The University of Reading

Foundations of Stochastic Diffusion Search

Kris De Meyer

Thesis submitted for the degree of Doctor of Philosophy

Department of Cybernetics – January 2004

Abstract

Stochastic Diffusion Search (SDS) was introduced by Bishop (1989*a*) as an algorithm to solve pattern matching problems. It relies on many concurrent *partial* evaluations of candidate solutions by a population of agents and communication between those agents to locate the optimal match to a target pattern in a search space. In subsequent research, several variations on the original algorithmic formulation were proposed. It also became evident that its main principles – partial evaluation and communication between agents – can be employed to problems outside the pattern matching domain.

The primary aim of this dissertation is to develop these expansive views further: SDS is proposed as a metaheuristic, a generic heuristic procedure for solving problems through search. Furthermore, it is proposed as a challenge to the dominant metaphor in computer science: sequential computation. The thesis proceeds in a structured way by first considering all questions that can be asked about a heuristic procedure like SDS: questions of a foundational nature, questions pertaining to mathematical analysis, questions about application domains and questions about physical implementation.

It is to the foundational issues that most attention is devoted. Analogies with selective processes in natural and social systems are investigated, as well as analogies with other metaheuristic techniques from artificial intelligence. An attempt is made to categorise potential variants, and to establish what kind of problems SDS would be the optimal problem-solving method for.

The work aims to provide an expanded but structured understanding of SDS, to give guidelines for future work, and to establish how progress in other scientific disciplines can be of use in the study of SDS, and vice versa.

Preface

All sciences characterise the essential nature of the systems they study. These characterisations are invariably qualitative in nature, for they set the terms with which more detailed knowledge can be developed.

A. Newell and H. Simon (Newell and Simon, 1976)

Cybernetics is the science of defensible metaphors.

G. Pask (von Foerster, 1995)

A personal note

When I started my PhD, I was given the freedom to choose a topic of my own interest, if only – as my supervisor expressed his hope – it was somehow related to Stochastic Diffusion Search. That freedom proved hard to handle: the choice – and the dilemmas created by it – turned out to be enormous. Four years later, my "explorations" in Stochastic Diffusion Search have produced this thesis, and the main question is: "why *this* one?"

Instead of posing a problem and then finding a method to solve the problem, it presents a method and then tries to find suitable problems. So why produce a thesis that turns normal scientific reasoning on its head? It was neither particulary easy to write, nor does it have a conventional structure.

A first reason is that previous work on SDS has seen applications to various problems, different kinds of mathematical models, new variants, with intuitions about its many links to other scientific disciplines, with intuitions about its strengths and weaknesses, but without a strong foundational framework to support all this. Secondly, in the 14 years since its incubation, SDS has not attracted much attention: it seems to have less intuitive appeal or is harder to understand than other, related methods. I therefore felt that – among all the choices I had – this was most needed: a work – to paraphrase Alan Newell and Herbert Simon – characterising the essential nature of the system; one that sets the terms with which more detailed knowledge can be developed.

By taking this unusual but interesting route, I have developed a deeper understanding of a large number of related topics. Although I have most likely expressed it inadequately in words, I hope that this work will generate a renewed interest in many aspects of its content. And, finally, following Gordon Pask, I would also like to present the thesis as a truly cybernetic one: in its pages, many metaphors are defended, while others are challenged.

Acknowledgements

First of all, I would like to express my gratitude towards my supervisor Dr. Mark Bishop and the Department of Cybernetics for giving me the opportunity to conduct this research. I would also like to thank him and Dr. Slawomir Nasuto, who acted voluntarily and with great dedication as co-supervisor, for their friendship, encouragement and motivating enthusiasm, and for the invaluable feedback they provided on earlier drafts of this document. In this respect I would also like to thank Stefan: our endless conversations have had great influence on both the content and clarity of this work. Darren, working on the same topic, has come up with several useful concepts that I was quick to assimilate into my own reasoning. Much appreciated are also the comments of Johan on the first chapter.

Secondly, I would like to thank my parents for taking care of their "lost son" during the last half year; without their support, I would not have been able to concentrate fully on writing the thesis. A special thank-you goes to Piraba, who has kindly harboured me on many occasions, and to the many other friends that have invited me to their homes after I left Reading.

Finally, there is the large number of friends that have made life in Reading a rewarding experience. In "chronological" order: Anne and Esther; Thomas; the early labrats Steve, Shabs, Ben, Rak, Matt (2x), Freddy, Farshid and Rui; the second wave with Robin, Oliver, Piraba, Guille, Alessio, Matt and Aysun; Ido and Gabi; the housemates from Hillside House, especially Rosminah, but also Gulay, Deep-ti, Aouda, and David; and the many people that spent shorter periods in Reading, or were friends of the friends listed here.

The Blind Men and the Elephant

John Godfrey Saxe

It was six men of Indostan
To learning much inclined,
Who went to see the Elephant
(Though all of them were blind),
That each by observation
Might satisfy his mind

The First approached the Elephant,
And happening to fall
Against his broad and sturdy side,
At once began to bawl:
"God bless me! but the Elephant
Is very like a wall!"

The Second, feeling of the tusk,
Cried, "Ho! what have we here
So very round and smooth and sharp?
To me 'tis mighty clear
This wonder of an Elephant
Is very like a spear!"

The Third approached the animal,
And happening to take
The squirming trunk within his hands,
Thus boldly up and spake:
"I see," quoth he, "the Elephant
Is very like a snake!"

The Fourth reached out an eager hand,
And felt about the knee.
"What most this wondrous beast is like
Is mighty plain," quoth he;
" 'Tis clear enough the Elephant
Is very like a tree!"

The Fifth, who chanced to touch the ear,
Said: "E'en the blindest man
Can tell what this resembles most;
Deny the fact who can
This marvel of an Elephant
Is very like a fan!"

The Sixth no sooner had begun
About the beast to grope,
Than, seizing on the swinging tail
That fell within his scope,
"I see," quoth he, "the Elephant
Is very like a rope!"

And so these men of Indostan
Disputed loud and long,
Each in his own opinion
Exceeding stiff and strong,
Though each was partly in the right,
And all were in the wrong!

Contents

Preface iii

Contents vii

List of Figures xi

List of Tables xii

List of Abbreviations xiii

1 Introduction 1

1.1 Context 1

1.1.1 The Unity of Knowledge 1

1.1.2 General Selection Theory 5

1.1.3 Artificial Intelligence 7

1.1.4 Computational Metaphors 9

1.1.5 Context – Conclusion 10

1.2 Subject 11

1.2.1 SDS as Algorithmic Process 13

1.2.2 SDS as a Metaphor for Computation 14

1.2.3 A Roadmap of Questions 15

1.3 Structure 18

1.4 Conclusion 20

2 Stochastic Diffusion Search 21

2.1 Origins 21

2.2 Standard SDS Operation 24
2.2.1 Problem Description and Terminology 24
2.2.2 Algorithmic Description of SDS 25
2.2.3 From Agent Operation to Population Behaviour 27
2.2.4 Examples 28
2.3 Previous Work on SDS 36
2.3.1 Foundations 37
2.3.2 Analysis 38
2.3.3 Applications 40
2.3.4 Implementations 41
2.4 Conclusion 41

3 Selective Processes in Natural and Social Systems 42
3.1 Biological Evolution 44
3.1.1 Natural Selection 45
3.1.2 Summary 48
3.2 Vertebrate Immune Systems 49
3.2.1 Clonal Selection 50
3.2.2 Summary 52
3.3 The Spread of Disease 53
3.3.1 Summary 54
3.4 The Social Insects 55
3.4.1 Recruitment in Ants 56
3.4.2 The Dance of the Honeybees 59
3.4.3 Summary 63
3.5 Culture and Society 64
3.5.1 Summary 67
3.6 Selective Processes 67
3.6.1 Perspectives 67
3.6.2 Variation + Selective Retention 70
3.6.3 General Selection Theory 74

3.7 SDS as a Selective Process 76
3.8 Rationale 78
3.8.1 What Does SDS Resemble? 78
3.8.2 Inspiration 78
3.8.3 General Selection Theory and Consilience 79
3.8.4 Why Selective Processes? 80
3.9 Conclusion 80

4 Problem Solving and Search **81**
4.1 Search 82
4.1.1 Different Meanings of Search 82
4.1.2 Search Terminology 83
4.2 Search Methods 87
4.2.1 Examples of Generic Search Methods 87
4.2.2 Search Methods: General Discussion 95
4.3 Search and Optimisation with SDS 100
4.3.1 Problems 100
4.3.2 Methods 102
4.4 Summary and Conclusion 105
4.4.1 Optimising Search Methods 105
4.4.2 Building Blocks 105
4.4.3 Conclusion 106

5 Properties of SDS **107**
5.1 Essential SDS 108
5.1.1 Algorithmic Properties 108
5.1.2 Process Properties 109
5.2 Dimensions of Choice 110
5.2.1 Activity 111
5.2.2 Diffusion 111
5.2.3 Initialisation 115

5.2.4 Termination . . . 115
5.2.5 Extraction . . . 117
5.2.6 Structural Properties of the Population . . . 118
5.2.7 Randomisation . . . 120
5.2.8 Adaptivity and Self-Adaptivity . . . 121
5.3 Conclusion . . . 122

6 Applicability **123**
6.1 Decomposable Functions . . . 124
6.1.1 Optimising Summations . . . 124
6.1.2 Other Types of Decomposable Functions . . . 129
6.2 Practice . . . 132
6.2.1 NFL Measures for SDS . . . 132
6.2.2 When to Use SDS . . . 135
6.2.3 When Not to Use SDS . . . 136
6.3 Conclusion . . . 137

7 Conclusions **138**
7.1 Summary and Contributions . . . 139
7.2 Directions for Future Research . . . 140
7.3 The Future of SDS . . . 141

A Examples **142**
A.1 Context-Sensitive SDS . . . 142
A.2 Deterministic Diffusion Search . . . 143
A.3 A Strategy of Errors . . . 146
A.4 Error Minimisation . . . 151
A.4.1 Error Minimisation with Standard SDS . . . 151
A.4.2 Activity as Belief . . . 154
A.5 Lattice Stochastic Diffusion Search . . . 157

References **164**

List of Figures

2.1 Hinton Mapping 23
2.2 Five playing cards 28
2.3 Stereo pair of images from Mars Explorer 33
2.4 Generating micro-features from grey-scale pixel values 34
2.5 Objective functions 35
2.6 Standard SDS search behaviour 35

A.1 Context-sensitive SDS search behaviour 145
A.2 Deterministic Diffusion Search 145
A.3 Effect of offset on copying accuracy 147
A.4 Strategy-of-errors search behaviour 148
A.5 Cumulative distributions of convergence times 149
A.6 Multiple-hypothesis clusters, $s = 4$ 150
A.7 Average Manhattan distance 152
A.8 Test scores generated from Manhattan distance 153
A.9 SDS behaviour for error minimisation 153
A.10 Average activity generated from Manhattan distance 156
A.11 Activity-as-belief search behaviour 156
A.12 Multiple-hypothesis clusters, $s = 2$ 157

List of Tables

2.1 Standard SDS operation 26
2.2 Test phase 26
2.3 Diffusion phase 27
2.4 Example 1: first test phase 29
2.5 Example 1: first diffusion phase 29
2.6 Example 1: second test phase 30
2.7 Example 1: second diffusion phase 30
2.8 Example 1: third test phase 30
2.9 Example 1: third diffusion phase 31
2.10 Example 1: fourth test phase 31

4.1 Standard GA operation 91

A.1 Activity-as-belief diffusion phase 155

List of Abbreviations

§xpy	Section x on page y
ACO	Ant Colony Optimisation
AI	Artificial Intelligence
AIS	Artificial Immune System
COP	Combinatorial Optimisation Problem
CSP	Constraint Satisfaction Problem
DDS	Deterministic Diffusion Search
DNA	Deoxyribo-Nucleic Acid
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolution Strategies
GA	Genetic Algorithm
GP	Genetic Programming
GRASP	Greedy Randomised Adaptive Search Procedure
MAXCSP	Maximum Constraint Satisfaction Problem
MOO	Multi-Objective Optimisation
NFL	No Free Lunch
PSO	Particle Swarm Optimisation
RGT	Random Generate-and-Test
RW	Random Walk
SA	Simulated Annealing
SDS	Stochastic Diffusion Search
SS	Scatter Search
TS	Tabu Search
TSP	Travelling Salesman Problem
V+SR	Variation + Selective Retention

Chapter 1

Introduction

1.1 Context

1.1.1 The Unity of Knowledge

Consilience

Belief in the unity of knowledge, the idea that the world can be explained by a small number of scientific principles, has appeared at different times and under different guises throughout the history of science. One of its most recent proponents, Wilson (1998), termed this alleged unity *consilience*. Historically, *consilience of inductions* was used to designate the fact that an induction from one class of observations coincides with an induction obtained from another class. Such consilience was seen as extra evidence for the validity of the theory derived from the inductions. In Wilson's (1998) work, the meaning of consilience becomes slightly more ambitious. It is used to describe the chain of cause-and-effect explanations that runs upwards through all disciplines of the natural sciences, from physics to chemistry, and from chemistry up to higher levels of biological organisation. Wilson argues that,

although there are still a few missing links, the natural sciences will eventually be made fully consilient. He also conjectures that this vertical integration along cause-and-effect relations can be extended into the social sciences and parts of the humanities, leading ultimately to the unification of science.

Wilson strongly advocates reductionism as a research methodology. However, the belief that concepts like mind, free will, culture and society can be investigated using a reductionist approach is not easily accepted by many humanistic scholars (Shweder, 2001). They regard the human sciences as concerned with something genuinely different, something not amenable to causal explanations in the biological terminology of genes and neurons. Conversely, Wilson himself does not believe that mind and culture are properties that are independent of the lower levels of biological organisation (Wilson, 2001).

Complexity

A different form of unity of knowledge was put forward by a school of thought that flourished in the 1980s and early 1990s: *complexity science* or the study of *complex (adaptive) systems* (Waldrop, 1992; Lewin, 1993). Scientists from several disciplines affiliated with the *Santa Fe Institute* to investigate to what extent common principles are at work in the systems studied by various branches of science. Complex systems consist of many relatively independent, but highly interconnected and interactive parts (Cowan, 1994). Such systems exist at a hierarchy of organisational levels in the real world: physical, chemical, biological, cognitive, social and economical. *Adaptation*, *self-organisation*, and *emergence of new properties* are but a few of the abstract principles studied in a cross-disciplinary fashion.

Complexity scientists are not primarily reductionists. Their principal research strategy is one of computer simulations and mathematical modelling from the bottom-up: starting from assumed properties of the parts and of their interactions, they try to generate and explain *emergent* properties on

the system level. This approach has been successful in some areas, but so far the complexity program has achieved few of its initial goals. The belief that the operation of all different complex systems can be explained by common principles seems to have all but evaporated. Wilson (1998) argues that this apparent failure originates from a lack of sufficient empirical knowledge that can serve as a basis for modelling. In his view, it is exactly a strong reductionist research program that is needed to supply this knowledge.

Cybernetics

Complexity science has its historical roots in two similar unifying movements that appeared almost fifty years earlier: *cybernetics* and *general systems theory*. During the Second World War, government funding in the United States and the United Kingdom had strongly stimulated scientific research, resulting in machines and technologies – most notably the general-purpose electronic computer – that would lead to new insights in the study of natural and social systems. Engineers, natural and social scientists had collaborated in interdisciplinary research teams, and developed common scientific interests and vocabularies. It was this climate that, in the aftermath of the war, brought together an interdisciplinary group of eminent scientists in a series of meetings: the "Macy conferences" on *Feedback Mechanisms and Circular Causal Systems in Biology and the Social Sciences* (Heims, 1991). In 1948, Wiener, one of the pivotal figures of the Macy conferences, published *Cybernetics or Control and Communication in the Animal and the Machine* (Wiener, 1961), and the new field soon became known as *cybernetics*.

Early cyberneticists were interested in the abstract principles of organisation, control and communication in social, natural and man-made systems. (Ashby, 1956; Pask, 1961; Heylighen and Joslyn, 2001). Their aim was to develop a transdisciplinary vocabulary and a set of mathematical formalisms that would allow a common description and explanation of these systems, re-

gardless of the *physical medium* of their implementation.[1] Cyberneticists saw as unifying principle the notion of *circular control – feedback* mechanisms – used for the *regulation* of systems. Such systems were considered *purposeful* or *teleological*: their behaviour can be interpreted as directed towards achieving or maintaining a certain *goal* (Rosenblueth, Wiener and Bigelow, 1943).

Cybernetic research became very popular in the 1950s, strongly influencing the development of control theory and control engineering, computer science, artificial intelligence, information theory, and certain areas of the social sciences. It anticipated much of the later work in neural networks, artificial life, robotics and complexity science. As a transdisciplinary movement, however, cybernetics was less successful: over the years, cybernetic research became more engineering-oriented and fragmented across subdisciplines; this slowly undermined the interests in the original research program. Many of the notions of cybernetics were assimilated in other disciplines without reference to their historical context; others were forgotten and some of them later reinvented independently. By the end of the 1960s, some of the remaining cyberneticists refocussed on the original notions of circular causality, autonomy and self-organisation, as well as on the role of the observer in modelling the system and the application of cybernetic principles to the study of cybernetics itself (von Foerster, 1995; Heylighen and Joslyn, 2001). This movement became known as *second-order cybernetics*. Even today, cybernetics as a transdisciplinary, theoretical framework is still being studied by a few dedicated groups (Ibid.).

Unifying Unity

The perspective on the unity of knowledge offered by cybernetics and complexity science is, in a sense, complementary to Wilson's notion of reductionist consilience. Whereas Wilson thinks of the unity of knowledge as a

[1] *General systems theory*, founded around the same period by von Bertalanffy (1950), had similar aims, but a different rationale behind it.

chain of cause-and-effect relationships grounding all scientific explanations in a few physical laws, cybernetics and complexity science perceive a unity in the operation of systems *at* each level. These two disparate views are not necessarily irreconcilable: both *reductionist* and *systemic* consilience might well prove to be valid. However, it is quite beyond the scope of this text to discuss the significance of the two perspectives, or to argue about the effectiveness of reductionist and systemic research methodologies: they fulfill different roles in the scientific process, and contribute to the growth of scientific understanding at different times and under different circumstances.[2]

1.1.2 General Selection Theory

Although cybernetics, from its inception, was interested in circular control mechanisms in natural and social systems, the question of *how* exactly control is exercised was less frequently asked. Often, a phenomenological description of the system as a control system was regarded as sufficient explanation. Nevertheless, it was well understood that biological control exists to ensure the survival of organisms (Ashby, 1956; Pask, 1961), and thus has to be the result of Darwin's process of *evolution* through *natural selection*. Charles Darwin formulated this theory in 1859 to explain the evolution and diversity of living organisms, but its principles have subsequently been applied to practically *all* processes of historical change (e.g., Dawkins, 1989; Plotkin, 1994; Dennett, 1995; Zimmer, 2002; Wheeler, Ziman and Boden, 2002).

Popper (1966), clearly influenced by the cybernetic thinking of that era, presented a view that connects this wider application of the natural selection paradigm with cybernetics: he envisaged each organism as a hierarchical system of flexible controls, brought about by natural selection, but *each level of control itself operating according to the trial-and-error-elimination principles*

[2]The same point about the roles of reductionist and systemic thinking in the context of biology has been thoroughly analysed and defended by Looijen (1999).

of natural selection. This idea was even more explicitly present in Campbell's (1974) *evolutionary epistemology.* Campbell regarded biological evolution as a *problem-solving* or *knowledge* process, assembling and incorporating knowledge about patterns and regularities in the world, and operating in continuity with other epistemic activities such as learning, thought and science. He abstracted Darwin's idea of natural selection to the essential mechanisms of *variation + selective retention* (V+SR). One important aspect of such a mechanistic description of the evolutionary process is its *substrate neutrality*: it is independent of the physical medium of its implementation. Campbell insisted that an abstract description in terms of V+SR mechanisms is applicable to more than just biological evolution: he called the selection paradigm the universal non-teleological explanation for all the seemingly teleological, goal-directed processes in natural and social systems. He identified a nested hierarchy of selective processes at work in natural and social systems: from biological evolution of locomotor and sensor organs, habit and instinct; over trial-and-error learning, observational learning, the development of language and culture; to, eventually, the process of science itself.[3] Each of these levels forms a *shortcut* in the evolutionary process, a *vicarious selector*, a control mechanism reducing wasteful trials of the selective processes on lower levels.

This evolutionary epistemology based on the apparent ubiquity of selective processes was later designated *general selection theory* (Campbell, 1997). General selection theory fits the two notions of consilience perfectly: systemic consilience because the systems at various levels of biological and social organisation all seem to be employing similar V+SR mechanisms; reductionist consilience because the hypothesis that all levels operate according to the principles of V+SR suggests a reductionist research program that allows the emergence of new properties on higher levels of the hierarchy to be explained from the operation of V+SR mechanisms on lower levels of the hierarchy. General selection theory thus conjoins the two notions of unity of knowledge.

[3] A similar view was developed, from a purely cybernetic perspective, by Turchin (1977).

1.1.3 Artificial Intelligence

Symbols and Search

Cyberneticists have always been especially interested in the purposiveness and intelligence emanating from animal and human brains. This interest translated itself into an active neurological research program, and especially into attempts at modelling and simulating the operation of the nervous system. From the mid-1950s, however, a view on the study of intelligence and cognition appeared that was antipathetic to the methodologies of cybernetics. This view would soon develop into the field of *artificial intelligence* (AI), situated on the boundary between *computer science* and *cognitive psychology*. AI researchers did not aim to unravel the operation of the nervous system as a means of understanding intelligence, but wanted to understand and *generate* intelligent behaviour by simulating human mentation at a higher level of abstraction (Andrew, 1983; Dreyfus and Dreyfus, 1990). They were inspired by the metaphor of *brain-as-computer*: not in the quality of computers as electrical devices (a metaphor that inspired cyberneticists), but in their quality of *symbol manipulators*. This early AI approach was effectively epitomised in the *physical symbol system hypothesis* (Newell and Simon, 1976): a necessary and sufficient condition for a system to exhibit intelligence is that it contains *symbols*, physical patterns that occur as components of *expressions*, and *processes* that operate on these symbols to produce other expressions. Symbol systems solve problems by *generating* tentative solutions (expressions) and *testing* them, i.e., by *searching* among all possible solutions to the problem; they demonstrate intelligence to the extent that *heuristic knowledge* limits the number of candidate solutions that need to be tested to solve the problem.

This insistence on generate-and-test processes as the driving force of intelligent action, though originating from an entirely different perspective,

concurs with Campbell's (1974) epistemic hypothesis on V+SR processes as the basis for all increases in knowledge. Although this is an example of consilience on an epistemological level – in its original meaning of "coming together of inductions" – there was a strong interpretational difference: firstly, the kind of search problems AI was primarily concerned with were *symbolic tree-search* problems. Secondly, influenced by the apparent *sequential* nature of human problem-solving strategies and the metaphor of *serial* computers, Newell and Simon (1976, p126) regarded all search as essentially sequential. These were unnecessary restrictions on the generality of their hypothesis, and they would in effect be removed by the course AI would take in later years.

Modern AI

In the 1980s, the weaknesses of symbolic AI became increasingly evident; in response to several different sources of criticism, AI gradually transformed into a hierarchy of theories that covers all epistemic activities at various levels of abstraction.[4] One approach – *connectionism* or the field of *artificial neural networks* (Haykin, 1999) – rejected the *symbolic* aspect of intelligence and returned to the cybernetic idea that intelligence is the result of *signal* processing in the nervous system. The approach of *intelligent agents* or *multi-agent systems* (Weiss, 1999; Ferber, 1999) criticised the *disembedded* aspect of symbolic AI: it emphasised the *situated* and *social* nature of intelligence by constructing *autonomous* agents that *interact* with their environment and with other agents. The interest in search problems remained, not only in the traditional representation of search trees, but in a more general form. A new generation of search algorithms, inspired by selective processes in natural and social systems, received increasing attention: algorithms modelled on biological evolution (e.g., Fogel, Owens and Walsh, 1966; Holland, 1992; Kennedy, Eberhart and Shi, 2001, Chapter 4), the behaviour of

[4]Modern textbooks of AI reflect this hierarchical structure (e.g., Luger and Stubblefield, 1997; Nilsson, 1998).

social insects (Bonabeau, Dorigo and Theraulaz, 1999), and human cooperative behaviour (Kennedy et al., 2001). These are not only applied to explicit search problems such as *combinatorial optimisation* problems, but also (in the paradigm of *artificial life*) used as tools for the design of intelligent systems. Finally, some of the classical techniques of symbolic AI were amended to allow for reasoning and decision making in the face of the uncertainties and "fuzziness"of the real world (Luger and Stubblefield, 1997, Chapter 7).

1.1.4 Computational Metaphors

The physical symbol system hypothesis – the dominant view on cognition for many years – was strongly influenced by other developments in computer science. Its assumption that all generate-and-test processes are essentially sequential was imposed by, what Stein (1999) calls, the *central computational metaphor*: computation regarded as a sequence of functional steps, calculating an end-result from some input. The dominance of this classical computational metaphor originated from two sides: theoretical computer science, where Turing (1936) had shown the expressive power of a class of abstract sequential computing machines, long before computers existed; and practical serial computer systems, based on the ideas of von Neumann (1945).

Recent years have seen the classical metaphor implicitly challenged by many changes in computer science. Firstly, the above paradigm shifts in AI led to new modes of problem solving, often inspired by parallel, population-based problem solving processes in natural and social systems. Secondly, the complexity of present-day software increasingly forces the use of *distributed* techniques in software development, resulting in computer programs that consist of many interacting components. Thirdly, the advent of *concurrent* programming methods introduced problems that had no equivalent within the sequential paradigm. Finally, the practical use of computer systems has shifted gradually from calculators-of-some-result towards *interac-*

tive machines that provide ongoing services without clear end-result. Stein (1999) concludes from these developments that the computational metaphor needs revision; she proposes a richer metaphor of *computation-as-interaction*: computation regarded as an ongoing process, socially interacting with its end-users, while the computational system itself can consist of many mutually interacting, concurrently operating entities.

The influence of the classical computational metaphor did not remain confined to symbolic AI and computer science. For many years – and even predating the computational metaphor itself – the dominant view in neuroscience and connectionism has been one of individual neurons as simple computational devices, calculating a non-linear output function of the sum of their inputs. Although individual neurons are arranged into networks, the emphasis in these older neural models has been mostly on the computational aspects of individual cells, and not on their mutual interaction. Nasuto, Dautenhahn and Bishop (1999) argue that, in light of many recent discoveries, this view does not capture the information-processing complexity of real neurons: they seem to behave more like spatiotemporal filters, selectively passing on certain signals, than as summation devices. Analogous to Stein's metaphor of computation-as-interaction, they propose a new metaphor for brain operation and cognition in terms of *communication*, emphasising the continuing interaction between neurons in the brain.

