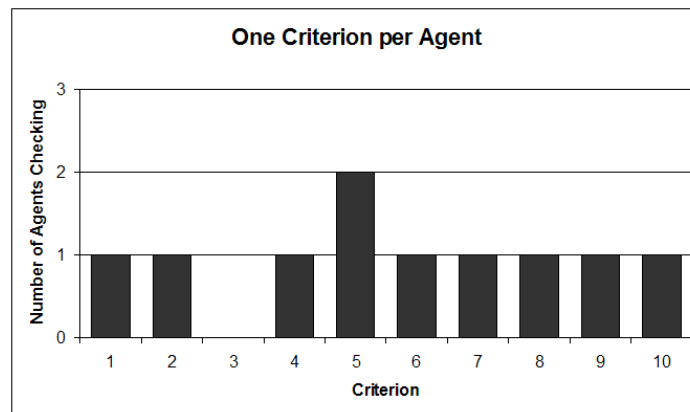


Figure 2.8. Number of agents checking each criterion for one criteria per agent case

configurations were not being effected by this bias.

2.3.1.7. Sum Rule. Sum rule is a variant of Borda's rule. In this case agents vote with their scores that they have given to a particular alternative. These scores are added to find the winner and the ordering. Sum voting gave exactly the same results as Borda for 'Number of Steps'=20 case (Figure 2.6).

2.3.1.8. Inverse Borda Rule. At each round Borda's Rule is applied. The alternative with lowest score is omitted until only one alternative is left. To find the rest of the preference ordering winner is removed from the list and procedure repeated until no alternative is left (Figure 2.9).

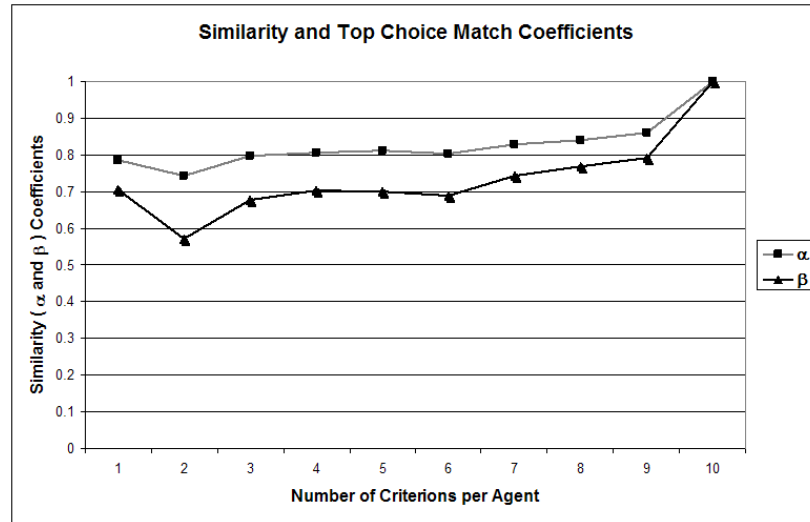


Figure 2.9. Averages of α and β coefficients for InvBorda algorithm at 'Number of Steps'=20

2.3.1.9. Nanson's Procedure. Borda's Rule is applied and the alternatives with scores lower than the average score are omitted. The procedure is repeated until one alternative is left. The ordering is determined by the removal sequence of alternatives. Last removed is most preferred (Figure 2.10).

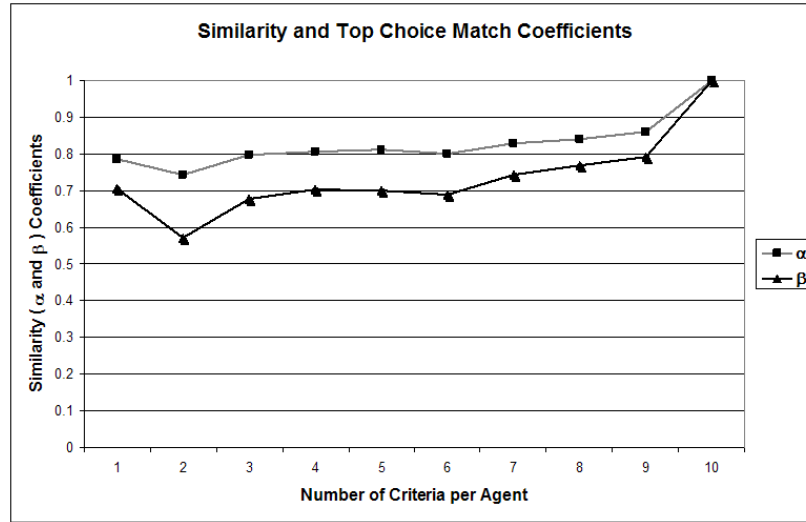


Figure 2.10. Averages of α and β coefficients for Nanson algorithm at ‘Number of Steps’=20

2.3.1.10. Coomb’s Procedure. Alternatives which are worst by the maximum number of agents are omitted until there is a winner with simple majority. Ordering is made by removing the winners from the profile and repeating the procedure until no alternative is left.

Coomb and CoombS algorithms are implemented for this case. CoombS counts only the agents who have tested an alternative for the simple majority. Both algorithms give exactly the same output for the case given in Figure 2.11.

2.3.2. Rules Using Majority Relation

2.3.2.1. Condorcet and Minimal Undominated Set. Condorcet is also called majority relation. Pairwise comparison of every alternative is made for each agent. Results are stored in a directed graph and this graph is compared with the directed graph of the correct ordering. This algorithm is designed to return two way edges for indifferent alternatives.

Two algorithms are implemented for this rule. **Condorcet** algorithm assumes that untested alternatives are least preferred; on the other hand **CondorcetS** algorithm

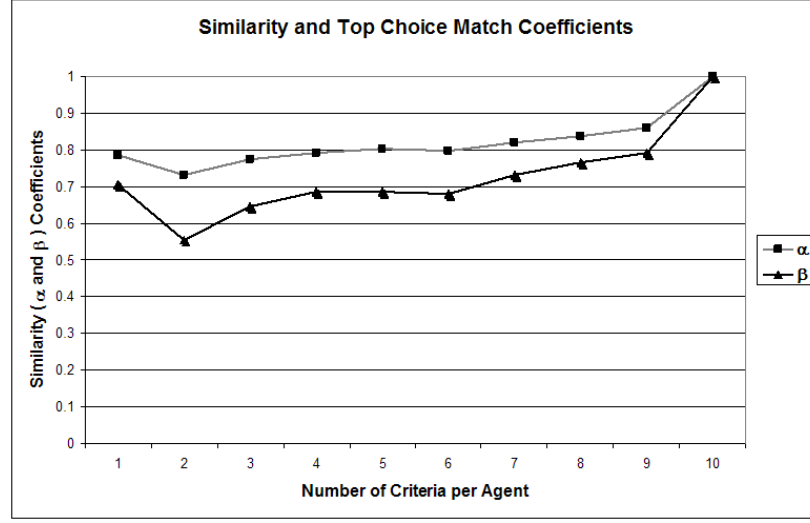


Figure 2.11. Comparing average α and β coefficients of Coomb algorithm at ‘Number of Steps’=20

does not take untested alternatives in to consideration when counting the pairwise comparisons.

The minimal undominated set algorithm finds the winner alternatives from the directed graph output of **Condorcet** and **CondorcetS** algorithms. A set is called undominated if no alternative outside of this set dominates an alternative within the set. This set is called minimal if there is no subset that is also undominated. If there is more than one such set then the result is the union of all such sets.

Algorithm

If V is the vertex set of a given majority relation, then the following algorithm will give the minimal undominated set.

1. Let Q and S be empty sets
2. For $f := 0$ to $n - 1$
 3. For all alternative x in V
 4. If $\text{card}(D(x)) = f$
 5. Add x to Q

6. Let $S = S \cup D(x)$
7. Next x
8. If $Q \supset S$ Return Q
9. Let $V = V \setminus Q$
10. Next f

Proof

Suppose $y \mu x$ (y dominates x) and above algorithm returns $x \in Q$ and $y \notin Q$. Then $y \in D(x)$. Because of step 6 S will include y and because of step 8 the algorithm will take the next f value until y is also included in Q . Either $y \notin Q$ or $y \mu x$ is false. Therefore Q will always be undominated.

Suppose that above algorithm returned a set Q and there is an undominated set $\acute{Q}$ where $\acute{Q} \supset Q$. Since $\acute{Q}$ is undominated by the rest of the members of V , all the members of Q will have a member of $\acute{Q}$ in their upper counter set and when that happens above algorithm would stop at step 8 because S is the union of upper counter sets of all the members of $\acute{Q}$. And since $\acute{Q}$ is a subset of Q it will have smaller upper counter set cardinality and therefore its members will be returned earlier. Therefore the set returned by the algorithm will be minimal.

Suppose that above algorithm returned Q as the result but there is another set W which is also a minimal undominated set. This is possible only if the members of Q and W are indifferent to each other and upper counter cardinality of the members of W is greater than the upper counter cardinality of the members of Q . However since our majority relation has two way edges for indifferent alternatives, members of Q will have W in their upper counter set and therefore it will not stop until f is big enough to include members of W . Therefore algorithm will return union of all minimal undominated sets.

Figure 2.12 shows results of **Condorcet** and **CondorcetS** algorithms at ‘Number of Steps’=20. When ‘Number of Steps’=20 agents test all the alternatives, therefore

results of Condorcet and CondorcetS are identical.

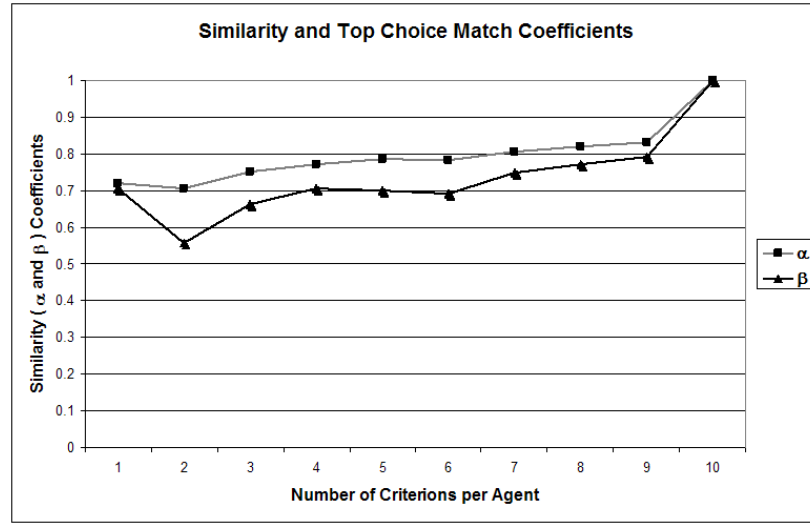


Figure 2.12. Averages of α and β coefficients for Condorcet and CondorcetS algorithm at 'Number of Steps'=20

There is a slight drop at similarity coefficient as the 'Number of Criteria' increase, at 'Number of Steps' values lower then 13 (Figure 2.13 for 'Number of Steps'=7 case). This same drop can also be seen at social choice rules (Fishburn, Uncovered I and II, Richelson and Copeland rules) that are using Condorcet within their procedures.

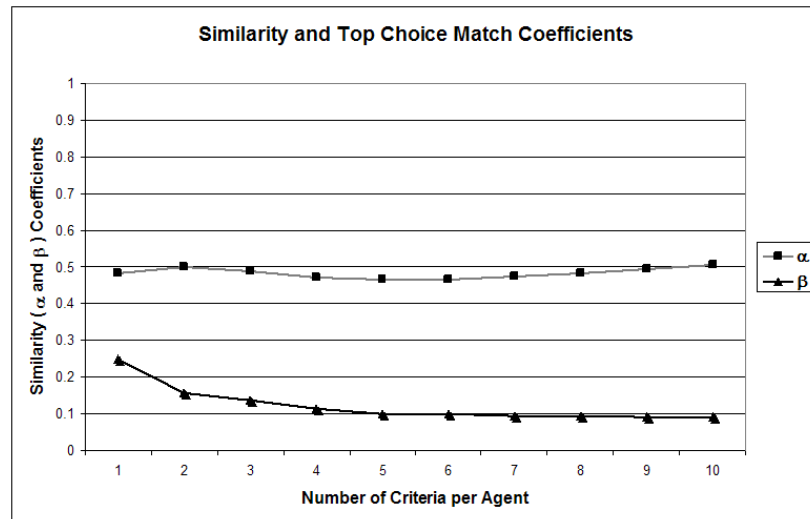


Figure 2.13. Averages of α and β coefficients for Condorcet algorithm at 'Number of Steps'=7

Since **Condorcet** algorithm assumes that untested alternatives have zero scores when ‘Number of Steps’ is a small number, untested alternatives create this drop. This drop is not seen in **CondorcetS** algorithm or the other algorithms using **CondorcetS** (Figure 2.14 shows the comparison of two algorithms).

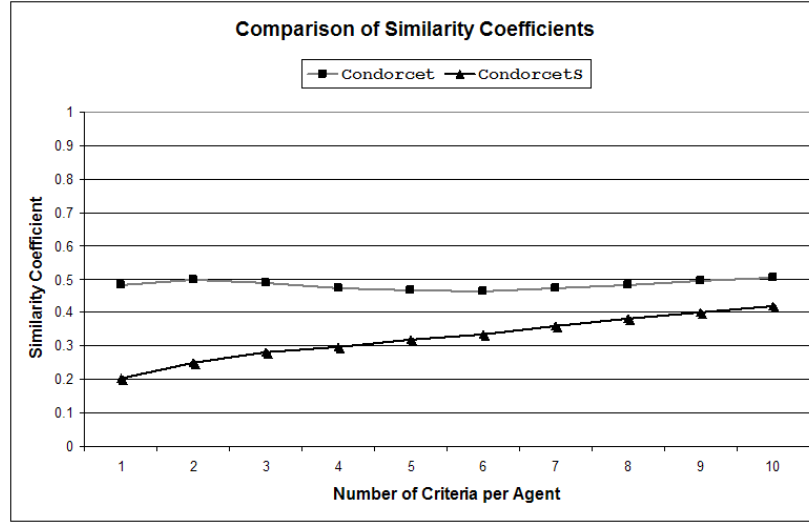


Figure 2.14. Comparing average similarity (α) coefficients of **Condorcet** and **CondorcetS** algorithms at ‘Number of Steps’=7

2.3.2.2. Fishburn’s Rule. A Condorcet vote is conducted and a directed graph where nodes are the alternatives is created. Edges in this directed graph are created if upper counter set of one of the nodes of the edge is a subset of the upper counter set of the other node in Condorcet result.

Two algorithms are implemented for Fishburn’s rule. **Fishburn** algorithm uses the **Condorcet** algorithm. On the other hand **FishburnS** algorithm uses **CondorcetS** algorithm (Figure 2.15). It can be seen from Figure 2.16 that **FishburnS** performs better at higher ‘Number of Crtiterions’ when agents are using the full criterion set.

2.3.2.3. Uncovered Set I. Similar to Fishburn’s Rule, a Condorcet vote is conducted and a directed graph where nodes are the alternatives is created. Edges in this directed graph are created if lower counter set of one of the nodes of the edge is a subset of the lower counter set of the other node in Condorcet result.

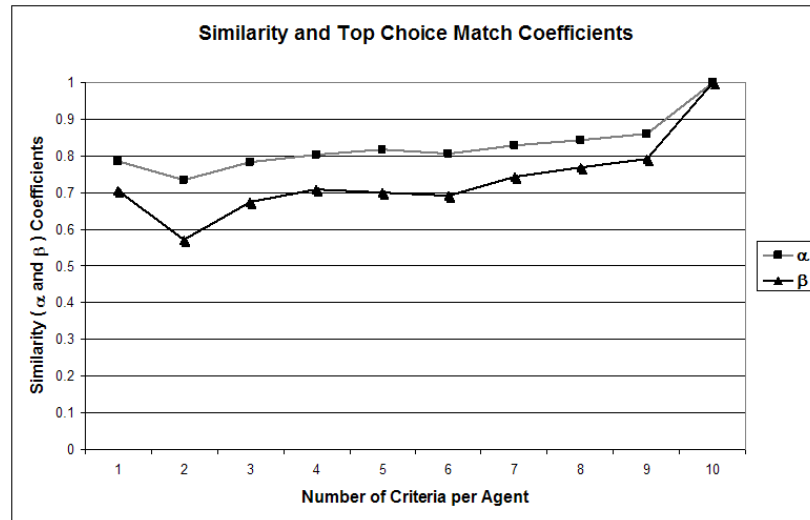


Figure 2.15. Averages of α and β coefficients for Fishburn algorithm at ‘Number of Steps’=20

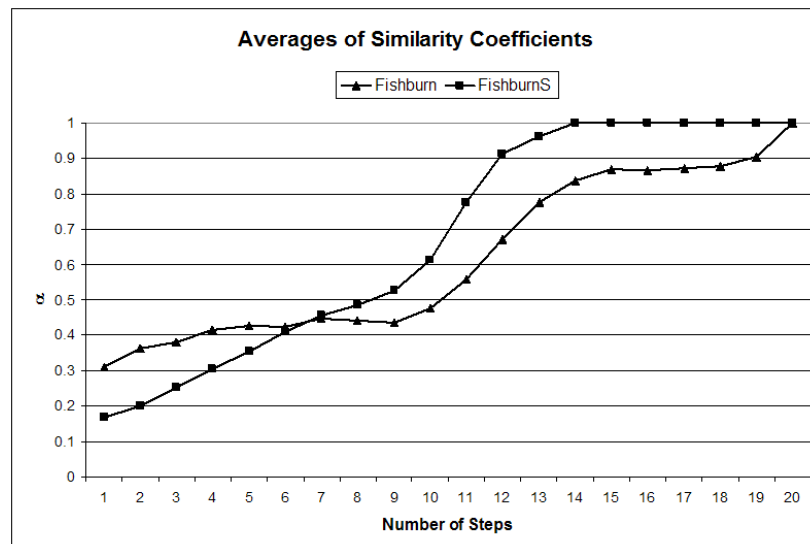


Figure 2.16. Comparing average similarity (α) coefficients of Fishburn and FishburnS algorithms at ‘Number of Criteria’=10

Two algorithms are implemented for Uncovered Set I. **Uncover1** algorithm uses the **Condorcet** algorithm. On the other hand **Uncover1S** algorithm uses **CondorcetS** algorithm (Figure 2.17). Uncovered Set I algorithm that skips untested alternatives performs better (Figure 2.18).