1.1.5 Context – Conclusion

The ideas outlined in the previous sections form the wider context – the epistemic framework – in which this work is embedded. It does not imply that these topics will be treated much further: this thesis is not about consilience, general selection theory, cybernetics, artificial intelligence, computer science or metaphors *per se*. However, these concepts play a part in the following sense: the author accepts the world-view of general selection theory, as op-

posed to a world-view based on, for example, *physical determinism*. This entails the acceptance of systemic consilience across a substantial part – if not all – of the scientific spectrum. Reductionist consilience is not rejected outright, but pure reductionism needs to be supplemented to understand the emergence of novel properties at higher levels of complexity, in a similar sense as proposed by Looijen (1999). By studying a class of very simple V+SR processes and their mathematical models, it is hoped that this work will result in a better understanding of such processes in the real world, making its own modest contribution to the unification of scientific knowledge.
Cybernetics and complexity science often provide deep insights and useful vocabulary. Many of the approaches from modern AI provide an interesting perspective; lying at the intersection of these different approaches, the discussion hopefully contributes to the unification of the study of intelligence itself. Finally, as a computational method the subject can be regarded as a radical exponent of the new metaphor of computation-as-interaction.

1.2 Subject

The primary subject of this dissertation is Stochastic Diffusion Search (SDS). Throughout the pages, the term 'SDS' will be used in a truly *polymorphic* sense. This is comparable to the everyday usage of a word like 'bread': sometimes it is used abstractly as a collective noun for all food that has flour as main ingredient and is baked in an oven; in other situations it is used to indicate one specific brand of large white, medium-sliced bread; or it is used to designate one particular *object* as belonging to the *category* of bread.
Different properties of bread are emphasised under different circumstances: for a baker, bread is a set of ingredients that need to be mixed together and baked in an oven; for a shopkeeper it is a sales-item with a weight, a price and a sell-by date; for a gourmand, the important properties are taste, texture and freshness. For each of these examples, 'bread' is considered a *subcategory*

of another category: a recipe, a sales-item, and a food-item.
Some of the properties of objects or categories are deemed *essential*: anything that is not made with flour of some kind, is quite unlikely to be called 'bread'. Some properties are *non-essential*: for instance, the presence or absence of salt in bread does not change its status of bread. Finally, many properties – essential or non-essential – can assume a *range* of *values*: 'weight' is an essential property of bread, but has a variable value.

What is essential and what is non-essential is often a matter of definition. Indeed, as Kennedy et al. (2001, p8, p30) remark, categories are linguistic conventions. Arguments about properties, or whether an object belongs to a category, or whether one category is a subcategory of another can sometimes lead to incessant discussions about the meaning of words. Whenever such matters arise in this work, they will only be pursued to the degree of what can be *gained* from assuming that something belongs to a certain category. This does not mean that categories and taxonomies of categories are unimportant: they are often fundamental in shaping scientific theories. However, it should not be assumed that categories always have static, clear-cut boundaries.

The question "what is SDS?" does not have a simple, straightforward answer. Some philosophers, such as Popper (Magee, 1973, p34, p106), even argue that questions of this kind are better not asked, since they rarely contribute to the progress of science. However, two short answers provide a good starting point for discussion: firstly, SDS is an *algorithmic process* solving certain types of problems through *search*; secondly, as a computational system, SDS fits into the new metaphor of computation-as-interaction. The discussion of these answers in Section 1.2.1 and Section 1.2.2 generates a new set of questions that are essential for developing a full understanding of SDS. These questions will be laid out systematically in Section 1.2.3. The combined answers to all these questions form the answer to the original "what is...?" question, and are provided – at least partially – in this dissertation.

1.2.1 SDS as Algorithmic Process

First and foremost, SDS is – what Dennett (1995) would call – an *algorithmic process*: a *process* described by an *algorithm*. Dennett, in his definition, implied that algorithmic processes are not algorithms in the narrow sense attributed to the word in theoretical computer science: procedures that specify with a finite number of instructions the transformation of some input into an output, with the output fully *determined* by the input (Prasse and Rittgen, 1998). Algorithmic processes often behave *non-deterministically* because of, for example, *randomised* steps within the procedure. Moreover, the observable effects of algorithmic processes are not restricted to what the underlying algorithm computes *stricto sensu*. For instance, chess programs are not algorithms "for" winning games of chess; at most, they are algorithms "for" deciding between legal chess moves. Winning a game – the ultimate goal of a chess program – is a side-effect of the particular decision strategy in combination with the moves made by the opponent.
This contrast is reminiscent of the distinction, often made in symbolic AI, between *algorithms* and *heuristics*[5] (Andrew, 1983): whereas an algorithm is a set of instructions producing a definite result, heuristics are procedures for *discovery*. They use prior knowledge to guide a *search* towards solutions to a certain problem. Heuristics are methods which work most of the time, but – in contrast with algorithms – are not *guaranteed* to work in every situation. The knowledge employed is usually very problem-specific, but recently a number of more generic heuristic search methods, applicable to different kinds of problems, have become popular in AI; they are often referred to as *metaheuristics*. Some of them are inspired by the problem-solving capabilities of algorithmic processes in natural and social systems.

SDS is clearly a metaheuristic, and comparison with other metaheuristics in Chapter 4 will confirm this point. However, 'algorithmic process' seems

[5]It is not implied here that algorithmic processes and heuristics are exactly the same.

an equally appropriate term, even if – for reasons stated above – somewhat confounding. It clearly emphasises the two sides of SDS: the algorithmic side and the process side. SDS is not a deterministic algorithm, as it usually contains several randomised steps.[6] Hence, with 'algorithmic' is meant that it is a *mechanistic procedure*, that it has a finite description in substrate-neutral terms detailing its operation. However, to fully understand how SDS solves problems, the algorithmic description is not sufficient: SDS is a *population-based* algorithmic *process*; it is to certain *statistical* regularities that can be observed on the level of the population as a whole that problem solving capabilities can be attributed.

1.2.2 SDS as a Metaphor for Computation

As an algorithmic process, SDS can be used to solve certain problems using computers. It does so in a radically different way from most other computational methods. In SDS, the need for a *total* ordering of computational steps, as in sequential programming, is rejected; instead, its operation is intrinsically *parallel* and *concurrent*[7]. SDS falls within the *generate-and-test* paradigm of AI, but categorically rejects the need for *complete* testing of candidate solutions: it performs many *partial* evaluations in parallel and in no specific order, and uses *communication* between the parallel search threads to produce a solution on the level of the system as a whole. Consisting of a population of interacting entities, SDS clearly departs from the classical computational metaphor of sequential computation, and exemplifies the new metaphor of interaction-based computation.

[6] Although the procedure can be *de-randomised* under certain circumstances.

[7] Concurrent programs define only a *partial* ordering of computational steps (Burns and Davies, 1993).

1.2.3 A Roadmap of Questions

Even without specifying details of its operation, the short discussion of the two answers to "what is SDS?" already generates several new questions: what kind of algorithmic processes in the natural world are similar to SDS? Are there any similar computational methods, particulary in modern AI? What kind of problems does SDS solve? What are its essential properties? How should it be implemented? If SDS can be regarded as part of the new computational metaphor, then how exactly does it relate to the old one? Is the new metaphor really an alternative for the dominant position of sequential computation, or is its influence confined to a small number of applications?

All key questions about SDS can be loosely organised in a few categories, and, although they cannot be answered in complete isolation from each other, this categorisation lays out a roadmap structuring the work presented here. Not all questions can be answered conclusively in the course of this text; the roadmap thus also attempts to provide a structure for future work, even if the list is probably not definitive. There are four main categories: questions about the foundations of SDS, questions pertaining to mathematical analysis of process behaviour and search performance, questions about practical applications, and questions about physical implementation. Finally, there is a fifth category of meta-level questions about the *study* of SDS itself.

Foundations

The foundational category is divided into three subcategories: questions about different interpretations of SDS, questions about algorithm and process properties, and questions about applicability and usefulness.

Interpretations SDS can be viewed from many different angles. In previous pages, it was already characterised as an algorithmic process and as a form of interaction-based computation. Three questions arise from there:

1. Which algorithmic processes in the real world are similar to SDS?

2. What does it do? Which problems does it solve?

3. Where does it fit in as a computational method? Which computational frameworks does it belong to and can it benefit from? As a stochastic process, which analytical frameworks can be of use?

Properties Many alterations can be made to the original formulation of SDS. This realisation gives rise to two questions about algorithmic properties:

4. What are the essential algorithmic properties of SDS? What distinguishes it from other, related search methods?

5. What are the choices that can be made within this framework of essential algorithmic properties?

Executing the algorithm results in a stochastic process that has certain observable properties. This gives rise to the question:

6. What are the important process properties of SDS? How are they influenced by the algorithmic properties?

Applicability Following from the answers to the questions about what SDS does, and what its essential properties are, two questions about its applicability can be raised:

7. What kind of functions can be optimised *in principle* using only partial function evaluations?

8. On what kind of problems can SDS outperform other search methods?

Analysis

SDS can be thought of as a discrete-time stochastic process. Analysing this process mathematically enables to understand and predict algorithmic performance. Three questions can be discerned here:

9. In general, what would be the interesting behaviour to analyse?
10. Given the algorithmic properties, which behaviour *can* be analysed?
11. Which types of mathematical methods can be employed for modelling?

Applications

The reason of existence for any problem solving algorithm like SDS is its application to real world problems. A few questions are of interest:

12. What classes of concrete problems is it good for?
13. Given a problem description, how can the problem best be solved? Can a generic algorithm-design methodology be developed?

Implementations

Given a certain problem description and a choice of algorithmic properties, there are still several possible choices about the *physical implementation*:

14. How to write an efficient and/or reusable software implementation for serial computers?
15. How to implement it on parallel computers or networks of computers?
16. Can it be efficiently implemented in programmable hardware?

17. Can it be implemented in application-specific integrated circuits?

18. Can some novel form of hardware be developed that is directly tailored to the massively interconnected nature of SDS?

Meta-Level Questions

Several questions can be asked about the history of SDS, about how the study of SDS is conducted, and how it is useful for other scientific disciplines:

A. What is the history and origin of SDS?

B. How much work has already been done? Which questions from the roadmap have already been answered?

C. How can the study of SDS benefit from progress in the scientific disciplines that it is related to?

D. How can these scientific disciplines benefit from the study of SDS?

E. What is the future of SDS-type methods?

1.3 Structure

Chapter 2 reviews the work on SDS that preceeds this thesis. It describes its origins, and explains the operation of the original variant – standard SDS – with a few simple examples. This concrete approach contrasts with the abstract view that was presented in Chapter 1. In later chapters, the abstract view will be gradually developed, in a top-down manner, into more concrete ideas. Referring to the concrete examples of Chapter 2 makes this top-down explanatory process significantly easier. The chapter also places all previous work on SDS within the framework of the roadmap. The chapter thus answers meta-level questions A and B.

Chapter 3 develops the abstract view of SDS as an algorithmic process into the view of SDS as *selective* process. The first part of the chapter gives an account of selective processes in natural and social systems. The second part analyses mechanisms of variation and selective retention, and discusses SDS within this framework. The chapter provides an answer to question 1 of the roadmap, and does so in the context of meta-level questions C and D.

Chapter 4 focusses on the problem-solving aspect of SDS. It defines a general terminology for discussing search problems, and gives an overview of generic search methods. The problem-solving aspects of Chapter 2 are recapitulated in the newly defined terminology, and SDS is compared with the other metaheuristics. The chapter mainly answers question 2 of the roadmap. By placing SDS within the framework of metaheuristics, it also provides a partial answer to question 3.

Chapter 5 gives a general definition of SDS by listing its essential algorithmic properties, and the process properties that ensue from them. It then proceeds by detailing the potential choices that can be made within the framework of this general definition. The effects that algorithmic properties have on process properties forms a recurring theme throughout the chapter. The chapter provides answers to questions 4, 5 and 6 of the roadmap.

Chapter 6 discusses the applicability of SDS. The first part demonstrates how in principle all summation functions can be optimised with standard SDS. Furthermore, some ideas are given about applicability towards other function types. The second part proposes measures and guidelines for establishing whether SDS is applicable in practice, i.e., whether it can outperform other search methods for particular types of problems. The chapter answers questions 7 and 8 of the roadmap; some issues are left open to future analysis.

Chapter 7 gives an executive summary of the thesis and its contributions. This summary forms a natural bridge to a discussion of possibilities for future research on SDS, and finally – meta-level question E – the future of SDS itself.

1.4 Conclusion

This introductory chapter has provided the contextual and structural framework for the remainder of the thesis. Contextually, the most important elements are the world-view of general selection theory – entailing systemic consilience – and the approaches of modern AI. Structurally, the roadmap leads from foundational to more practical questions about SDS. It is especially the foundational issues that will be addressed in the following chapters.

SDS itself has only been discussed briefly and rather abstractly: it has been characterised as an algorithmic process and as an example of interaction-based computation. Furthermore, a few important properties have already been mentioned: SDS is a randomised algorithm, intrinsically parallel, and relying on *partial* evaluations and internal communication to solve problems. In the next chapter, the discussion will assume a more concrete form, by addressing some of the historical issues surrounding SDS.

Chapter 2

Stochastic Diffusion Search

Having set a contextual and structural framework in the previous chapter, this chapter answers some of the historical questions about SDS. Section 2.1 briefly outlines the algorithms that formed its inspiration. Section 2.2 describes in detail the operation and behaviour of standard SDS – the original variant – and illustrates this with some elementary examples. Finally, Section 2.3 situates earlier work on SDS within the roadmap of Section 1.2.3.

2.1 Origins

SDS has its origins in two methods for *invariant pattern recognition*, the task of classifying a pattern or identifying it within a larger data structure, irrespective of the transformations it has undergone. Examples of such transformations are translation or shift, rotation, and isotropic or anisotropic scaling. The first method is a sequential algorithm called *Template Matching*; the second is a connectionist model called *Hinton Mapping*.

Template Matching is most often used in the context of 2D image matching. A template image is available and needs to be identified within a larger input image. For each position that the template can assume in the input

image and for each additional transformation it can undergo, the correlation between the transformed template and the corresponding region of the input image is calculated. The solution is simply the transformation yielding the highest correlation. In its original brute-force formulation, the main disadvantage of this algorithm is the calculation of the entire correlation for each admissible transformation, making it computationally intensive for large-scale problems. Several modifications for reducing the workload have been proposed, like sub-sampling of the template image (Nair and Wenzel, 1999) or multi-resolution template matching (Rosenfeld and Vanderburg, 1977).

Although humans seem particulary adept at recognising patterns under various kinds of transformations and distortions, invariance has long been a fundamental problem for neural networks (Rumelhart and McClelland, 1987, Chapter 4). Traditionally, it was assumed that man's ability is due to a normalisation process that occurs prior to recognition of the pattern, but it was not clear how such normalisation could be made to work *within* the network. Therefore, most neural networks (like those based on back-propagation) overcome the problem by an *external* pre-normalisation of the input, forcing a *sequential* component unto otherwise parallel methods. That connectionist models can handle invariance at least in principle was demonstrated by Hinton (1981). Hinton Mapping solves the problem by processing all admissible transformations in parallel. Interactive activation and competition between *mapping units* arranged in a network ensures that the transformation yielding the best match will receive most activation. The network performs inverse transformation to a normalised, object-based representation and recognition *simultaneously*, without evaluating all transformations sequentially and exhaustively (see Figure 2.1). In terms of resources, the main disadvantage of this model is that each admissible transformation needs one mapping unit in the network, making it impractical for anything but demonstration purposes. Bishop (1989*a*), Bishop and Torr (1992) and Nasuto (1999) describe in more detail the analogies between Hinton Mapping and SDS.

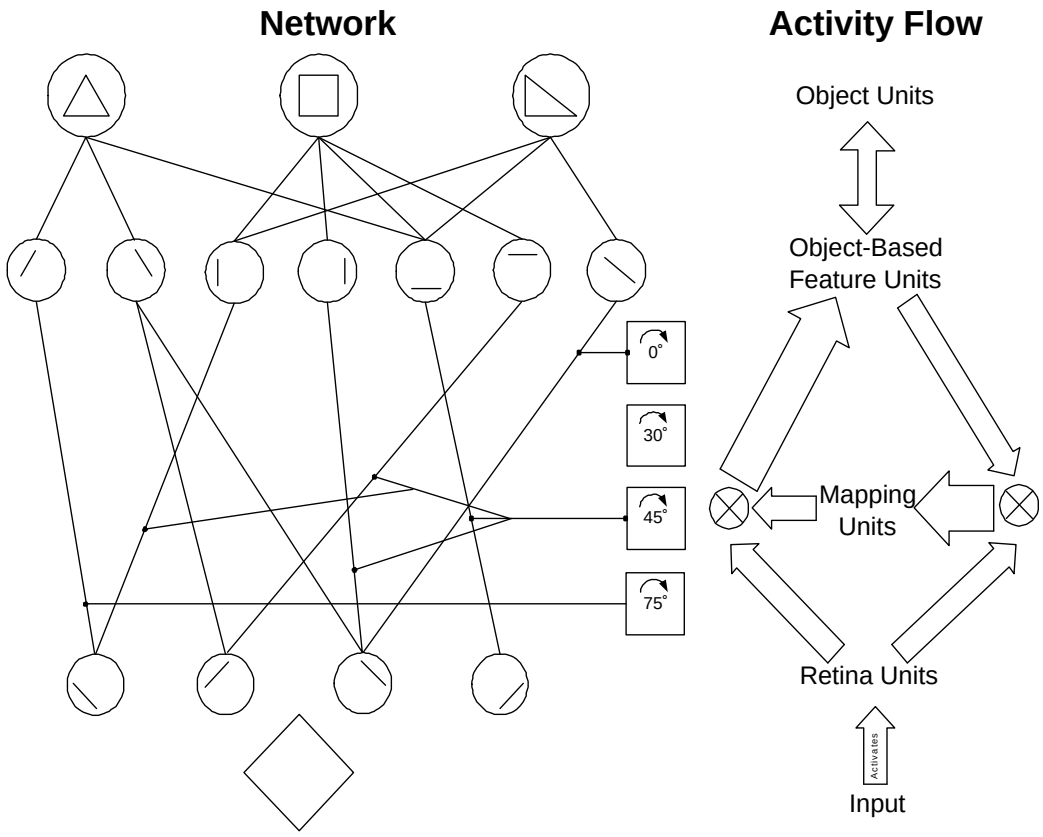

Figure 2.1: *Hinton Mapping: retina-based feature detectors are activated by a 2D input pattern – a square rotated by 45°. Mapping units control a set of channels (not all are shown) from retina units to normalised, object-centred feature units. Channels are bidirectional and multiplicative: the activity of retina units multiplied by the activity of mapping units activates object-based feature units. In turn, the activity of object-based feature units multiplied by the activity of retina units stimulates mapping units. Combinations of object-based feature units stimulate object units, each representing a particular object; object units, when activated, can further stimulate the constituent object features. The positive feedback loops in the network produce a runaway effect, mutually reinforcing those mapping and object units whose combined activity is in agreement with the activity of the retina units. Negative feedback by means of competitive inhibition between units (not shown) suppresses the activity of less successful object and mapping units.*

Template Matching and Hinton Mapping can be regarded as diametrical opposites: sequential Template Matching suffers from long running times for practical problems, whereas parallel Hinton Mapping has large memory requirements. Bishop (1989*a*; 1989*b*) proposed SDS as the middle ground between these two extremes by adopting characteristics from both methods.

2.2 Standard SDS Operation

2.2.1 Problem Description and Terminology

In the original *invariant pattern matching* description of SDS, a number of *agents* process information from the *search space* in order to identify the best match to a specified *target pattern*. Search space and target pattern need to be decomposable into *micro-features* from a pre-defined set or alphabet. For instance, in a simple 1D string matching problem both the search space and target string are composed of a one-dimensional list of characters. In a 2D image matching problem where target and search space are grey-scale bitmap images, micro-features could be (naively) thought of as single pixels; alternatively, if some form of feature extraction is applied to the images, then higher level features like lines, angles, semicircles etc. could be used as well. Micro-features of target and search space do not need to be from the same alphabet: the aim in 3D object recognition is to identify an object, specified by a 3D model, in a 2D image. Micro-features from the search space could be pixels or lines, whereas micro-features from the target could be the polygons used in the 3D representation of the object.

During operation each agent maintains a *hypothesis* about the correct solution – the transformation producing the best match between target and search space. For example, in 1D string matching, the hypothesis could be a single parameter x, indicating the position of the first character of the target

string in the search space. In a 2D image matching problem, the hypothesis could be a vector (x, y, θ, s), describing the x and y position, rotation θ and isotropic scaling s of the target in the search space. Specification of all admissible domain values for these hypothesis parameters defines the *solution space* in which each point corresponds to a set of transformation parameters mapping the target into the search space. A measure of similarity between the mapped target and the corresponding micro-features from the search space, defined over all points of the solution space, forms an *objective function*. Searching for the target in the search space can thus be translated into an optimisation problem in the solution space: finding the global maximum of the objective function; an example of such an objective function can be seen in Figure 2.5 (p35). Building an explicit representation of the entire objective function or searching the solution space exhaustively is often unfeasible. This would practically be equivalent to brute-force Template Matching. In SDS individual agents never calculate values of the objective function explicitly. In each iteration, agents evaluate a hypothesis only *partially* by comparing one or a few micro-features from target and search space. Through repeated communication they share information about the perceived quality of a hypothesis, and also the hypothesis itself. Good hypotheses are evaluated and communicated more frequently, and clusters of agents with a common hypothesis start to form. The solution to the optimisation problem can eventually be identified from these clusters. Although the computational abilities of agents are insufficient to allow them to decide on the optimal solution individually, SDS *as a system* eventually discovers the optimal solution.

2.2.2 Algorithmic Description of SDS

Agents in the original SDS algorithm operate synchronously. They undergo various stages of operation, which are summarised in Table 2.1.

```
Initialise(Agents);
repeat
   Test(Agents);
   Diffuse(Agents);
until (Halting Criterion)
```

Table 2.1: *Standard* SDS *operation.*

Initialise As a first step, agents' hypothesis parameters need to be initialised. Different initialisation methods exist, but their specification is not needed for the basic understanding of the algorithm.

Test All agents evaluate their hypothesis by randomly selecting one or a few micro-features from the target, mapping them into the search space using the transformation parameters defined by their hypothesis, and comparing them with the corresponding micro-features from the search space. Based on the outcome of the comparison, agents are divided into two groups: *active* or *inactive*. Active agents have successfully located one or more micro-features from the target in the search space; inactive agents have not. The test phase is described in pseudo-code in Table 2.2.

```
for agent = 1 to All Agents
   mf1 = Pick-Random-MF(target);
   mf2 = Find-Corresponding-MF(mf1,agent.hypothesis);
   if (mf1 == mf2)
      agent.activity = TRUE;
   else
      agent.activity = FALSE;
   end
end
```

Table 2.2: *Test phase with comparison of a single micro-feature.*

Diffuse During the *diffusion phase*, each inactive agent chooses at random another agent for communication. If the selected agent is active, then the selecting agent copies its hypothesis: *diffusion* of information. If the selected agent is inactive, then there is no flow of information between agents; instead, the selecting agent adopts a new random hypothesis. Conversely, active agents do not start a communication session in standard SDS.

```
for agent = 1 to All Agents
  if (agent.activity == FALSE)
    agent2 = Pick-Random-Agent(Agents);
    if (agent2.activity == TRUE)
      agent.hypothesis = agent2.hypothesis;
    else
      agent.hypothesis = Pick-Random-Hyp();
    end
  end
end
```

Table 2.3: *Standard* SDS *diffusion phase.*

Halt Several halting criteria exist; their specification is not needed for the understanding of the algorithm and will be postponed until a later chapter.

2.2.3 From Agent Operation to Population Behaviour

The algorithmic description of individual agent operation is insufficient to understand how SDS solves optimisation problems. Therefore, it is necessary to consider what happens with the population as a whole. By iterating through test and diffusion phases individual agents explore the whole solution space. Since tests succeed more often in points in the solution space with a good match between target and search space than in regions with a

poor match, agents will spend on average more time examining high-quality solutions, at the same time attracting other agents, which in turn attract even more agents. This positive feedback ensures that potential matches are identified by clusters of agents at certain points in the solution space.

2.2.4 Examples

A few examples will clarify the differences between individual and group-level behaviour: the first example, finding a particular playing card in a deck of cards, describes step-by-step the algorithmic operation of a small number of agents. The second example, a more realistic image matching problem, is used to illustrate the overall search behaviour of a large population of agents.

The Queen of Hearts

The task of finding a target pattern in a search space is analogous to finding a desired object among a set of objects. Consider the example of finding a particular card among the five playing cards depicted in Figure 2.2. Each playing card is unambiguously described using two properties (micro-features) that can be evaluated separately: 'type' and 'value'. The domain values for 'type' are 'hearts', 'diamonds', 'spades' and 'cups'. The possibilities for 'value' are 'ace', 'two' ... up to 'ten', 'jack', 'queen' and 'king'.

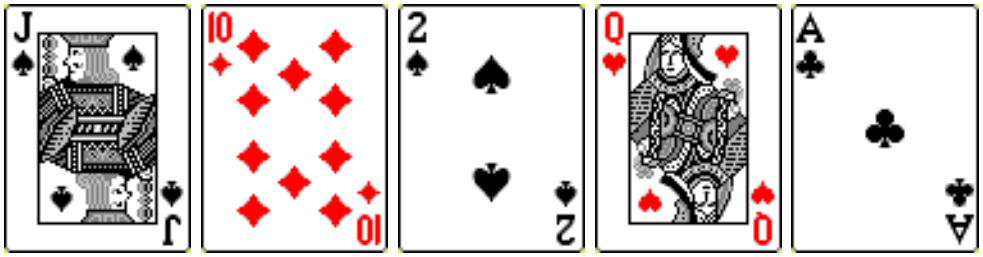

Figure 2.2: *Find the Queen of Hearts among these five playing cards.*

The solution space consists of the set $\{1, 2, 3, 4, 5\}$, the position of each of the five cards in Figure 2.2 counting from the left. The task is to find the Queen of Hearts, card number 4. All agents are inactive at the start and initialised with a randomly chosen hypothesis from 1 to 5. In the test phase, each agent randomly evaluates either the type or the value of the card indicated by its hypothesis. If the type is 'hearts' or the value is 'queen', then the agent will be in state active during the next diffusion phase. In all other cases, the agent will be inactive. In the diffusion phase, each inactive agent contacts one randomly chosen agent. If the contacted agent is active, the selecting agent copies the hypothesis from the contacted agent. Otherwise, the selecting agent generates a new random hypothesis. The algorithm halts when all agents are active. In the following step-by-step description 3 agents are used. Tables 2.4 to 2.10 provide all the details.

Agent	Card	Property	State
1	3	type	inactive
2	1	type	inactive
3	5	value	inactive

Table 2.4: *Agents at the end of the first test phase. No agent evaluated type='hearts' or value='queen', so all of them remain inactive.*

Agent	State	Contact	New card
1	inactive	2	4
2	inactive	1	2
3	inactive	1	1

Table 2.5: *Agents at the end of the first diffusion phase. Each agent contacted another inactive agent, thus randomly generated a new hypothesis.*

Agent	Card	Property	State
1	4	value	active
2	2	type	inactive
3	1	value	inactive

Table 2.6: *Agents at the end of the second test phase. Agent 1 evaluated value='queen' and becomes active. Agent 2 evaluated type='diamonds' and agent 3 evaluated value='jack'; both agents remain inactive.*

Agent	State	Contact	New card
1	active	–	–
2	inactive	1	4
3	inactive	2	5

Table 2.7: *Agents at the end of the second diffusion phase. Agent 1 was active and did not start a communication session. Agent 2 contacted active agent 1 and copied its hypothesis:* diffusion *of information. Agent 3 contacted inactive agent 2 and generated a new random hypothesis.*

Agent	Card	Property	State
1	4	type	active
2	4	value	active
3	5	type	inactive

Table 2.8: *Agents at the end of the third test phase. Agent 1 evaluated type='hearts' and agent 2 evaluated value='queen'; both are now active. Agent 3 evaluated type='spades' and remains inactive.*

Agent	State	Contact	New card
1	active	–	–
2	active	–	–
3	inactive	2	4

Table 2.9: *Agents at the end of the third diffusion phase. Agents 1 and 2 are active. Agent 3 contacted active agent 2 and copied its card number.*

Agent	Card	Property	State
1	4	value	active
2	4	value	active
3	4	type	active

Table 2.10: *Agents at the end of the fourth test phase. All agents evaluated value='queen' or type='hearts' and are now active. All of them have as hypothesis card 4, the Queen of Hearts.*

Stereo Matching on Mars

This section presents an example of standard SDS behaviour on an elementary image matching problem. The task is to locate the small image of Figure 2.3 within the larger left and right stereo-pair images.[1] The inset is taken from the left image; a perfect match between some part of the left search space and target can thus be expected. Due to a slightly different camera position, the target cannot be expected to have a perfect match in the right search space. Suitable micro-features are generated using a sampling method similar to the one used in (Bishop and Torr, 1992) (see Figure 2.4 for details). Only

[1]The method, although using real images, does not attempt to be a robust stereo matching algorithm. Rather, it is used to generate some interesting objective functions, more interesting than can be easily generated in an artificial manner.

translational transformations in two dimensions are taken into account. The transformation parameters (x, y) indicate the position of the top left corner of the target window in the search space. The objective functions over the left and right solution spaces are shown in Figure 2.5; they are generated by counting the number of micro-features in common between the target and the equally sized and sampled window with top left corner (x, y) in the larger image. The values of the objective function at each point in the solution space divided by the total number of micro-features give the relative frequency with which tests of that hypothesis result in agents being active: this is the *test score*. The objective functions are rather flat with an average over the whole solution space of approximately 25% of the maximum value and a sharp, single peak of higher values close to the correct solution.

The search behaviour of a population of 1000 agents for 1000 iterations is shown in Figure 2.6. Depicted are the total number of active agents at each iteration, and the number of active agents supporting the optimal hypothesis. The left graph corresponds to a search of the left solution space in Figure 2.5, where a perfect solution exists. The search behaviour of the system can be divided into distinct stages: at the start of the search, activity fluctuates around 280 agents; this is the *background* activity, caused by the overall level of the objective function. After 80 iterations, a reasonable solution is discovered, and activity suddenly increases to 500 agents; this is called a *convergence* stage. Shortly thereafter, a new convergence stage occurs, and activity jumps to and stabilises around 700 agents. The system can be said to have *converged* upon the solution. The stable stage is termed *quasi-equilibrium* in (Nasuto, 1999). Activity rarely ventures far from this quasi-equilibrium value, unless a better solution is discovered. Just before iteration 200, the correct solution is discovered, and all agents converge rapidly upon that one. During the entire search, the system automatically *allocates resources* to solutions in a manner dependent on their quality: the better the quality of a match, the more resources are allocated to the corresponding

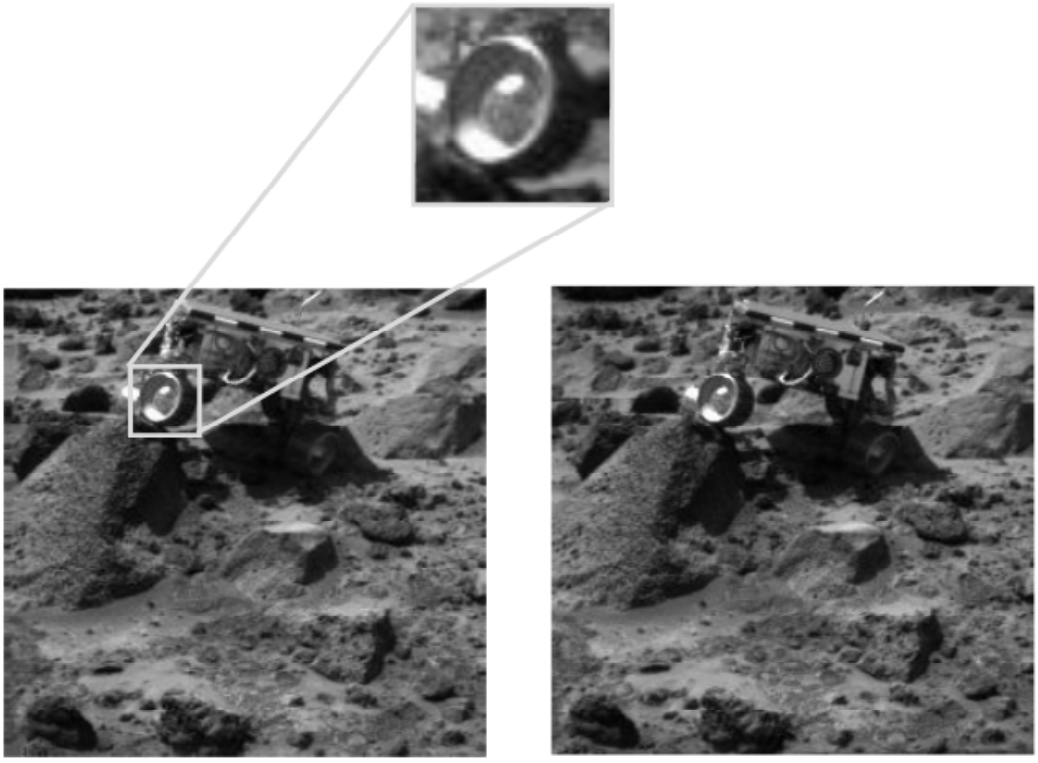

Figure 2.3: *Stereo pair of images from Mars Explorer. The small inset is the target pattern to be identified within the two images, and is taken from the left image. The resolution of the large images is* $248 * 256$ *pixels, the resolution of the target image is* $40 * 40$.

solution. A dynamical balance between a further *exploration* of the search space and *exploitation* of already discovered good solutions emerges from interaction between agents and interaction of agents with the environment.

The right graph of Figure 2.6 corresponds to the right solution space, and has only approximate matches. Here too, different convergence and quasi-equilibrium stages are present. Around iteration 550 the optimal match is discovered, attracting most but not all of the resources. A small percentage of the agents remain exploiting suboptimal matches, whereas a larger fraction keep exploring the search space for possible improvements.

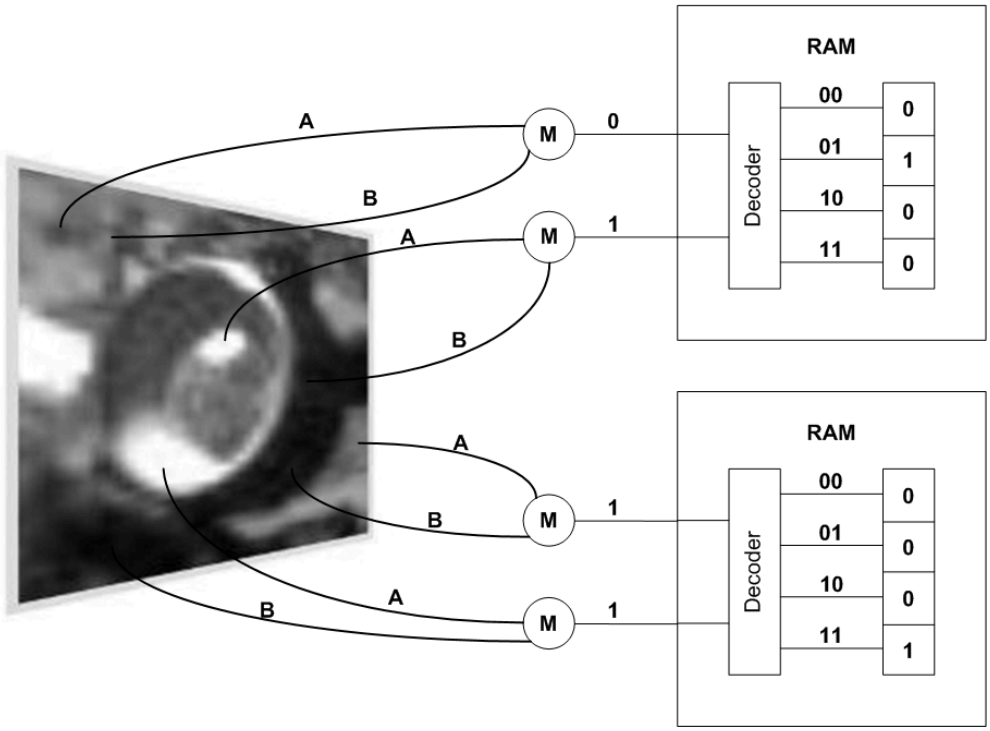

Figure 2.4: *Generating suitable micro-features from grey-scale pixel values.* Minchinton cells *(Bishop et al., 1990) transform the 8-bit grey-scale values of the pixels into binary values: each cell takes two randomly chosen pixels as input; if value A > value B, then the output of the cell is* 1*, else it is* 0*. These binary outputs are then used as address lines into an n-tuple network (Aleksander and Stonham, 1979) with* RAM *of size 2: a '1' is stored at the location addressed by two Minchinton cells taken together. Even though this is an unusual way of using n-tuple networks (the sampling can hardly be called* training *of the network since only one image is available), it still provides some generalisation and results in more suitable micro-features than the output of single Minchinton cells. Furthermore, although a* RAM *consists of 4 memory locations, it contains only a single '1'. It can thus be represented by a number from* 0 *to* 3*, designating the memory location where the '1' is stored. The 1600 pixels from the target are sampled by 1600 Minchinton cells, resulting in 800* RAM *that can assume values from* 0 *to* 3*. These form the micro-features of the image matching problem.*

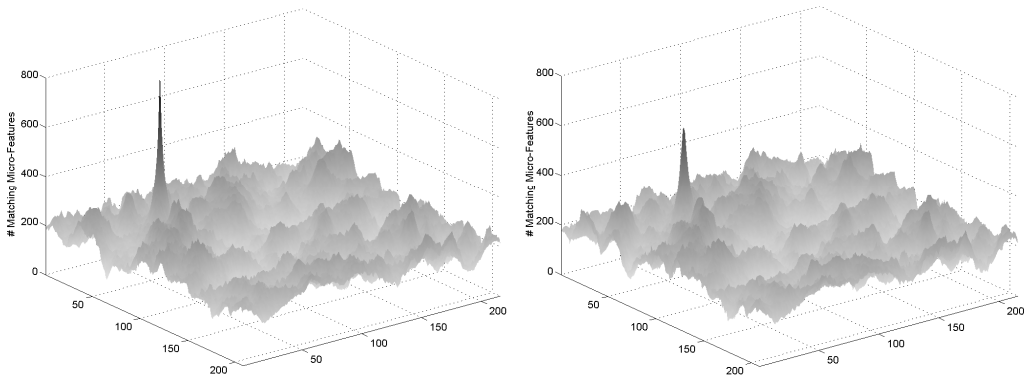

Figure 2.5: *Objective functions obtained by matching of the target with left and right images of Figure 2.3 for all possible* (x, y) *values. The target image consists of 800 micro-features, as explained in the caption of Figure 2.4. The left objective function has a maximum of 800, corresponding to the perfect match; the right objective function has a maximum of 565, corresponding to a test score of* 0.71*. The size of the solution space is* $208 * 216 = 44928$*.*

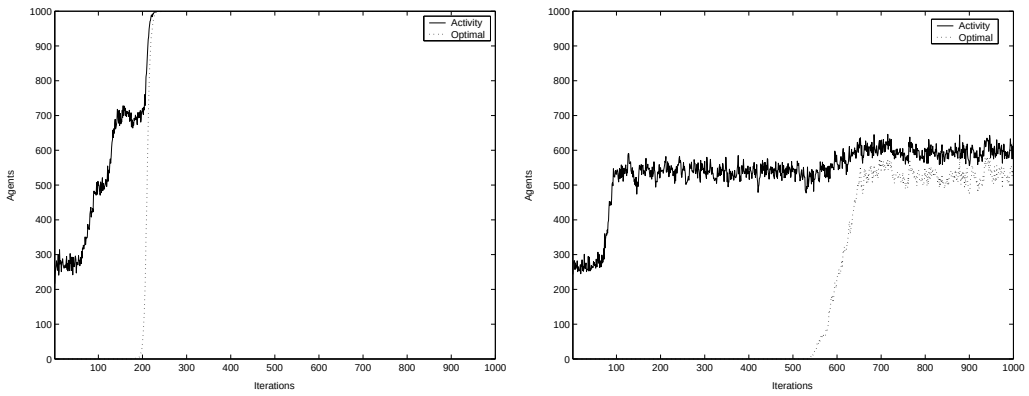

Figure 2.6: *Search behaviour of a population of 1000 agents for 1000 iterations on the left and right objective functions of Figure 2.5. Depicted are the total agent activity and the number of agents supporting the optimal hypothesis. During each test phase, each agent compares a single micro-feature from the target with the corresponding micro-feature of the search space.*

Some positive and negative aspects of SDS are already apparent from these simple experiments: standard SDS devotes most of its resources to the best match discovered so far. This is good when one is only interested in the best solution, but less desirable if also suboptimal solutions are of interest. When waiting long enough, standard SDS will always converge upon the optimal match. However, it is possible that for long periods of time, the system will focus on a suboptimal solution. This is mainly because standard SDS does not use an explicit *hill-climbing* mechanism. This is a strength as well as a weakness: it makes SDS a good choice for search problems without useable gradient information in the objective function; however, the downside is that standard SDS has no way of improving on a solution other than discovering a better one by chance. A suboptimal solution can even hinder system performance, by attracting some of the system's resources and preventing them from discovering better solutions. In spite of this, the computational gain relative to ordinary Template Matching should be obvious: it took 44,928 positions $*$ 800 micro-features = 35,942,400 comparisons to generate the complete objective functions of Figure 2.5, whereas even the naive and straightforward 1,000 iterations by 1,000 agents require 1,000,000 comparisons (and with a suitable halting criterion convergence can be determined even earlier). The number of agents needed is only weakly dependent on the properties of the target and search space: the use of 100 or 2000 agents would have given qualitatively similar behaviour. It is thus less memory intensive than Hinton Mapping that would have needed 44,928 mapping units.

2.3 Previous Work on SDS

In accordance with the road map proposed in Section 1.2.3, all the previous work on SDS can be discussed along the four main avenues: foundations (Section 2.3.1), analysis (Section 2.3.2), applications (Section 2.3.3) and implementations (Section 2.3.4).

2.3.1 Foundations

Foundational questions concerning SDS have not received much systematic treatment so far. The most complete overview of interpretations was given by Nasuto (1999), linking SDS to the domains of pattern matching, randomised search and optimisation algorithms, and multi-agent architectures. An interesting resemblance between *attentional* phenomena in SDS and attentional processes in the brain has been explored in (Nasuto and Bishop, 1998; Bishop and Nasuto, 1999; Nasuto et al., 1999; Summers, 1998). This work led to the implementation of SDS in a neural architecture with biologically inspired neurons (Nasuto et al., 1999; De Meyer, Bishop and Nasuto, 2000; Morey, De Meyer, Nasuto and Bishop, 2000). Emergent synchronisation across a large population of neurons in this network can be interpreted as a mechanism of *attentional amplification* (De Meyer et al., 2000); the formation of dynamic clusters can be interpreted as a mode of *dynamic knowledge representation* (Bishop, Nasuto and De Meyer, 2002).

Many variations on standard SDS have been proposed: Grech-Cini (1995) describes the *secret optimist*, an agent that does not always discard a hypothesis after a failed test; and the *hermit*, an agent that does not always want to communicate a seemingly good hypothesis to other agents. Nasuto (1999) proposes two modified diffusion mechanisms: in *context-free* SDS, each active agent communicates during the diffusion phase with one randomly chosen agent and switches to inactive when the other agent is also active; in *context-sensitive* SDS, an active agent switches to inactive if the contacted agent is active and additionally maintains the same hypothesis. Both mechanisms result in a substantially different resource allocation. Beattie (2000) introduces *focused* SDS, a mechanism that does not attempt to find the best solution directly, but instead seeks the region in solution space that contains good solutions and then focusses on a correct solution by sub-dividing the region into ever smaller parts. To this end, many new features are introduced into the operation of the algorithm, the most important of them the *focus level*, a

parameter that dynamically alters aspects of the testing phase. Hurley and Whitaker (2002) use a similar strategy of repeated self-focusing runs. Other variations have also been investigated: not communicating activity states, or asynchronous instead of synchronous operation (De Meyer, 2000).
In *lattice* SDS, inter-agent communication is restricted to a limited number of neighbours, with as main rationale a more efficient implementation in silicon hardware. The effects of such restrictions on the resource allocation properties are investigated in (De Meyer, Bishop and Nasuto, 2002), which is included in Appendix A.5 (p157).

The only work to date addressing the question of what type of solution spaces or objective functions SDS is good for is Grech-Cini's (1995). He argues that SDS is useful in large spaces with little or no surface information in the objective function to guide a search towards a correct solution. He also proposes a systematic method to check whether the objective function of a specific matching problem is suitable for SDS: correct solutions should have test scores higher than a certain threshold r_c, where r_c is dependent on the test scores of all incorrect solutions (see also Equation 2.1).

2.3.2 Analysis

Bishop (1989*a*) investigates SDS performance with a mixture of probabilistic reasoning and simulations. The number of agents in his description is fixed to the number of micro-features in the target. Results are mostly reported in terms of the *time ratio*, the ratio of target size to solution space size. In the general case, when the number of agents differs from the target size, the important factor is the ratio of number of agents to solution space size.
Bishop and Torr (1992) give a proof of convergence in case of no noise and infinite alphabet size. The proof is based on the formulation of the homogeneous Markov Chain describing system behaviour under these conditions.
Grech-Cini (1995) develops a probabilistic model to investigate the *critical*

response, the minimal test score for which a point in the solution space can attract a cluster of agents, given the *background response* of the whole solution space. For background response r_b, the critical response is given by:

$$r_c = \frac{1}{2 - r_b} \tag{2.1}$$

Sustainable, stable populations cannot form at locations with $r < r_c$. Although it was later shown by Nasuto (1999) that even at these locations transient, short-lived populations can form, r_c effectively forms a critical signal-to-noise ratio of what can be easily detected and what not.[2]

An interesting property investigated empirically in (Grech-Cini, 1995) is the *efficiency* of SDS: the average number of micro-feature evaluations before convergence. For the specific problem described in (Grech-Cini, 1995), the system has an efficiency equivalent to 5.2 tests per admissible transformation, independent of the size of the target.

Analysis of some important properties of SDS was completed using Markov Chain theory and Ehrenfest Urn models. Nasuto and Bishop (1999) prove convergence of all agents to the correct solution in the presence of one perfect match and in absence of noise; and also prove that in other situations, convergence can be understood as approaching an equilibrium in a statistical sense. Nasuto, Bishop and Lauria (1998) prove the convergence time of standard SDS to be sub-linear in solution space size in absence of noise. Nasuto (1999) develops a general model for the resource allocation of standard, context-free and context-sensitive SDS. Expressions for the mean and standard deviation of the overall activity in steady state are derived for various problem settings. The byproduct of the model is the generalisation of convergence (in a statistical sense) to the optimal solution in noisy search spaces.

[2]In fact, all states of the Markov Chain describing standard SDS are transient when no perfect matches are present; this means that all clusters are transient and would disappear given a long enough time period. However, the system behaves very differently on either side of r_c, suggesting that the notions sustainable and transient apply at least in practice.

2.3.3 Applications

Bishop (1989*a*) describes the application of SDS to two types of problems: locating an array of m digits in a search space of M digits; and classification of an image of m micro-features into 1 of M classes.

The first real-world application of SDS was to locate eyes within grey-scale images of human faces (Bishop and Torr, 1992). The method consisted of a combination of SDS with an n-tuple network (Aleksander and Stonham, 1979) as a pattern classifier or low-level feature extractor. The system attained perfect performance on the data on which it was trained, and 60% performance on unseen data; its performance was not limited by SDS itself but by the rejection and generalisation characteristics of the n-tuple classifier.

The same combination of n-tuple classifier and SDS was used by Grech-Cini (1993; 1995) for the tracking of the eye-nose region in video images of human faces (the location of the mouth region was inferred relative to the location of the eye-nose region). After a good but maybe suboptimal solution is identified by SDS, it is refined using a hill-climbing algorithm. The combination of the three systems was found to be fast and very reliable.

Beattie and Bishop (1998) and Beattie (2000) developed the *Focused Stochastic Diffusion Network* to handle the very large solution spaces of the self-localisation problem of an autonomous wheelchair. They report a set of results for simulated and real environments. For the largest, most-demanding industrial environment, the system located itself accurately in 58% of the trials and was robust to environment noise. Closer inspection of the results reveals that the system suffers from a well-known problem in robotics, *perceptual aliasing*: the problem that distinct places look the same to the robot's limited perception apparatus.

Hurley and Whitaker (2002) apply the main principles of SDS to a problem outside the pattern matching domain: site selection for wireless networks. These are problems with costly objective functions, and eliminating the need for complete evaluations has a great potential of reducing solving times.

2.3.4 Implementations

Although SDS is inherently parallel, most applications so far have been implemented in software on single serial computers. Only once has it been implemented on a multi-processor machine (Morey et al., 2000). The main purpose of that implementation was not speed nor efficiency, but results indicate that assigning a single agent to each processor leads to communication overhead that could make the process run slower than on a serial machine.

2.4 Conclusion

This chapter has answered some of the historical questions about SDS, in particular meta-level questions A and B from the roadmap in Section 1.2.3. It has also concretised the abstract discussion of the previous chapter by explaining in great detail the operation of standard SDS. Leaving these details aside, the main principles of SDS can be recapitulated: it is a population-based, randomised algorithmic process. The algorithm relies on *partial* rather than complete evaluations of hypotheses, and on sharing of information about the quality of hypotheses. An observer looking at the system as a whole perceives behaviour – the clustering of agents in certain points in the solution space – that was not explicit in the algorithmic description of individual agent operation, but can only be understood when probabilistic arguments are taken into account. It is to this behaviour on the population level that problem solving capabilities can be attributed.

In the next chapter it will be argued that many biological and social systems operate according to similar algorithmic processes. It forms a first step towards a structured and wider understanding of what SDS is.

Chapter 3

Selective Processes in Natural and Social Systems

Whereas the previous chapter focused on the historical context of SDS, this chapter is concerned with part of the wider scientific context: it discusses processes in a number of natural and social systems that consist, just like SDS, of many interacting entities. These systems are implemented in a variety of physical substrates, but their operation can be abstracted into algorithmic descriptions that have several elements in common. Each of these algorithmic processes also has elements in common with SDS on one or more levels of abstraction. Although the different levels of abstraction or perspectives of observation are not independent from one another, and although it would be impossible to disentangle them, it is useful to understand what they are:

1. The algorithmic description of *individual* behaviour: it is abstracted from the physical implementation and substrate-neutral.

2. The *group dynamics* perspective: the dynamical process on a collective level that is the result of actions and interactions of individuals.

3. The *historical* perspective: the long-term picture of historical change on the system level.

4. The *problem solving* perspective: individuals and the system as a whole can be regarded as solving certain problems. The problems solved by the system can be different from the ones solved by individuals.

There are several reasons for addressing the relevant details of these processes. They will be treated at length in Section 3.8, but are outlined here:

1. Discussing these processes provides a partial answer to question 1 from the roadmap in Section 1.2.3: "What does SDS resemble?"

2. Meta-level questions C and D from the roadmap suggest two reasons: answers to question 1 indicate where to look for possible answers to *other* questions from the roadmap. Conversely, answers developed in the study of SDS can be of use in the fields of study described here.

3. Discovering common mechanisms of operation on different levels of biological and social organisation is an example of systemic consilience. This consilience can be used to generate hypotheses about the operation of other, less-well understood systems, and result in the specification of reductionist research programs to investigate these hypotheses.

4. With seemingly similar mechanisms operating at different levels of biological and social organisation, the more fundamental question arises of *why* this is the case. Dependent on the answer to that question, it can be argued that these mechanisms should be applied more systematically. Understanding *how* they work can provide safeguards against improper use and wrongful application.

The processes in the following sections are described in as much detail as necessary in the context of this work, meaning that they are simplified to a large extent. Moreover, certain processes are not covered at all; this can be for several reasons: they could be less relevant, similar to other processes that have been covered, less-well studied by the scientific community, or

unknown to the author. The descriptions that have been included, though, have the intent of conveying the message that many real-world systems have fundamental principles of operation in common, and that knowledge of those common principles is of substantial interest in the study of SDS.

Section 3.1 starts with the process that is the precursor to all others in a historical sense: biological evolution. Following this, Section 3.2 describes the immune system's response to intruders in the animal body. Section 3.3 then takes a look at how and when diseases spread through a population of individuals. Section 3.4 details the recruitment behaviour in integrated societies of social insects. Section 3.5 briefly describes processes of cultural and social change. After these accounts of processes in natural and social systems, Section 3.6 then highlights what all of them have in common. Section 3.7 discusses how these common principles relate to the operation of SDS. Finally, Section 3.8 reviews the four reasons for undertaking the chapter in the first place.

3.1 Biological Evolution

The diversity of living organisms and their often astonishing adaptations to surrounding environments are thought to be the results of the process of *Darwinian evolution – adaptation* through *natural selection*.[1] Prior to Darwin's publication of *The Origin of Species* in 1859, the prevailing doctrine stated that the Creator had designed each and every species for a particular purpose, ruling out the possibility of evolutionary change of the original creation. In *The Origin of Species*, Darwin argued for the *existence* of evolution in the natural world, resulting in the diversity, but also in the genealogical connection among all earthly organisms. He also proposed a mechanism – *natural selection* – to explain the *causes* of evolutionary change.

[1] Unless specified otherwise, this section is based on (Campbell and Reece, 2002).

Darwin had not been the first to perceive the existence of evolution; half a century earlier, Lamarck had published *Philosophie Zoologique*, proposing a theory of evolution through the *inheritance of acquired characteristics* (Barthélemy-Madaule, 1979). In this concept of heredity, an individual organism can undergo modifications during its lifetime (through the effects of *use* and *disuse* of organs) and pass them on to offspring. Selection mechanisms did not occupy any significant position in Lamarck's theory, but they are not necessarily incompatible with his mechanism of inheritance. However, Weismann's discovery in 1883 of the *barrier* between *somatic* and *germ* cells in sexually reproducing multicellular organisms rules out the possibility that modifications occurring in somatic cells during an organism's life can be transferred to the germ cells that are responsible for reproduction. Since then, it has become widely accepted that *Lamarckian* inheritance is impossible in the physical substrate of the biological reproduction process.

3.1.1 Natural Selection

The theory of natural selection is based on a few simple observations: firstly, that all living organisms produce more offspring than can possibly survive on the *limited resources* of the environment. Secondly, that individuals within a population *vary* extensively. Thirdly, that much of this variation is *heritable*. From these facts Darwin inferred that, since only some offspring can survive, the survivors are more likely to be those variants that are better adapted to the conditions of the local environment. If those variations are heritable, they will be passed on to organisms of the next generation. Over time, better adapted variants will become more and more prevalent, and the population will, on average, become better adapted to local conditions. Different local conditions lead to the prevalence of different variants in different locations, and eventually to the formation of distinct species. In short: differential success in survival and reproduction on the *individual* level leads to adaptation of the *population*, and eventually to evolution into distinct species.