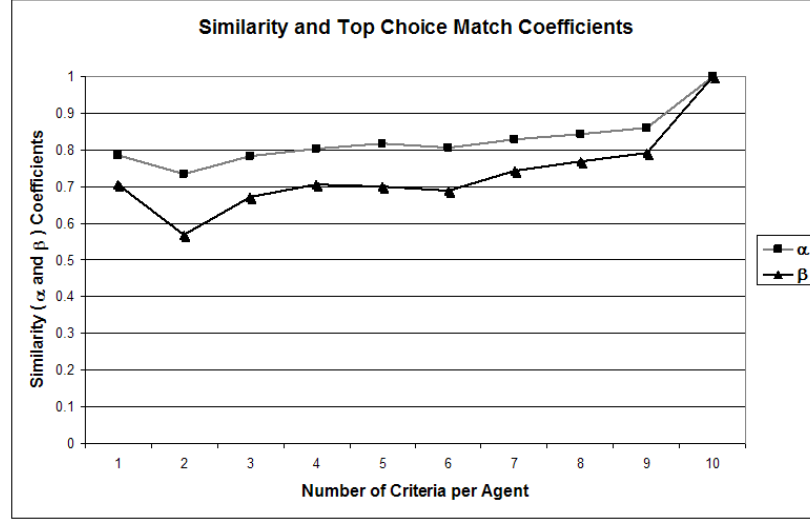


Figure 2.17. Averages of α and β coefficients for **Uncover1** algorithm at ‘Number of Steps’=20



Figure 2.18. Comparing average similarity (α) coefficients of **Uncover1** and **Uncover1S** algorithms at ‘Number of Criteria’=10

2.3.2.4. Uncovered Set II. A Condorcet vote is conducted and a directed graph where nodes are the alternatives is created. Edges in this directed graph are created if upper counter set of one alternative is a subset of the other alternative or equal; and

alternative which is a subset beats the other in the Condorcet result.

Two algorithms are implemented for Uncovered Set II. `Uncover2` algorithm uses the `Condorcet` algorithm. On the other hand `Uncover2S` algorithm uses `CondorcetS` algorithm.

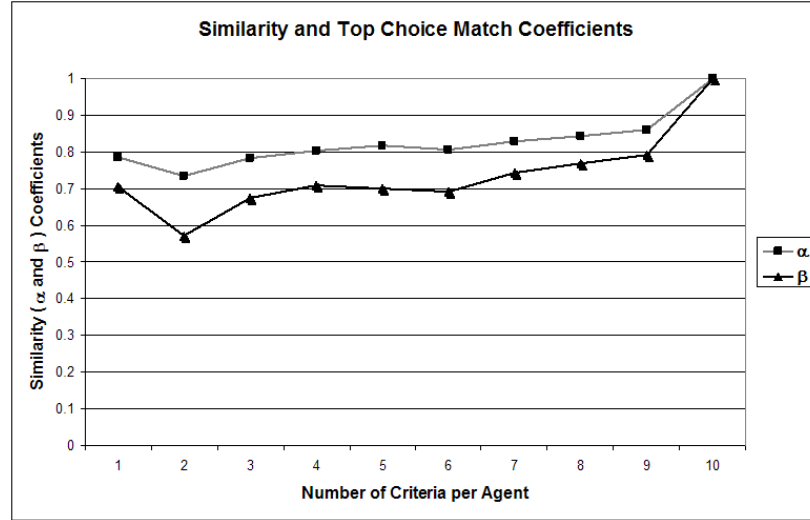


Figure 2.19. Averages of α and β coefficients for `Uncover2` algorithm at ‘Number of Steps’=20

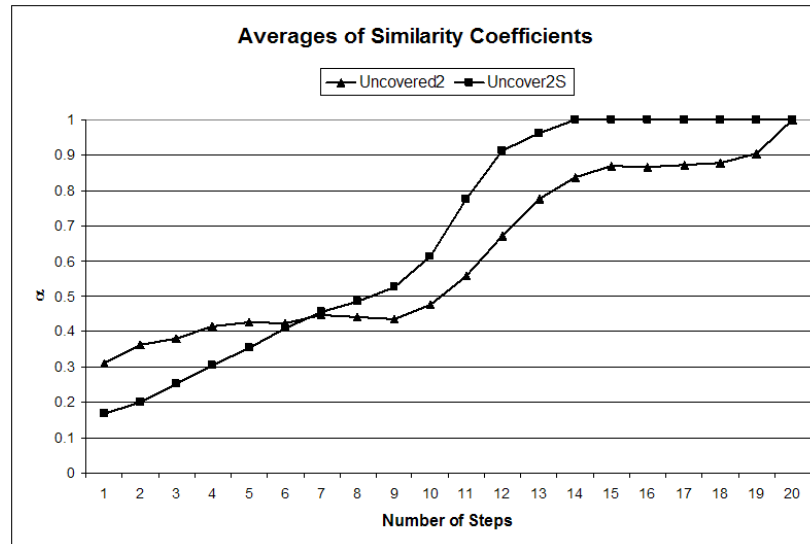


Figure 2.20. Comparing average similarity (α) coefficients of `Uncover2` and `Uncover2S` algorithms at ‘Number of Criteria’=10

2.3.2.5. Richelson’s Rule. A Condorcet vote is conducted and a directed graph where nodes are the alternatives is created. If a and b are two nodes of this directed graph

and $L(x)$ defines lower counter set of x and $D(x)$ defines upper counter set of x then there will be an edge between a and b if

$$aPb \Leftrightarrow (L(a) \supseteq L(b)) \wedge (D(a) \subseteq D(b)) \wedge [(L(a) \supset L(b)) \vee (D(a) \subset D(b))] \quad (2.7)$$

Two algorithms are implemented for Richelson's rule. **Richelson** algorithm uses the **Condorcet** algorithm (Figure 2.21). On the other hand **RichelsonS** algorithm uses **CondorcetS** algorithm (Figure 2.22). Therefore **RichelsonS** does not compare untested alternatives while **Richelson** algorithm regards untested alternatives as least preferred.

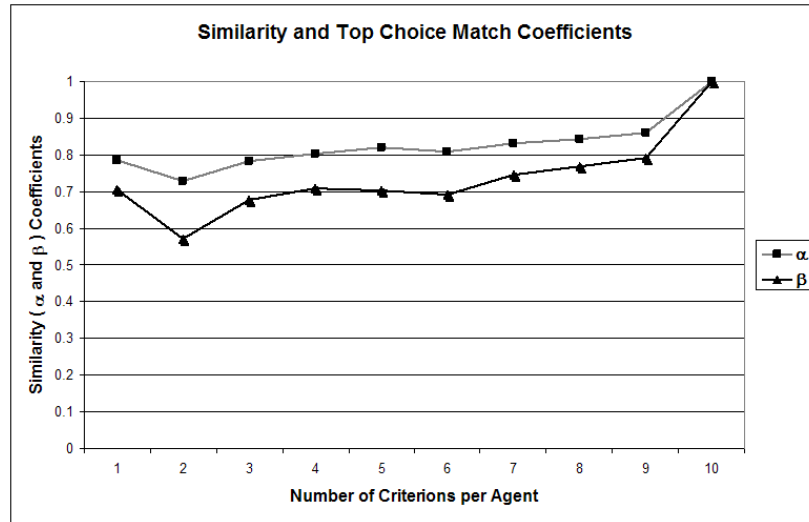


Figure 2.21. Averages of α and β coefficients for **Richelson** algorithm at ‘Number of Steps’=20

There is a strange drop when ‘Number of Steps’ increase from 2 to 10 in Fishburn, uncovered set I, uncovered set II and Richelson rules (Figures 2.16, 2.18, 2.20, and 2.22).

The reason of this drop can be explained using a small example where there are a , b , c , and d alternatives and a majority relation between these alternatives that is given in Figure 2.23 where aPb , aPd , cPd , aIc , bId , and bIc . When Fishburn, uncovered set and Richelson algorithms are applied to (a, c) , (c, b) , and (b, d) the results in Table 2.2 can be found.

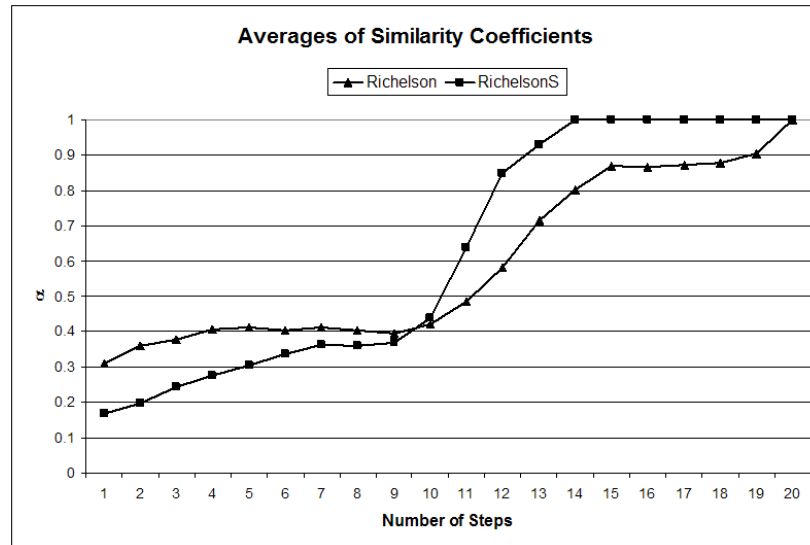


Figure 2.22. Comparing average similarity (α) coefficients of Richelson and RichelsonS algorithms at ‘Number of Criteria’=10

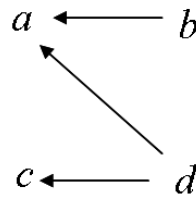


Figure 2.23. A majority relation between a , b , c , and d .

Table 2.2. Results according to Fishburn, uncovered set I, II and Richelson rules

Fishburn	Uncovered Set I	Uncovered Set II	Richelson
aIc	aPc	aIc	aPc
cPb	cPb	cPb	cPb
bPd	bId	bPd	bPd

When ‘Number of Steps’=1 all alternatives will be indifferent to each other since each agent tests only one alternative therefore they will not be able to compare two alternatives. When ‘Number of Steps’ = 2 agents will be comparing only two alternatives and they will be indifferent to all the others. Depending on the random configuration of the alternatives this would mean nearly half of the space will be tested and majority relation will have single edges that are not connected. As the ‘Number of Steps’ increase from 3 to 8 edge groups will start to form patterns like in Figure 2.23. As seen from Table 2.2 in such a case indifference relations at majority relation could be wrongly interpreted with Fishburn, uncovered set I, uncovered set II and Richelson rules. This will create a drop in similarity coefficient. As ‘Number of Steps’ continue to increase majority relation graph will be more connected and similarity coefficient will also increase.

2.3.3. Rules Using Value Function

2.3.3.1. Copeland’s Rule I. Condorcet vote applied and preferences ordered by (incoming degree - outgoing degree) of each alternative.

Two algorithms are implemented for Copeland’s rule I. **CopeNet** algorithm uses the **Condorcet** algorithm. On the other hand **CopeNetS** algorithm uses **CondorcetS** algorithm (Figures 2.24 and 2.25).

2.3.3.2. Copeland ’s Rule II. Condorcet vote applied and preferences ordered by outgoing degree of each alternative.

Two algorithms are implemented for Copeland’s rule II. **CopeOut** algorithm uses the **Condorcet** algorithm (Figure 2.26). On the other hand **CopeOutS** algorithm uses **CondorcetS** algorithm (Figure 2.27).

2.3.3.3. Copeland’s Rule III. Condorcet vote applied and preferences ordered by incoming degree of each alternative.

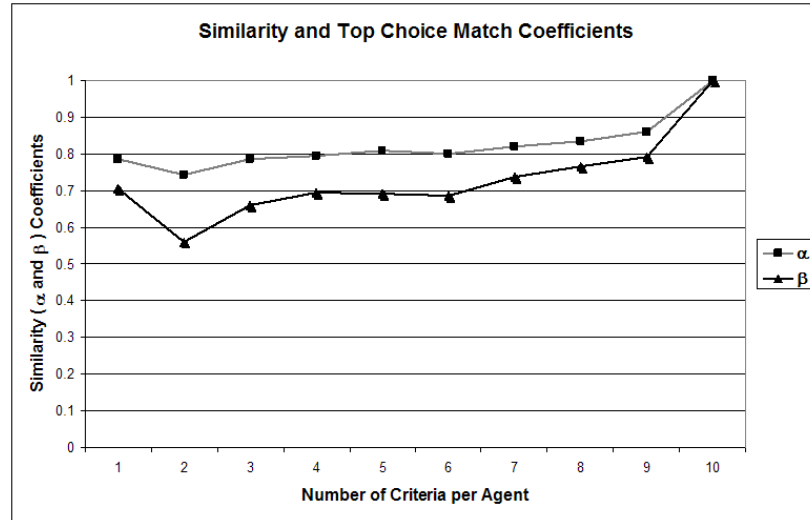


Figure 2.24. Averages of α and β coefficients for CopeNet algorithm at ‘Number of Steps’=20

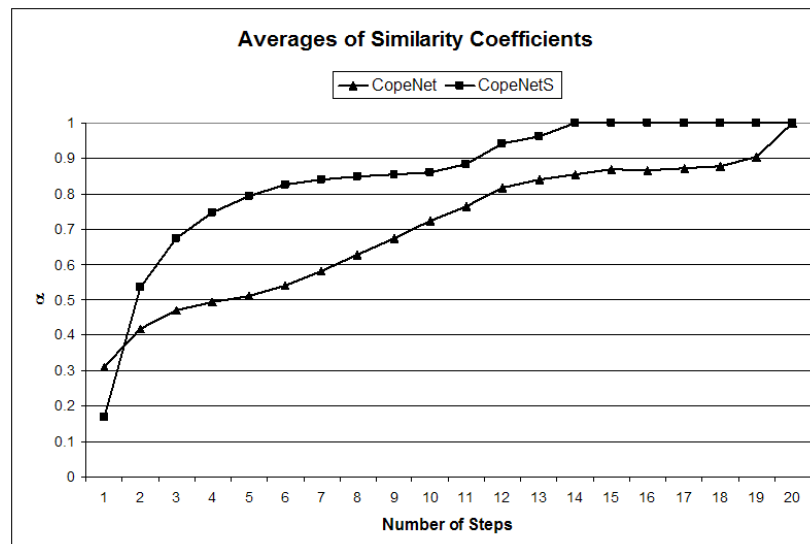


Figure 2.25. Comparing average similarity (α) coefficients of CopeNet and CopeNetS algorithms at ‘Number of Criteria’=10

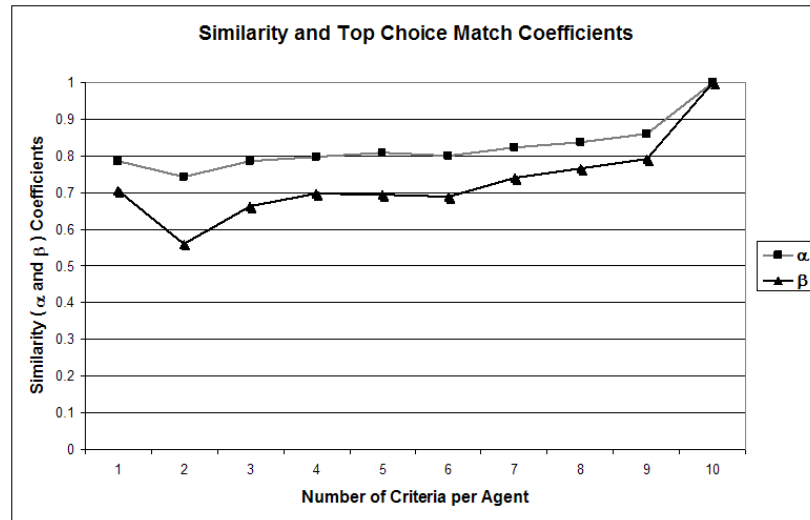


Figure 2.26. Averages of α and β coefficients for CopeOut algorithm at 'Number of Steps'=20

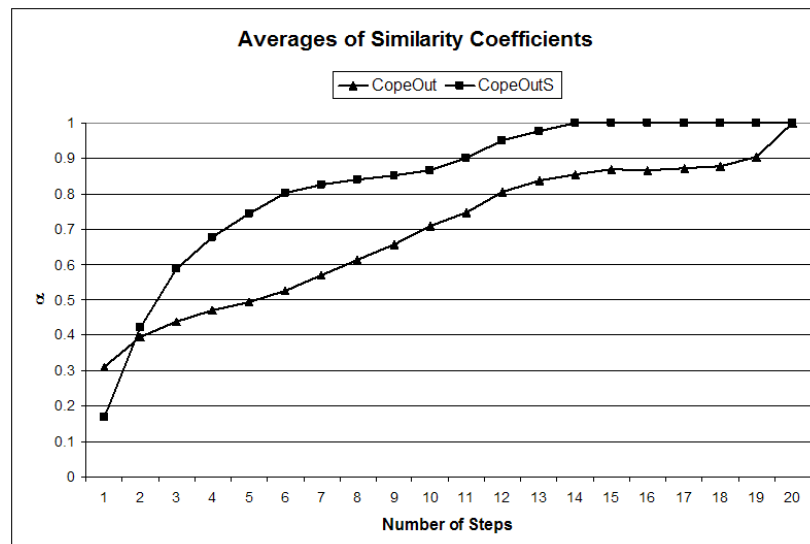


Figure 2.27. Comparing average similarity (α) coefficients of CopeOut and CopeOutS algorithms at 'Number of Criteria'=10

Two algorithms are implemented for Copeland's rule III. CopeIn algorithm uses the Condorcet algorithm (Figure 2.28). On the other hand CopeInS algorithm uses CondorcetS algorithm (Figure 2.29).