The mechanism of inheritance and the causes of variation were unknown in Darwin's age. It was not until much later that their molecular basis was uncovered: *genes* form the discrete units of inheritance. They are the functional parts of DNA molecules in the organism's cell(s) that (indirectly) specify cell development and operation and thus affect the characteristics of the organism. An organism passes its DNA on to its offspring through asexual or sexual reproduction. With some minor exceptions, inheritance seems to be solely based on the replication of DNA in the reproduction process, confirming Weismann's hypothesis and ruling out Lamarckian inheritance. Genetic variation – and hence: variation in characteristics – is generated by two random processes during reproduction: *mutations* arise from copying errors during the replication process of a DNA molecule. It is the only process that gives rise to new types of genes. They can change a single base in the DNA molecule (*point* mutations) or affect larger parts. Mutations are rare in general and most of them have either negative or no effects on the survival and reproduction chances of an organism. Occasionally, however, a mutant gene makes its bearer better adapted to the environment and improves its reproductive success. The second random process, *sexual recombination*, occurs, in conjunction with mutation, in many multicellular species. Sexually reproducing organisms give rise to offspring that have unique combinations of genes of the two parents. Recombination does not introduce new genes into the *gene pool* – the total set of genes present in a population at a certain moment. It merely shuffles and randomly deals genes of both parents to determine the genetic makeup of the individual. Its primary evolutionary advantage is thought to lie in an increased resistance to parasites (Ridley, 1994).

The natural selection process has different effects on different timescales. On a generation-to-generation timescale, a change in gene frequencies in the gene pool can occur through the combined effects of random fluctuations, mutation, and natural selection. This is evolution on the smallest scale, and therefore often referred to as *micro-evolution*. The cumulative effect of all

those small changes over long periods of time, together with the effects of exposing populations to different environmental conditions, leads to diverging morphologies and the formation of new species. Over even larger time spans, it is thought to account for the historical diversity of bio-organic life. This is evolution on a grander scale: *macro-evolution*. Whether random mutations and natural selection are sufficient to generate this diversity is highly debated, but not a critical issue in the context of this work.[2]

Primary interest goes to the effects of natural selection on a micro-evolutionary timescale: the fluctuation of gene frequencies in the gene pool. At any given moment, a number of *alleles* (different variants of a gene) can be present in the gene pool of a population. One of these alleles might be *fitter* than others, because the organism endowed with it is better adapted and has a higher chance of surviving until it can reproduce.[3] Such an allele is more likely to add copies of itself to the gene pool than less fit alleles, and over time it will become more and more prevalent.
By itself, natural selection would soon push most of the less viable alleles out of the gene pool. Competition for limited environmental resources has the effect that individuals carrying a weaker allele are more likely to die before reproduction can take place. Whereas mutation introduces variation into the gene pool, selection reduces it. However, additional aspects of the physical implementation of the reproductive process can preserve variation; they are based on the fact that most multicellular organisms have two or more copies of a specific gene. It allows some of the variation to be "hidden" from selective pressures. In other cases the combination of two *different* alleles might benefit its bearer more than the possession of two *identical* alleles.

In order to better understand the adaptive side of the natural selection process, it can be useful to visualise it as individuals living and reproducing

[2]Dennett (1995) gives an overview of many criticisms; he either rejects them as irrelevant or accepts them as refinements of the original theory of natural selection.

[3]There exist genes that influence replication in different ways.

on a rugged surface of *selective* values or *fitness landscape*. Such landscapes have been used both as metaphor and analytical tool, with fitness values a function of either genes or characteristics (Cruzan, 2001). The higher the value or *fitness* of individuals, the more they contribute on average to the gene pool of the next generation. Individuals constitute points on the landscape, and populations form clouds of points. Adaptation can be regarded as a population moving gradually towards a nearby peak on the landscape.

Although natural selection can improve the average fitness of a population, it does not create perfection. Biological evolution is strongly constrained by its own historical course: mutations and selection cannot easily build complex structures from scratch, but adapt existing structures to new situations. Moreover, adaptations are often compromises to the many different conditions faced by an organism during its lifetime.

3.1.2 Summary

The relevant aspects of natural selection operating on bio-organic entities can be summarised as follows:

- Individual organisms can vary with respect to their genetic make-up. This leads to variation in the characteristics of the organisms.
- Some genetic variations are more advantageous to its carrier than others. These become more prevalent in the gene pool of a population. Variations that are less favourable tend to disappear.
- Variation is introduced by mutation and reduced by selection. Other aspects of reproduction can preserve variation.
- The population adapts to the local environment.

3.2 Vertebrate Immune Systems

Biological evolution has not produced a variety of species evolving in isolation from each other, but has shaped an intricate and constantly changing web of interdependent organisms. One such interdependency is that of *parasites* needing the resources and reproductive mechanisms of *host* organisms for their own reproduction. The environment contains a multitude of microbes that can *infect* and disrupt the metabolism of specific organisms, leading to disease and, if left unchecked, eventually to the death of the host. There is no malice involved on the part of the parasites; they do what they do well because natural selection favours genes that are good at replicating, whatever the means. On the part of the hosts, selection responds by favouring organisms capable of defending themselves against such attacks. Parasites then are under selective pressure to evolve mechanisms that evade these defenses, in turn giving rise to hosts with even more effective defenses. At the same time, natural selection also has a preference for less lethal parasites (Stolley and Lasky, 1995, p208). Indeed, a parasite that kills its host too quickly might die with it before being able to spread to other susceptible hosts, and is thus not very effective at reproducing. This co-evolutionary process leads to a dynamical balance between the efficiency of attack and defense mechanisms, a process known as the *Red Queen* effect (Ridley, 1994).

The most sophisticated defense mechanisms against parasites can be found in the *vertebrae*.[4] The vertebrate *immune system* consists of a relatively small set of rather general, non-adaptive mechanisms, called *innate* immunity. It also comprises a very large number of *specific* immune defenses: *acquired* immunity. Instead of evolving new defenses every time parasites escape the guard of the old mechanisms, natural selection shaped a highly diversified and adaptive mechanism that can recognise and improve recogni-

[4]The material in this section is based on (Campbell and Reece, 2002, Chapter 43) and (Roitt, Brostoff and Male, 2001).

tion of an enormously wide variety of micro-organisms. The adaptive immune system is modified through exposure to certain parasites over the life span of an individual animal, but no evidence suggests that these modifications are passed on to offspring, as in Lamarckian inheritance.

Any immune response to infection involves two tasks: firstly, recognition of the intruder, and secondly a reaction to eliminate it. This latter reactive stage is a complex cooperation of all parts of the immune system, innate and acquired alike. Recognition, on the other hand, is largely performed by the adaptive immune system, and more specifically by cells called *lymphocytes*. Each lymphocyte carries surface *receptors* that can recognise a specific *antigen*, a molecule that is part of a parasite but foreign to the host. As undifferentiated cells develop into lymphocytes, gene segments specifying the antigen receptors are recombined in an almost unique fashion for each individual cell. This process creates an enormous *diversity* of lymphocytes with *specific* receptors, long before any contact with foreign antigen occurs.

Having such a wide variety of lymphocytes means that the body cannot maintain sufficient amounts of each variation to react rapidly and effectively against all possible parasites. This ability however is assured and improved by *clonal selection*, a process akin much to natural selection.

3.2.1 Clonal Selection

There are several different types of lymphocytes, each capable of undergoing clonal selection, but of special interest here are the so called *B-cells*. When B-cells become activated upon recognition of a specific antigen, they start to multiply and to differentiate into two clones. One clone consists of a large number of *plasma* cells, short-lived cells that combat the antigen-bearing parasite by secreting *antibodies*, receptor molecules in soluble form. Antibodies can bind to the antigen that initially activated the B-cell, labelling the antigen-bearing parasite as 'foreign substance', so that other cells of the

immune system can eliminate it. The other clone consists of *memory* cells, long-lived cells that enable a faster response upon subsequent infection with the same antigen-bearing parasite. Memory cells are receptor-carrying B-cells, and can undergo further clonal selection.

Upon the first infection with a particular microbe, only a small amount of all the B-cells in the body has a type of receptor that fits the shape of one of the antigens on the parasite, and only these cells will become activated and start multiplying. However, it is quite unlikely that any of these fits is perfect: the *affinity* of the receptor for the antigen is said to be low. Some of the B-cells might have a slightly higher affinity than others. These cells can bind to an antigen more easily and for a longer amount of time before thermodynamical disruptions destroy the relatively weak, *non-covalent* bonds. They will produce on average more plasma and memory cells than other B-cells. After several days, plasma cells with the highest affinity will become abundant enough to produce enough well-fitting antibodies so that other mechanisms of the immune system can eliminate the parasites.
In addition, mutation during cell cloning plays a role in fine-tuning the B-cell response. Parts of the genes responsible for the production of the receptor molecules are especially prone to point mutations during cloning. These mutations can result in a receptor with a higher affinity for a particular antigen, giving the corresponding B-cells a selective advantage for subsequent cloning. The *highest* affinity of receptors and antibodies for a particular antigen will thus increase over time through this process of *affinity maturation*.

A subsequent infection with a parasite carrying the same antigen will be greeted by a host of memory cells from the previous infection, each with a high affinity for the antigen. Production of plasma cells and secretion of antibodies occurs faster and in higher quantities than during the previous infection, resulting in a more effective response. It is this memory that is thought to cause immunity against subsequent infections, by detecting and eliminating the parasites before they succeed in making the host ill.

Perelson and Oster (1979) introduced the concept of *shape space* in the study of antibody repertoire size. They assumed that it is possible to describe the antibody features relevant for antigen binding by a number of *shape parameters*; geometric quantities such as size and physical quantities such as dipole moment are among the possible candidates. If combined into a vector, they form an N-dimensional vector space $\mathcal{S}$, in which antibody and antigen are represented as points. A distance metric on $\mathcal{S}$, not necessarily Euclidean, can be used as a measure of affinity. It is assumed that a foreign antigen is likely to activate a B-cell if it falls within a ball with radius ε around its antibody. For an immune system to recognise all possible intruders, the total volume of the balls should cover the whole shape space. In that case, at least one type of B-cell would become activated for each possible antigen. Mutations during cell-cloning fine-tune the response, producing cells with an ever higher affinity. Affinity maturation can thus be thought of as a cloud of antibodies moving gradually closer to a parasitic antigen in the shape space.

3.2.2 Summary

Some of the main principles of natural selection also seem to be at work in clonal selection, but some of the details are substantially different. Summarising the relevant facts on different levels of observation gives:

- Individual B-cells vary in genetic make-up, leading to large variation in antigen receptors, even before infections have occurred.
- Infection with an antigen activates the reproduction of certain cells more than others. These cells become more prevalent. The other variants remain low in number or disappear.
- Upon infection, the total population of lymphocytes increases temporarily, and large groups of cells with similar receptors form to combat the infection.

- The immune system adapts to the infection.
- Mutations fine-tune the response.
- Memory assures a more effective response in the future.

3.3 The Spread of Disease

When parasites enter the body of an animal or a human, they will usually trigger a reaction from the immune system. Sometimes that response is too slow or inadequate to prevent the micro-organisms from multiplying extensively and getting a foothold in the host: the individual becomes *infected*.[5] All diseases caused by micro-organisms are called *infectious*; if the disease can spread from one infected host to another, it is said to be *communicable*. There are various transmission routes that communicable diseases can take: there is transmission through different forms of *direct* contact, and there are *indirect* routes through the environment.

In addition to one or more characteristic transmission routes, each communicable disease also has a characteristic *risk of transmission* when contact between an infected and uninfected individual occurs, and a typical *infectious period*: the time span during which an infected individual can transmit the disease. How fast a disease spreads through a population does not only depend on these factors, but also on the contact patterns between individuals. In modern human society, where contact patterns are changing rapidly due to increased mobility, this is a factor of growing concern, and the main cause for the increasing occurrence of previously unknown or rare diseases. A factor with a negative influence on transmission is the immunity of a fraction of the population because of vaccination, previous infection, or mutations protecting its carrier from infection. The combined effect of all these factors

[5]Definitions in this section are based on (Giesecke, 2001).

taken together can be seen in the *reproductive rate* of a disease – the average number of individuals infected during the infectious period of one individual.

On a collective level, the question of concern is whether the occurrence of a disease in one or more individuals will lead to a major *outbreak* or *epidemic* affecting a substantial part of the population. It can be easily seen that for a disease entering a population, the necessary condition for the occurrence of an epidemic is that the reproductive rate $R > 1$. As an infection spreads through a population, the number of susceptible individuals starts to decline (because more of them are either immune or infected), with the effect that the reproductive rate of the infection decreases, until $R < 1$. Since each individual now infects less than one new individual on average, the epidemic will soon disappear or stabilise. Other ways of reducing R is by blocking the transmission route or by changing contact patterns between individuals.

3.3.1 Summary

The approach of this section has been somewhat different from the two previous ones: the emphasis was more on the short-term *dynamics* of the spreading process than on its causes. The important principles can be summarised as:

- Communicable diseases can spread from one individual to others.
- The reproductive rate of a communicable disease is influenced by factors such as contact patterns and level of immunity in the population.
- On the population level, epidemics can be observed if $R > 1$. The higher R, the faster the disease will spread through a population, and the more individuals will be affected.
- Epidemics can be brought to a halt by reducing $R < 1$.

3.4 The Social Insects

Ants, termites and certain species of wasps and bees live together in highly organised colonies, usually consisting of many *workers* and one or more reproductive *queens*. These insects, *eusocial* as they are called, have three traits in common (Wilson, 1971): individuals of the same species cooperate in caring for the young; more or less sterile individuals work on behalf of fecund individuals; and there exists an overlap of at least two generations capable of contributing to colony labour.[6]

Colonies require many different tasks to be performed in order to maintain themselves: reproduction, brood care, nest building, nest defense, foraging for food etc. Division of labour among these tasks, but also the allocation of workers to the exploitation of different food sources and the choice of new nest sites are problems encountered by colonies during their life cycle. These problems are not solved using a centralised control or decision making centre, but in a highly distributed fashion through the interaction of many individuals reacting to limited and local information. Natural selection has shaped a wide variety of behavioural responses and direct or indirect communication mechanisms to enable efficient cooperation. In the context of this work, the most relevant of these mechanisms is *recruitment*. It is defined as communication that brings nestmates to some point in space where work is required (Wilson, 1971) for joint efforts in food retrieval or during migration to new nest sites. Recruitment generally serves two different subtasks: individuals need to be *stimulated* to perform a certain task, and *orientated* to the location where the task needs to be performed. Often these two functions are mixed: the same signal both stimulates and orientates other individuals; however, in some cases separate signals are used for each function.

[6]Eusociality has also evolved in mammals: for instance, in naked and normal mole-rats (Campbell and Reece, 2002).

3.4.1 Recruitment in Ants

Ants are the most widely distributed of the eusocial insects, and their ecological and social adaptations are highly diversified.[7] Food specialisation and nesting habits differ widely from one species to the next. Such a diversity is also present in the many recruitment strategies employed by different species of ants for cooperative foraging and nest relocation. Communication through the use of chemical signals or *pheromone trails* constitutes the primary form of recruitment. It is assumed that this form of indirect communication has evolved many times independently from more elementary mechanisms, and intermediate evolutionary stages still exist.

The most elementary strategy of recruitment seems to be *tandem running*: a successful foraging ant will, upon its return to the nest, attract a single ant (different strategies exist: chemical, tactile or through motor display) and physically lead this ant to the food source. The leading ant may or may not have deposited pheromones along the way, but these have no stimulative effect by themselves; they are merely used as orientational signs by the leading ant. In so called *group recruitment*, an ant summons several ants at a time, then leads them to the target area; pheromones have only for the leader ant an orientational function, and no stimulative effect. In more advanced recruitment strategies, successful scouts lay a pheromone trail from the food source to the nest; this trail in itself does not have a stimulative effect. However, ants that are stimulated by, for instance, motor display in the nest can follow the trail to the food source without additional cues from the recruiter. Finally, the most developed form is *mass recruitment*. Stimulation and orientation occur indirectly: worker ants encountering a pheromone trail will follow it without the need for additional cues or stimulation.

Two examples will clarify how simple behavioural responses to limited

[7] Unless stated otherwise, the contents of this section follows Hölldobler and Wilson (1990), which provide a detailed account of all aspects of ant physiology and behaviour.

and local information coupled with recruitment can lead to collective decision making and flexible resource allocation.

Ants of the primitive species *Leptothorax albipennis* live in small colonies with only a few hundred workers. These ants recruit via tandem running or *social carrying*. They nest in preformed rock crevices and frequently need to relocate when the old nest deteriorates or becomes too small. Choosing a new nest site typically consists of a two stage process, carried out by a small group of active scouting ants (Mallon, Pratt and Franks, 2001; Pratt, Mallon, Sumpter and Franks, 2002). When a scouting ant discovers a potential nest site, she evaluates it at length and, if satisfied with the result, starts to lead tandem runs to the potential site. Some of her recruits also lead tandem runs, and the population at the new site gradually increases. The perceived quality of a nest site influences the *latency* of individual recruitment behaviour: ants having discovered mediocre nest sites seem less eager to recruit and wait on average longer before starting than ants at superior nest sites. This means that the buildup of a population goes faster at superior sites than at mediocre sites. Once a quorum of ants is present at a nest site, the active workers switch to recruiting the passive majority of the colony via transports, in which remaining nestmates and brood are simply carried to the new site. The switch from tandem running to the faster transport phase can be regarded as a decision point: a sufficient number of workers have agreed on the future nest site. In addition to the distributed decision making process, Mallon et al. (2001) claim at least a partial role for individual decision making: scouting ants that visit both a mediocre and superior nest site often prefer to recruit for the superior site. Whether this trend is really caused by an individual's capacity to make explicit comparisons between two sites is debatable and remains an open question. Large differences in the number of ants visiting two sites, reported for different colonies in (Mallon et al., 2001), do not lead to a noticeable difference in the dynamics of the decision process. This indicates that individual decision making does not play a significant role in the observed colony migrations (Pratt, personal communication).

Ants of the species *Solenopsis invicta* form colonies containing up to 200,000 workers that use mass recruitment for foraging purposes. When returning from a food source, individual ants deposit an amount of pheromones along the way, dependent on variables like perceived quality or type of the food source. Trails of individual workers evaporate within a few minutes. However, the combined trail laying activity of many ants may lead to the establishment of a stable trail. A stable allocation of workers at a single food source is achieved as follows: initially the buildup of workers at a newly discovered food source is exponential since the outflow of workers from the nest is linearly dependent on the amount of pheromone discharged by foragers already in the field. When workers become crowded on the food source, newly arriving workers are unable to reach the food source and turn back without laying trails. As a result, the number of workers stabilises at a level which is a linear function of the area of the food mass. When the food source is of low quality or far away, the number of workers may stabilise at a lower level.

Different foraging and recruitment strategies induce different quantitative performances: empirical results from (Chadab and Rettenmeyer, 1975) show that tandem running is slower than group recruitment, which in turn is slower than mass recruitment. Furthermore, the degree of accuracy – how many ants reach the food source for which they have been recruited – is dependent on the type of communication used and differs significantly from species to species. It varies from as low as 20% for group recruitment up to 70% for pure mass recruitment. Deneuborg, Pasteels and Verhaeghe (1983) argue that communication in ants is essentially probabilistic and that this "strategy of errors" may have adaptive advantages in that it allows a colony to discover previously unexploited food sources. Another important issue is how rapidly a colony adapts its forager allocation to a changing environment. Results from a number of sources, summarised in (Bonabeau, Theraulaz and Deneuborg, 1998), show that tandem running and group recruitment strategies have no problem in shifting resources towards a superior food source

that is introduced some time after a poor food source has been discovered. Colonies using pure mass recruitment, however, are often trapped at the poor site. Whatever the exact details of the recruitment behaviour, it is likely that each strategy is well adapted to the food specialisation and needs of the colony. It leads to a balance between exploration for new food sources and exploitation of discovered food sources that is sufficient for the survival of the genes causing the observed behaviour.

3.4.2 The Dance of the Honeybees

Recruitment mechanisms in honeybees (*Apis mellifera*) have been the most intensely studied of the social insects so far. The colony's food collection and nest site selection processes have proven to be very amenable to empirical analysis. Colonies of honeybees, containing up to 20,000 workers, are easily kept in observation hives, so that the normally hidden activities of individual bees can be studied in detail. Moreover, it is possible to label each bee for individual identification, enabling the study of the relationship between individual and colony-level behaviour.

Honeybees forage for nectar and pollen from flowers.[8] Like ants, they use recruitment to regulate a colony's exploitation of discovered food sources. Unlike ants however, communication is not chemical but primarily via motor display: the principle mechanism of recruitment (having both a stimulative and orientational function) is the *waggle dance*, a miniaturised reenactment of the journey to a patch of flowers. It is a descriptive and truly symbolic message, separated in space and time from the actions on which it is based and the behaviours it will guide. A dance consists of one or more straight *waggle runs* in which the recruiter vigorously waggles her abdomen, followed by turns looping back to the starting point. The direction of the waggle run

[8] All aspects of the food collection process are discussed in (Seeley, 1995), where a more detailed account can be found.

indicates the direction of the flower patch in relation to the sun. The duration of a single waggle run is a measure of the distance to the source. Dances are performed on the *dance floor*, a part of the hive near the entrance.

Nectar is the colony's prime source of energy. Having discovered a nectar source, a forager seems to be able to estimate quite correctly its energetic profitability, taking into account factors such as sugar concentration, distance to the hive, accessibility etc. The total quantity of the source, however, is information that is not readily available to individual foragers. Nectar source profitability influences the probability that a bee will advertise it on the dance floor, and also the length of the total dance: the more profitable, the more waggle runs in a single dance – numbers range from as low as 1 up to 100 runs. There is strong variation among individual bees in the dance response to a source. This makes the total length of a dance a poor predictor of nectar source profitability. However, unemployed forager bees – if not scouting for new food sources themselves – sample the dance of a single, randomly chosen recruiter for only a few waggle runs, meaning that no information about the source's profitability is transmitted. They then try to find the food source reported by the dance. If successful, a recruited forager can in turn perform waggle dances to advertise her location. The more waggle runs are performed for a specific location, the more likely it is that this location will attract recruits that can perform even more waggle runs; the number of foragers at the site thus increases until overcrowding starts to reduce profitability or competition from other sites has drastically reduced the number of unemployed foragers. Despite large individual differences in dance behaviour, rich food sources receive on average more advertisement and are more actively exploited than poor food sources.

Although an employed forager sometimes abandons a poor food source, she does not use information from the dance floor to make that decision. Employed forager bees indirectly estimate the overall nectar intake of the colony in order to determine whether they should abandon a food source or change their amount of advertisement. An individual bee estimates overall nectar

intake by measuring the time before she is unloaded by a *food-storer* bee. Longer waiting times indicate higher nectar intake. If overall intake is high, only very profitable sources are advertised on the dance floor; if intake is low, exploitable but less profitable sources are advertised more vigorously.
The combination of these information feedback mechanisms allows for a flexible and context-dependent allocation of foragers among *all* discovered flower patches. Seeley (1995, p151) even argues that, despite the fact that individual bees possess only limited information about their *own* flower patch, the steady-state allocation pattern is as efficient as the one that would result from foragers that each have complete knowledge about *all* nectar sources.

When compared to the stimulative function of recruitment strategies in ants, honeybees can be said to practice group recruitment: each bee can directly recruit several other bees during its time on the dance floor. The orientational function is very different: whereas ants either lead the follower to the food source – which is time consuming – or leave chemical signposts along the way, honeybees do neither. Natural selection has evolved a communication mechanism that is more adapted to their specific living conditions.

The physical implementation of the recruitment process determines its quantitative performance. For instance, it seems that a colony has little difficulties in rapidly adapting its allocation pattern to a changed environment (Seeley, 1995, p134). Communication in bees is inherently probabilistic and the accuracy of recruitment is not perfect: Michelsen, Andersen, Kirchner and Lindauer (1991) describe an experiment with four sugar-water feeders in the same direction of the hive but at different distances. They measure the number of bees that arrive at each feeder when recruitment is restricted to one of the four feeders. Roughly half of the bees arrive at the feeder for which they have been recruited, with the other half divided between the remaining feeders. Seeley (1995, p126) reports that in more natural foraging conditions, only 1 in about 4 dance-guided searches are successful. Imperfect recruitment could have similar adaptive advantages for bees as for ants.

Similar recruitment mechanisms operate when a colony chooses its future nest site (Seeley and Buhrman, 1999; Seeley and Buhrman, 2001). This is a problem not of resource allocation, but of distributed decision making. When a colony outgrows its hive it divides itself by swarming. The mother queen and about half the workers leave the hive to establish a new colony, while a daughter queen and the remaining workers stay behind. After leaving the hive, the bees gather at an interim site to choose their future nest. The decision process is performed by only a small part of the colony: several hundred scout bees fly from the swarm to search for potential nest sites. On return, they report their findings by means of waggle dances; direction and duration of a waggle run indicate direction and distance of the site. The estimated quality of a potential nest site determines the number of waggle runs in a dance. Unemployed scouts will, just as in the food collection process, follow the dance of one randomly chosen bee. Better nest sites, being advertised by longer dances, have a higher probability of attracting scouts which, after inspection of the site, possibly perform even more recruitment dances. A dozen or so discovered sites could be advertised on the dance floor during the 2 to 3 day process. Unanimity between the dancing bees before take-off of the swarm seems to be a requirement. This is achieved by a combination of two behavioural responses: a small number of bees switch their allegiance from one site and start dancing for another; more important however is the propensity of individual bees to reduce dancing over time or stop dancing altogether and leave the decision making to new scouts. Mediocre sites that do not recruit new scouts fast enough to compensate for this abandonment, soon disappear out of the decision process altogether.

As in the case of nest relocation in *Leptothorax albipennis*, individual bees often visit more than one potential nest site, indicating that direct comparisons between sites might play a role in the decision making strategy. However, preventing bees from making comparisons (Visscher and Camazine, 1999) did not prevent or delay swarms from arriving at a decision, meaning that direct site comparison does not have great influence on the decision process.

Recruitment as a Selective Process

Seeley (1995, p136) compares the process of forager allocation to a natural selection process operating on nectar sources: variation is introduced by foragers discovering new food sources. Reproduction occurs in the form of recruitment to a food source. The finite number of available recruits induces competition between flower patches, resulting in the "survival" of only the most profitable food sources. On the population level, this process leads to an allocation pattern that is well adapted to the changing environment. The nest site relocation process has a similar interpretation as selective process.

There exist, of course, many differences between the recruitment process and "real" natural selection. Nectar sources and nest sites take the place of genes as units of replication. They are not unalterably bound to the organism for its entire lifetime like genes in DNA, but exist as modifiable patterns of nervous system activity. This means, among other things, that Lamarckian inheritance is not impossible in the physical implementation of this system, and a "strategy of errors" could well be interpreted as such.

3.4.3 Summary

The focus in this section has mostly been on the global problem solving abilities that arise from the interaction of poorly informed individuals. Whether it is about ants or bees, chemical or visual signals, forager allocation or nest relocation, the following points summarise the important facts:

- An individual has only limited information about a single location. This information elicits a recruitment response, which correlates on average in strength or duration with some measure of quality.
- A single site attracts a number of recruits in proportion to a combination of measures such as distance, surface (quantity), and quality.

- *Cooperation* between a *finite* number of ants or bees gives rise to a *competitive*, selective process between sites.
- Slightly different mechanisms lead to either a context-dependent and flexible allocation pattern among different sites, or to an unanimous decision about the best site.

3.5 Culture and Society

Selectionist reasoning has been applied to many aspects of cultural and social change. Darwin himself interpreted the historical development of languages within a selectionist framework (Plotkin, 1994, p62).[9] The development of science, an integral part of western culture, has often been phrased in evolutionary terminology: Popper (1972, p261), for example, argued that the gradual growth of scientific knowledge is the result of a *natural selection of hypotheses*. Kuhn advocated a similar view on the role of selection mechanisms, but his process is one of revolutionary episodes of dramatic *paradigm shifts* interspersed with longer periods of *normal science* (Hull, 1988, p12). Evolutionary theorising in economics has an even longer history: Adam Smith's *The Wealth of Nations* (1776) can be easily interpreted as an evolutionary account *avant la lettre*.[10] An overview of evolutionary thinking in economics can be found in (Nelson, 2002). Dennett (1995) discusses the development of morality in evolutionary terms, and refers to the implicit evolutionary elements in Hobbes's *Leviathan* (1651), a treatise on the creation of the state and the appearance of morality. The idea of evolutionary change has also been applied several times to cultural change at large, the most popular of these accounts starting with the introduction of the *meme* concept, a cultural

[9] It has even been suggested that William Jones's theory (1786) on the development of Indo-European languages formed a direct source of inspiration for Darwin's original formulation of the natural selection theory (Kennedy et al., 2001, p245).

[10] Smith's work too has often been named as a source of direct inspiration for Darwin.

equivalent to genes (Dawkins, 1989). Memes are ideas, beliefs, elements of culture that can be passed on from individual to individual mainly through *imitation*, giving rise to a process of cultural change through a differential success in survival and replication of different meme variants. Imitation is a form of social learning that only humans seem to be particulary good at.[11] It is widely accepted that it gives an adaptive advantage to individuals, and thus can be the product of biological evolution. However, Blackmore (1998) argues that, once imitation has appeared, the things that can be imitated – the memes – start leading a life of their own, and that their success is to a large degree decoupled from biological advantage or disadvantage.

The analogy between genes and memes is not always regarded as very fruitful (Jablonka, 2002; Midgley, 2002), and serious doubts exist whether *memetics*, the study of memes, can develop into a sound scientific discipline. There are many reasons for this scepticism: first of all, the concept 'meme' covers a wide variety of cultural elements. Plotkin (2002) distinguishes between at least 6 different functional groups: actions, methods, gossip, artifacts, concepts and social constructions. With such a wide variety, it is difficult to imagine a common physical substrate for implementation. It is usually believed that memes exist in the brain as patterns of activity or connectivity, but this hypothesis has proven so far to be untestable. Memes can also exist *exosomatic* – outside the brain – in books, on CD's, in computer memories etc., and can continue to exist in this form even if forgotten by all humans. There are other differences than just implementational: memes can spread *horizontally* from individual to individual, much like a disease spreads through a population, or be transmitted *vertically* from parents to offspring, like genes (Blackmore, 1998). Variation arises from more than just copying errors: memes can be modified during and after transmission through reinterpretation of the imitating individual (Jablonka, 2002), meaning cul-

[11] There are other, less powerful forms of social learning that some animal species have mastered (Alonso, d'Inverno, Kudenko, Luck and Noble, 2001).

tural inheritance is more *Lamarckian* than *Weismannian*. A further difficulty arises from the fact that evolution occurs simultaneously on many levels of social organisation: informal interactions between individuals lead to the appearance of social institutions that can exert downwards causal influences on the evolution of culture itself (Runciman, 2002).

Although these objections indicate that the analogy between genes and memes should not be taken too literally, they do not invalidate the basic premises of the idea: that imitation leads to the replication of cultural elements, some more readily than others; that variation is introduced in a number of ways; and that the limitedness of resources (human brains, books etc.) introduces selective pressures. However, because of the more complex mechanisms involved, it will prove difficult to move evolutionary theory in the social sciences from *appreciative* to *formal*[12] status.

Another way of looking at the processes that arise from human interaction is the problem-solving perspective. An overview of this approach is given by Kennedy et al. (2001); they argue that cultural evolution has the same result as biological evolution: adaptation. Individual humans are envisaged as searching for solutions through abstract, high-dimensional problem spaces. They evaluate their own ideas and beliefs, and compare with and learn from the experience of neighbouring individuals. What is perceived as better knowledge spreads faster through society, resulting in the overall growth of knowledge and cognition. In this view, individual cognition is considered the product of culture, whereas it was classically believed that culture is the result of individual cognition. In reality, cognition and culture have probably reinforced each other in a circular causal fashion.

[12]Terminology used by Nelson and Winter (Nelson, 2002) to describe the different levels of abstraction in the social sciences: appreciative theory is relatively informal, a verbal account specifying key causal mechanisms; formal theory is more abstract, more rigourous, farther away from empirical substance. Ideally, appreciative and formal theories should be in accordance in a particular field of study, but this is not always so: e.g., in economics most appreciative theories are evolutionary, whereas formal theories are *neoclassical*.

3.5.1 Summary

Processes of historical change in culture and society have been described using the general terminology of biological evolution, although details are different and many issues remain unresolved. Most relevant to this work is the problem solving perspective, which can be summarised in short as follows:

- When searching for solutions to problems, individuals often learn from the experience of others.
- Cooperation between a finite number of humans leads to competition between different facts of knowledge.
- Better knowledge spreads faster through society.
- Cooperation leads to adaptation: it improves knowledge and cognition.

3.6 Selective Processes

The previous sections have presented appreciative accounts of processes occurring in natural and social systems. There are several aspects that they have in common: all of them are population-based processes that can be viewed from a number of perspectives or levels of abstraction. These perspectives, already announced in §3p42, will first be reviewed. Following that, it will be argued that from one of these perspectives all processes can be seen as functioning according to V+SR mechanisms. Finally, it will be discussed how they fit the hierarchical framework of general selection theory.

3.6.1 Perspectives

The processes described in the previous sections are implemented in a range of physical substrates. The physical substrate determines what is possible in a

system: for instance, Weismann's hypothesis (§3.1p45) rejects inheritance of acquired characteristics in biological evolution because of the barrier between somatic and germ cells in sexually reproducing organisms, but in cultural evolution "acquired characteristics" can be transferred during imitation learning (§3.5p66). The physical substrate also affects the pace of each process: biological reproduction takes more time than imitation learning, meaning that biological evolution is generally slower than cultural evolution.

Differences in physical implementation thus lead automatically to disanalogies between the different processes. However, a number of analogies between all these processes exist that are more significant than the disanalogies resulting from the physical implementation. These analogies appear when describing the systems on several different levels of abstraction: firstly, there is the substrate-neutral, algorithmic description of the behaviour of individual elements in the system; secondly, the relatively short-term group dynamics on the system level; thirdly, the long-term perspective of historical change; and finally, the description as a problem-solving process.

Individual Behaviour

In a system consisting of many entities, algorithmic descriptions that "explain" the behaviour of the system as a whole are often in terms of behaviour of the individual entities. These substrate-neutral descriptions are abstractions and idealisations that allow a more simplified reasoning about the mechanisms and processes involved. For instance, the concept of genes as discrete units of information and the description of natural selection in terms of their variation and replication are abstractions of physical reality. However, it is commonly assumed that it is logically possible to construct abstract descriptions that capture the aspects necessary for explanation of the underlying material processes.[13]

[13]Not everyone would agree: Searle (1990), for instance, rejects the possibility of such explanations for the phenomena of human understanding and intentionality.

In the systems described in previous sections, 'individual behaviour' can actually be separated into *two* different perspectives: the perspectives of *replicator* and *interactor* (Hull, 1980). The former is defined as an informational entity whose structure is transferred largely unchanged during replication; the latter is defined as an entity that somehow contains the replicators and interacts with its environment in such a way that replication is differential. A description on the level of the individual can adopt the perspective of the interactor: the emphasis in this description is on the actions and interactions of individuals. Or the description can take the perspective of the replicator and specify mechanisms of variation and selection. It is from this perspective that the processes are most aptly called 'selective' or 'selectionist'. For instance, the process of natural selection was described twice: once from the perspective of organisms, the interactors (§3.1.1p45); and once from the perspective of genes, the replicators (§3.1.1p47). Recruitment in bees has consistently been described from the perspective of interactors, except where it is described as a selective process operating on nectar sources (§3.4.2p63).

Short-term System Dynamics

Certain relatively short-term dynamical effects that result from combined individual behaviour can be observed on a collective level or the level of the system as a whole. The effects are often described as *diffusive*: replicators diffusing across a population of interactors, such as the spread of genes, disease or memes in a population; or as a group of interactors *clustering* in a region of a certain space, such as in insect recruitment, the description of organisms on fitness landscapes (§3.1.1p48) and immune cells in shape spaces (§3.2.1p52). Sometimes, the effects can be observed in the formation of certain physical structures, like pheromone trails in mass-recruiting ants (§3.4.1p58). On this level, properties of the system as a whole (and not of its individual elements) are often modelled with stochastic or deterministic differential equations.

Historical Perspective

The historical perspective is concerned with the long-term picture of historical change that is the result of many iterations of V+SR mechanisms. In biology it is adopted in the reconstruction of the lineages of common ancestry of species; in linguistics in the research of the development of different languages from common precursors. It is from this perspective that the processes can best be described as 'evolutionary'.

Solving Problems

The systems described in this chapter can all be regarded as solving problems or assembling knowledge about the external world; they are therefore often called 'adaptive'. This adaptiveness or problem solving on the system level can be quite unrelated to the problems that are solved by the individual elements themselves: for instance, biological organisms face the continuing problem of individual survival; natural selection results in a population whose successive members become progressively better at surviving. Individual bees solve problems of finding food, navigating and surviving; cooperation in a group of bees results in a resource allocation pattern on the colony level that is well adjusted to the local environment (§3.4.2p61).

3.6.2 Variation + Selective Retention

It is from the replicator perspective that the processes of previous sections are most homogeneously described in terms of Campbell's (1974) framework of variation + selective retention (§1.1.2p6). Each process relies on:

1. Mechanisms that introduce variation.
2. Consistent (on average) mechanisms of selection.
3. Mechanisms for retaining and/or replicating selected variations.

Variation

Here are a few examples of the variation mechanisms that have been described in previous sections:

- Relatively small DNA mutations in natural selection (§3.1.1p46).
- The almost arbitrary recombination of gene segments during the development of undifferentiated cells into lymphocytes (§3.2p50)
- Fine-tuning point mutations in clonal selection (§3.2.1p51).
- Introduction of independently discovered nectar sources on the dance floor by forager bees (§3.4.2p60).
- Discovery of food sources in the neighbourhood of already exploited sources through a "strategy of errors" in ants (§3.4.1p58) and bees (§3.4.2p61).

It can be seen that there are roughly two types of variational mechanisms: those that introduce only small variations on the information that is already present in the system; and those that introduce variation that is relatively independent of existing knowledge. If a process relies mainly on small-variational mechanisms, then it can be said to be *historically constrained*: its evolutionary course is very much dependent on its own past; this is the case in biological evolution. Conversely, small-variational mechanisms can take advantage of certain regularities in the world that would be overlooked by large-variational mechanisms: for instance, there is a reasonable chance that profitable food sources are located in each other's proximity; hence the existence of a "strategy of errors" in addition to the independent scouting efforts of a fraction of the unemployed ants and bees.

In his original description of V+SR mechanisms, Campbell strongly emphasised the *blindness* of variational mechanisms. By this he meant that,

although a selective process can be regarded as solving a certain problem, the generation of variation cannot be *a priori directed* towards the solution of that problem. To use his own words: "In going beyond what is already known, one cannot but go blindly" (Campbell, 1974, p422). Whenever variation *does* seem to be directed, it has to be the consequence of some *previous*[14], more fully blind V+SR process, and/or the result of blind V+SR processes on *other* levels. For example, during clonal selection only parts of a B-cell's genome are especially prone to point mutations, namely those parts that are responsible for the structure of the cell's antigen receptors (§3.2.1p51). This *targeting* of variation is the result of a V+SR process that operates on mutation rates of different DNA segments.[15] Bees decide to advertise a nectar source on the dance floor, based on its profitability and some limited information about the overall nectar intake of the colony (§3.4.2p60). This means that the variation introduced on the dance floor is somewhat directed, since not all discovered food sources are advertised. However, the information used by an individual bee to come to its decision is very limited, and obtained by a process that itself consists of a random sampling of both the food source and the overall nectar intake of the colony. Moreover, a colony maintains a large variance in individual dance thresholds. This is the result of a natural selection process that operates on genes of individual bees; the *composition* of the colony in terms of individuals with large differences in dance thresholds is the result of a natural selection process operating on *colonies*. It can therefore be seen that all knowledge used to direct the variation on the dance floor is itself the result of other, more fully blind selective processes.

[14]Small-variational mechanisms lead to historical constraints on the generation of new variation, but what is meant here is more general.

[15]Strictly speaking, this example does not even violate Campbell's often-misunderstood notion of 'blind', since mutations in other parts of the genome would barely alter the shape of the receptors; hence, they would not affect the course or outcome of the clonal selection process, but affect the operation of the cell in other, possibly harmful ways. Nevertheless, the difference in mutation rates needs to be explained somehow: a sensible hypothesis seems that it is the result of selective processes on other levels of biological organisation.

Selection

From the interactor perspective, interactions can be competitive, cooperative, or a mixture of both. Whether competitive or cooperative, they will always result in some form of selective pressure on the variance of replicators, due to *limitedness of resources*. For example, competition between animals for a limited commodity in the environment leads to selective pressure on different alleles in the gene pool, with only the fittest alleles prevailing (§3.1.1p47). Cooperation between a limited number of honeybees leads to selective pressure on nectar sources, with only the most profitable ones surviving on the dance floor (§3.4.2p63). Here are a few examples of selection mechanisms:

- Death of organisms because of maladaptation to the environment or because of direct competition with other organisms.
- Mate choice in sexually-reproducing organisms.
- Selection of B-cells by binding to an antigen on a parasite (§3.2.1p50).
- Selection of a dancing bee by an unemployed forager (§3.4.2p60).
- Abandonment of a food source by a forager bee (§3.4.2p60).

It can be inferred from the examples that there are at least two kinds of selection mechanisms: *positive* and *negative*. Positive selection leads to replication or reinforcement of the selected variants; negative selection leads to elimination of the selected variants. In reality, each selective process most likely uses a number of different positive and negative selection mechanisms.

Replication and Retention

Finally, there are mechanisms that ensure that positively selected variants come to dominate in number or strength:

- Replication of DNA through reproduction of organisms (§3.1.1p46).
- Cloning of activated B-cells (§3.2.1p50).
- Transmission of disease (§3.3p53).
- Recruitment to a food source or nest site (§3.4p55).
- Imitation learning (§3.5p66).

The mechanisms of replication and variation are sometimes linked, like copying errors that occur during replication of DNA. However, on average replication needs to be fairly accurate; if not an evolutionary system would not be able to stabilise and exploit the knowledge it has accumulated about regularities in its environment. In some mechanisms, selected variants are not explicitly replicated, but only retained or reinforced.

3.6.3 General Selection Theory

According to Campbell (1974, p421), the mechanisms of V+SR are fundamental to all genuine increases in knowledge, or to all increases in fit of system to environment. He called the natural selection paradigm – following in the tracks of Popper (1966) – "...the universal non-teleological explanation of teleological achievements, of ends-guided processes, of "fit"" (Campbell, 1974, p420). It provides a plausible explanation for the apparent purposiveness of the natural and social control systems of cybernetics. For example, one could easily be misguided in thinking that the near-optimal allocation of foragers to nectar sources needs to be the result of a purposeful controlling unit with complete knowledge about the profitability and quantity of all available nectar sources. Yet interaction in a population of bees, with each bee using a limited repertoire of behavioural responses and communication mechanisms, gives rise to a selective process that achieves exactly the same *without* central control.

The "parameters" or criteria of the recruitment process – for instance, the composition of a colony in terms of individual dance thresholds – are but "trials" of the natural selection process on the level of the genes. In turn, the recruitment process affects the evolutionary course of the natural selection process itself: since nectar-producing flowers often use bees as pollinators, they are dependent on them for procreation. A selective process operating on the dance floor thus leads to selective pressure on the reproductive success of patches of flowers in the neighbourhood of the hive. It is this interplay of upwards and downwards causal forces generated by V+SR mechanisms that led Campbell to the hypothesis of general selection theory (§1.1.2p6): all truly creative knowledge processes in the natural world form part of a nested hierarchy of V+SR processes, each level in the hierarchy a *shortcut* or *vicarious selector* of the "original" natural selection process. General selection theory does not imply a perfect analogy of all V+SR processes to the neo-Darwinian theory of biological evolution. And it does certainly not treat all selective processes as isolated or existing in their own right: they are all the product of a selective process operating on a biochemical level.

The selective processes described in this chapter cover only part of Campbell's hierarchy, but it can be easily seen how they fit in: clonal selection, a selective process in single vertebrate bodies, occurring in somatic time; the diffusive process of disease propagation as the short-term dynamical result of differential contact patterns and infectiveness; the recruitment process in colonies of ants and bees. All are processes whose criteria have been set by natural selection, and at the same time they are genuine selective processes themselves that are the result of repeated interactions between individuals. And the hierarchy of selective processes extends higher, into the realm of the social sciences: natural selection has produced an animal with the ability of imitation learning; imitation has lead to evolving cultures that give rise to languages, a scientific process, technological innovations, societal institutions and commercial firms that all evolve through V+SR mechanisms themselves.

3.7 SDS as a Selective Process

In the first chapter, SDS was defined – rather abstractly – as a population-based algorithmic process (§1.2.1p13). The second chapter gave a concrete example of an SDS variant; a clear distinction was drawn between the algorithmic description of individual agent operation (§2.2.2p25) and the dynamical properties and problem-solving capabilities of the population as a whole (§2.2.3p27). In this chapter, after having discussed several concrete examples of selective processes and selective processes in general, it will now be argued conclusively that SDS itself can be regarded as a selective process.

Algorithmic descriptions of SDS variants are usually presented from the perspective of the interactors: the individual SDS agents. Contrastingly, the *hypotheses* maintained by these agents are the replicators, and from their perspective SDS can be described in terms of V+SR mechanisms. For instance, for standard SDS, variation is introduced in two ways: in the initialisation phase of the algorithm; and at the end of each diffusion phase when inactive agents that contacted other inactive agents adopt new random hypotheses. These new hypotheses are independent of the hypotheses already present in the population, meaning that standard SDS does not utilise small-variational mechanisms. Negative selection occurs at the end of the test phase, when agents that failed the test discard their present hypothesis and become inactive. Positive selection occurs in the diffusion phase, when inactive agents pick active agents for communication. Positive selection leads to replication, the copying of the hypothesis parameters of the active agent by the inactive agent. This replication mechanism is completely accurate. Retention occurs when agents pass a test and keep their present hypothesis.

The combination of these mechanisms gives rise to a stochastic process exhibiting dynamical behaviour that is typical of the selective processes discussed in this chapter: the behaviour can be interpreted as hypotheses diffusing across a population of agents, much like the spread of diseases or memes.

Conversely, it can also be interpreted as agents clustering in points of the solution space, much like the effects of insect recruitment. The problem solving capabilities of SDS as a system are attributed to these effects – diffusion or clustering – on the collective level: since good solutions are on average retained and replicated more often than poor solutions, they can attract larger clusters; good solutions can then be identified from clusters of agents in the solution space. Each variant of SDS exhibits its own distinctive dynamical behaviour. For instance, standard SDS is greedy: the best solution discovered so far will attract a large cluster of agents, while slightly poorer solutions are not able to simultaneously attract a stable population. Standard SDS also exhibits *punctuated equilibrium* behaviour, as observed in Figure 2.6 (p35): long periods of quasi-equilibrium are interspersed with short periods of rapid convergence.

Although SDS has some elements in common with all of the selective processes described in this chapter, the ones it resembles most are the recruitment processes in honeybees, especially on an abstract level of how interactions between individuals give rise to information feedback mechanisms on the colony level that reduce the uncertainty experienced by single bees. During foraging, a honeybee seems to behave as a mixture of a hermit and a context-free SDS agent (§2.3.1p37): hermit because a bee does not always advertise its foraging location on the dance floor; context-free because she bases the probabilities of dancing and abandonment on limited information about the overall foraging activity of the colony. The outcome of this process is an allocation pattern that reflects the quality and quantity of different nectar sources in the environment. The nest site relocation process of honeybees is more like standard SDS: the greediness of the process results in a clear decision about the best site. One of the real conceptual differences between recruitment processes and SDS variants described so far is the accuracy of replication: SDS agents do not have the equivalent of a "strategy of errors" to exploit regularities in the solution space.

3.8 Rationale

At the start of this chapter (§3p43), 4 reasons were mentioned for discussing the details of selective processes. These reasons are reviewed in this section.

3.8.1 What Does SDS Resemble?

This chapter has provided a partial answer to question 1 of the roadmap (p15). It has abstracted the operation and behaviour of several natural and social systems to an extent that comparison with SDS became possible. As a result of the comparison, the abstract definition of SDS as a population-based algorithmic process has become more concrete: SDS is now defined as a selective process that can be characterised in terms of V+SR mechanisms.

3.8.2 Inspiration

This chapter provides a partial answer to meta-level questions C and D of the roadmap (p18). Of the four perspectives on selective processes (§3.6.1p67), the historical perspective is the least relevant to the study of SDS. The other three perspectives – individual, group-dynamics and problem-solving – have corresponding perspectives in the study of SDS and lead naturally to answers to some of the questions of the roadmap: the descriptions of individual behaviour correspond to the level of algorithmic descriptions of SDS variants. Mechanisms of variation and selection can provide inspiration for algorithm development, and therefore answer question 5 of the roadmap (p16). For instance, small-variational mechanisms that fine-tune the adaptation of a system seem common in selective processes. Introducing a similar mechanism in SDS could solve – in certain solution spaces – the problem observed in Figure 2.6 (p35) and discussed in §2.2.4p36: the long periods of time that standard SDS spends examining sub-optimal solutions. An example of how such a mechanism may operate is found in Appendix A.3 (p146).

The stochastic process underlying SDS can be analysed using models from fields such as insect ethology, epidemiology or population genetics, thus providing the tools for answering questions 6, 9, 10 and 11 of the roadmap (p17). Conversely, mathematical models of SDS, such as developed in (Nasuto, 1999), can be of use in these and other disciplines.
Finally, the problem-solving perspective is related to question 2 (p15): selective processes have been described as leading to adaptation, resource allocation, and decision making. This point will be addressed in the next chapter.

3.8.3 General Selection Theory and Consilience

General selection theory – a form of systemic consilience – can be used to speculate about systems that are not yet completely understood. These speculations lead to scientific hypotheses that can then be investigated in a reductionist manner, in order to establish the exact nature of the V+SR mechanisms involved (if any). The road from systemic speculations towards reductionist research has been taken several times in the past: for instance, an immunologist whose work confirmed the selectionist theory of clonal selection, Edelman, went on to develop a selectionist theory for brain development and memory formation (Edelman, 1987).[16] Similarly, the resemblance of phenomenological properties of attention to dynamical properties of SDS led to the speculation that the neurobiological mechanisms of attention are in an abstract sense similar to SDS operation (Nasuto and Bishop, 1998; Nasuto et al., 1999). It can now be understood that this is because both may be the product of similar V+SR mechanisms. Although the systemic consilience of general selection theory is not a rational argument for the validity of this theory, it does make it a more enticing hypothesis to investigate.

[16]The theory is controversial and has therefore not been treated in this chapter.

3.8.4 Why Selective Processes?

One might ask the question *why* selective processes reoccur at so many organisational levels. A first answer comes from Campbell himself: it seems to be the only non-teleological explanation for teleological achievements. A second answer is that they simply *have* to occur: wherever a population of interacting elements exists, subjected to some form of limitedness of resources, selective pressures arise. Finally, to answer the question why distributed selective processes are so much more abundant than centralised directive processes is that they are much more information-efficient: a colony of interacting bees – each with limited information – achieves the same foraging efficiency as could be achieved by a centralised control unit with unlimited information. A distributed, capitalist economy is more effective in producing goods and services than a centralised plan economy, because it is virtually impossible for the central planner to obtain all necessary information.[17]

3.9 Conclusion

This chapter has concretised the definition of SDS from 'algorithmic process' to 'selective process'. Selective processes have been described in great depth, offering inspiration for the design of SDS variants and indicating how the study of SDS can cooperate with other scientific disciplines. The problem solving perspective of selective processes featured strongly throughout the chapter: adaptation, resource allocation and decision making were terms frequently used. In the following chapter, the problem-solving perspective of SDS will be analysed in more detail.

[17]This argument should not be interpreted as justifying unregulated capitalism: economic competition has as result the survival of companies trying to maximise their short-term profits, and is not concerned with environmental effects or well-being of individuals.

Chapter 4

Problem Solving and Search

The previous chapter described a number of natural and social processes, and explained in what respects SDS resembles them. From one perspective, these processes can all be regarded as solving certain problems through the mechanisms of variation, selection, and replication or retention. Alternatively, they can be regarded as solving problems by *generating* candidate solutions and *testing* them.[1] In this interpretation, a clear correspondence exists between the selective processes of general selection theory and the generate-and-test or search processes of the physical symbol system hypothesis in AI, even if some strong interpretational differences arose from the original meaning of 'search' in AI (§1.1.3p7). In recent years, this meaning has been expanded to incorporate different kinds of search problems and search methods.

Section 4.1 gives an overview of these different meanings of search and defines search problems in general. Section 4.2 gives examples of a few common search methods or metaheuristics in AI, followed by a general discussion of generic search methods and their effectiveness. Section 4.3 then discusses in what sense SDS performs search. Finally, Section 4.4 concludes the chapter.

[1] This is, in fact, how Plotkin (1994) and Dennett (1995) refer to these processes.

4.1 Search

4.1.1 Different Meanings of Search

Search is such a prevalent activity in daily life that it can be easily understood intuitively as the act of attempting to find or discover something. On more formal grounds, it is possible to distinguish between different types of search problems in several ways. For instance, Mitchell (1996) argues that in computer science the word 'search' has at least three (overlapping) meanings:

1. *Search for stored data* (*data search*) requires finding a piece of data in a larger collection of explicitly stored data. This notion encapsulates the classical meaning of search in computer science (Knuth, 1973).

2. *Search for paths to goals* (*path search*) is the problem of finding a list of steps that will move from a given initial state, through some intermediate states, to a given goal. This form of search was central to many problem solving techniques of early symbolic AI (Winston, 1992). Paths are not explicitly stored, but are created as the search proceeds.

3. *Search for solutions* (*solution search*) is the problem of finding a solution in a large space of candidate solutions. In this case it is not the steps leading to a solution that are important, as in path search, but the solution itself. The solutions are not explicitly stored; rather, candidate solutions are created as the search process proceeds.

Each of the above meanings has unique properties, and quite distinct techniques exist for solving problems in either of the three cases. However, they do overlap considerably. Solution search is the most general category of the three. It subsumes path search, since every path can always be encoded as a candidate solution. It also includes data search, since the location of each individual piece of data in the larger collection can be interpreted as a candidate solution to the search problem.

Under the umbrella notion of solution search, a different criterion distinguishes between two types of search problems. Here the distinguishing feature is whether the problem is one of *existence* or *optimality* of a solution:

1. *Constraint Satisfaction Problems* (CSP) require finding a solution fulfilling a set of constraints. The problem is solved when such a solution is found, or when it is established that no solution exists.

2. *Optimisation Problems* require finding as good a solution as possible within some limits on time and computation. The definition of 'good' is dependent on the specific problem.