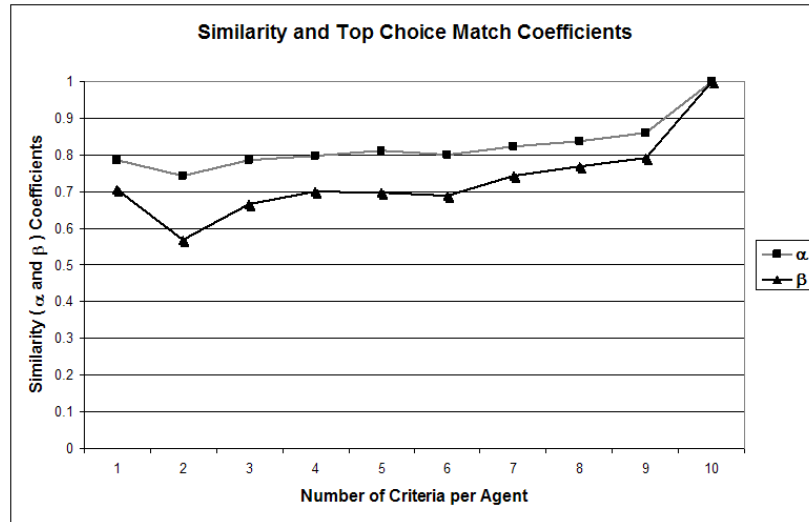


Figure 2.28. Averages of α and β coefficients for CopeIn algorithm at 'Number of Steps'=20

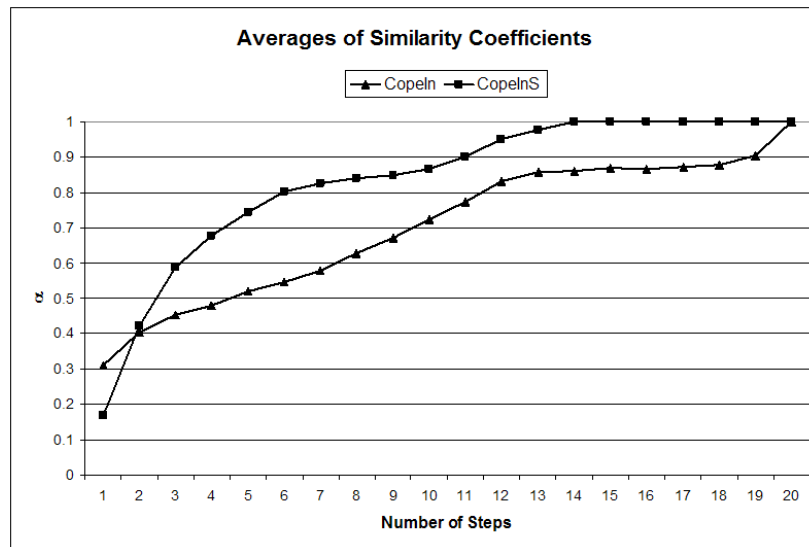


Figure 2.29. Comparing average similarity (α) coefficients of CopeIn and CopeInS algorithms at 'Number of Criteria'=10

2.3.4. Rules Using Tournament Matrix

2.3.4.1. Simpson's Procedure (Maxmin Procedure). A tournament matrix S^+ is constructed.

$$\begin{aligned} \forall a, b \in X, S^+ &= (n(a, b)) \\ n(a, b) &= \text{card}\{i \in N \mid aP_i b\}, \quad n(a, a) = \infty \end{aligned} \quad (2.8)$$

and to find the winner

$$x = \arg \max_{a \in A} \min_{b \in A} (n(a, b)) \quad (2.9)$$

To find the rest of the ordering, remove the winner(s) from the profile and repeat the procedure.

Figure 2.30 shows the results for Maxmin procedure.

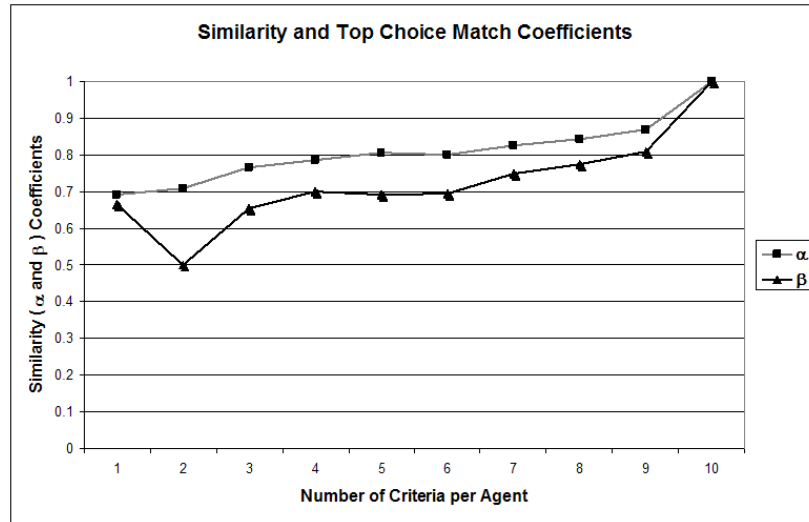


Figure 2.30. Averages of α and β coefficients for Maxmin algorithm at 'Number of Steps'=20

As can be seen Maxmin method would return all the alternatives as the winners because of 0 values of aPc and cPa counts.

2.3.4.2. Minmax Procedure. Construct matrix S^+ such that

$$\begin{aligned} \forall a, b \in X, S^- &= (n(a, b)) \\ n(a, b) &= \text{card}\{i \in N \mid aP_i b\}, \quad n(a, a) = -\infty \end{aligned} \quad (2.10)$$

and to find the winner

$$x = \arg \min_{b \in A} \max_{a \in A} (n(a, b)) \quad (2.11)$$

To find the rest of the ordering, remove the winner(s) from the profile and repeat the procedure.

Results for **Minmax** are exactly the same for ‘Number of Steps’=20 case (Figure 2.30. However **Minmax** is a better performing method compared to **Maxmin**. Figure 2.31 shows comparison of both methods for ‘Number of Criteria’=7 case.



Figure 2.31. Averages of α and β coefficients for **Minmax** algorithm at ‘Number of Steps’=20

2.3.5. q-Paretian Rules

2.3.5.1. Strong q-Paretian Simple Majority Rule. The social choice rule is defined as: select the alternative which is in between top alternatives for each agent in at least one simple majority coalition. If there is no such alternative, then the social choice is defined by increasing q by 1 until it is not empty. The ordering is done by sorting for q .

$$C(A) = \bigcup_{I \in \mathfrak{S}} \bigcap_{i \in I} f(\vec{P}; i; q) \quad (2.12)$$

Where $f(\vec{P}; i; q) = \{x \in A \mid \text{card}(D_i(x)) \leq q\}$, $\vec{P}$ is the preference profile, I is a simple majority and $\mathfrak{S}$ is the family of simple majority coalitions.

StrongQ algorithm counts all the agents when calculating the simple majorities even if an agent didn't test the alternative under question (Figure 2.32). **StrQSkip** algorithm takes only the agents that have tested the alternative for a simple majority. This algorithm skips agents that didn't test an alternative when a simple majority is being constructed for that alternative. It can be seen from Figure 2.33 that **StrQSkip** algorithm is able to find exact solutions ($\alpha = 1.0$) at small 'Number of Steps' values when 'Number of Criteria' is 10.

2.3.5.2. Strong q-Paretian Plurality Rule. This rule is similar to rule 2.3.5.1 with the following addition. If several options are chosen for a given q , then for each alternative number of coalitions that choose this alternative will be counted. Then alternatives will be ordered according to this index within each q .

Similar to strong q-Paretian simple majority rule there are two implementations of this rule. **StrongQP** algorithm takes all agents when calculating the simple majorities even if they haven't tested the alternative (Figure 2.34). On the other hand **StrQPSkip** algorithm skips the agents when calculating the simple majority if they haven't tested the alternative under question. Figure 2.35 shows a comparison of both rules.

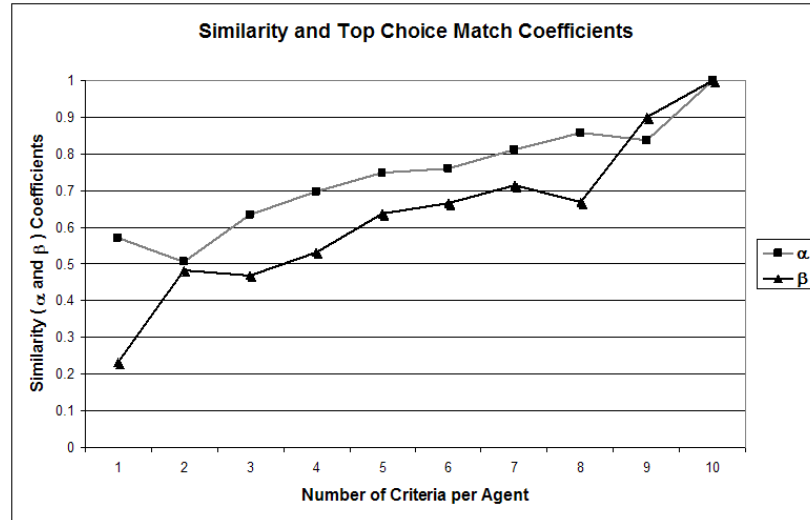


Figure 2.32. Averages of α and β coefficients for StrongQ algorithm at ‘Number of Steps’=20

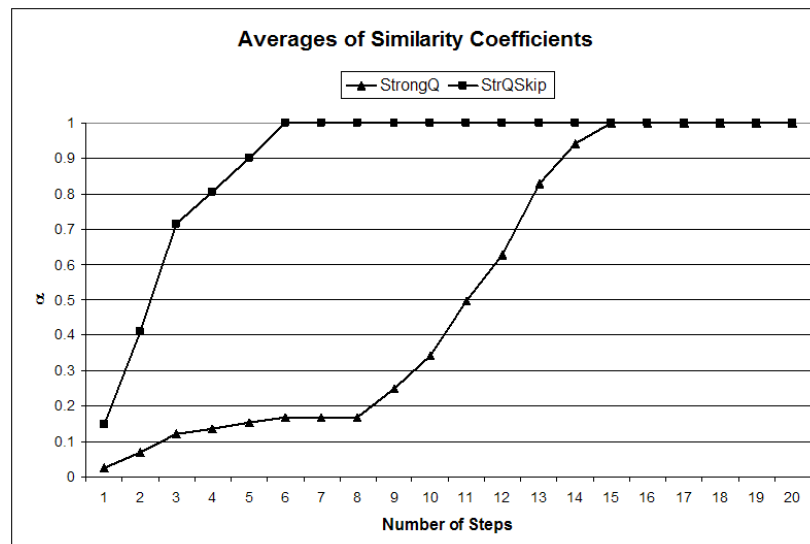


Figure 2.33. Comparing average similarity coefficients of StrongQ and StrQSkip algorithms at ‘Number of Criteria’=10

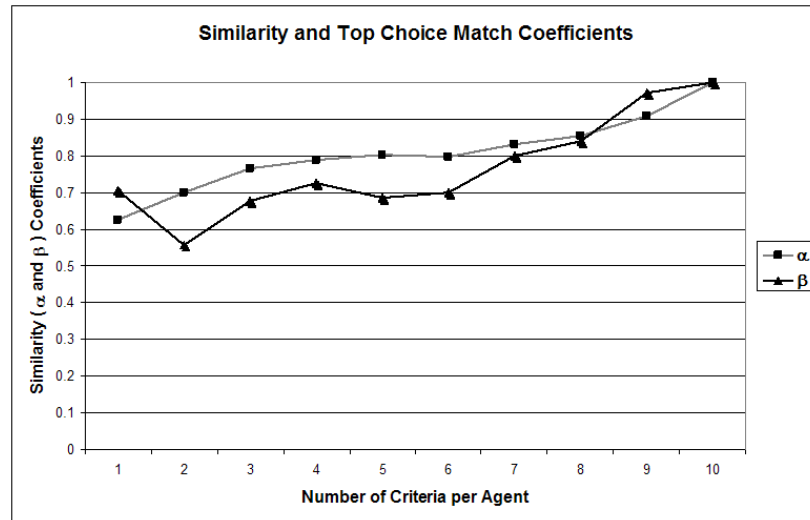


Figure 2.34. Averages of similarity coefficients for StrongQ algorithm at ‘Number of Steps’=20

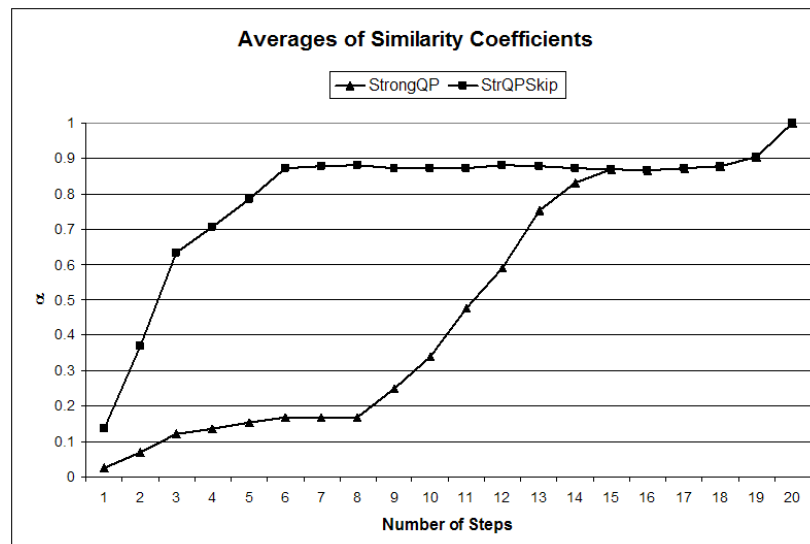


Figure 2.35. Comparing average similarity coefficients of StrongQ and StrQSkip algorithms at ‘Number of Criteria’=10

2.3.5.3. Strongest q-Paretian Simple Majority Rule. The social choice is the alternative which is Pareto optimal in each simple majority coalition with $q = 0$. If there is no such alternative then $q = 1, q = 2$, etc. is considered. Ordering is done by sorting by q .

$$C(X) = \bigcap_{I \in \mathfrak{S}} f(\vec{P}; I; q) \quad (2.13)$$

Where $f(\vec{P}; I; q) = \left\{ x \in A \mid \text{card}\left(\bigcap_{i \in I} D_i(x)\right) \leq q \right\}$, $\vec{P}$ is the profile and I is the simple majority coalition.

Strongest q-Paretian simple majority rule uses all simple majority coalitions. If the number of agents are n then the number of coalitions to be calculated can be found with the following equation.

$$\text{card}(\mathfrak{S}) = \frac{n!}{(n/2 - 1)!} \quad (2.14)$$

Number of all possible coalitions quickly becomes very large as the size of population increases. Strongest q-Paretian simple majority rule is not practical in these cases.

There are three implementations for strongest q-Paretian simple majority rule. **StrgestQA** algorithm calculates all possible simple majorities. **StrgestQ** algorithm takes only 100 simple majorities randomly. **StrgestQAS** algorithm again calculates all simple majorities but it skips the agents that haven't tested the alternative in question. Algorithms **StrgestQAS** and **StrgestQA** gives the same results for 'Number of Steps' = 20 therefore only **StrgestQA** and **StrgestQ** are shown in Figures 2.36 and 2.37. Figure 2.38 compares all three rules at 'Number of Criteria'=10.