4.1.2 Search Terminology

A single search problem can often be interpreted in different ways, and each interpretation has its own importance. The following paragraphs define a disambiguating terminology for use in this thesis, consistent with the terminology used in Chapter 2. These terms do not have a universally accepted meaning, and their definitions can differ from the ones used in other work.

Search Space, State Space and Solution Space

Search space is used to designate the large collection of data that is processed in data search. The search space consists of a finite number of explicitly stored data items, each item possessing one or more properties. The items in the search space are arranged, ordered or labelled in a suitable manner so that they can be indexed unambiguously. Often the ordering or labelling is used to search more efficiently: for example, a *sorted* array of numerical values is searched more efficiently than an unsorted one; or a *hash* function can be used to find a complex entry in a database by a simple key value.

State space describes the structure searched during path search. It is usually a directed graph, where the possible states are nodes in the graph and links between nodes designate potential operations leading from one state to a successor state. During search, the state space is incrementally constructed as a *search tree*: the initial state is the *root node* of the tree. The goal states are *leaf nodes* of the tree. Often, a *heuristic score* function imposes a measure on the graph in an attempt to guide the search process towards the goal.

Solution space is the structure searched in solution search. In its most abstract form it is a (possibly infinite) set of candidate solutions, but often the set forms an n-dimensional, real-valued parameter space or a graph with some "natural" neighbourhood relationship between connected nodes.

Problem and Representation

Often, a distinction exists between a problem itself and its *representation* that is defined as one tries to solve it. For instance, finding the minimal value of a continuous function over a real-valued domain is a valid problem, even though a discrete representation is always adopted when solving it on a digital computer. For this reason, Jones (1995) introduced a distinction between the *object space* – the set of candidate solutions, independent of representation – and the *representation space* – a set of candidate solutions in a representation that can be more easily manipulated by the search process. This distinction will not be made explicitly here. Whether a problem is described in terms of object or representation space should be clear from context. It will also be assumed that, if a search problem is described in terms of object space, a conversion towards a suitable representation space can be made at solving time. This work is not concerned with such conversions, but finding suitable representations is one of the most important aspects of the problem solving process: problems, unsolvable due to time constraints in one representation, are sometimes easily solved using an alternative representation.

Constraint Satisfaction versus Optimisation

Constraint Satisfaction Problems can be defined as (Roli and Milano, 2002):

1. a set of n *variables* $\{x_1, \ldots, x_n\}$;
2. a set of *domain values* for each variable, $D_i \; \forall i = 1, \ldots, n$;
3. a set of m *constraints* $c_j(Y_j)$, $Y_j \subseteq \{x_1, \ldots, x_n\}$, $\forall j = 1, \ldots, m$.

When values are assigned to all variables in a constraint, it is either *fulfilled* (evaluates to *true* or 1) or *unfulfilled* (evaluates to *false* or 0). Solving a CSP means finding an assignment for all variables so that all constraints are fulfilled, or determining that no such assignment exists.
In what is usually designated by the term CSP, the constraints are fairly simple expressions in the variables itself. However, it is possible that constraints involve more complicated symbolic relations. For instance, the *Queen of Hearts* example (§2.2.4p28) can be reformulated as a CSP as follows:

> Find x, with $D_x = \{1, 2, 3, 4, 5\}$, such that the constraints
> (value(x) = *queen*) and (type(x) = *hearts*) are fulfilled.

In this case the functions value() and type() can only be defined through enumeration over the entire (finite) solution space. In data search, this enumeration forms the explicitly stored search space itself.
All problems regarding the *existence* of a solution, such as finding a desired object among a set of objects, as well as *exact* string and pattern matching problems, can be written in the form of CSP.

In their most general form, optimisation problems are defined by the three properties of CSP, in conjunction with a fourth property:

4. an *objective function* in the variables $\{x_1, \ldots, x_n\}$ to be optimised (minimised or maximised).

If the domains of the variables are finite or countable sets, then the problem is called a *Combinatorial Optimisation Problem* (COP); otherwise the optimisation is called *continuous*. Although methods applicable to COP can often be applied to continuous solution spaces, they are quite distinct flavours, and their fields of study have diverged significantly (Papadimitriou and Steiglitz, 1982). Issues specific to continuous optimisation will not be discussed in the remainder of this thesis.

A hybrid class of problems is formed by *Maximum Constraint Satisfaction Problems* (MAXSCP). They are problems with the properties of CSP, but with the difference that they potentially lack a solution that fulfills *all* constraints; they require finding a solution fulfilling *as many constraints as possible*. Alternatively, they can be regarded as maximisation problems for which the objective function is the number of fulfilled constraints. The *Matching on Mars* example (§2.2.4p31), where the task is to find the solution maximising the number of corresponding micro-features, can be regarded as a MAXCSP.

Another hybrid is *Multi-Objective Optimisation* (MOO). The classical notion of 'optimality' becomes ambiguous when a problem has more than one objective function: variable values optimising one of the objective functions might not optimise the others. Therefore, the purpose in MOO is to find all or some of the *Pareto-optimal* solutions. A solution is called Pareto-optimal if no other solution exists in which at least one objective function has a better value and no objective functions have worse values.

Objective Functions and Test Scores

The notion of 'objective function' in optimisation problems was introduced above. Each point in a solution space has an associated value, defined by the objective function. The solution space is thus formally defined by the pair $(\mathcal{S}, f)$, where $\mathcal{S}$ is the set of solutions, and $f : \mathcal{S} \rightarrow \mathcal{Y}, \mathcal{Y} \subseteq \mathbb{R}$ the objective function. An example of an objective function was given in Figure 2.5 (p35).

A related measure in the context of SDS is the *test score*. It is the relative frequency with which the outcome of the test procedure results in an agent being active. It is not only dependent on the values of the objective function, but also on the particular test procedure used. Formally, the test score forms a mapping $t : \mathcal{S} \rightarrow \mathcal{T}, \mathcal{T} \subseteq [0, 1]$. 'Test score' is a most important concept in the context of SDS, since it – and not the values of the objective function – determines the behaviour of a population of SDS agents. Choosing a good test procedure is thus almost as important as choosing a good representation.

4.2 Search Methods

4.2.1 Examples of Generic Search Methods

A large number of generic search methods (*metaheuristics*) for constraint satisfaction and optimisation problems have been developed over the years. Several of them use the selective processes of Chapter 3 as metaphor, others have been developed in different contexts. The following sections present a non-exhaustive overview of relevant search methods; emphasis is on those aspects that distinguish a method from other methods.

Random Generate-and-Test and Systematic Search

One of the simplest of search methods, *random generate-and-test* (RGT), generates random solutions and evaluates them, one after the other (Kennedy et al., 2001, p71). The best solution found is returned at the end of the search.

Its deterministic variant, *systematic search*, generates solutions in a strict, predefined order. This excludes the possibility that a solution is evaluated twice. An example of systematic search is Template Matching (§2.1p21).

Random Walk

Another simple search method is *random walk* (RW) (ibid.). A first solution is generated and evaluated. Subsequent solutions are generated by modifying a random small part of the current solution. The RW search process can be envisaged as moving randomly and in small steps through the solution space. Methods exploring the neighbourhood of the current solution are often called *local search* methods. For example, with solutions encoded as binary strings of length n, a common RW strategy is to randomly pick one of the bits and flip it, moving from the current node to one of the neighbouring nodes on the n-dimensional hypercube. For certain problems, local search requires only partial evaluations to update the value of the objective function after a step. For instance, in MAXCSP problems, only those constraints that contain the variable whose value has been changed need to be re-evaluated.

Hill-climbing

Hill-climbing strategies are local search methods that modify a solution and only accept the changes if the modified solution is better than the current solution (ibid., p72). For instance, *steepest-ascent* hill-climbing starts from a randomly chosen position and always moves towards the *best* neighbouring solution. The *first-improving* variant moves to the first evaluated solution that is better than the current one. *Gradient ascent* uses information present in the derivatives of the objective function (when computable) to decide on the direction of the next step. The effectiveness of hill-climbing methods is dependent on the properties of the objective function: when an objective function contains many local optima with respect to the neighbourhood investigated by the search strategy, hill-climbing gets easily trapped in one of the local optima. Hill-climbing is therefore often used in conjunction with a *restart* strategy: if trapped in a local optimum, restart the search from a randomly chosen position in the solution space.

Simulated Annealing

Simulated Annealing (SA) draws on a metaphor from physics: the slow cooling (*annealing*) of molecules into a regular crystal (Kirkpatrick, Gelatt and Vecchi, 1983). As optimisation algorithm, it follows a strategy similar to first-improving hill-climbing, with the main distinction that it sometimes accepts solutions that are *worse* than the current one. If a newly evaluated solution is better than the current solution, it is always accepted. If a new solution is worse than the current one, it is accepted with probability:

$$p(\Delta E) = \exp\left(-\frac{|\Delta E|}{T}\right) \tag{4.1}$$

where ΔE is the difference between the current and new solution, and T acts as a control parameter (a hypothetical temperature of the system). When T is high poorer solutions are more easily accepted than when T is low. The algorithm is completed with the specification of an *annealing schedule*, a recipe for lowering the temperature from high values towards 0 as the search progresses. Unlike simple hill-climbing strategies, SA can escape local optima without the need for a restart, especially early on in the search.

Tabu Search

Tabu Search (TS) adds the concept of *memory* to local search methods like RW and SA: a list of already visited but poor locations in the solution space – the *tabu list* – prevents the search process from needlessly returning to these locations (Taillard, Gambardella, Gendreau and Potvin, 2001).

GRASP

In the *Greedy Randomised Adaptive Search Procedures* (GRASP) metaheuristic, each iteration of the algorithm consists of two phases: the *construction* phase in which a starting solution is constructed – as opposed to random

selection of starting solutions – followed by a local search phase such as hill-climbing (Resende and Ribeiro, 2003). The construction phase starts with an empty solution. In each step of the construction phase, the current partial solution is extended with a single element. The incremental costs of all candidate elements are evaluated, and one of the extended partial solutions is selected for the next step of the construction phase. In the *greedy* variant, it is simply the best extended partial solution that is retained, but other choice heuristics are possible. When a solution is completed, a local search method is used to improve it. When the local search method reaches a local optimum, the process is restarted in the construction phase.

Evolutionary Computation

The paradigm of *Evolutionary Computation* (EC) (Kennedy et al., 2001, Chapter 4) is inspired by the principle of adaptation through natural selection (§3.1p44). Most EC methods are population-based; in other words, a group of individuals explore the solution space in parallel. The main mechanisms of natural selection are implemented: variation, selection and reproduction; a fitness measure determines the probability of reproduction. The metaphor of fitness landscapes (§3.1.1p48) is useful to envisage the search process: the population adapts by gradually moving towards regions with high fitness values. Four areas developed quasi-independently: *Genetic Algorithms*, *Evolutionary Programming*, *Evolution Strategies* and *Genetic Programming*. Recently, those four areas have started to converge; many *hybrid* algorithms have been developed, and aspects that were originally representative for one approach have been incorporated into other approaches. The following brief exposition describes the four areas mainly in their historical context.

Genetic Algorithms (GA) are the best known of the Evolutionary Computation paradigm. They draw on the metaphor of genetic inheritance at the level of the individual. Candidate solutions constitute an individual's

Initialise;
repeat
  Calculate fitness for each individual;
  Select individuals for next generation;
  Apply crossover and mutation;
until (halting condition is met)

Table 4.1: *Standard* GA *operation.*

genetic material (the *chromosome*); originally, GA used only fixed-length binary encodings. These chromosomes are allowed to evolve over a number of generations. Fitness is calculated for each chromosome from the objective function under optimisation; often, it is the raw value of the objective function itself, or proportional to it. The operation of a standard GA is summarised in Table 4.1. A typical population size is 20-200.

Selection is based on the fitness of chromosomes: the higher the fitness, the higher the probability that a chromosome is selected for the next generation. In *roulette wheel* selection, the probability of selection is proportional to the fitness value. *Elitist* strategies ensure that the solutions with highest fitness are always selected. In *tournament* selection, individuals have to compete directly with one or a few other individuals to determine selection.

Variation is introduced through the mechanisms of *mutation* and *crossover*. A common mutation strategy is to flip each bit in the chromosome with a small probability p_m; this is a small-variational mechanism, resulting in small steps in the solution space. Crossover occurs between two randomly-paired individuals: with a certain probability p_c, the chromosomes are broken into two or more parts; offspring is generated by combining parts from both parents. Crossover can result in large variation between parents and offspring. As the search progresses, the variational effect of crossover decreases: the population becomes more uniform by converging on a small region of the solution space.

Evolutionary Programming Evolutionary Programming (EP) does not simulate the mechanisms of genetic inheritance, but rather the adaptiveness of behaviour. Originally, individuals were finite state machines, mainly used to solve prediction problems. Fitness of individuals is calculated by exposing them to the environment and evaluating their prediction performance.
EP relies on mutation to generate variation, but the probability of mutation is typically much higher than in GA. Each parent usually produces 1 child; children are added to the current population, and half of the population is selected for the next iteration, either directly based on fitness ranking, or through tournament selection: an individual is paired with a number of other individuals and scores points with a probability based on its own fitness relative to the fitness of its opponent.

Evolution Strategies In Evolution Strategies (ES), solutions are usually encoded as real-valued parameters. Variation is generated by mutation, as well as by *recombination* (similar to crossover in GA). Parameters are mutated by adding normally distributed random numbers with zero mean value to the current values, so that individuals of the next generation explore the area in the solution space around the solutions encoded by the parents. The amount of mutation (controlled by the variances of the normal distribution) itself is subjected to variation and selection, leading to *self-adaptation* of the algorithm. Recombination can involve two parents (*local* recombination) or the entire population (*global* recombination), and is implemented in one of two ways: it is either *discrete*, meaning that a parameter value is selected from only one of the parents; or *intermediate*, meaning that the parameter value of the child is a *linear combination* of the parameter values of the parents. Typically around 7 children are produced for each parent. Selection occurs as in GA or EP through fitness-proportional or tournament strategies.

Genetic Programming From one perspective, *Genetic Programming* (GP) can be regarded a subfield of GA. GP evolves executable programs represented as tree structures instead of the binary string representations of GA. Additionally, in GP the structures evolved are generally not of fixed length. The fitness of individuals is calculated by executing the evolved programs and measuring their performance in solving the problem.

Artificial Immune Systems

Artificial Immune Systems (AIS) model aspects of vertebrate immune systems to solve pattern recognition and optimisation tasks (de Castro and Timmis, 2002; de Castro and Timmis, 2003). One approach is to use the affinity maturation concept of the clonal selection theory (§3.2.1p50). A few differences with EC approaches are the use of 'affinity' to measure the quality of a solution, together with the metaphor of shape space (§3.2.1p52); the sole reliance on mutation to generate variation; a reproduction rate that is proportional to affinity; and a mutation rate inversely proportional to affinity.

Ant Colony Optimisation

Among a number of optimisation methods based on social insect behaviour (Bonabeau et al., 1999), *Ant Colony Optimisation* (ACO) is the most established. It is based on the metaphor of trail-laying, trail-following ants (§3.4.1p58), and has been used to solve different types of COP, of which the Travelling Salesman Problem (TSP) is the best known. TSP consist of the task of finding the shortest closed tour in a fully connected graph, visiting all nodes exactly once. In ACO, artificial ants construct a tour by visiting nodes. At completion, they each lay an amount of artificial pheromone on the links travelled, proportional to the overall quality of the tour. Links belonging to high-quality tours thus receive on average higher amounts of pheromone than links in poor-quality tours. During the construction process, each ant selects

the next node based on a probability distribution reflecting the amount of pheromone on all admissible links. Pheromone evaporates at a fixed rate, preventing poor links from being amplified by accident. ACO is a population-based constructive metaheuristic; it is best used in conjunction with a local search method to improve the solution found by the constructive phase.

Memetic Algorithms

Memetic Algorithms (MA) combine individual local search methods with a population-based global search (Moscato, 1989). They are based on the metaphors of scientific or cultural evolution: phases of individual improvements to ideas or theories alternate with phases of cooperative and competitive interactions on the population level. In the original formulation of MA, a number of individuals (16 in this case) is arranged on a ring. Each individual starts by adopting a random solution and improves it through a local search method such as SA. Individuals then compete with neighbouring individuals or cooperate with distant individuals. Competition results in one individual copying the solution of its neighbour; cooperation results in exchange of information through GA-type crossover operators. The local and global search operations are repeated until a stopping criterion is satisfied.

Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) is an abstraction of individual and cooperative problem solving in human society (§3.5p66) (Kennedy et al., 2001). The search process is envisaged as a group of particles flying through solution space. The position and velocity of particles is updated based on prior individual experience and the experience of other, neighbouring particles.

Scatter Search

Scatter Search (SS) is a population-based metaheuristic that is sometimes classified under the EC paradigm, although it originated from a different context (Glover, Laguna and Marti, 2003). As in GA and ES, new solutions are generated by combining others, but the combination strategies are different and more diverse. SS also contains explicit mechanisms to ensure that a sufficient diversity in the population is maintained during the search process.

4.2.2 Search Methods: General Discussion

The previous pages presented a short overview of some well-known search methods. This overview is far from complete: new methods and hybridisations of two or more approaches continue to appear on a daily basis. The aim of the overview – rather than being exhaustive – is to highlight general aspects of search methods, and to provide a catalogue of ideas that can be used for the design of efficient search algorithms; that aim will be developed further in this section. Subsequently, some theoretical remarks on the effectiveness of metaheuristics are discussed at the end of this section.

Properties of Search Methods

Based on their operation, the search methods of the previous section can be divided into two broad categories:

1. *Motive* methods in which the search moves through the solution space under control of a search strategy or search operator. At each position in the solution space, the search process performs a certain amount of work to evaluate the current solution or to decide where to go next. Examples include RGT, local search methods and GA.

2. Methods that *construct* a solution in a number of steps. In each step an amount of work is performed to decide how to extend partial solutions.[2] Constructive methods are often combined with motive methods. Examples from the previous section are GRASP and ACO.

Another important feature is the amount of work performed when evaluating a solution. Most methods rely on *complete* evaluations of candidate solutions. Exceptions exist for local search methods applied to certain solution spaces: only part of the objective function needs to be re-evaluated when making a small modification to the current solution. In some EC methods, fitness is implicitly or explicitly based on partial knowledge of the objective function value (e.g., Ochoa-Rodriguez, 1997; Hahn, 1997). A final exception is the class of constructive methods, where at each step in the construction phase only *incremental* costs are evaluated. Some other distinguishing properties of search methods are:

- *Individual versus population-based searches.* Methods like hill-climbing usually perform single walks through the solution space. Such individual methods also exist in parallel forms (Eksioglu, Pardalos and Resende, 2002), but their operation is not explicitly based on parallelism. Conversely, other methods perform multiple *interacting* walks through the solution space. This means that the operation of these methods relies specifically on the interaction between individuals. Even if these procedures are executed on single serial machines, they are still in essence population-based, parallel methods. Examples include GA, MA and SS.

[2]These constructive search methods can be regarded as a somewhat more inclusive class than the heuristic path-search methods of symbolic AI. Similar to path-search methods, they can be regarded as motive too: the space of all partial solutions forms a tree with the empty solution as the root node and all complete solutions as leaf nodes; a directed link from parent to child node exists if the child solution can be obtained by extending the parent solution with a single element. Constructing a solution can then be envisaged as descending the tree from the root node towards a leaf node.

- *Randomised versus deterministic.* Methods can contain randomised steps or be completely deterministic. However, determinism or randomness are rarely essential features of a particular search method: deterministic strategies can easily be randomised and randomised steps can be replaced by deterministic ones. In practice, randomised methods use (deterministic) pseudo random-number generators anyway.

- *Unguided versus guided.* RGT and RW are unguided searches: they do not employ information from the objective function in the search strategy. Most other metaheuristics attempt to exploit regularities and use information from the objective function to guide the search.

- *Directive versus selective.* For guided methods, there is a subtle distinction between methods that perform evaluations before deciding where to move next, and methods that generate new solutions and then select to retain some solutions after evaluation. Examples of the former are steepest-ascent and gradient-ascent hill-climbing, examples of the latter include GA and AIS. This distinction runs parallel to the discussion of blind and directed variational mechanisms in selective processes (§3.6.2p71). Ultimately, it reduces to a distinction between methods with *explicit* hill-climbing *strategies*, and methods with *implicit* hill-climbing *behaviour*.

- *Memoryless, implicit and explicit memory.* TS maintains an explicit history of locations that have been visited. On the other side of the spectrum, RGT and RW methods are completely memoryless. In between are a number of metaheuristics that contain *implicit* memory mechanisms: for instance, for GA and SS a history of the search process is implied in the population; for ACO, pheromones constitute the memory. This concept of memory was proposed as a unifying perspective on metaheuristics in the framework of *adaptive memory programming* (Taillard et al., 2001).

- *Non-adaptive, adaptive and self-adaptive.* For each metaheuristic, a number of parameters affecting the search strategy need to be chosen. Methods that have search parameters fixed for the entire search are non-adaptive: e.g., a simple hill-climber. Methods that allow parameters to change as search proceeds, in response to information from the search process, are adaptive: e.g., the temperature in SA and the variable mutation rate in AIS. Methods that subject the search parameters itself to a search are self-adaptive. An example of the last type is ES, where the mutation rate itself is subjected to mutation and selection.[3]

- *Convergent versus non-convergent.* Many population-based metaheuristics are convergent: as the search proceeds the individuals of the population converge upon a limited region of the solution space; examples include GA and PSO. Other methods are non-convergent because they lack interaction or because they explicitly attempt to maintain diversity in the population; an example of this last case is SS.

Exploration versus Exploitation

Search methods are characterised by a balance between a wide *exploration* of the solution space, and an *exploitation* of regularities that the objective function is supposed to exhibit; exploitation usually results in concentrated efforts in parts of the solution space that are regarded as "promising" by the search strategy. Designing a successful search method for a particular problem means finding a strategy that – throughout the search – maintains a good balance between exploration and exploitation for the objective function under consideration. It involves making a number of decisions of the kind: motive versus constructive; how much work to perform in each step of

[3]In the present context, the notions of non-adaptive and adaptive refer to the *parameters* of the search strategy, and not to the nature of the search method itself. A method can be called non-adaptive in the first context, but adaptive in the context of Chapter 3: namely, that it is a selective process leading to adaptation.

the algorithm; individual or population-based; non-adaptive or adaptive etc. Decisions about other details need to be made later in the design process: movement operator, mutation rate, population size etc.

Problems, Search Methods and No Free Lunch

There are currently no established guidelines for designing good search methods for particular classes of problems. Also, there are no standardised procedures to assess the performance of search methods. A common approach is to provide results of search performance on some common benchmark problems, and compare them with the results for other types of methods.
Theoretical models predicting search performance for a particular method are scarce, and often presented for simplified variants that are not used in practice. There exists, however, a theoretical framework connecting search methods with search problems in general: the *No Free Lunch* (NFL) theorems (Macready and Wolpert, 1996; Wolpert and Macready, 1997). NFL states that, when averaged over *all possible* search problems, all search methods have the same performance, and no search method outperforms RGT (or similarly, systematic search). NFL does not imply that a search method cannot be superior on a specific class of search problems. However, if it is superior on some problems, then there must be other problems on which it performs worse than alternative methods.
It is still unclear what the consequences of NFL are. In practice, generic search methods like GA and hill-climbing have proven to be at least moderately effective on a wide variety of search problems. This discrepancy between theory and practice has led to the suggestion that real-world problems belong to a restricted class for which efficient generic search methods do exist (Sharpe, 1999). In response to NFL, Christensen and Oppacher (2001) investigate what makes objective functions efficiently searchable in practice. They observe that each metaheuristic attempts to exploit certain regularities of the objective function; most methods (e.g., local search and selective methods

with small variational mechanisms) exploit *self-similarity*: nearby points in the solution space have similar function values. One can proceed by defining a measure that reflects the regularities, exploited by a search method, over the space of all possible objective functions. Establishing whether a search method is suitable for a class of search problems then reduces to evaluating that measure for a number of problems of that class.[4]

4.3 Search and Optimisation with SDS

4.3.1 Problems

In §2.2p25, it was argued that standard SDS searches for a target pattern in a search space (*data search*) by performing optimisation of an objective function in a solution space (*solution search*). §2.2.4p31 then gave an example of an objective function consisting of the number of corresponding micro-feature pairs. This is a problem of the MAXCSP type. The test score for this example was defined as the number of corresponding micro-features (the value of the objective function) divided by the total number of micro-features; since the test procedure evaluates only 1 randomly chosen micro-feature pair, this measure gives the probability of agents being active after the test phase.

The double interpretation of SDS in the pattern matching context – as performing data *and* solution search – is explained by the fact that pattern matching consists of two related subproblems: the *correspondence* and *transformation recovery* problem.[5] The correspondence problem involves finding a correspondence between micro-features from target and search space; in the transformation recovery problem, the goal is to find transformation pa-

[4] In fact, a similar approach has been proposed many times before in more restricted contexts, e.g., by Jones and Forrest (1995).

[5] Beveridge (1993) termed these the *correspondence* and *pose recovery* problem in the context of 3D object recognition.

rameters for the optimal correspondence. The former gives rise to an interpretation of data search, the latter to an interpretation of solution search.

The work on the steady state resource allocation of standard SDS (Nasuto, 1999), which forms the basis for the convergence proof in noisy search spaces, was formulated in the context of string matching, a form of MAXCSP. However, the results are in principle valid for *all* MAXCSP, since the results of Nasuto (1999) are dependent on the properties of the test score, and not on the properties of the objective function or the origin of the problem itself. This claim will be validated in Chapter 6.

Figure 2.6 (p35) demonstrates that, when a single solution fulfilling all constraints is present in the solution space, all agents eventually converge on it. This indicates that SDS also solves CSP (as a special case of MAXCSP). However, standard SDS cannot prove that an assignment fulfilling all constraints does not exist. The algorithm is said to be *incomplete*. Nasuto and Bishop (1999) proved that the probability of not solving a soluble problem converges to 0 as the run-time approaches infinity. This property is sometimes called *Probabilistic Approximate Completeness* (PAC) (Hoos, 1999).[6]

SDS performs optimisation without calculating values of the objective function explicitly. Instead, it performs repeated *partial evaluations* of the objective function. This raises a fundamental question: what type of optimisation problems – besides MAXCSP – can be *in principle* solved using only partial evaluations? Intrinsically linked to this is the question: what type of solution spaces and objective functions is partial evaluation especially suited for? Both these questions will be treated in Chapter 6. An example of how an objective function other than MAXCSP can be optimised solely through partial evaluations is presented in Appendix A.4.

[6]This is not directly related to the concept *Probably Approximately Correct Learning* (PAC learning) in the context of Machine Learning.

4.3.2 Methods

Standard SDS is a motive search method, i.e., the agents in the population move through the solution space under control of a search strategy. Focused variants of SDS (Beattie, 2000; Hurley and Whitaker, 2002) can be regarded as a mixture of motive and constructive: in these hierarchical searches, agents start with testing approximate solutions in reasonably small solution spaces; as the search proceeds, these solutions are refined until they have the complexity of the desired solutions. The search process can thus be envisaged as constructing ever more complex solutions, while simultaneously moving around in the solution spaces defined by these approximate solutions.[7]

For standard SDS the amount of work performed by a single agent in a single iteration is very small: agents perform only partial evaluations of the objective function. No single agent at any moment during the search has access to a full function value. It is this property that most distinguishes SDS: many other methods use complete function evaluations at each point in the solution space. Local search methods that perform only partial evaluations to update the value of the objective function still have access to full function values during the entire search process. Fitness-based methods like GA allow in theory for partial evaluations, but this option has been rarely used.
Partial evaluation is the core element of the operation of SDS: it is the source of its efficiency for certain problems, its "justification" as a separate metaheuristic. Given the conclusions of NFL, it also has to be the source of its limitations: certain objective functions cannot be easily optimised using only partial evaluations; this issue will be addressed in Chapter 6. Furthermore:

- SDS can be classified as a population-based method: its operation depends on the interaction of many individuals. Since single agents per-

[7] 'Approximate solution' does not equal 'partial solution'. A solution can be approximate because it contains a subset of all the variables x_i, but also because it operates on a subset of the domain values D_i for each variable x_i.

form only small amounts of work, population sizes are typically much larger than for MA or even GA (usually in the range of 100 to 1000).

- SDS is a randomised method, although it is possible to de-randomise the procedure (see Appendix A.2).

- Standard SDS takes an intermediary position between guided and unguided search methods: in the beginning of the search, when most or all agents are inactive and little diffusion of information takes place, standard SDS behaves very much like parallel RGT. Later, when good solutions are discovered, agents guide each other towards those solutions. Standard SDS does not attempt to exploit any self-similarity in the objective function; the addition of a small-variational mechanism to the diffusion phase, as in Appendix A.3, does weakly exploit self-similarity and increases the degree of guidance.

- Standard SDS is certainly not directive: since agents have no access to information about the form of the objective function, they cannot use hill-climbing strategies.[8] It is a selective search method, generating solutions first and then retaining the promising ones. In one iteration of SDS, candidate solutions are tested only a fraction of the amount they are tested in other selective methods such as GA. SDS can thus be regarded as operating on a different timescale from GA.

- SDS maintains an implicit history of the search in the clusters of agents. Since standard SDS allocates most of its resources to the single best solution discovered so far, the amount of information memorised in this case is very limited. Other variants, like context-sensitive SDS (see Appendix A.1), can support more than one cluster of agents, and therefore maintain a larger history of the search process.

[8] SDS can easily be combined with a hill-climbing method, as in (Grech-Cini, 1995): standard SDS is used to quickly find a reasonable starting solution; this solution is then improved using a simplex method, essentially a form of hill-climbing.

- Standard SDS is non-adaptive, since all the search parameters are fixed; variants like focused SDS (Beattie, 2000; Hurley and Whitaker, 2002) are adaptive, since the testing phase is altered as the search proceeds.

- Standard SDS is a convergent method: agents cluster together in points in the solution space. Convergence is an essential aspect: because single agents have no ability to assess and compare the quality of solutions, it is the group behaviour that indicates where good solutions are.

The meaning of 'exploitation' in the context of SDS is somewhat special: for SDS variants without small-variational mechanisms, 'exploitation' refers to the clustering of agents in *single points* in the solution space, whereas for other metaheuristics it refers to concentrated efforts in *regions* of the solution space. For SDS variants that exploit self-similarity in the objective function, the meaning of exploitation shifts more towards its usual meaning: instead of considering clusters of agents in single points in the solution space, all agents in *nearby* points could be considered as belonging to the same cluster.
In any case, the conflicting demands of exploration and exploitation are even more important for SDS than for other methods: since individual agents cannot compare the quality of solutions, exploitation (clustering) is needed to indicate that good solutions have been discovered. However, as observed in §2.2.4p36, too much exploitation conflicts with continuing exploration, and can possibly hinder system performance. When applying SDS to a particular problem, the prime task is thus to find a variant that combines a maximum of continuing exploration with a level of exploitation that is sufficient to indicate the presence of good solutions. Due to its greedy nature, standard SDS is not necessarily an optimal candidate for many problems.

The conclusions of NFL are of course valid for SDS: if standard SDS is more efficient than Template Matching (a form of systematic search) on some problems (§2.2.4p36), then it must perform worse than systematic search on other problems. To assess what kinds of problems SDS is suitable for, the

approach as proposed in (Christensen and Oppacher, 2001) can be adopted: define a measure that reflects the regularities the search method is trying to exploit, and evaluate that measure for typical examples of the problem class. A further treatment of this approach is deferred to Chapter 6.

4.4 Summary and Conclusion

4.4.1 Optimising Search Methods

NFL demonstrates that matching specific problems with suitable metaheuristics is the prime task of researchers interested in problem solving. This actually has been happening on an informal "trial-and-error" basis for the last 50 years: search methods were proposed, evaluated on benchmark problems and compared with other methods. The most successful methods were adopted by other researchers; variations and hybridisations were proposed, evaluated and eventually retained or rejected. The entire process is an example of how science itself is an adaptive process operating through the mechanisms of V+SR (§3.5p64). The work of Christensen and Oppacher (2001) is a formalisation of this process: specifying a measure over the space of all objective functions, and then searching for regions where this measure predicts good performance for a specific search method, is an optimisation problem in itself.

4.4.2 Building Blocks

The previous sections have implicitly advocated the approach that search methods can be designed by combining "building blocks", aspects of search methods that attempt to speed up the search, often by exploiting regularities of objective functions: memory, adaptiveness, parallelism, interaction, randomness, search operators exploiting various kinds of self-similarity etc.

The main contribution of SDS to this collection is a new building block: the principle of partial evaluation.

A building-block approach to search methods blurs categorical boundaries. One can easily imagine individual SDS agents with tabu lists; phases of SDS intertwined with hill-climbing; GA, MA or PSO relying on partial evaluations etc. It is this approach that leads to the structure of Chapter 5: rather than attempting to formally categorise SDS variants according to their properties, a number of "dimensions of choice" are presented that will allow a more efficient matching of search problem to search method in the future.

4.4.3 Conclusion

This chapter has translated the interpretation of SDS as a problem-solving selective process into the language of AI and computer science: SDS solves problems by searching through a solution space. The text focused on problems that are explicitly formulated as search problems; however, many problems that are formulated in different contexts – such as machine learning, control, resource allocation or decision making – are essentially solved through search. SDS – or its most fundamental building block: partial evaluation – can also play a role in all of these derived problems, even if this subject has not been discussed. Search *methods* were presented with a similar rationale as the discussion of selective processes in Chapter 3: to serve as inspiration for the design of SDS variants.

Referring back to the roadmap of questions, this chapter has mainly provided an answer to question 2 (p15): "what does SDS do?" Meta-level questions C and D (p18) have also had an important influence on the content of the chapter: inspiration from other search methods can be seen as a partial answer to question C; SDS's main contribution to the field of metaheuristics – the principle of partial evaluation – forms an answer to question D.

Chapter 5

Properties of SDS

Chapter 4 advocated a building block approach to the design of metaheuristics: a specific search method can be constructed from a combination of ideas from other methods. This approach makes a strict classification of search methods impossible: for instance, an SDS variant that performs cross-over of hypotheses during the diffusion phase can be regarded as either SDS or as a GA using partial evaluation and tournament selection. Nonetheless, for many search methods like GA, MA, ACO or SDS itself, it is possible to define "archetypical" examples. These examples are built around properties that are considered essential to the search method. Flexible category definitions can then be based on these essential properties.

Chapter 1 defined SDS as an algorithmic process (§1.2.1p13), and stressed the contradistinction between the algorithmic side – the mechanistic procedure detailing the operation of SDS agents – and the process side, more specifically the statistical regularities on the population level. Consequently, when discussing properties of SDS, the same distinction should be made between algorithmic properties and process properties. Section 5.1 discusses the essential algorithmic properties of SDS and the process that ensues from them. Section 5.2 then lists alternatives that can be combined to instantiate specific SDS variants, and evaluates how they affect the process dynamics.

5.1 Essential SDS

5.1.1 Algorithmic Properties

The previous chapters have described SDS as a *population-based* generate-and-test method trying to reduce the computational cost of searching by performing *partial evaluation* of hypotheses only. In this section, SDS is defined in more detail by listing *essential* algorithmic properties:

Definition 5.1. *An* SDS *algorithm consists of a population of independently operating agents. Each agent has associated with it:*

- *A* ***hypothesis*** *about the optimal solution to the problem.*
- *A procedure for partial evaluation of the hypothesis, the* ***test*** *procedure.*
- *A state variable, called* ***activity****, whose value is based on the outcome of the test procedure.*
- *A procedure specifying whether to retain or alter the hypothesis, the* ***diffusion*** *procedure. It includes acts of direct communication with other agents during which information about hypothesis and activity parameters is exchanged.*

An SDS *algorithm should also contain the following specifications (associated to either individual agents or the population as a whole):*

- *A strategy for* ***initialising*** *hypothesis and activity parameters.*
- *Usually, but not necessarily, a criterion for* ***termination****.*
- *A strategy for* ***extracting*** *a solution from a population of hypotheses.*
- *A specification of* ***structural properties*** *of the population.*

*Furthermore, it is assumed that agents react **rationally** (on average) to information obtained during the test procedure and from other agents.*

The rationality assumption excludes variants where activity values do not correlate with the outcome of the test procedure, and variants where agents react in nonsensical ways to information obtained from other agents. It is unlikely that such variants can produce behaviour useful for search purposes.

5.1.2 Process Properties

When some or all of the algorithmic steps of SDS are randomised, the spatial and temporal allocation of agents in the solution space[1] is governed by a stochastic process sampling from a number of probability distributions. For example, the test procedure can be regarded as sampling random numbers from probability distributions whose means are given by the test score. Similarly, the algorithmic steps of communication and information-sharing in the diffusion procedure give rise to a sampling process whose effects can be translated into a weak stochastic coupling between agents.

For certain combinations of sampling processes (or combinations of parameter values of the processes), the allocation pattern of agents in the solution space exhibits a strong *nonlinear* effect: a limited number of points or regions in the solution space attracts a disproportionately large number of agents, in the form of relatively stable clusters of agents. For other combinations, the nonlinear effect is virtually absent, and stable clusters do not form. The change in behaviour from nonlinear to linear occurs abruptly as parameters of the sampling processes are varied. For example, the allocation pattern of standard SDS changes abruptly from nonlinear to almost linear when the test score of the best solution approaches the critical response (§2.3.2p39).[2]

[1] Or, alternatively, the temporal distribution of hypotheses over the agents.

[2] Although the change is abrupt, it is not discontinuous: in a small region around the critical response, the stability of clusters changes rapidly. (see also Nasuto, 1999)

Given the assumption of (on average) rational agent behaviour, the points or regions that attract clusters are expected to contain the best solutions. It is this nonlinear aspect of the SDS allocation process that is essential to its success as an optimisation method: good solutions can be extracted from stable clusters of agents in the solution space. This property of the SDS process is *not* an algorithmic property: the algorithmic description of SDS does not include or predict the formation of clusters. Conversely, algorithmic properties are not necessarily process properties: for instance, for the process side of SDS it is quite irrelevant that the test score is the averaged result of randomly-selected partial evaluations; the uncertainty expressed by this distribution could equally well be the result of uncertainty in the problem definition or some form of measurement noise. Hence, partial evaluation is not an essential *process* property; it is only an essential *algorithmic* property.

The spatial and temporal allocation of agents reflects the dynamical balance between exploration and exploitation (§4.3.2p104). Too much exploration without exploitation (clustering) means that the search is unsuccessful; too much exploitation means that the search settles quickly on possibly suboptimal solutions.[3] For a given problem, the balance between exploration and exploitation can be manipulated by selecting algorithmic properties from a number of alternatives, some of which are discussed in the next section.

5.2 Dimensions of Choice

The current section concentrates on choices that can be made in the general framework provided by Definition 5.1 (p108). It does not attempt to discuss combinations of SDS with other search methods, such as SDS agents with tabu lists or combinations of SDS with hill-climbing. This is mainly because it wants to emphasise on the many potential alternatives within the setting of *pure* SDS algorithms, and how alternatives influence process behaviour.

[3]This does not necessarily imply that it gets permanently stuck in these solutions.

5.2.1 Activity

Activity is based on the outcome of the test procedure. It is usually a scalar, but – if needed – could also be a vector: for instance, instead of storing only the present activity level, the activity level from the previous iteration could also be retained, resulting in a 2-tuple activity.

The original formulations of SDS used binary activity levels. Alternatively, activity levels could be chosen to span the interval $[0, 1]$. Activity can then be interpreted as a *probability*, the *belief* of an agent in the quality of its hypothesis, given evidence from the test procedure; or it could be interpreted in the language of fuzzy-set theory and possibility theory. More generally, activity levels can also be integer-valued or real-valued numbers.

The reason for using more complex types of activity levels is that, for optimisation problems other than MAXCSP, the output of the test procedure contains more information than just 0 or 1. Compressing that information to a 0 or 1 value would waste possibly valuable information. Appendix A.4.2 (p154) gives an example of this idea.

5.2.2 Diffusion

The test procedure (Chapter 6) and the structural properties of the population (§5.2.6p118) both have a decisive influence on the behaviour of the SDS allocation process. Since the former is largely determined by the objective function to be optimised, and the latter constrained by aspects of the physical implementation, it is in the diffusion procedure that the balance between exploration and exploitation can be most actively manipulated.

Agents communicate directly with other agents during the diffusion phase. They compare activity and hypotheses parameters, and then decide whether to retain or alter their own hypothesis, possibly replacing it with the hypothesis of another agent. The steps of communicating, comparing and altering

can be combined in many different ways. The influence of these combinations on the allocation process can best be discussed from the perspective of the *replicators* – the hypotheses – in the framework of V+SR (§3.6.2p70).

Selection Mechanisms

In SDS, agents choose to reject, retain or replicate a hypothesis, based on information received from one or at most a few other agents.[4] A few potential selection strategies exist: these are based on the comparison of activity levels, hypotheses parameters, or both. In the case where agent A contacts one agent only – agent B – the following combinations exist for activity-based selection:

1. The activity levels of agent A and B are both low. Because it is unlikely that either of the two agents has a good hypothesis, agent A should reject its own hypothesis and not accept the hypothesis of agent B.

2. Agent A has a low activity level, but agent B has a high activity level. Because it is likely that agent B has a better hypothesis, agent A should reject its own hypothesis and accept the hypothesis of agent B instead.

3. Agent A has a high activity level, while agent B has a low activity level. Since the hypothesis of agent B is likely to be worse, it would be irrational of agent A to replace its hypothesis with agent B's hypothesis. The best strategy for agent A is to simply retain its own hypothesis.

4. Both agents A and B have high activity levels, and are therefore both likely to maintain a good hypothesis. There are two potential strategies here: agent A decides to retain its hypothesis; or agent A abandons its own hypothesis to resume exploration (context-free SDS).

[4] This is an important distinction with centralised selection mechanisms in EC (4.2.1): fitness ranking (p92), roulette wheel and elitist selection strategies (p91). The selection mechanisms of SDS can be seen as an extreme form of tournament selection (p92).

The first situation is biased towards exploration; the second scenario results in exploitation; the third does not affect levels of exploration and exploitation; the fourth either maintains levels, or results in exploration.

For binary activity levels, these four situations unambiguously cover all possible scenarios. For other types of activity levels, there is a certain freedom in deciding what constitutes high and low levels. Alternatively, selection in this case can be based on the *difference* of two activity levels, rather than on a high-low comparison. An example thereof is given in Appendix A.4.2 (p154). Selection can also be made probabilistic, dependent on the difference between the two activity levels.

There is only one rational hypothesis-based selection strategy: when agent A and agent B have similar hypotheses, agent A can choose to abandon its own hypothesis, based on the idea that agent B is already advertising it. This is the context-sensitive approach, resulting in an increased exploration.

Replication Mechanisms

A hypothesis is replicated when one agent copies it from another agent. The hypothesis of a single agent may be copied many times by different agents, but usually one act of copying results in one new copy of the hypothesis. Alternatively, one act of copying could be chosen to result in multiple copies, enhancing exploitation and thus the stability of clusters. This mechanism results in a growing population; a context-sensitive selection mechanism can be used to stabilise overall population size, by removing the excess active agents from the population.

Variational Mechanisms

The original formulations of SDS generate variation that is uncorrelated to the hypotheses already present in the population: new hypotheses are chosen randomly from all admissible ones. In combination with the probabilistic

outcome of the test procedure, this mechanism ensures that SDS continuously explores the entire solution space and can never become permanently trapped in a suboptimal solution.
Knowledge about the objective function may be incorporated to *bias* variation towards exploitation of certain areas of the solution space. For instance, many objective functions based on real-world problems exhibit self-similarity (§4.2.2p100). This means that a good location in the solution space has a non-negligible probability of being close to even better locations. It is sensible to bias the generation of variation towards the detection of these nearby better locations by employing small-variational mechanisms. However, if uncorrelated variational mechanisms are completely replaced by strongly biased ones, then the ability of SDS to escape local optima may become impaired.

In the original formulation of SDS, new variation is generated after old hypotheses are rejected. However, variational mechanisms can also be applied in the replication steps of the diffusion procedure: for instance, if a small random offset is added while copying a hypothesis – as in Appendix A.3 (p146) – then resources of large clusters are diverted to neighbouring locations in the solution space; exploitation of one solution is exchanged for exploitation of a few neighbouring solutions. Another option for generating variation during replication is the recombination operators known from ES and SS: for example, a hypothesis can be copied only *partially*; or one hypothesis can be replaced by a linear combination of hypotheses. Finally, variation may also be generated when hypotheses are simply retained: an agent remaining in the same location for some time may decide to add a small random offset to its hypothesis parameters. This mechanism should be implemented with care, since it could make sufficient exploitation impossible.

5.2.3 Initialisation

In general, agents need to be initialised with a starting hypothesis. If prior knowledge about the problem is available, then this knowledge can be used to bias the assignment process towards certain regions or points of the solution space. In the absence of prior knowledge, hypotheses can be sampled uniformly at random from the solution space. Alternatively, hypotheses could be assigned using a *low-discrepancy sequence*, a pseudo-random sequence of points that fills the sampled space more regularly than uniform random sampling; or hypotheses could be assigned evenly and deterministically.

The primary reason for spacing the initial hypotheses more evenly is that for problems with self-similarity, the optimal value may lay on a peak of non-negligible width; the probability of missing the peak altogether in the initial assignment is lower with evenly spaced sequences than with uniform random sequences. In combination with a small-variational mechanism, inducing hill-climbing behaviour, this approach can improve algorithmic performance.

5.2.4 Termination

There are several criteria for terminating the execution of an SDS algorithm:

1. No stopping criterion: the algorithm continues to run until interrupted by the user or a different process. This is the preferred setting for *dynamically changing* problems, where SDS is used to track the maximum or minimum of a dynamically changing objective function.

2. Time-based criteria: the algorithm is run for a fixed number of iterations, or until a fixed amount of real computational time is elapsed.

3. Activity-based criteria: the process is terminated if the overall activity exceeds a certain level; this level can be predefined by the user or calculated from the overall activity level during the first iterations of the

algorithm (the activity level is then the result of the background response). Alternatively, the process can be terminated if overall activity of the population stabilises for a number of iterations after a sudden increase; or a combination of these two criteria can be used.

4. Cluster-based criteria: execution of the algorithm can be terminated when the formation of stable clusters has been observed.

Activity-based and cluster-based stopping criteria introduce a serial computational bottleneck into an otherwise completely distributed algorithm. This poses a problem for efficient distributed implementation of SDS; for large numbers of agents, it also affects computational efficiency in general.
The problem can be alleviated by implementing the termination criterion as a random sampling process: for example, a cluster-based termination procedure may wish to monitor the hypotheses of a small proportion of the population until it encounters the same hypothesis more than once. This forms partial evidence that a cluster has been formed. The procedure can then choose to increase the size of the sample taken from the population.
Such a random sampling procedure could eventually be translated into an SDS algorithm itself: a small population of stopping agents compares the hypotheses of pairs of the original searching agents; the activity state of these stopping agents should be based on the similarity of the two hypotheses. A cluster-based criterion for a large population can thus be translated into an activity-based criterion for a much smaller population.[5]
The termination procedure does not need to operate in the same time frame as the SDS agents: for instance, it is possible to execute the termination procedure only every x iterations, or every x seconds of computational time.

[5]In some situations these two populations can even be merged: for instance, active context-sensitive agents already compare hypotheses. A small proportion of agents can then simply be designated as 'stopping agents', and be given an extra activity state.

5.2.5 Extraction

After termination of an SDS algorithm, and also *during* tracking of the optimum of a dynamically changing objective function, a single solution needs to be extracted from the population of agent hypotheses. For greedy variants such as standard SDS, it is simply the most common hypothesis that is proposed as the optimal solution. For variants like context-sensitive SDS that can support multiple clusters of similar size (see Appendix A.1 (p142)), it can be necessary to do a few complete function evaluations to extract the optimal solution from the cluster information. When small-variational mechanisms – such as the strategy of errors in Appendix A.3 (p146) – are employed, clusters often contain a few neighbouring hypotheses. If more than one of such clusters is formed, it might be required to perform some standard clustering analysis on the population to determine the relative size of clusters.

The problem of extracting a solution from a population of hypotheses is an instance of the generic problem of *learning from secondary data* (Kearns and Seung, 1995): a learning algorithm must combine a collection of potentially poor hypotheses into a final hypothesis. In the more restricted *population learning* model for function approximation, Kearns and Seung (1995) assume that the hypotheses are statistically independent, an assumption that certainly does not apply to SDS. Nevertheless, the theoretical analysis of population learning leads to some interesting conclusions: firstly, that there exist combinations of problems and population learners for which the final hypothesis is expected to be better than any of the original hypotheses; secondly, that the class of general population learners is strictly more powerful than the class of *population predictors*, population learners based on voting schemes (Nakamura, Takeuchi and Abe, 1998).[6] As SDS variants are applied to new types of problems, these conclusions become potentially relevant for the study of SDS.

[6]Taking the most common hypothesis can be regarded as a voting scheme.

5.2.6 Structural Properties of the Population

Guidelines for Population Size

Population size is determined by the conflicting demands of computational efficiency versus stability of clusters: the temporal and/or spatial computational requirements for simulating the population increase with population size; from this perspective the number of agents should be kept as low as possible. However, from the perspective of cluster stability, a reasonable population size is required to guarantee sufficient stability. For standard SDS, a population size of a few hundred probably provides sufficient stability (and this irrespective of the size of the solution space). For variants such as context-sensitive SDS, population size often needs to be higher.

The number of agents used in SDS is often one or even two orders of magnitude higher than in methods like GA, MA or PSO. However, in many cases the computational cost of an SDS iteration is still lower than for these methods, due to the partial evaluation of the objective function and the cheap selection mechanism following from direct one-to-one communication.

Temporal Mode of Operation

SDS agents can operate synchronously or asynchronously. In synchronous mode, all agents perform test and diffusion procedures as dictated by a central clock; this is the operational mode of the original formulations of SDS. Additionally, there exist two different modes of *asynchronous* operation:

1. In each iteration, one agent is chosen at random to perform a specified sequence of test and diffusion procedures. Its de-randomised counterpart – the *systematic sequential* mode – updates all agents one for one in a deterministic sequence. These two modes of operation are – like

the synchronous mode – suitable for software implementation on serial computers. For many variants, asynchronous process behaviour is the same as for the synchronous mode, except for the time scale; however, asynchronous process behaviour is considerably easier to analyse mathematically than the synchronous process (De Meyer, 2000).

2. Each agent operates in its own time. This mode of operation is specifically suited towards distributed implementations, with the purpose of implementation on parallel hardware. The process properties are likely to be the same as for other modes of operation, except for the timescale.

Communication Structure of the Population

In the original formulations of SDS, agents can communicate with all other agents in the population. This is the best and most simple communication structure for software implementations on single serial machines. However, for spatially distributed types of implementation, this means that agents either need to be connected through a shared communication channel (a *bus*), or fully connected by direct one-to-one communication channels.

When neither of these possibilities is acceptable, the following alternatives can be considered, dependent on the type of physical implementation:

1. Limit the number of agents that each agent can communicate with. This can be achieved by arranging agents on a regular lattice and allowing communication with nearby neighbours only. Although this approach is ideally suited for implementation in parallel hardware, a potential problem is that, for increasing lattice sizes, clusters form more slowly and are less stable than for fully-connected variants. Empirical evidence suggests that for *small-world*[7] networks, process behaviour

[7]Small-world networks consist of a regular lattice of local connections with relatively few long-range connections added (Watts and Strogatz, 1998).

is much closer to the behaviour of fully-connected variants, even for a modest number of connections, providing the optimal trade-off between process behaviour and network complexity (De Meyer et al., 2002) – see also Appendix A.5 (p157).

2. Agents are divided into several subpopulations that are internally fully connected, but have no or relatively few external connections. This approach is particulary suitable for implementation on multi-processor machines or clusters of single-processor machines, limiting the amount of communication taking place between processors or machines.

Compositional Properties

Populations can consist of one type of agents, or of multiple types of agents. For instance, mixing standard and context-sensitive agents results in a population with an allocation pattern that is somewhere in between the allocation pattern for purely standard and purely context-sensitive populations.

5.2.7 Randomisation

Randomisation itself was not listed as an essential algorithmic property of SDS. This is because almost any of the algorithmic steps in the test and diffusion procedure can be either randomised or de-randomised. An example of a completely de-randomised version of standard SDS is given in Appendix A.2 (p143): all randomised operations are replaced by operations that guarantee a maximum amount of variation in actions undertaken by different agents.

The balance between exploration and exploitation can also be manipulated by further randomising some of the algorithmic steps: for instance, the *secret optimist* (Grech-Cini, 1995) only rejects a poor hypothesis with a certain probability $p < 1$, shifting the balance towards more exploitation. The *hermit* (ibid.) prefers generating new hypotheses rather than working

in a team, and only attempts to communicate with other agents a certain proportion of all iterations, shifting the balance towards more exploration.

Biasing the uniform random distributions used in the original SDS variants is another way of manipulating the allocation process: examples have already been given of bias in variational mechanisms in the diffusion and the initialisation procedure. A further example is the choice of the contacted agent: rather than choosing it uniformly at random, the probability can be biased by the activity levels of other agents, resulting in more exploitation.

5.2.8 Adaptivity and Self-Adaptivity

Many algorithmic properties of SDS can be made adaptive and even self-adaptive (§4.2.2p98). Some of the diffusion mechanisms are *implicitly* adaptive, i.e., their effect is dependent on the status of the search process: for instance, a context-sensitive mechanism has almost no effect as long as good solutions have not been discovered; only when clusters start forming does the probability increase that agents with the same hypothesis communicate.
However, it is also possible to introduce *explicit* adaptivity: for instance, focused SDS variants (Beattie, 2000; Hurley and Whitaker, 2002) dynamically alter aspects of the test procedure in response to information feedback from the search process. Parameters of the diffusion procedure can also be made adaptive by modifying them during the search and self-adaptive by including them in the search itself: for instance, a secret optimist can decrease its probability of rejecting a poor hypothesis when there is too much exploration.

The main purpose of making the diffusion procedure adaptive is to allow the search process to adjust itself automatically to the properties of the test score: the ideal problem for SDS is one where the test scores of all unacceptable solutions fall below the critical response, and the test scores of acceptable solutions are higher than the critical response. The critical response can be changed by manipulating the diffusion procedure: for instance, for a popula-

tion of secret optimists with binary activity retaining a failed hypothesis on average i iterations, the critical response is given by (Grech-Cini, 1995):

$$r_c = \frac{1 + ir_b}{2 + i - r_b} \tag{5.1}$$

For $i = 0$, this equation reduces to the equation for standard SDS (p39). Making i modifiable gives control over the critical response, resulting in a variant that still converges when standard SDS would fail, and for which a population of secret optimists with fixed i might converge too quickly.