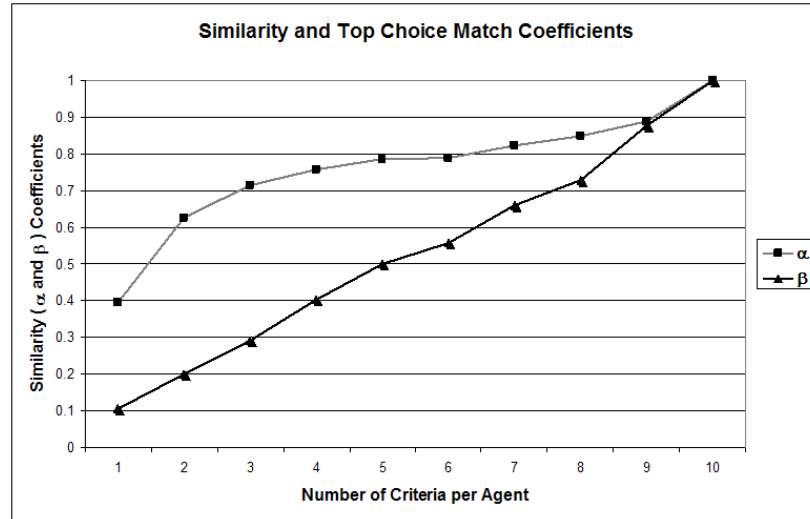


Figure 2.36. Averages of similarity (α) coefficients for StrgestQA and StrgestQ algorithm at 'Number of Steps'=20

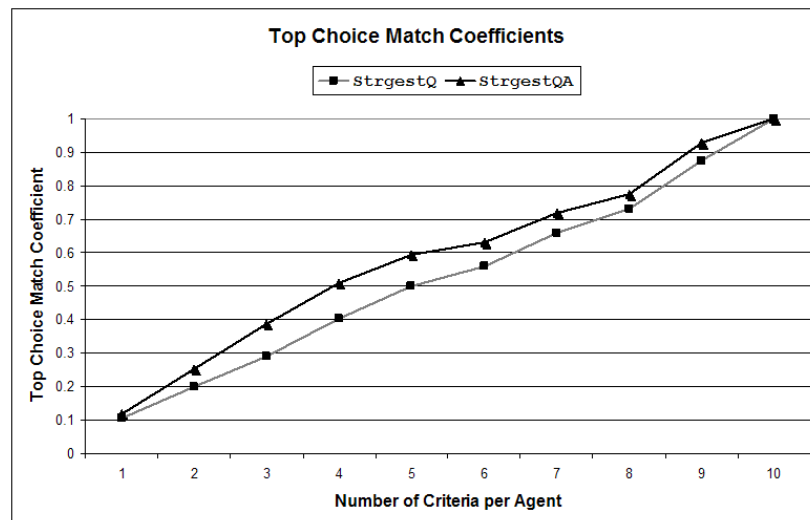


Figure 2.37. Averages of similarity (α) coefficients for StrgestQA and StrgestQ algorithm at 'Number of Steps'=20

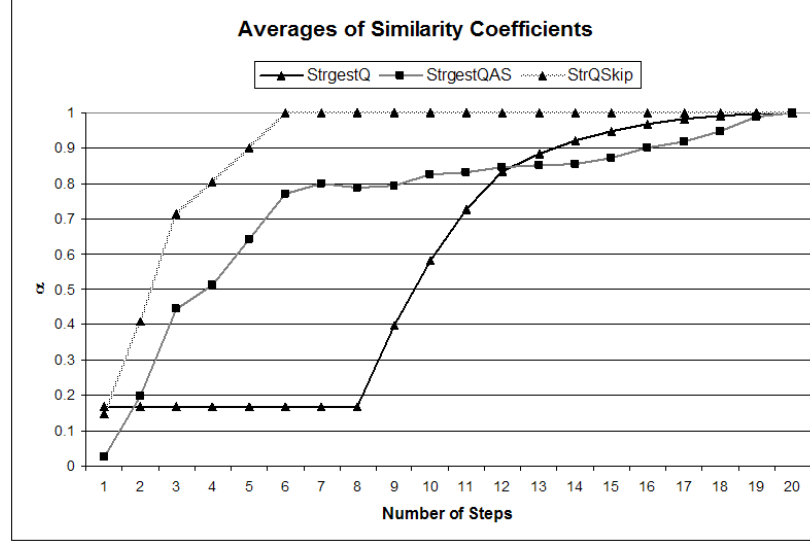


Figure 2.38. Comparing average similarity coefficients of **StrgestQ**, **StrgestQA**, and **StrgestQAS** algorithms at ‘Number of Criteria’=10

2.4. Conclusion

Tables 2.3 and 2.4 show the overall winners for each ‘Number of Step’ and ‘Number of Criterion’ of α and β respectively. Bold cells in both tables indicate exact matches. When averages of all the number of steps and all the number of criteria are calculated it can be seen that **CopeNetS** algorithm is the best performing algorithm for α and **StrQSkip** for β (Table 2.5). In pattern recognition usually all algorithms test the whole space, therefore averages of all number of criteria at number of steps = 20 are also calculated (Table 2.6). In this case the winners are **Sum** rule for α and **StrQPSkip** for β . It can be seen from Table 2.3 that for ‘Number of Steps’ = 20, Strong q-Paretian Simple Majority and Strong q-Paretian Plurality methods perform better on α when number of criteria per agent increases. Same social choice rules also perform better on β when ‘Number of Steps’ = 20 and number of criteria increase (Table 2.4).

Another result of this study is that if number of criteria is above a certain value then there is no need for agents to check all the search space for some of the social choice rules. Figure 2.39 shows that similarity coefficients of **StrQSkip**, **Coomb** and **CopeNetS** algorithms does not improve as ‘Number of Steps’ is increasing when ‘Number of Criteria’=7. If these agents were pattern recognition algorithms with similar

Table 2.3. Best performing social choice algorithms according to α for each number of criterion and number of steps.

		NUMBER OF CRITERIONS PER EACH AGENT									
		1	2	3	4	5	6	7	8	9	10
NUMBER OF STEPS IN SEARCH SPACE	1	Nanson	Nanson	Nanson	Hare	Hare	Hare	Hare	Hare	Hare	Hare
	2	Nanson	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS
	3	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip
	4	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip
	5	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip
	6	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	StrQSkip
	7	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	StrQSkip	StrQSkip
	8	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	StrQSkip	StrQSkip
	9	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	StrQSkip	StrQSkip
	10	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	Minmax	StrQSkip
	11	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	Minmax	StrQSkip
	12	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	StrQSkip	Minmax	StrQSkip
	13	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeOutS	CopeInS	Minmax	StrQSkip
	14	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeOutS	FishburnS	RichelsonS	RichelsonS	Group 1
	15	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeOutS	CopeOutS	FishburnS	RichelsonS	StrongQ	Group 2
	16	CopeNetS	CopeNetS	CopeNetS	CopeNetS	CopeInS	CopeInS	RichelsonS	RichelsonS	StrongQ	Group 2
	17	CopeNetS	CopeNetS	CopeNetS	CopeOutS	CopeOutS	FishburnS	RichelsonS	RichelsonS	RichelsonS	Group 2
	18	CopeNetS	CopeNetS	CopeNetS	CopeOutS	FishburnS	RichelsonS	RichelsonS	RichelsonS	StrgestQA	Group 2
	19	CopeNetS	CopeNetS	Sum	Uncover1S	RichelsonS	RichelsonS	RichelsonS	StrQSkip	StrgestQAS	Group 2
	20	Group4	Group 5	Sum	Sum	Group 6	Group 6	Sum	Group 7	Group 8	Group 3

Group 1: StrQSkip, CopeNetS, CopeOutS, CopeInS, Minmax

Group 2: Group1, StrongQ

Group 3: Group 2, Borda, Sum, Nanson, CoombS, Fishburn, Uncover1, Uncover2, Richelson, FishburnS, Uncover1S, Uncover2S, RichelsonS, CopeNet, CopeOut, CopeIn, StrongQP, StrQPSkip, StrgestQ, StrgestQA, StrgestQAS, Maxmin

Group 4: Approval, Borda, Sum, CoombS, Fishburn, Uncover1, Uncover2, Richelson, FishburnS, Uncover1S, Uncover2S, RichelsonS, CopeNet, CopeOut, CopeIn, CopeNetS, CopeOutS, CopeInS

Group 5: Borda, Sum, Nanson

Group 6: Richelson, RichelsonS

Group 7: StrongQ, StrQSkip

Group 8: StrongQP, StrQPSkip

Table 2.4. Best performing social choice algorithms according to β for each number of criterion and number of steps.

		NUMBER OF CRITERIONS PER EACH AGENT									
		1	2	3	4	5	6	7	8	9	10
NUMBER OF STEPS IN SEARCH SPACE	1	Group 1	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	2	Borda	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	3	Borda	CopeNetS	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	4	Borda	CopeNetS	CopeNetS	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	5	Borda	Minmax	CopeNetS	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	6	Minmax	CopeNetS	CopeNetS	Minmax	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	7	Minmax	Minmax	StrQSkip	Minmax	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	8	Minmax	Group 2	CopeInS	Minmax	Minmax	StrQSkip	StrQSkip	StrQSkip	StrQSkip	StrQSkip
	9	Minmax	CopeInS	CopeInS	Minmax	Minmax	Minmax	StrQSkip	Minmax	Minmax	StrQSkip
	10	CopeNetS	CoombS	CopeInS	Minmax	Minmax	CopeInS	StrQSkip	Minmax	Minmax	StrQSkip
	11	Minmax	CopeInS	CopeInS	CopeInS	Minmax	CopeInS	StrQSkip	Minmax	Minmax	StrQSkip
	12	Minmax	CopeInS	CopeInS	CopeInS	Minmax	CopeInS	StrQSkip	Minmax	Minmax	StrQSkip
	13	Minmax	CopeNetS	CopeNetS	CopeNetS	CopeInS	CopeInS	Minmax	StrongQ	StrongQ	StrQSkip
	14	Minmax	CopeOutS	CopeOutS	CopeOutS	CopeOutS	FishburnS	FishburnS	StrongQ	StrongQ	Group 3
	15	Minmax	CopeInS	CopeOutS	CopeOutS	FishburnS	FishburnS	RichelsonS	StrongQ	StrongQ	Group 4
	16	Minmax	CopeInS	FishburnS	FishburnS	FishburnS	FishburnS	RichelsonS	CondorcetS	StrongQ	Group 4
	17	CopeInS	FishburnS	FishburnS	FishburnS	FishburnS	RichelsonS	RichelsonS	CondorcetS	StrongQ	Group 4
	18	CopeInS	FishburnS	FishburnS	FishburnS	RichelsonS	RichelsonS	CondorcetS	CondorcetS	StrongQ	Group 5
	19	FishburnS	FishburnS	FishburnS	RichelsonS	RichelsonS	RichelsonS	StrongQP	StrongQP	StrongQ	Group 5
	20	Approval	Borda	Borda	StrQPSkip	Borda	StrQPSkip	StrQPSkip	StrQPSkip	StrongQP	Group 6

Group 1: Plurality, Borda, Sum, InvBorda, Condorcet, Fishburn, Uncovered1, Uncovered2, Richelson, FishburnS, Uncovered1S, Uncovered2S, RichelsonS, CopeNet, CopeOut, CopeIn

Group 2: Approval, CoombS

Group 3: CondorcetS, CopeNetS, CopeOutS, CopeInS, StrQSkip, Minmax

Group 4: Group3, StrongQ, StrgestQA

Group 5: Group 4, StrgestQAS

Group 6: Group4, Approval, Borda, Sum, Nanson, CoombS, Condorcet, Fishburn, Uncovered1, Uncovered2, Richelson, FishburnS, Uncovered1S, Uncovered2S, RichelsonS, CopeNet, CopeOut, CopeIn, StrongQP, StrQPSkip, Maxmin

classifiers (as 7/10) then checking all the search space would not be improving the results significantly.

Table 2.5. Averages of α and β coefficients over all ‘Number of Criteria’ and all
‘Number of Steps’

Averages of Averages

Similarity		Top Choice Match	
SC Rule	Avr. α	SC Rule	Avr. β
CopeNetS	71.125%	StrQSkip	51.431%
CopeInS	69.173%	Minmax	50.945%
CopeOutS	69.168%	CopeInS	49.718%
Nanson	66.126%	CopeNetS	49.143%
StrQPSkip	64.138%	CopeOutS	47.567%
CopeNet	63.762%	StrQPSkip	47.011%
CopeIn	63.749%	CoombS	46.591%
Minmax	63.558%	Approval	46.476%
Coomb	63.336%	Uncover1S	46.053%
StrQSkip	63.233%	Nanson	42.733%
CopeOut	62.921%	InvBorda	42.586%
Sum	62.738%	Plurality	42.055%
Borda	62.524%	FishburnS	41.884%
InvBorda	62.220%	Uncover2S	41.884%
CoombS	61.907%	Coomb	41.530%
Hare	61.740%	CopeIn	41.243%
RunOff	60.934%	Uncovered1	41.122%
Uncovered1	58.542%	Fishburn	40.806%
Fishburn	58.249%	Uncovered2	40.806%
Uncovered2	58.249%	Richelson	40.618%
StrgestQAS	58.219%	RichelsonS	40.556%
Condorcet	57.388%	Sum	38.865%
Richelson	56.970%	CopeNet	38.779%
Uncover1S	52.516%	CopeOut	38.365%
FishburnS	52.512%	Borda	37.883%
Uncover2S	52.512%	Hare	37.782%
Approval	51.538%	Condorcet	35.899%
RichelsonS	48.990%	RunOff	35.773%
CondorcetS	48.281%	StrongQP	34.654%
StrgestQA	45.792%	StrongQ	34.484%
Plurality	44.482%	StrgestQAS	32.448%
StrgestQ	43.980%	CondorcetS	31.796%
StrongQP	41.311%	StrgestQA	30.994%
StrongQ	40.240%	Maxmin	30.905%
Maxmin	39.242%	StrgestQ	26.570%
InvPlrty	36.114%	InvPlrty	10.722%

Table 2.6. Averages of α and β coefficients over all ‘Number of Criteria’ and at

‘Number of steps’ = 20

Averages of Averages at 20

Similarity		Top Choice Match	
SC Rule	Avr. α	SC Rule	Avr. β
Sum	83.152%	StrQPSkip	77.124%
Borda	83.149%	StrongQP	76.658%
Nanson	82.731%	Borda	74.652%
InvBorda	82.724%	Sum	74.652%
Richelson	82.593%	Richelson	73.600%
RichelsonS	82.593%	RichelsonS	73.600%
Fishburn	82.573%	Fishburn	73.547%
Uncovered2	82.573%	Uncovered2	73.547%
FishburnS	82.573%	FishburnS	73.547%
Uncover2S	82.573%	Uncover2S	73.547%
Uncovered1	82.569%	Nanson	73.502%
Uncover1S	82.569%	InvBorda	73.499%
CopeIn	82.424%	Uncovered1	73.444%
CopeInS	82.424%	Uncover1S	73.444%
CopeOut	82.421%	Condorcet	73.315%
CopeOutS	82.421%	CondorcetS	73.315%
Coomb	82.387%	CopeIn	73.285%
CopeNet	82.298%	CopeInS	73.285%
CopeNetS	82.298%	CopeOut	73.052%
CoombS	81.996%	CopeOutS	73.052%
Maxmin	80.939%	CopeNet	72.916%
Minmax	80.939%	CopeNetS	72.916%
StrQPSkip	80.827%	Coomb	72.427%
StrongQP	80.721%	CoombS	72.427%
Condorcet	79.737%	Maxmin	72.384%
CondorcetS	79.737%	Minmax	72.384%
StrgestQAS	78.225%	Approval	72.032%
StrgestQA	78.213%	StrongQ	63.006%
RunOff	76.743%	StrQSkip	63.006%
StrgestQ	76.268%	Hare	59.610%
Hare	74.912%	StrgestQAS	59.215%
StrongQ	74.181%	StrgestQA	59.171%
StrQSkip	74.181%	RunOff	54.893%
Approval	59.046%	Plurality	54.727%
Plurality	43.089%	StrgestQ	53.210%
InvPlrty	32.099%	InvPlrty	10.061%

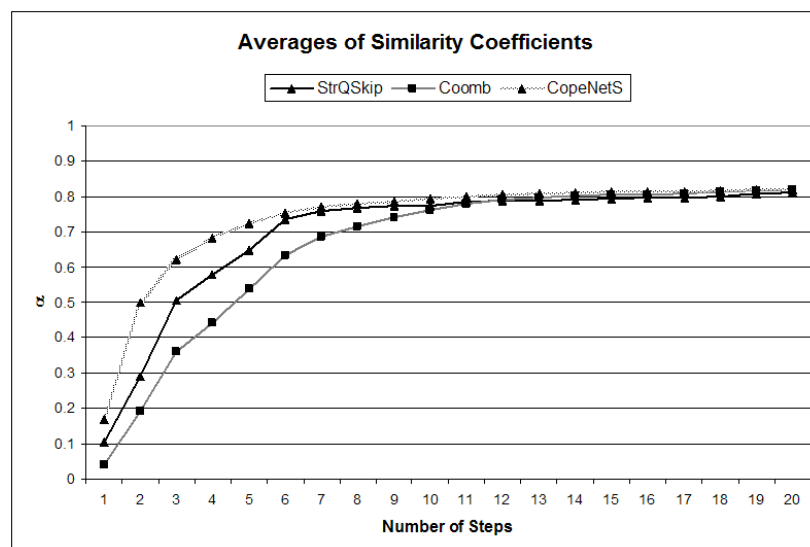


Figure 2.39. Comparing averages of similarity coefficients for StrQSkip, Coomb, and CopeNetS at ‘Number of Criteria’=7

3. STOCHASTIC DIFFUSION SEARCH

3.1. Introduction

It is not possible to find a solution of a combinatorial search with a systematic search method in a reasonable time frame. Therefore many stochastic local search methods had been proposed.

Parameterized stochastic local search methods, simulated annealing (Kirkpatrick *et al.*, 1983) and tabu search (Glover, 1989) are single agent approaches to this problem.

Population based methods use several agents to do the search with competitive and cooperative rules that govern their relations. These methods are gaining popularity due to increased use of parallel computers. These methods include genetic algorithms (Mitchell, 1998), ant foraging algorithms (Coloni *et al.*, 1991), memetic algorithms (Moscato, 1989) and particle swarm optimization (Kenedy and Ebehart, 1995).

Stochastic Diffusion Search (SDS) is a population based search algorithm introduced by Bishop (1989a). The original idea of SDS comes from Hinton Mapping (Hinton, 1981) however it has similarities to nest search of ants and honeybees where insects communicate with each other by tandem running or waggle dance (Franks *et al.*, 2002).

It consists of agents that test one location at a time with one partial criterion from a decomposable criterion set. If a solution exists in the search space agents quickly converge to that location (Nasuto and Bishop, 1999). An analogy of SDS is given as follows (De Meyer *et al.*, 2003):

A group of delegates attends a long conference in an unfamiliar town. Each night they have to find somewhere to dine. There is a large choice of restaurants, each of which offers a large variety of meals. The problem the group faces is to find the best restaurant, that is the restaurant where the maximum number of delegates would enjoy dining. Even a parallel exhaustive search through the

restaurant and meal combinations would take too long to accomplish. To solve the problem delegates decide to employ a Stochastic Diffusion Search.

Each delegate acts as an agent maintaining a hypothesis identifying the best restaurant in town. Each night each delegate tests his hypothesis by dining there and randomly selecting one of the meals on offer. The next morning at breakfast every delegate who did not enjoy his meal the previous night, asks one randomly selected colleague to share his dinner impressions. If the experience was good, he also adopts this restaurant as his choice. Otherwise he simply selects another restaurant at random from those listed in ‘Yellow Pages’.

SDS has a simple cycle of testing and diffusion until the halting criteria is matched. Each agent keeps a part of the criteria that is being sought and a hypothesis about the location of where this criteria would be satisfied in the search space. During the test phase agents test their hypothesis with their partial criteria. Agents are marked as active if this test returns true. At diffusion phase inactive agents randomly select another agent. The inactive agent copies the hypothesis of the other if the other is active. If the other agent is also inactive then this agent assumes another random hypothesis. There are other diffusion strategies and agent configurations that are already proposed and tested in the literature (De Meyer, 2003).

By changing their hypothesis, agents are performing a kind of voting in original SDS. When the processes halts, hypothesis which is held by the the most number of active agents is selected as the winner. This voting process resembles plurality vote.

Chapter 2 of this thesis showed that plurality vote is very limited in performance. In this chapter a framework to test SDS will be given followed by a short discussion of original SDS algorithm. Finally alternative voting schemes will be discussed and performance of new SDS algorithms will be compared to original SDS using a string search and an image search testing framework.

3.2. Testing Framework

3.2.1. String Search Framework

A string search similar to the test framework in Chapter 2 is used for testing different algorithms.

Search spaces and the search key that was generated randomly for the voting experiments of Chapter 2 were used. Tests are made for 10,000 search spaces of 20 characters long and one search key of 10 characters long.

Different variants of SDS algorithms are then tested for their speed and performance of matching the correct ordering. Similarity coefficient and top choice match coefficient are used (Equations 2.3 and 2.4) to measure the correctness.

3.2.2. Image Search Framework

Image search is made by using 640×480 pixel pictures provided by the Computer Vision Laboratory, University of Ljubljana, Slovenia¹ (Solina *et al.*, 2003).

Serious expression, smiling with and without showing teeth images of the first ten individuals in the library are converted to 8-bit gray scale. Right eye sections framed in a 50×40 pixel rectangle of serious expression pictures of the first five individuals are used as models and these models are searched in the three pictures each of first ten individuals. Therefore a total of 150 tests are made at each image test.

The averages of the distances of the results (if there are more than one results) are compared with the correct locations of the right and the left eye for each image.

¹CVL Face Database license agreement requires the database to be downloaded from: <http://www.lrv.fri.uni-lj.si/facedb.html>

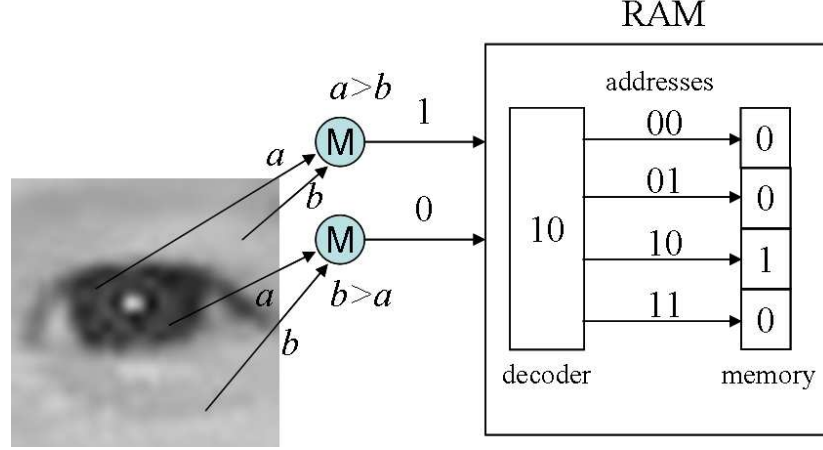


Figure 3.1. Results of Minchinton Cells are used as addresses for RAM

$$d = \frac{\sum_{i \in A} [(x_r - x_i)^2 + (y_r - y_i)^2]^{1/2}}{n} \quad (3.1)$$

where x_r and y_r are known coordinates of the eye location; A is the set of results found by the social choice rule; x_i and y_i are member coordinates of A ; n is the number of locations returned by the social choice rule.

This average is introduced in order to catch possible multiple correct answers. However it is seen that all of the correct matches in this thesis were single results.

If the distance is less than or equal to 16 pixels which is $1/4$ of the 50×40 frame diagonal, it is considered to be a correct match.

The search criteria are created using Minchinton cells (Bishop, 1990). A Minchinton cell compares the gray level of two locations in the picture. If the first location is greater than the second it returns 1 otherwise returns 0. Results of Minchinton cells are then placed in a RAM (Random Access Memory) where address lines are the results of Minchinton cell (Figure 3.1).

In the image case criteria of agents are RAMs. RAMs are initialized with the

model images taken from the serious looking images of the first five individuals. During the search each agent tests its hypothesis with its RAM. If the hypothesis returns true this would mean that Minchinton cells in this hypothesis are giving the same results as the model.

Using more Minchinton Cells in a RAM would make the criteria more strict, therefore in this study one Minchinton cell per RAM is used.

3.3. Original SDS

Original SDS consists of the following steps (Bishop, 1989b).

1. Initialize agents
2. Start Loop
 - (a) Test Hypothesis of agents
 - (b) Diffuse agents
3. Until halting criteria achieved
4. Extract solution

Agents are initialized with the search criteria and each one randomly assumes a hypothesis about the location of the solution. These hypotheses are tested and the agents who found that their hypothesis is satisfied are named active while others are named inactive. Inactive agents randomly chose another agent from the population. If the chosen agent is active then its hypothesis is copied and both agents test another part of the hypothesis (chooses new meal from the menu as in the above example). If the chosen agent is also inactive both agents assume new hypothesis randomly. This is done repeatedly until a halting criterion is reached

3.3.1. Agent Properties

Each agent has a hypothesis about the location of the solution and a test procedure for the evaluation of the problem. A state variable indicating whether the agent is

active or not is altered with this test procedure. This variable could be Boolean to indicate whether the agent is active or not or a real value between 0 and 1 in optimization problems.

Agents also keeps a diffusion procedure describing how to retain or alter the hypothesis when it is communicating with another agent. Agents can be made to communicate with another agent form the entire population or only its neighbors in a graph. Studies with small world and lattice shaped networks show that these shapes are giving results close to fully connected shapes (De Meyer, 2003).

3.3.2. Diffusion

In the diffusion phase inactive agents randomly select another agent and check its status. There are different types of diffusion strategies.

Passive diffusion lets the inactive agent to chose another agent at random and copies its hypothesis if the other is active. Active diffusion uses active agents for selecting random agents. On the other hand mixed diffusion uses both active and inactive agents during the diffusion phase.

3.3.3. Halting Criteria

Several halting criteria options are possible.

- a. User monitors the program and decides when to stop.
- b. Program stops when a pre determined time interval or number of iterations is reached.
- c. If the number of agents that are active is above a threshold, program stops.
- d. Program stops when agents are densely populated in a cluster.
- e. To keep the parallel nature of the program stopping agents could be introduced.

Stopping agents monitor a random sample of search agents and compare the hypothesis of these agents. If stopping agents found that hypothesis of random

sample are the same then they would stop the process.

3.3.4. Variants of SDS in Literature

Several variations on SDS had been proposed in the literature. Grech-Cini (1995) proposed secret optimization where agents keep their hypothesis even they are failed. A new category of agents called hermit agents keep good hypothesis as secret for a while.

Nasuto (1999) proposed context-free and context sensitive SDS. In context-free SDS each active agent communicates with one randomly chosen agent and if other agent is also active turns itself to inactive. Context-sensitive SDS additionally requires that both agents to have the same hypothesis. Context free SDS ensures that population of active agents will be maximum 50 per cent of the total population. In other words 50 per cent of agents will continue to search for new hypothesis (Jones, 2003).

Beattie (2000) proposes a focused SDS where process does not attempt to find the solution directly but first looks at the possible solutions and later focuses in these possible solutions in detail.

3.3.5. Applicability of SDS

SDS can only work with decomposable criteria. It is different from most other search and optimization algorithms in this respect.

De Meyer (2003) gives conditions of when to use and not use SDS as follows.

- Criteria function decomposability. SDS requires functions that are decomposable. If the criteria function is not decomposable the diffusion phase in SDS will be very costly.
- Computational difficulty of criteria function. SDS is more successful with rich criteria functions. If the criteria function is relatively simple or have less number

of components it will be costly to use SDS.

- Criterion similarity. If criterions are similar to each other then satisfying one criterion will be predictive that the whole criteria is satisfied. Therefore highly similar criteria is not suitable for SDS.
- Signal to noise ratio. Signal to noise ratio is the maximum score the search space can give divided by the average score. If signal to noise ratio is low SDS will not be able to converge to the correct choice in a reasonable time. SDS requires a high signal to noise ratio.

3.3.6. Comparison of SDS with Other Search Methods

SDS is a motive based search method like genetic algorithms, local search methods and random or systematic search methods. Motive search methods search for the solution with a search strategy or search operator. Variants of SDS can be made to include step that construct a solution in each steps similar to agent colony optimization algorithms (De Meyer, 2003).

The most distinguishing property of SDS is that its agents test only parts of the criteria at each step. Partial criteria testing is the core characteristic of efficiency of SDS however it is also its weakness since some of the functions cannot be easily optimized with partial evaluations.

SDS is very similar to genetic algorithms. Genetic algorithms search for the solution by generating random solutions and using the best of such solutions to obtain the next generation of solutions with cross-over and mutation. Genetic algorithms allow partial evaluations but this feature is rarely used in practice.

A detailed comparison of SDS and other search methods can be found in De Meyer's thesis. (2003).

3.3.7. Applications of SDS

SDS was first applied to find eyes in human faces by Bishop and Torr (1992). Bishop and Torr were able to match 60 per cent of the cases in finding the locations of eyes when the face of a different individual were given.

A SDS and hill climbing combination is used to detect eye and nose area tracking in video images with success (Grech-Cini, 1995).

A variant of SDS is used for self localisation of an autonomous wheelchair. In most demanding real world scenerios algorithm was successful 58 per cent of the time (Beattie, 2000).

Hurley and Whitaker (2002) used SDS for site selection for wireless networks where partial evaluation of criteria was a big advantage.

3.4. SDS with Shared Memory

SDS algorithm finishes with a solution extraction phase where the location which is held by the most number of active agents is selected as the winner. This voting process is similar to plurality voting. In plurality voting, agents vote for their best choices. However since agents do not have a memory in original SDS their votes are counted only if they are active.

To overcome this limitation of original SDS a blackboard (Engelmore and Morgan, 1988) implementation is created. Although a shared memory allows the use of only a few social choice rules it is a very simple addition to the original.

A shared memory is used as a blackboard where all agents mark their findings after each test phase. The rest of the algorithm is based on context free SDS. Context free SDS ensures at least 50 per cent of the agents to continue the search by limiting convergence to a solution. This gives an added benefit in shared memory case, since

as the search continues in different locations of search space, density of marks will be increasing on the blackboard for the locations that agents are converging to. Therefore as the search continues a better evaluation of the search space will be made. In original SDS this effectiveness could be achieved with more number of agents and/or longer iterations. Original SDS would require longer iterations to escape local optimums and in the case of context free SDS it would need more number of agents for the cases of multiple global optimums. Lower numbers of agents have the risk of not being able to distinguish finer orderings between two or more close ranked alternatives.

Process could be stopped using any of the halting criteria given in 3.3.3 and results are obtained from the blackboard.

Two types of blackboards are developed and tested. First one is a simple table of locations and scores, the second is a directed graph.

3.4.1. SDS with Shared Count Memory

This variant of SDS with shared memory uses a simple table as a blackboard. Two counting methods are developed for this case.

During the testing phase an agent adds one to the score of the location that it is testing if the result of the test is positive. On the other hand shared count net memory version subtracts one if the agent is inactive and adds one if it is active.

Winner is the location with the highest score.

3.4.2. SDS with Shared Graph Memory

SDS with shared graph memory uses a directed graph as the blackboard. During the diffusion phase if one agent is active and the other is inactive an edge is created on shared directed graph. If there is already an edge stored in the directed graph pointing to the opposite direction it is going to be removed.

To find the result Copeland rules I, II and III (defined in Section 2.3.3) and page rank are used.

Page rank is a metric used by Google to sort web pages for their relevance. Page rank looks similar to Copeland III but it gives more weight to the nodes who has higher page ranks or incoming degrees. Therefore page rank is a recursive measure and needs to be repeated until each node converges to a value.

$$P(i) = d \sum_{j \in I_i} \frac{P(j)}{|N_j|} + (1 - d) \quad (3.2)$$

3.4.3. String Test Results

For the string search case original standard SDS with passive diffusion strategy and original context free SDS are compared with count, count net and directed graph versions of SDS with blackboard architecture.

Shared directed graph version uses Copeland rules and page rank to aggregate the results. Finally an algorithm with no diffusion is also included for benchmark purposes. In no diffusion algorithm agents only take random locations and random criteria at each cycle.

Comparisons are made with similarity coefficient α (Equation 2.3) and top choice match coefficient β (Equation 2.4).

Three test runs of 10, 100, and 200 iterations are made with 10 agents. Random search spaces and random search key created in Chapter 2 is used.

It is found that there is no difference on α between no diffusion, standard SDS and context free SDS. This is expected because SDS aims to find the top choices not the complete orderings.

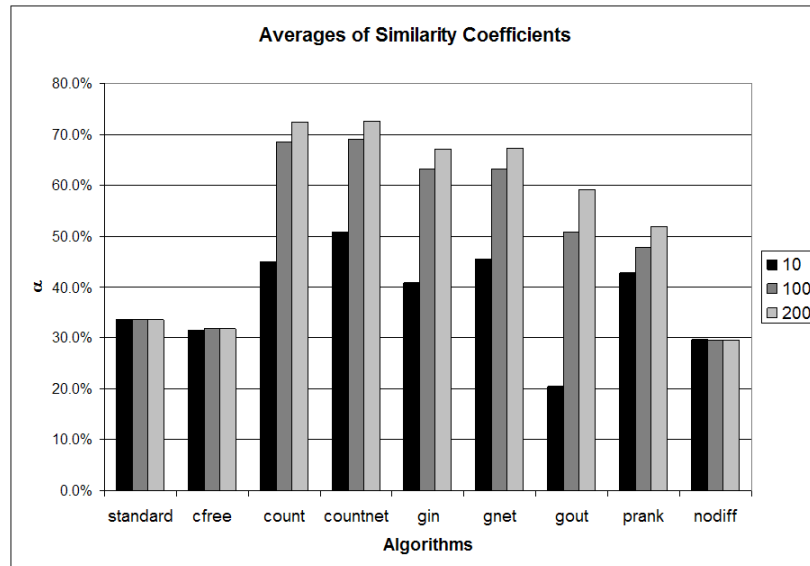


Figure 3.2. Similarity coefficients comparing standard and context free SDS with blackboard SDS and no diffusion algorithms

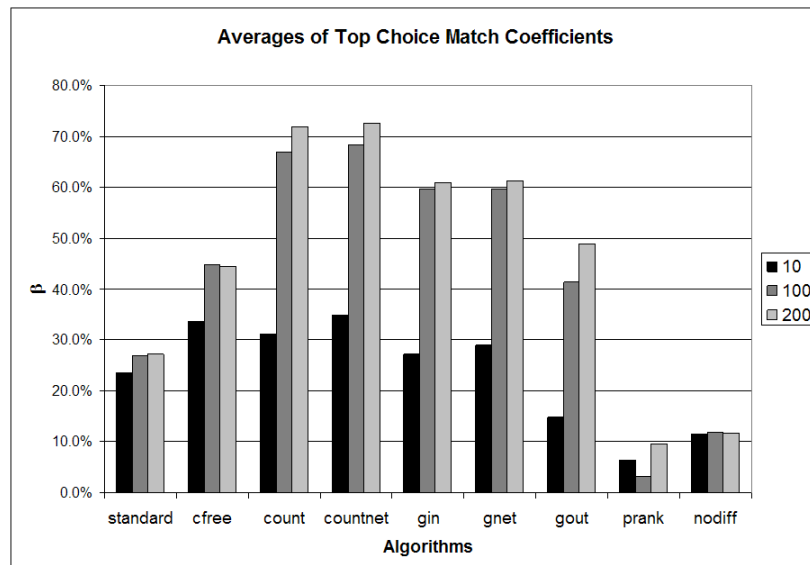


Figure 3.3. Top match coefficients comparing standard and context free SDS with blackboard SDS and no diffusion algorithms

SDS with blackboard architecture performs much better compared to standard and context free SDS. The only exception being β value of pagerank (Figures 3.2 and 3.3)

Results for α showed an average 72.5 per cent similarity to the correct ordering for the SDS with shared count and count net memory variants while original SDS could only reach 33.6 per cent. The success of SDS with blackboard architecture is greater for top choice match coefficient β . In this case SDS with shared net count achieved 72.5 per cent in average of 10,000 test runs while the original context free SDS gave 27.2 per cent.

3.4.4. Image Test Results

A total of 150 tests with CVL Face Database showed that only Copeland III was successful in SDS with graph memory case. Similar to text search a no diffusion algorithm is also tested but it was not successful to find neither the right nor the left eye. Therefore Copeland I and II and no diffusion results are not shown.

Two test runs with 100 and 1000 iterations are made. In each case 1000 agents are used and number of correct matches are counted using Equation 3.1. Results are shown in Figures 3.4 and 3.5.

Results show that even at 100 iterations blackboard algorithms perform much better compared to context free SDS. SDS with shared count and count net variants were able to find 15 of the 150 cases tested. Original context free SDS was able to find 8.

At 1000 iterations SDS with shared count net was able to find the right eye correctly in 37 cases out of 150 while original context free stayed at 18 correct matches.

SDS with blackboard architecture do not create a big overload to the process as can be seen from averages of elapsed time in Figure 3.6. Blackboard with page rank

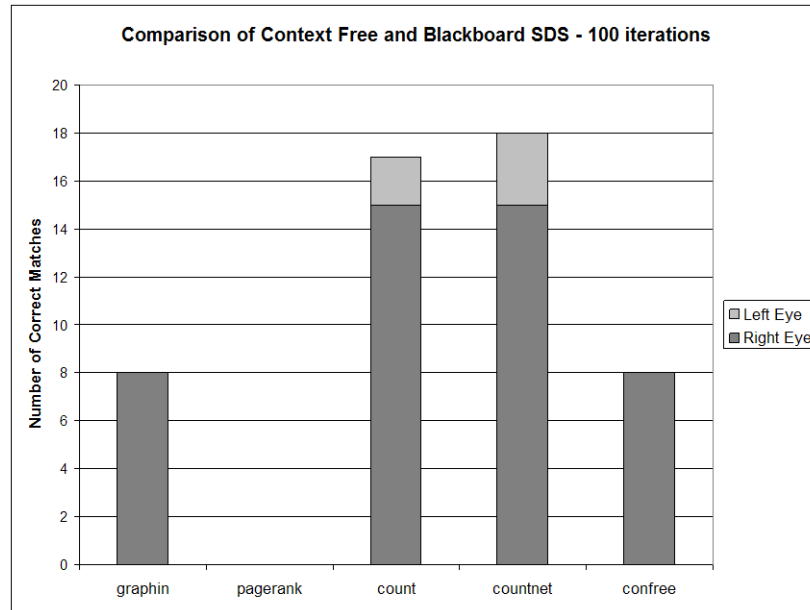


Figure 3.4. Comparing performance of Context Free SDS and SDS with blackboard architecture for image search at 100 iterations

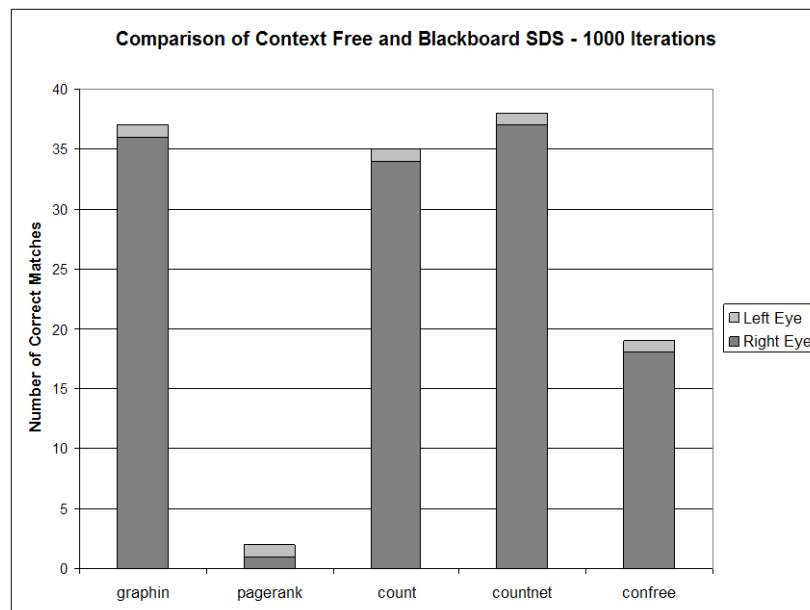


Figure 3.5. Comparing performance of Context Free SDS and SDS with blackboard architecture for image search at 1000 iterations

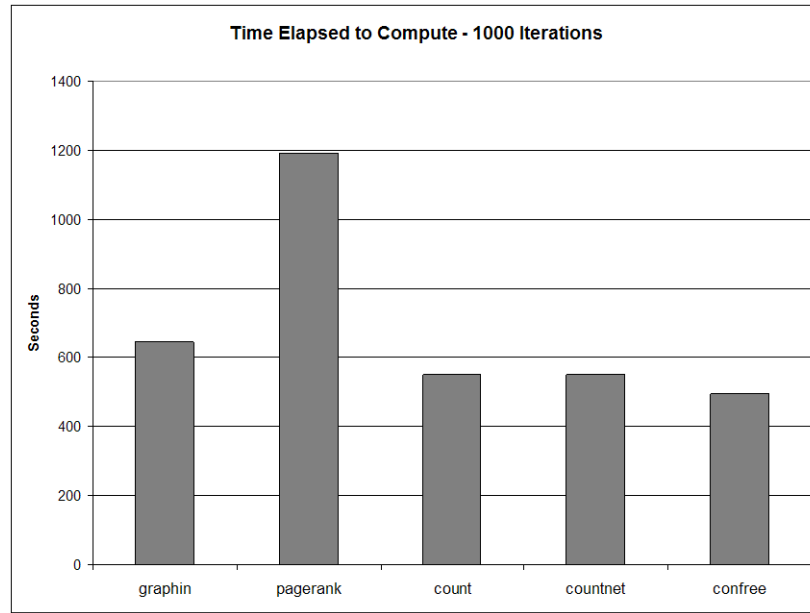


Figure 3.6. Comparing averages of time elapsed to calculate the original context free SDS and SDS with blackboard architecture for image search at 1000 iterations

takes about twice the time to finish because of page rank calculations.

3.4.5. Combining Results of Multiple SDS Algorithms

Similar to the approach of ensembles of pattern recognition results of multiple SDS algorithms are combined using several social choice rules.

Tests are made with ten SDS algorithms that are run independently with 100 iterations. Results (Figure 3.7) show that compared to a single SDS run combination of ten SDS runs gave superior results. It is also found that sum and q-Pareto rules were the best performing social choice rules.

3.5. SDS With Individual Memories

Individual memories for each agent is necessary to be able to test all social choice rules. For this variant of SDS a simple, table style memory is added to each agent. Despite single criterion in each agent, they are able to create complex preference orderings because of changing the criterion and ability to visit the same location more than

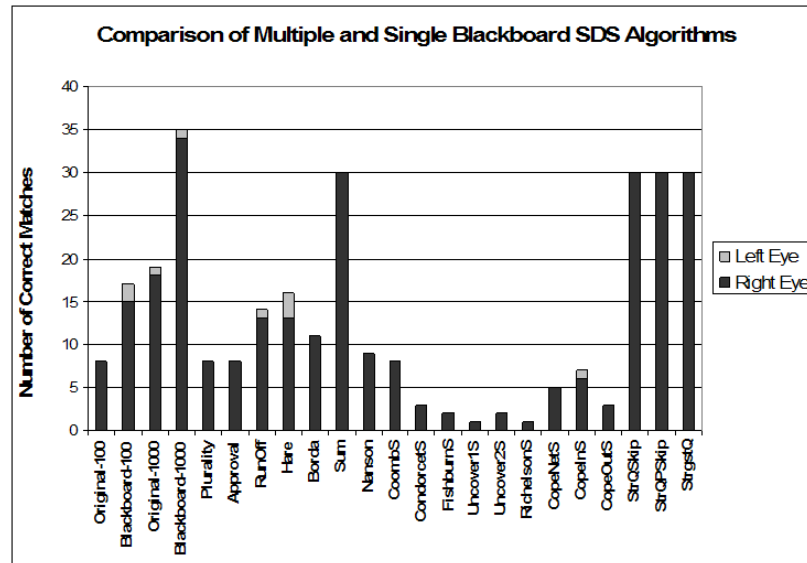


Figure 3.7. Performance of multiple blackboard SDS algorithms aggregated using different social choice rules

once. In this variant memories of agents can be thought as multiple blackboards in the system since they change their criterion from a common pool. However this variant allows to test all social choice rules.

Configurations with multiple criteria were also tested but with no success. In these configurations several threshold values were set to determine whether the agent is active or not. If number of criteria were above a certain limit agent was defined as active. These configurations did not give any successful results.

Context free SDS is used again for SDS with individual memories to keep at least half of the agents continue to search. At each test cycle agents add one to the score of the visited location if the they are active. Process stoped after a pre-determined number of iterations and social choice rules are applied to find the winner.

3.5.1. String Test Results

Tests on strings again use the same random search spaces and the search key created in Chapter 2. A no diffusion algorithm is used for benchmark purposes.

Search spaces and search key had 20 and 10 characters respectively. Tests are made with 10, 100 and 200 iterations. Comparisons are made using the α (Equation 2.3) and β (Equation 2.4) coefficients. Results of SDS with individual memories (denoted with CF) and no diffusion (denoted with ND) algorithms are shown in Tables 3.1, 3.2, and 3.3.

Table 3.1. α and β values for context free SDS and no diffusion with individual

CF - 10 Iterations		ND - 10 Iterations		CF - 10 Iterations		ND - 10 Iterations	
Algorithm	Avr. α	Algorithm	Avr. α	Algorithm	Avr. β	Algorithm	Avr. β
StrQPSkip	19.60%	Hare	46.40%	Sum	31.11%	Sum	19.13%
Sum	19.37%	RunOff	45.51%	Approval	30.26%	Borda	19.12%
Hare	19.36%	CopeNet	45.26%	Plurality	29.95%	Fishburn	18.63%
RunOff	19.33%	CopeIn	44.67%	Richelson	29.92%	Uncover2	18.63%
Borda	18.45%	CopeOut	44.44%	Fishburn	29.91%	Richelson	18.61%
CopeNet	18.28%	Sum	43.99%	Uncovered2	29.91%	Uncover1	18.45%
StrQPSkip	18.28%	Borda	43.91%	Uncovered1	29.85%	Nanson	18.34%
CopeIn	17.87%	Nanson	43.91%	Condorcet	29.77%	CopeIn	18.24%
Nanson	17.80%	StrQPSkip	43.72%	CopeIn	29.75%	CopeOut	17.58%
CopeOut	17.65%	Fishburn	42.48%	Nanson	29.61%	Approval	17.51%
Approval	17.25%	Uncovered2	42.48%	Borda	29.54%	CopeNet	17.45%
CoombS	17.11%	Uncovered1	42.43%	CopeOut	29.52%	Condorcet	17.31%
Uncovered1	16.90%	Richelson	42.28%	CoombS	29.48%	CoombS	17.16%
Fishburn	16.90%	CoombS	42.26%	CopeNet	29.43%	StrQPSkip	15.65%
Uncovered2	16.90%	Condorcet	39.39%	Hare	28.98%	Hare	15.09%
Richelson	16.86%	Approval	39.10%	RunOff	28.57%	Plurality	14.78%
Plurality	16.64%	CopeOutS	36.87%	StrongQ	28.27%	StrongQ	14.61%
CopeOutS	16.16%	CopeNetS	36.37%	StrongQP	28.19%	StrongQP	14.43%
CopeNetS	15.73%	CopeInS	35.37%	StrQPSkip	27.41%	StrQPSkip	14.39%
Condorcet	15.02%	CondorcetS	35.22%	StrQSkip	27.29%	RunOff	13.50%
CopeInS	15.02%	FishburnS	33.68%	StrgstQA	27.06%	StrgstQA	10.84%
CondorcetS	14.96%	Uncover2S	33.66%	StrgstQ	23.68%	Minmax	10.09%
FishburnS	14.38%	Plurality	32.63%	Coomb	20.40%	FishburnS	10.08%
Uncover2S	14.22%	MinMax	31.91%	StrgstQAS	16.58%	StrgstQ	10.08%
StrgstQAS	14.16%	Uncover1S	29.89%	FishburnS	16.26%	Uncover2S	10.07%
Coomb	13.62%	InvPlrty	29.09%	Uncover2S	15.72%	StrgstQAS	10.00%
Uncover1S	13.10%	RichelsonS	28.58%	RichelsonS	14.62%	InvPlrty	9.93%
RichelsonS	12.44%	StrQSkip	28.34%	Minmax	11.11%	RichelsonS	9.49%
InvPlrty	11.55%	Coomb	27.30%	Uncover1S	10.78%	InvBorda	9.32%
MinMax	10.90%	StrgstQA	21.89%	InvBorda	10.55%	Maxmin	9.31%
StrngstQA	10.74%	StrgstQAS	21.64%	Maxmin	10.52%	Uncover1S	7.49%
StrgstQ	10.16%	StrgstQ	19.02%	CopeOutS	10.05%	CondorcetS	6.06%
StrongQP	9.73%	StrongQP	18.39%	CondorcetS	9.86%	CopeOutS	6.06%
StrongQ	9.71%	StrongQ	18.38%	InvPlrty	8.47%	CopeNetS	5.62%
Maxmin	7.77%	Maxmin	17.39%	CopeInS	7.71%	CopeInS	5.51%
InvBorda	6.36%	InvBorda	15.16%	CopeNetS	7.47%	Coomb	4.54%

It is found that in general no diffusion algorithm was more successful to find the full ordering (α coefficient) especially on tests with 10 iterations. However SDS with individual memories performed better to find the correct top choices compared to no diffusion. This is due to the ability of SDS to converge on optimum solutions. When there is no diffusion, algorithm does not converge on good results. For a simple text search it may be able to find the complete ordering with no diffusion algorithm.

Table 3.2. α and β values for context free SDS and no diffusion with individual

CF - 100 Iterations		ND - 100 Iterations		CF - 100 Iterations		ND - 100 Iterations	
Algorithm	Avr. α	Algorithm	Avr. α	Algorithm	Avr. β	Algorithm	Avr. β
Sum	61.89%	Sum	73.29%	StrgstQ	70.01%	Sum	44.45%
Borda	61.37%	Borda	73.16%	StrgstQA	68.48%	Borda	43.45%
StrQPSkip	61.01%	CopeNet	72.39%	StrgstQAS	68.37%	Fishburn	42.65%
RunOff	60.80%	Nanson	72.29%	Richelson	67.54%	Uncovered2	42.65%
Hare	60.79%	CopeIn	72.21%	Condorcet	67.50%	Richelson	42.34%
Nanson	60.38%	CopeOut	72.13%	Fishburn	67.42%	Uncovered1	41.95%
CopeNet	60.37%	Hare	70.91%	Uncovered2	67.42%	CopeOut	41.68%
CopeIn	60.22%	StrQPSkip	70.73%	Uncovered1	67.36%	CopeIn	41.51%
CopeOut	60.07%	RunOff	69.79%	Plurality	67.29%	Nanson	41.09%
StrQSkip	59.17%	Uncovered1	68.75%	Approval	67.28%	CopeNet	41.02%
Uncovered1	59.09%	Fishburn	68.67%	CopeOut	67.14%	StrgstQA	39.88%
Fishburn	59.06%	Uncovered2	68.67%	CopeIn	67.10%	StrgstQAS	39.88%
Uncovered2	59.06%	Condorcet	68.44%	Nanson	67.03%	StrQPSkip	38.86%
CoombS	58.87%	Richelson	67.47%	CopeNet	66.95%	Approval	37.80%
Richelson	58.77%	StrgstQ	67.29%	Sum	66.59%	StrQSkip	37.37%
Condorcet	57.97%	CoombS	67.22%	Borda	66.56%	CoombS	36.73%
StrgstQAS	55.96%	Coomb	66.56%	FishburnS	65.92%	Coomb	36.61%
StrgstQ	55.77%	StrQSkip	64.23%	Hare	65.91%	Hare	36.30%
StrgstQA	55.75%	StrgstQAS	63.27%	CopeOutS	65.79%	Condorcet	35.89%
StrongQ	55.63%	StrgstQA	59.99%	RunOff	65.73%	FishburnS	35.87%
StrongQP	51.32%	CopeNetS	57.29%	CondorcetS	65.52%	Plurality	35.28%
CopeOutS	50.76%	CopeOutS	56.86%	CoombS	65.47%	CopeOutS	35.15%
Coomb	49.88%	StrongQP	55.53%	Coomb	65.24%	CondorcetS	34.74%
CopeNetS	47.76%	CopeInS	54.37%	StrQSkip	65.09%	Maxmin	34.08%
CopeInS	42.67%	Approval	52.69%	StrQPSkip	64.66%	StrgstQ	33.98%
FishburnS	41.40%	Plurality	46.88%	StrongQP	51.49%	RunOff	33.64%
CondorcetS	38.14%	FishburnS	46.08%	Uncover2S	36.37%	Minmax	30.55%
Minmax	37.75%	CondorcetS	43.32%	Minmax	32.34%	Uncover2S	25.11%
Uncover2S	36.38%	Uncover2S	42.27%	RichelsonS	30.32%	RichelsonS	23.39%
Approval	34.66%	Uncover1S	41.79%	CopeNetS	23.19%	Uncover1S	19.05%
Uncover1S	34.21%	RichelsonS	41.16%	Uncover1S	20.76%	CopeNetS	18.98%
Plurality	33.95%	Minmax	40.89%	StrongQ	19.99%	CopeInS	18.15%
RichelsonS	33.21%	Maxmin	30.49%	CopeInS	19.58%	StrongQP	16.73%
InvPlrty	22.86%	InvPlrty	23.80%	Maxmin	17.85%	StrongQ	9.05%
Maxmin	20.41%	StrongQ	21.36%	InvBorda	9.25%	InvBorda	8.81%
InvBorda	15.60%	InvBorda	16.99%	InvPlrty	2.89%	InvPlrty	4.62%

Table 3.3. α and β values for context free SDS and no diffusion with individual

CF - 200 Iterations		ND - 200 Iterations		CF - 200 Iterations		ND - 200 Iterations	
Algorithm	Avr. α	Algorithm	Avr. α	Algorithm	Avr. β	Algorithm	Avr. β
Sum	69.54%	Sum	77.63%	StrgstQ	75.83%	Sum	54.55%
Borda	69.13%	Borda	77.46%	StrgstQA	74.15%	Borda	53.27%
RunOff	68.60%	CopeIn	76.74%	StrgstQAS	74.09%	Fishburn	53.07%
Hare	68.60%	CopeNet	76.71%	Condorcet	73.47%	Uncovered2	53.07%
StrQPSkip	68.49%	CopeOut	76.70%	Richelson	73.32%	Richelson	52.88%
Nanson	68.47%	Nanson	76.62%	Fishburn	73.21%	StrgstQAS	52.37%
CopeNet	68.43%	StrQPSkip	75.51%	Uncovered2	73.21%	Uncovered1	52.30%
CopeIn	68.41%	Uncovered1	75.45%	Uncovered1	73.13%	CopeOut	52.15%
CopeOut	68.26%	Fishburn	75.38%	Plurality	72.97%	CopeIn	51.76%
Uncovered1	67.83%	Uncovered2	75.38%	CopeIn	72.94%	StrgstQA	51.70%
Fishburn	67.73%	Hare	75.23%	CopeOut	72.92%	Nanson	51.42%
Uncovered2	67.73%	Coomb	74.92%	Nanson	72.89%	CopeNet	51.37%
Richelson	67.60%	CoombS	74.92%	Approval	72.88%	CoombS	49.86%
StrQSkip	67.38%	Richelson	74.78%	CopeNet	72.75%	Coomb	49.86%
CoombS	66.91%	RunOff	74.39%	CondorcetS	72.23%	StrQSkip	49.06%
StrgstQA	66.66%	StrgstQ	74.38%	FishburnS	72.23%	StrQPSkip	48.71%
StrgstQ	66.58%	StrQSkip	73.52%	CopeOutS	72.22%	Maxmin	48.59%
Condorcet	66.58%	Condorcet	73.51%	Borda	72.02%	Condorcet	48.19%
StrgstQAS	66.33%	StrngstQAS	69.62%	Sum	71.84%	FishburnS	48.06%
Coomb	62.52%	StrngstQA	68.26%	CoombS	71.78%	Approval	47.88%
StrongQP	59.64%	CopeNetS	60.89%	Coomb	71.77%	CopeOutS	47.69%
CopeOutS	57.18%	CopeOutS	60.53%	RunOff	71.41%	Minmax	47.44%
CopeNetS	54.80%	CopeInS	58.05%	Hare	71.41%	CondorcetS	47.42%
CopeInS	49.77%	Approval	53.17%	StrQSkip	71.17%	Hare	46.47%
FishburnS	47.56%	FishburnS	51.31%	StrQPSkip	70.76%	StrgstQ	46.19%
CondorcetS	42.21%	Plurality	48.83%	Minmax	50.76%	Plurality	46.16%
Uncover2S	41.38%	RichelsonS	46.25%	Maxmin	42.23%	RunOff	43.90%
Minmax	40.04%	Uncover2S	46.22%	Uncover2S	38.01%	Uncover2S	30.02%
RichelsonS	39.89%	CondorcetS	46.05%	RichelsonS	34.78%	RichelsonS	28.81%
Uncover1S	39.87%	Uncover1S	45.37%	StrongQP	32.29%	Uncover1S	22.06%
StrongQ	39.48%	Minmax	41.76%	CopeNetS	24.66%	CopeNetS	21.35%
Approval	33.49%	StrongQP	34.51%	Uncover1S	23.43%	CopeInS	19.79%
Plurality	33.13%	Maxmin	29.64%	CopeInS	21.40%	StrongQP	10.25%
Maxmin	25.59%	InvPlrty	18.70%	StrongQ	13.79%	StrongQ	8.84%
InvPlrty	20.76%	StrongQ	17.57%	InvBorda	9.01%	InvBorda	8.81%
InvBorda	16.37%	InvBorda	16.99%	InvPlrty	0.95%	InvPlrty	1.42%

However, in combinatorial search with a vast search space, finding complete orderings could be impossible in a reasonable time frame.

Results of similarity coefficient for SDS with individual memories show similar results to the results in Chapter 2 (Table 2.6 on Page 40). In both cases **Sum** and **Borda** are best performers. Unlike the results on Chapter 2, strong q-Paretian plurality voting is among the best social rules for the SDS with individual memories case.

Results of top choice match coefficient showed that strongest q-Paretian simple majority rules are the best social choice rules for SDS with individual memories. These rules were very poor performers when social choice rules were compared in the previous chapter. The reason for increased performance is random criterion change in SDS. Because of this each agent tend to have similar preference orderings. This in return allows these alternatives to be able to come out of the outer intersection in the definition (Equation 2.13 on Page 34) of strongest q-Paretian simple majority.

When these results are compared to the results of the SDS with shared memory it can be seen that similar performance α and β scores are obtained (Figure 3.2 and 3.3) for 100 and 200 iterations. Performance of SDS with shared memory is more than twice of the SDS with individual memory for 10 iterations. The similarity coefficient α of no diffusion individual memory increases when number of iterations increased from 10 to 100 and value of α is 2.4 times higher compared to no diffusion shared memory case. No diffusion with individual memories also showed slightly better results for β coefficient.

One very strange result is that **StrngstQ** algorithm was the best performer for β at 100 and 200 iterations. **StrngstQ** algorithm was based on the strongest q Paretian simple majority rule with the modification to use a given number of random simple majority coalitions. This algorithm was found to be better compared to **StrngstQA** and **StrngstQAS** that use all simple majority coalitions.

To further understand the phenomena **StrngstQ** algorithm was tested with 1, 10,

and 1000 random simple majority coalitions in addition to 100 coalitions which was used in Tables 3.1, 3.2, 3.3.

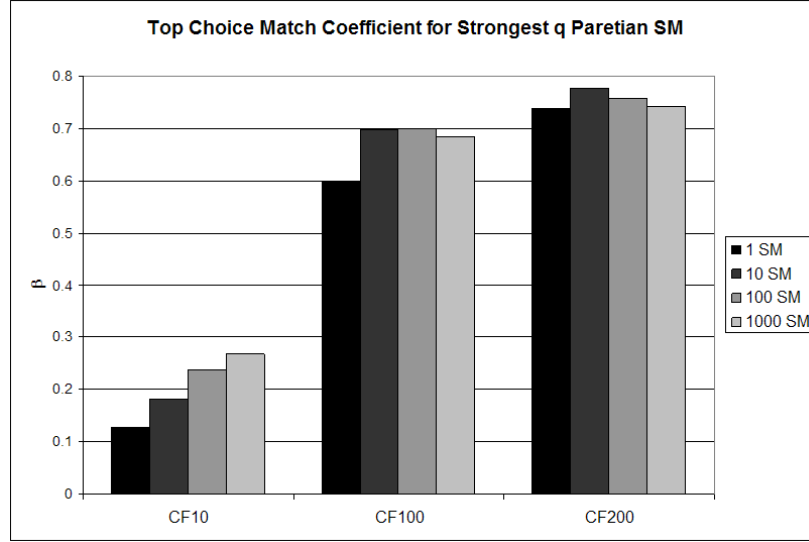


Figure 3.8. Top choice match coefficient comparison for various strongest q Paretian simple majority coalitions and context free SDS iterations

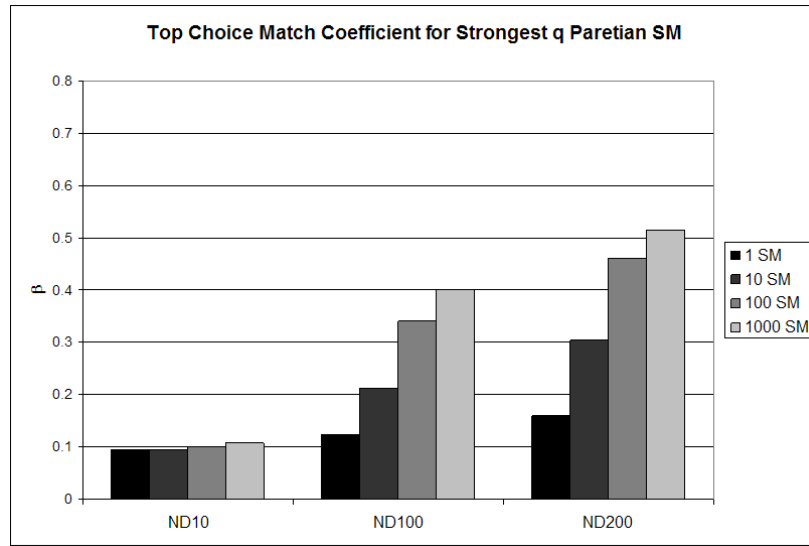


Figure 3.9. Top choice match coefficient comparison for various strongest q Paretian simple majority coalitions and no diffusion iterations

Results (Figure 3.8) show that as the number of iterations in SDS increases peak point of β is shifting from higher to smaller number of simple majority coalitions. This effect cannot be seen with no diffusion algorithm (Figure 3.9). No diffusion with strongest q-Paretian simple majority gives nearly half the performance of context free SDS.

3.5.2. Image Test Results

Image test of SDS with individual memories are made using the same face library as in Section 3.4.4 with 1,000 agents. Most of the social choice rules were not tested in image test because of high memory and very long calculation time requirements due to the high number of alternatives and agents.

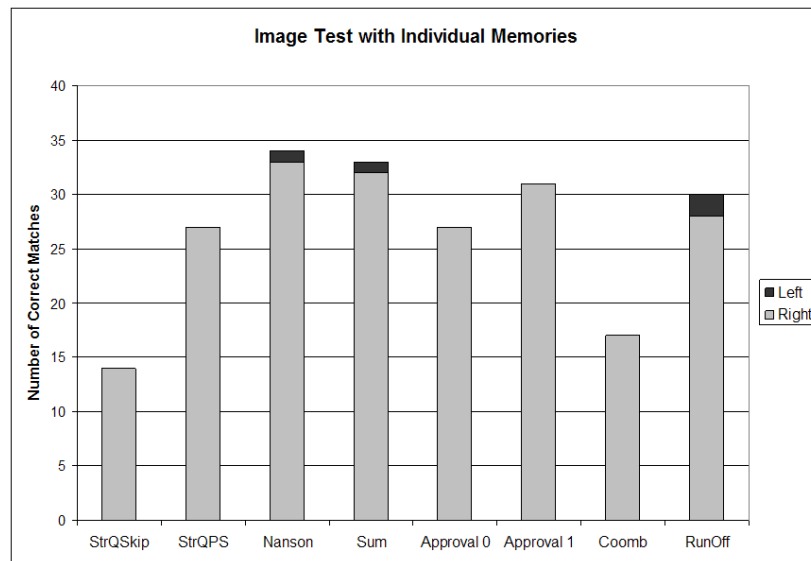


Figure 3.10. Number of correct matches for right and left eye using SDS with individual memories

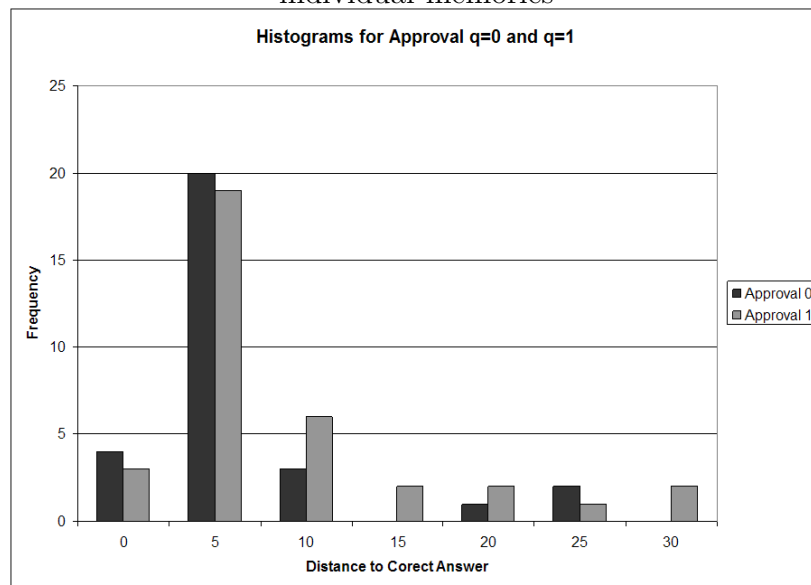


Figure 3.11. Histogram to compare Approval algorithm with $q = 0$ and $q = 1$

Results show that **Nanson** and **Sum** algorithms performed better compared to others. Agents were able to find 33 right eye images and one left eye image with **Nanson**

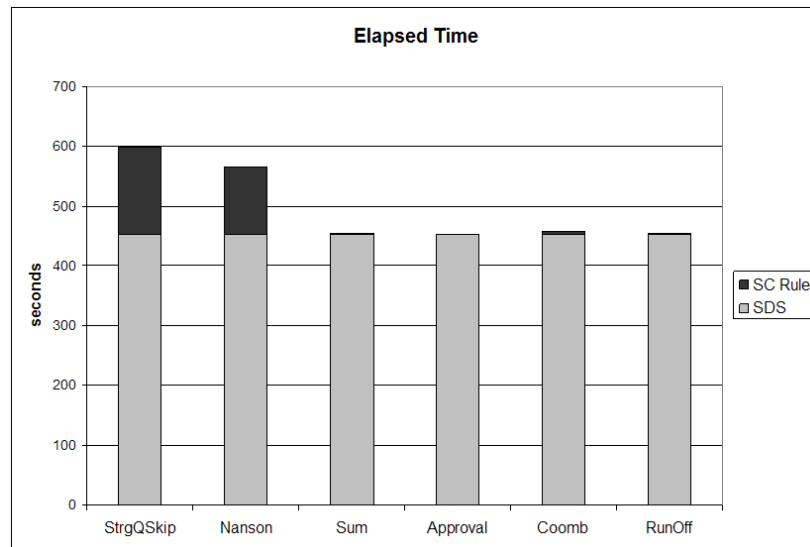


Figure 3.12. Elapsed time comparison for social choice rules in SDS with individual memories

algorithm. **Approval** algorithm with $q = 1$ was the third best algorithm (Figure 3.10). It is found that increasing q for **Approval** increases only the number of distant matches as can be seen from the comparison of histograms in Figure 3.11.

As shown in Figure 3.12 after SDS returns its results, the social choice rules took relatively very little time to compute.

3.6. Conclusion

It is found that SDS with shared and individual memories developed in this work increases the performance more than two times. Using a shared memory is found to be the best performing algorithm. In image tests original SDS were able to locate 18 right eye images correctly. In comparison SDS with shared memory was able to find 37 while SDS with individual memories was able to locate 33 images correctly. The SDS variants developed in this study give considerably higher performance without a increase in calculation time.

4. KNOWLEDGE STRUCTURE

4.1. Instability and Value Restriction

Pairwise majority voting (please see Condorcet voting in Section 2.3.2.1 on page 15) over three or more alternatives can lead to cyclical results even when the preferences of voters are transitive (Arrow, 1963). A cyclical result is a situation where the winner of a voting cannot be determined because of a top cycle violating the transitivity. Analytical solutions and simulation results show that when number of alternatives and voters increase the probability of a cycle in a majority vote will also increase (Jones *et al.*, 1995; Kelly, 1986).

Cyclical results are not desirable in most applications. Starting with Condorcet (1785) there has been a great interest to avoid the cycles (Saari, 2003). Black (1948) proved that if a profile (a set of individual preference orderings) obeys a condition called single-peakedness then the results of the majority voting of the profile would be free from cycles. Single-peakedness requires preferences to be ordered in such a way that when an alternative is a top choice the rest of the alternatives will be preferred less and less as they go further away from the top choice.

Sen (1966) proved a more general condition known as triplewise value restrictedness. Sen's theorem says if that for every triple of distinct candidates x_1, x_2, x_3 , there exists $x_i \in \{x_1, x_2, x_3\}$ and $r \in \{1, 2, 3\}$ such that no voter ranks x_i as its r -th preference among x_1, x_2, x_3 then the resulting majority voting will be acyclic. (Sen, 1966; Elsholtz and List, 2002).

4.2. Knowledge Structures

Knowledge structure is another restriction to avoid cycles. Knowledge structure is a connected graph $K(A, e)$ where nodes $A = \{a_1, \dots, a_n\}$ are the alternatives and e are the edges. It enforces a limitation to the individual preferences of agents. Each

agent has a unique most preferred top choice which is called the ideal point a_i . The rest of the choices are ordered according to the ideal point and the knowledge structure.

The ordering is done by comparing the number of edges between the two alternatives and the ideal point. If a_t is the ideal point of this agent then a_x will be preferred to a_y if the number of edges from a_t to a_x is less than the number of edges from a_t to a_y (Richards *et al.*, 1998).

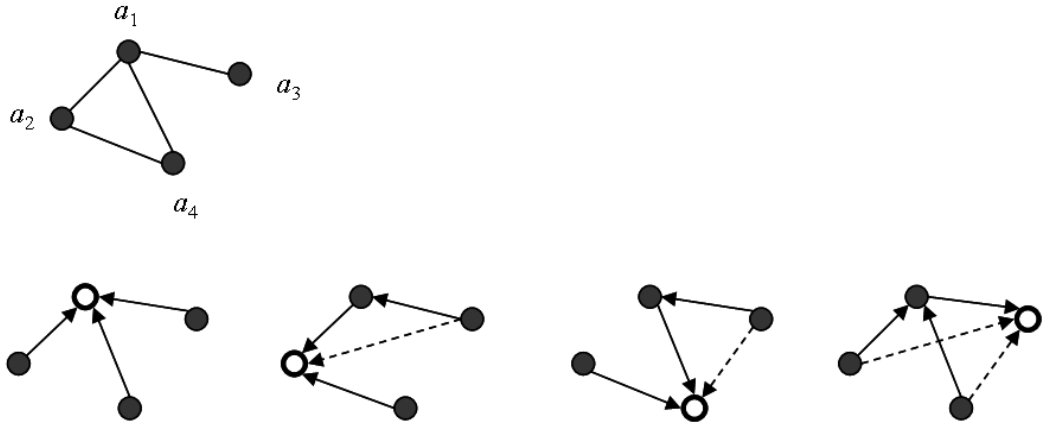


Figure 4.1. Four preference orderings derived from a knowledge structure

An edge between two alternatives in the knowledge structure means that there will be a directed edge between these alternatives when the individual preferences are to be created. If a_2 in Figure 4.1 is the ideal point for an agent, then the alternatives a_1 and a_4 connected to a_2 in the knowledge structure will give $a_2 \succ a_1$ and $a_2 \succ a_4$. Since a_3 is connected to a_1 it will give $a_1 \succ a_3$. Original definition of knowledge structure requires a unique ideal point. A more general algorithm that accepts more than one ideal points is given in Table 4.1.

Richards *et al.* (1998) describes knowledge structures as a way to represent “rules of thumb” or regularities in alternatives. Giving examples of frequency of vocal sounds made by animals of different sizes and categories of movies. Richards *et al.* propose that knowledge structures can be viewed as a tool to map similar alternatives in a choice set. Just as “...small animals with smaller vocal cavities make higher pitched sounds” most of the cases in physical, social and cognitive area include such regularities.

Table 4.1. Algorithm to find individual preference from a knowledge structure

Require: Knowledge structure K and a set of top alternatives A_t

```

1: Create a directed graph  $G$ 
2: for  $j = 1$  to number of nodes in  $K$  do
3:    $d_j :=$  minimum shortest path from  $a_j$  to  $x \in A_t$ 
4:   if  $d_j = 1$  then
5:     create an edge from  $a_j$  to all the members of  $A_t$  in  $G$ 
6:   end if
7:   for  $k = j + 1$  to number of nodes in  $K$  do
8:     if  $a_k \in A_t$  then
9:       take next  $k$ 
10:    end if
11:     $d_k :=$  minimum shortest path from  $a_j$   $x \in A_t$ 
12:    if  $d_k < d_j$  then
13:      create an edge from  $a_j$  to  $a_k$  in  $G$ 
14:    else if  $d_j < d_k$  then
15:      create an edge from  $a_k$  to  $a_j$  in  $G$ 
16:    end if
17:  end for
18: end for
19: return  $G$ 

```

Two alternatives connected with an edge in a knowledge structure are interpreted as having a smaller difference relative to the other alternatives that are further away. This difference is the “rule of thumb” that is assumed to be accepted by all the agents.

Richards *et al.* (2002) showed that forcing agents with a knowledge structure reduces the probability of top cycles. They also found that some shapes of knowledge structures produce more top-cycles than others. Yet some graphs guarantee stable outcomes. The known stable graphs are (1) all covered graphs, (2) trees, and (3) complete k-partite graphs (Richards *et al.*, 1998).

4.3. Problems with Knowledge Structures

Converting a profile to a connected knowledge structure is not always possible. If there are two voters with the same ideal point x but different preference orderings of $x \succ y \succ z$ and $x \succ z \succ y$ knowledge structure will not be connected. From the first ordering there shall be two edges on knowledge structure connecting y to z and x to y and there shall be no edge between x and z . The second ordering on the other hand will require an edge connecting x to z and no edge between x to y . Therefore removing edges (x, z) and (x, y) will result an unconnected graph.

Tests with SDS showed that it is very common to have different orderings with same ideal points. This result was expected because agents with different criteria could easily have different orderings.

An unconnected knowledge structure cannot guarantee a cycle free majority voting. Therefore a general knowledge structure had been defined to ensure connected knowledge structures where alternatives with the same distance to the ideal point are not forced to be indifferent but could also be preference relations in both directions. Sen’s triplewise value restrictedness is tried without success to prove that general knowledge structure would avoid cycles.

Research showed that SDS does not give connected knowledge structures and

general knowledge structure does not guarantee cycle free pairwise majority vote. More importantly research on performance of different social choice rules showed (Chapter 2) that other voting methods could be better alternatives.

5. CONCLUSIONS

A comparison of known social choice rules are made and results are given for all number of criteria and number of steps. It is found that less frequently used methods like strong q-Paretian rules are best performers on certain important number of criterion and step configurations. It is shown that the selection of a social choice rule depends coverage of the search space and difference of criteria of each agent. Results showed that social choice rules performed differently under complete preference similarity and top choice similarity coefficients.

The population based search method, stochastic diffusion search is studied with two main modifications.

SDS with blackboard architecture used a shared memory to store information about the agent activity. Two implementations of blackboard architecture are made. The first implementation stored scores of locations in a table and the location with the highest location were selected as the winner.

Second implementation used a directed graph where edges were created between active and inactive locations. Copeland rules and page rank metric are used to find the winner in directed graph case. It is found that Copeland's rule III is a good performer but page rank metric failed.

Tests are made with string and image frameworks and it is found that blackboard architecture with both modifications performed considerably better compared to the classical SDS. Scores of the blackboard architecture are two times higher compared to the classical SDS.

Second SDS implementation used individual memories for each agent. The string test framework tested all social choice rules and it is found that strongest q-Paretian rule was the best among all. Only the rules with applicable calculation times are used

for the image test framework. Nanson and sum algorithms were found to be better social choice rules for this case. SDS with individual memories performed slightly worse than the shared memory version but it was again considerably better compared to the classical SDS.

It is also found that the new variants did not bring a considerable extra calculation time.

Voting methods tested in this work are created for the use of humans. Therefore they are designed to satisfy the requirements (e.g. manipulability) of human elections. As can be seen from the results of this work certain voting methods perform better compared to others for SDS. Designing a new voting method that is optimized for SDS would be considered as a future work.

Voting methods tested in this work are borrowed from the literature of Social Choice Theory. As a result, they are typically designed so as to satisfy criteria of social welfare. Whether these criteria coincide with those that would maximize SDS performance is a question subject to further research.

REFERENCES

- Aleskerov, F. and E. Kurbanov, 1998, “On a Degree of Manipulability of Collective Choice Rules”, *Automation and Remote Control*, Vol. 10, pp. 134–146.
- Arrow, K. J., 1963, *Social Choice and Individual Values*, Yale University Press.
- Beattie, P., 2000, *The Design and Implementation of a Focused Stochastic Diffusion Network to Solve the Self-Localisation Problem on an Autonomous Wheelchair*, Ph.D. Dissertation, University of Reading.
- Bishop, J. and P. Torr, 1992, in R. Linggard, D. Myers, and C. Nightingale (eds.), *Neural Networks for Images, Speech and Natural Language*, Chapman & Hall.
- Bishop, J. M., 1989a, *Anarchic Techniques for Pattern Classification*, Ph.D. Dissertation, University of Reading.
- Bishop, J. M., 1989b, “Stochastic Searching Network”, in *Proc. 1st IEE Conf. ANNs*, London.
- Bishop, J. M., 1990, “A Hybrid Network for Feature Extraction”, in *Proc. INNC’90*.
- Black, D., 1948, “On the Rationale of Group Decision-Making”, *Journal of Political Economy*, Vol. 56, pp. 23–34.
- Coloni, A., M. Dorigo, and M. V., 1991, “Distributed Optimization with Ant Colonies”, in *Proc. of Ecal91 - European Conference on Artificial Life*, pp. 134–142.
- de Condorcet, M., 1785, *Essai sur l’Application de l’Analyse à la Probabilité des Décisions Rendues à la Probabilité des Voix*, Paris: De l’Imprimerie Royale.

- De Meyer, K., 2003, *Foundations of Stochastic Diffusion Search*, Ph.D. Dissertation, The University of Reading.
- De Meyer, K., J. M. Bishop, and S. J. Nasuto, 2003, “Stochastic Diffusion: Using Recruitment for Search”, in *Symposium on Evolvability & Interaction*, Queen Mary, The University of London.
- Dormer, B., B. Raphael, K. Shea, and I. F. C. Smith, 2003, “A Study of Two Stochastic Search Methods for Structural Control”, *Journal of Computing in Civil Engineering*, Vol. 17, pp. 132–141.
- Eaves, W., 1999, *Collective Choice Theory in Collaborative Computing*, <http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/9905003>.
- Elsholtz, C. and C. List, 2002, *A Simple Proof of Sen’s Possibility Theorem on Majority Decisions*, <http://www.nuff.ox.ac.uk/Politics/papers/2002/w18/value-restriction.pdf>.
- Engelmore, R. and T. Morgan (eds.), 1988, *Blackboard Systems*, Addison-Wesley.
- Franks, N. G., C. P. Stephen, E. B. Mallon, N. Britton, and D. J. T. Sumpter, 2002, “Information Flow, Opinion Polling and Collective Intelligence in House-Hunting Social Insects”, *Phil. Trans. R. Soc. London*, Vol. 357, pp. 1567–1583.
- Glover, F., 1989, “Tabu Search Part I”, *ORSA Journal of Computing*, Vol. 1, pp. 190–206.
- Grech-Cini, H., 1995, *Locating Facial Features*, Ph.D. Dissertation, University of Reading.
- Grierson, D. E. and S. Khajepour, 2002, “Method for Conceptual Design Applied to Office Buildings”, *Journal of Computing in Civil Engineering*, Vol. 16, pp. 83–103.

- Hinton, G., 1981, “A Parallel Computation that Assigns Object-Based Frames of Reference”, in *7th Int. Joint Conference on Artificial Intelligence*, Vancouver, Canada.
- Hurley, S. and R. Whitaker, 2002, “An Agent Based Approach to Site Selection for Wireless Networks”, in *2002 ACM Symposium on Applied Computing*.
- Jones, B., B. Radcliff, C. Taber, and R. Timpone, 1995, “Condorcet Winners and the Paradox of Voting: Probability Calculations for Weak Preference Orders”, *American Political Science Review*, Vol. 89, pp. 137–144.
- Jones, D., 2003, “Constrained Stochastic Diffusion Search”, in *In Proc. SCARP 2003*, pp. 185–187, Reading, UK.
- Kelly, J. S., 1986, “Condorcet Winner Proportions”, *Social Choice and Welfare*, Vol. 3, pp. 311–314.
- Kenedy, J. and R. Eberhart, 1995, “Particle Swarm Optimization”, in *Proc. IEEE International Conference on Neural Networks*.
- Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, 1983, “Optimization by Simulated Annealing”, *Science*, Vol. 220, pp. 671–770.
- Maier, H. R., A. R. Simpson, A. C. Zecchin, W. K. Foong, K. Y. Phang, H. Y. Seah, and C. L. Tan, 2003, “Ant Colony Optimization for Design of Water Distribution Systems”, *Journal of Water Resources Planning and Management*, Vol. 129, pp. 201–209.
- Mitchell, M., 1998, *An Introduction to Genetic Algorithms*, The MIT Press.
- Moscato, P., 1989, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts*, Caltech Concurrent Computation Program Report 826.

- Nasuto, S. and J. Bishop, 1999, "Convergence Analysis of Stochastic Diffusion Search", *Parallel Algorithms and Applications*, Vol. 14, No. 2, pp. 89–107.
- Nasuto, S. J., 1999, *Resource Allocation Analysis of the Stochastic Diffusion Search*, Ph.D. Dissertation, University of Reading.
- Richards, D., W. A. Richards, and B. D. McKay, 1998, "Collective Choice and Mutual Knowledge Structures", *Advances in Complex Systems*, Vol. 1, pp. 221–236.
- Richards, W. A., B. D. McKay, and D. Richards, 2002, "The Probability of Collective Choice with Shared Knowledge Structures", *Journal of Mathematical Psychology*, Vol. 46, pp. 338–351.
- Saari, D. G., 2003, "Capturing the "Will of the People"", *Ethics*, Vol. 113, pp. 333–349.
- Sen, A. K., 1966, "A Possibility Theorem on Majority Decisions", *Econometrica*, Vol. 34, pp. 491–499.
- Sen, A. K., 1970, *Collective Choice and Social Welfare*, Holden-Day Inc..
- Solina, F., P. Peer, B. Batagelj, S. Juvan, and J. Kovac, 2003, "Color-Based Face Detection in The "15 Seconds of Fame" Art Installation", in *Mirage 2003, Conference on Computer Vision/Computer Graphics Collaboration for Model-Based Imaging, Rendering, Image Analysis and Graphical Special Effects*.
- Valentini, G. and F. Masulli, 2002, "Ensembles of Learning Machines", *Lecture Notes in Computer Science*, Vol. 2486, pp. 3–19.
- Van Erp, M., L. Vuurpijl, and L. Schomaker, 2002, "An Overview and Comparison of Voting Methods for Pattern Recognition", in *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR.8)*, pp. 195–200.