5.3 Conclusion

By discussing properties of SDS, this chapter has answered several of the questions of the roadmap. Section 5.1 answered questions 4 and 6 (p16): the most distinguishing essential algorithmic properties are partial evaluation of hypotheses, and direct one-to-one communication between agents; the latter leads to a computationally-cheap, distributed selection mechanism. On the process side, it is the explicit reliance on the formation of clusters that distinguishes SDS from other search methods.

Section 5.2 partially answered question 5 (p16). Rather than giving a taxonomy of SDS variants, it proposed how algorithmic properties can be chosen from a number of alternatives. When making alterations to the original algorithmic formulation, it should be carefully considered what is expected from them: some are justified in light of an efficient physical realisation of the algorithm; other modifications are mainly used to manipulate the balance between exploration and exploitation, so that stable clusters can form on the best locations in the solution space and not on unwanted solutions.

The chapter did not consider potential modifications to the test procedure, since this issue is intrinsically linked to the question of what type of objective functions can actually be optimised by partial evaluation only. This question will receive due attention in the following chapter.

Chapter 6

Applicability

Chapter 5 defined the essential properties of SDS, and discussed potential variations within this framework. The test procedure was defined as a "procedure for partial evaluation", but details of its operation were not discussed. This issue will be treated in the current chapter, as it is very much related to the question of what can be optimised using partial evaluations only.

What can be achieved in principle with a method like SDS, and how well it works in practice are two very different questions. The NFL theorems, introduced in Chapter 4, state that for every class of problems where a particular method outperforms random search, there must be other classes where the reverse is true. Establishing whether a particular class of problems is well-suited for optimisation with SDS is therefore a question of great importance.

Section 6.1 addresses the first of these two questions. It demonstrates how in principle all summation functions can be optimised using standard SDS, and gives some ideas about objective functions that are not summations. Section 6.2 then concentrates on the question of practical usability: it proposes a number of measures that can be used to judge the suitability of a problem class for optimisation with SDS, and identifies a number of problem scenarios for which SDS should or should not be attempted.

6.1 Decomposable Functions

Partial evaluation presupposes, first and foremost, that the objective function is *decomposable* into components that can be evaluated independently. For many optimisation problems, the objective function can be expressed as a summation, which is naturally decomposable into its individual terms:

$$f(x) = \sum_{i=1}^{n} f_i(x) \tag{6.1}$$

All summation functions can in principle be optimised using standard SDS by evaluating individual terms of the summation in the test procedure; this is explained in more detail in Section 6.1.1. For objective functions that are not summations, it remains an open question what can be achieved, although some general ideas are presented in Section 6.1.2.

6.1.1 Optimising Summations

What exactly does standard SDS optimise?

The stochastic models of standard SDS in (Nasuto, 1999) were formulated in the context of string matching: dependent on the presence or absence of a target character in the search space, the output of the test procedure is either 1 or 0, resulting in either an active or inactive agent. These stochastic models are not based on specific details of string matching, but only on probabilities of producing active and inactive agents. Their validity is therefore not limited to the string matching domain. They demonstrate that – irrespective of any objective function – the largest cluster of standard SDS agents eventually forms at the location in the solution space that has the highest probability of producing active agents, i.e., at the maximum of the test score $t(\cdot)$ (§4.1.2p87).[1] Hence, standard SDS can be regarded as finding:

[1] Additional conditions, such as the critical response (§2.3.2p39), are not considered at this point. These issues will be addressed further on.

$$\max_{x \in \mathcal{S}} t(x) = \max_{x \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} t_i(x) \tag{6.2}$$

with $t_i(\cdot)$ deterministic $\{0, 1\}$-valued test functions, and n the total number of different test functions. For example, in the *Queen of Hearts* problem (§2.2.4p28), there are two test functions: value$(\cdot)$ and type$(\cdot)$; for the *Matching on Mars* example (§2.2.4p31), there are 800 test functions $t_i(\cdot)$, evaluating the presence or absence of each of the 800 micro-features. If the $t_i(\cdot)$ functions are themselves probabilistic, then standard SDS converges to:

$$\max_{x \in \mathcal{S}} t(x) = \max_{x \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}(t_i(x)) \tag{6.3}$$

with $\mathrm{E}(\cdot)$ the expected value; $\mathrm{E}(t_i(x))$ is the proportion of times that $t_i(x)$ has 1 as outcome. Equations 6.2 and 6.3 assume that all $t_i(\cdot)$ are equally likely to be evaluated. When this is not the case, standard SDS locates the maximum:

$$\max_{x \in \mathcal{S}} t(x) = \max_{x \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} w_i \, \mathrm{E}(t_i(x)) \tag{6.4}$$

where w_i is the relative frequency of evaluation of $t_i(\cdot)$ ($\sum_i w_i = 1$).

Strictly Order-Preserving Mappings for Summation Functions

Standard SDS converges to the maximum of the test score; if the process of mapping objective function values to test score values (taking place in the test procedure) does not change the locations of the optima, then SDS can also be said to optimise the objective function itself. A sufficient condition to ensure this is that the operation $f(\cdot) \to t(\cdot)$ is *strictly order-preserving*:

$$\forall\, x_1, x_2 \in \mathcal{S} \qquad \begin{cases} f(x_1) = f(x_2) \Rightarrow t(x_1) = t(x_2) \\ f(x_1) < f(x_2) \Rightarrow t(x_1) < t(x_2) \end{cases}$$

or

$$\forall\, x_1, x_2 \in \mathcal{S} \qquad \begin{cases} f(x_1) = f(x_2) \Rightarrow t(x_1) = t(x_2) \\ f(x_1) < f(x_2) \Rightarrow t(x_1) > t(x_2) \end{cases}$$

For several types of summation functions, individual terms of the summation $f_i(\cdot)$ can be straightforwardly mapped to test functions $t_i(\cdot)$ while strictly preserving order. For instance, for objective functions of the MAXCSP type:

$$\max_{x \in \mathcal{S}} \sum_{i=1}^{n} c_i(x) \qquad c_i : \mathcal{S} \rightarrow \{0, 1\} \tag{6.5}$$

the identity mapping $c_i(\cdot) \rightarrow t_i(\cdot)$ results in the case described by Equation 6.2. When not all constraints are equally important, the optimisation problem – called *weighted* MAXCSP – has an objective function of the form:

$$\max_{x \in \mathcal{S}} \sum_{i=1}^{n} \tilde{w}_i c_i(x) \qquad c_i : \mathcal{S} \rightarrow \{0, 1\} \tag{6.6}$$

Using identity mapping $c_i(\cdot) \rightarrow t_i(\cdot)$, after normalisation of importance weights $\tilde{w}_i$:

$$w_i = \frac{\tilde{w}_i}{\sum_i \tilde{w}_i} \tag{6.7}$$

the optimisation of Equation 6.6 is reduced to the case of Equation 6.4.

For maximisation of summation functions with [0,1]-valued components:

$$\sum_{i=1}^{n} u_i(x) \qquad u_i : \mathcal{S} \rightarrow [0, 1] \tag{6.8}$$

the component functions $u_i(\cdot)$ can be mapped to probabilistic functions $t_i(\cdot)$, with $\mathrm{E}(t_i(x)) = u_i(x),\ \forall x \in \mathcal{S}$, transforming Equation 6.8 in Equation 6.3. For *minimisation* of the same type of summation functions, the mapping $u_i(x) \rightarrow t_i(x)$ should ensure that $\mathrm{E}(t_i(x)) = 1 - u_i(x),\ \forall x \in \mathcal{S}$.

Summation functions in which all components have the same range $\mathcal{Y}$:

$$\sum_{i=1}^{n} f_i(x) \qquad f_i : \mathcal{S} \rightarrow \mathcal{Y},\ \mathcal{Y} \subset \mathbb{R} \tag{6.9}$$

can be transformed into summation functions with [0,1]-valued components, by mapping component functions $f_i(\cdot)$ to functions $u_i(\cdot)$:

$$u_i(x) = \frac{f_i(x) - \min \mathcal{Y}}{\max \mathcal{Y} - \min \mathcal{Y}} \qquad u_i : \mathcal{S} \to [0, 1] \tag{6.10}$$

The $u_i(\cdot)$ can then be mapped to probabilistic $t_i(\cdot)$ as for Equation 6.8. One often occurring variant of this type is *error minimisation*; an example of standard SDS on such a problem can be found in Appendix A.4.1 (p151).

For general summation functions consisting of components with different but known ranges $\mathcal{Y}_i$, functions $f_i(\cdot)$ can be mapped to the interval [0,1] following Equation 6.10. However, since the different magnitudes of the ranges can contribute differently to the overall sum, the frequency of evaluation of a component needs to be weighted by its relative contribution:

$$w_i = \frac{\max \mathcal{Y}_i - \min \mathcal{Y}_i}{\sum_j^n (\max \mathcal{Y}_j - \min \mathcal{Y}_j)} \tag{6.11}$$

With this scaling and weighting of component functions, the mapping of $f_i(\cdot) \to u_i(\cdot)$ is guaranteed to be strictly order-preserving, because:[2]

$$f_i(x) = C w_i u_i(x) + \min \mathcal{Y}_i, \quad C = \sum_j^n (\max \mathcal{Y}_j - \min \mathcal{Y}_j) \tag{6.12}$$

The optimisation of general summation functions with known component ranges is then transformed – via a mapping of $u_i(\cdot)$ to probabilistic functions $t_i(\cdot)$ – into Equation 6.4. An example of this type of problem is TSP (§4.2.1p93), where for each node i its closest and its most distant neighbour give $(\min \mathcal{Y}_i)$ and $(\max \mathcal{Y}_i)$. When the bounds of the component functions cannot be obtained exactly, a conservative estimate can often still be made.

Approximate Mappings

The requirement of a strictly order-preserving mapping between component functions $f_i(\cdot)$ and test functions $t_i(\cdot)$ can sometimes be relaxed, since the

[2] Multiplication with constants ($\neq 0$) and adding constants are strictly order-preserving.

mapping essentially only needs to preserve the locations of the optima, and not the complete topology of the objective function. Hence, when a priori knowledge about the likely value of the optimum is available, it can be incorporated into the mapping: for instance, for error minimisation problems, it can often be assumed that the optimal value v_i for each component function $f_i(\cdot)$ is likely to be $0 \leq v_i \leq c \ll \max \mathcal{Y}$. Values within the restricted interval $[0, c]$ can then be mapped to a test score of $(0, 1]$, whereas values of $[c, \max \mathcal{Y}]$ can be mapped to 0. An example of this principle is given in Appendix A.4.1 (p151), where – although $\mathcal{Y} = [0, 255]$ – for $f_i(x) > 50 \Rightarrow t_i(x) = 0$.

The Critical Response

To ensure the formation of stable clusters on the maxima of Equations 6.2, 6.3 or 6.4, the test score needs to fulfill an additional condition: if the signal-to-noise ratio of the maximum test score to the test scores of the entire solution space is too low, then the standard SDS process is in its linear mode of operation (§5.1.2p109), and only small, unstable clusters can form.

Chapter 5 proposed various modifications to the diffusion procedure that alter the value of the critical response. However, modifications can also be made to the test procedure; these leave the critical response unchanged, but alter the values of the test score for the entire solution space. One potential modification is to increase the amount of work performed in a single test phase: instead of evaluating a single component function $f_i(\cdot)$ in the test phase, multiple components may be evaluated. Dependent on how these multiple results are combined into a single activity level, values of the single-evaluation test score are either increased or decreased: for instance, for the *Matching on Mars* problem (§2.2.4p31), 2 micro-feature pairs can be evaluated in each test phase instead of 1. If the activity level is determined by performing the logical OR operation of the two results, then the overall test score for each solution is increased; if the activity level is determined

with AND, then the test score for each solution is decreased. In general, any strategy for combining evaluation results may be used, provided that it attempts to decrease the uncertainty in the estimation provided by a single test phase (in accordance with the rationality assumption of §5.1p109).

Modifying the test procedure in this way alters the total number of test functions: for instance, when 2 randomly-selected micro-feature pairs are evaluated in the *Matching on Mars* problem, then component functions are mapped to test functions by the substitution $(c_i(\cdot), c_j(\cdot)) \rightarrow t_{ij}(\cdot)$, resulting in 800^2 possible test functions instead of 800. A different way of manipulating the test score is by using different (approximate) mappings $f_i(\cdot) \rightarrow t_i(\cdot)$: for instance, the test scores of Figure A.8 (p153) and Figure A.10 (p156) are obtained from the same objective function, depicted in Figure A.7 (p152), but use different mappings from $f_i(\cdot)$ to probabilistic test functions $t_i(\cdot)$.

6.1.2 Other Types of Decomposable Functions

For objective functions that are not simple summations, it is difficult to reach firm conclusions about what can or cannot be achieved with partial evaluations. The reason for this is that there are two strategies that potentially make such a function optimisable with SDS, and that these strategies can be applied in a wide variety of circumstances. The first strategy is to replace the objective function with a summation function, and then perform optimisation of that summation instead; the second strategy is to modify the ensemble of test and diffusion procedure in such a way that the resulting SDS algorithm does not converge to $\max_{x \in \mathcal{S}} \sum_i t_i(x)$, but converges to what is required by the optimisation problem. Both strategies can have as a consequence that what SDS converges to are not the optima of the original objective function; their effects should therefore be considered with great care. The following pages give a few examples of how these strategies can be applied.

Replacing General Objective Functions with Summations

Sometimes, an objective function can be replaced by a summation function that can then be optimised by standard SDS or related variants. This replacement function should reach its optima for the same argument values as the original function. This condition is easily fulfilled for strictly-order preserving replacements. For instance, if all $f_i(\cdot) \geq 0$, then the function:

$$\sqrt{\sum_{i=1}^{n} f_i(x)} \tag{6.13}$$

can be optimised by optimising Equation 6.1 instead, which reaches its optima for the same arguments $x \in \mathcal{S}$. *Product* functions form another example:

$$\prod_{i=1}^{n} f_i(x) \tag{6.14}$$

If all $f_i(\cdot) > 0$, then taking the natural logarithm of the product transforms the objective function into a summation without moving the optima:

$$\ln \prod_{i=1}^{n} f_i(x) = \sum_{i=1}^{n} \ln(f_i(x)) \tag{6.15}$$

When a strictly order-preserving replacement cannot be found, it is often possible to find a replacement that works well in practice. For example, an objective function often used in Template Matching – the normalised correlation of n-element vector a and larger vector b – is, for all starting positions x in b, given by:

$$\frac{\sum_i^n \sum_j^n (a(i) - a(j))(b(x+i) - b(x+j))}{\left[\sum_i^n \sum_j^n (a(i) - a(j))^2 \sum_i^n \sum_j^n (b(x+i) - b(x+j))^2 \right]^{1/2}} \tag{6.16}$$

Because of the denominator, this function cannot be easily decomposed into small component functions. However, the average of repeated evaluations of Equation 6.16 for randomly-selected, very small subsets of points (pairs,

triples, quadruples, ...) estimates the location of the maximum of the full correlation correctly for many large-scale image matching problems.[3]

Modifying Test and Diffusion

If SDS is used to optimise types of problems that cannot easily be reduced to summations, then the test and diffusion procedure may be adapted so that it performs the correct optimisation. For instance, a MINIMAX problem requires to find the solution x for which the worst-case $f_i(\cdot)$ is minimal:

$$\min_{x \in \mathcal{S}} \max_{i} (f_i(x)) \tag{6.17}$$

MINIMAX – and the related MAXIMIN, MAXIMAX and MINIMIN problems – are a class of objective functions that are very distinct from summation functions. However, if test and diffusion procedure are modified such that an agent memorises the maximum $f_i(x)$ found so far for its hypothesis x, bases its activity on this memory, and communicates the worst-case $f_i(x)$ together with hypothesis x in the diffusion procedure, then it is conceivable that clusters will form on the MINIMAX solution.

MOO problems (§4.1.2p86) can be decomposed in several ways, resulting in different types of solutions found by SDS. For instance, if single objectives constitute the individual components, then the MOO problem can be mapped to a summation function, and from there optimised using standard SDS or one of its variants. The optimum found in that way is not guaranteed to be Pareto-optimal. Alternatively, if the individual objectives themselves are decomposable, then the test procedure can evaluate one or a few components for each of the objective functions. Additional changes to the test procedure may be necessary to ensure that SDS converges to Pareto-optimal solutions.

[3]The evaluation of Equation 6.16 for pairs of randomly-selected points is equivalent to the use of Minchinton Cells (Figure 2.4 (p34)).

6.2 Practice

A more important question than what can be optimised *in principle* with SDS is for what types of problems it is expected to outperform other methods. Section 6.2.1 first identifies a few measures that can be used to judge the suitability of SDS algorithms. Section 6.2.2 then uses these measures to establish what type of objective functions are suitable for optimisation with SDS. Finally, Section 6.2.3 outlines a few problem scenarios where SDS is almost certainly outperformed by other methods.

6.2.1 NFL Measures for SDS

Christensen and Oppacher (2001) stated – in the context of the NFL theorems – that a search method can be expected to outperform random search when the objective function exhibits the regularities that the search method attempts to exploit. For local search methods, the exploited regularity is self-similarity in the "natural" topology of the objective function (§4.2.2p100). For search methods using recombination operators such as GA, ES and SS, or other methods using large-variational mechanisms, it is self-similarity in the neighbourhood structure defined by the search operator (Jones, 1995).

SDS algorithms with uncorrelated variational mechanisms attempt to improve search performance by exploiting a different aspect: evaluation of part of an objective function can be predictive for the full function value, while being computationally cheaper than the complete evaluation. SDS algorithms using small-variational mechanisms, as employed in Appendix A.3 (p146), and focused SDS variants also attempt to exploit self-similarity.

The identification of the aspects that SDS algorithms attempt to exploit, namely predictive power and reduced computational cost of partial evaluations, leads to concepts and measures that can be used as guidelines when

judging the suitability of SDS for concrete classes of problems. A few of these measures are proposed in the following pages.

Component Similarity

Predictive power of partial evaluations translates into the concept of *component similarity*: when component functions $f_i(\cdot)$ have values that are on average similar to each other, evaluation of one or a few components is predictive for the values of the entire function.[4] Component similarity can for instance be measured by the *variance* of the $f_i(\cdot)$'s:

$$\sigma^2(x) = \frac{1}{n}\sum_{i=1}^{n}(f_i(x) - \overline{f}(x))^2 \tag{6.18}$$

with $\overline{f}(x) = \frac{1}{n}\sum_j f_j(x)$ their mean.[5] In general, it can be stated that a lower component variance – especially for the optima – leads to less uncertainty and greater predictive power of partial evaluations.

Measures on the Objective Function

Component similarity is not the only useful measure for the performance of SDS. Another requirement is that the response of acceptable solutions needs to be high in comparison with the response of all other solutions. Because there are limits to how the topology of the objective function can be transformed into a test score with suitable characteristics for SDS, and limits to how the diffusion procedure can manipulate the critical response of the SDS process, a few measures on the structure of the objective function itself can give valuable information about the suitability of SDS for a given problem class. One measure is the ratio of the maximum (or minimum) objective

[4]This is true for summations as well as for other types, such as MINIMAX problems.

[5]For summation functions, $\overline{f}(x) = \frac{1}{n}f(x)$; for other types of objective functions, such as MINIMAX, this is not the case; however, even in these cases $\sigma^2(\cdot)$ can be calculated.

function value to the mean of objective function, effectively a signal-to-noise ratio:

$$\frac{\max_{x \in \mathcal{S}} f(x)}{\frac{1}{|\mathcal{S}|} \sum_{x} f(x)} \tag{6.19}$$

The higher this measure (or lower for minimisation problems), the easier it is to develop an SDS algorithm that performs well on the problem. The measure gives a good indication of potential search performance only if the variance of $f(x)$ values is low. If the variance is high, then there are potentially many solutions with objective function values that are quite close to the optimal value. In this case it is better to estimate the number of solutions that have function values within a certain range from the optimal value. For instance, if a solution space contains 10000 solutions within 5% of the best objective function value, then it is more difficult to develop an SDS algorithm that performs well on finding the global optimum than if only 100 such solutions exist.

Values for these measures can be obtained by either evaluating the solution space exhaustively for some example problems of that class, or – if this is infeasible – by sparse random sampling of the solution space.

Evaluation Gain and Test/Diffusion Ratio

The ratio of the computational cost of a full evaluation to the cost of a partial evaluation – the *evaluation gain* – gives an estimate of what can be gained by performing partial evaluations. For instance, for the *Matching on Mars* example, the evaluation gain is 800. The larger this gain, the larger the potential of SDS for outperforming methods that rely on full evaluations.

The evaluation gain gives a theoretical estimate of what can be gained in computational efficiency by using SDS. The true amount of evaluation work

performed by an SDS algorithm is of course higher than the evaluation gain suggests, and depends also on component similarity and signal-to-noise ratio.

The computational cost of a partial function evaluation can still be large in comparison with the computational cost of the diffusion procedure. It is therefore also useful to define the *test/diffusion ratio*, the computational cost of a single test phase relative to the cost of a single diffusion phase. For large values of this measure, the computational cost of the diffusion procedure can be neglected, and the overall computational cost of SDS estimated from the true amount of evaluation work performed.

6.2.2 When to Use SDS

Based on the measures developed in the previous section, a number of problem scenarios can be identified for which it seems sensible to try SDS:

- Problems with costly-to-evaluate, decomposable objective functions: a high evaluation gain means there is a large potential for improving search performance through partial evaluation. If the evaluation gain is not very high, but a full function evaluation is computationally very expensive, then it may still be worthwhile to use partial evaluations.
- Problems without any form of self-similarity in the objective function: the only possible improvement of search performance, relative to RGT or systematic search, can come from reducing the amount of evaluation work performed for each solution. SDS variants without small-variational mechanisms should be used.
- Problems with high signal-to-noise ratios: the more pronounced the ratio of the best objective function values to the remaining values, the easier it is to devise a test procedure that converts the objective function values into a test score for which the SDS process converges quickly to the maximum value.

- Problems with good component similarity, but with low signal-to-noise ratios: a partial evaluation is predictive for the value of the full function evaluation, but the difference between the best solution and the large number of suboptimal solutions is so small that, if the result of the partial evaluation is compressed to a binary activity level, many partial evaluations would have to be performed to distinguish between suboptimal and optimal solutions. In this case, it may be better to use a mechanism that directly compares the outcome of test procedures, as explained in Appendix A.4.2 (p154).

6.2.3 When Not to Use SDS

A few problem scenarios for which SDS is almost certainly outperformed by other search methods can be identified:

- Problems with indecomposable objective functions: the evaluation gain for these types equals 1. Because of the computational cost of the diffusion procedure, even GTR is likely to outperform SDS on this type.
- Problems with computationally-cheap objective functions: the evaluation gain is too small to counterbalance the computational overhead of the diffusion procedure and the requirement of convergence. Methods performing full function evaluations are likely to outperform SDS on this type of problems.
- Problems for which it is known a priori that the ideal solution exists in the solution space: any evaluation of a component function that gives a result deviating from the ideal solution removes all uncertainty regarding the status of that hypothesis. Unless there is a large degree of self-similarity in the objective function, methods performing partial evaluations without communication are likely to outperform SDS.

- Problems for which it is known a priori that only a single solution x has $t(x) \neq 0$: any positive evaluation of a solution removes all uncertainty regarding the status of that hypothesis. Methods performing partial evaluations without communication can easily outperform SDS.

- Problems for which local search algorithms can obtain the full objective function value of a solution without the need for a complete function evaluation: for problems of this type, a local search algorithm needs to perform one complete function evaluation at the start of each run; after each step, the new objective function value is obtained by updating the old value at a very low computational cost. Multi-start local search methods seem better tailored towards these problems, since they do not need to cope with the uncertainty caused by the partial evaluations, while performing similar amounts of evaluation work as SDS algorithms.

- Problems with few local optima, or with large basins-of-attraction for the global optimum: simple hill-climbing algorithms are likely to find the optima more easily than SDS algorithms, since the latter do not exploit the information in the topology of the objective function.

6.3 Conclusion

The chapter has provided partial answers to questions 7 and 8 from the roadmap (p16), namely on the usability of SDS methods in principle and in practice. It establishes procedures to convert objective functions into a format optimisable by SDS, and guidelines for testing the practical suitability of problems for optimisation with SDS. These answers conclude the foundational issues concerning SDS. The next chapter will conclude the thesis.

Chapter 7

Conclusions

The preface related the story of the six blind men of Indostan who went to "see" an elephant, and left with very different perceptions. The parable – originating from ancient Jainist or Buddhist scriptures – has been used many times before as metaphor. To these metaphorical interpretations, the author would like to add two more: firstly, if only these six men had cooperated, they could have constructed a more accurate picture of reality than each of them had individually. In this respect, the parable serves as a metaphor for the operation of SDS: a group of agents can construct an answer to a question for which each of the agents individually would fail. Secondly, by emphasising the many-sidedness of things, the parable serves as a metaphor for the thesis itself: SDS has subsequently been described as an algorithmic process, as a metaphor for computation, as a selective process, as the principle of partial evaluation, as a population-based metaheuristic using partial evaluation and direct communication, and as a nonlinear resource allocation process.

Section 7.1 briefly summarises the thesis and highlights its contributions. Section 7.2 discusses open questions and directions for future research. Finally, Section 7.3 addresses meta-level question E, the future of SDS itself.

7.1 Summary and Contributions

This dissertation has answered many of the foundational questions of the roadmap. In doing so, it has brought structure to the study of SDS, hopefully facilitating future applied research in this area. Chapter 3 has situated SDS within the wider context of selective processes, clarifying the connections of SDS with many scientific areas: for instance, the loose analogy between resource allocation processes in SDS and attentional processes in the brain (e.g., Nasuto et al., 1999) can now be understood in the larger context of general selection theory. Cross-fertilisation of the study of SDS with other fields is stimulated through these links: for instance, the mathematical techniques employed in models of SDS in (Nasuto, 1999) and (De Meyer, 2000) can, for instance, be used to model the behaviour of social insect colonies. Chapter 3 also prepared the way for the problem solving perspective of the next chapters: by introducing the framework of V+SR, it facilitated the discussion of metaheuristics in Chapter 4, and the discussion of potential modifications to the algorithmic formulation of SDS in Chapter 5. Chapter 4 itself compared SDS with a wide variety of metaheuristics. It was argued that SDS adds an important concept to the "library of ideas" in optimisation: the principle of partial evaluation. The strong distinction between algorithmic and process properties, made in Chapter 5, is essential to understanding what modifications can be made to the original algorithm, and how these changes affect the stochastic process that ensues from them. Chapter 6 significantly enlarged the repertoire of objective functions that are known to be optimisable in principle through partial evaluation. It also proposed measures to judge the suitability of particular classes of problems for optimisation with SDS. Using these measures, algorithms can be applied to problem classes in a more structured manner, by more careful matching of problem type with particular SDS variant; this will hopefully accelerate the design of real-world SDS applications in the future, and improve their performance.

Apart from the contributions to the study of SDS itself, at least two other levels of contributions can be discerned within the thesis: firstly, the discussion of a wide variety of selective processes, their integration within the common framework of general selection theory, and the detailed analysis of analogies and differences, hopefully makes its own modest contribution to the unification of science. Secondly, the building block approach towards the design of metaheuristics, advocated in Chapter 4, contributes to the unification and hybridisation of the field of metaheuristics.

7.2 Directions for Future Research

The foundational work does not end here. Although answers were given to many important questions, some remain unanswered or only partially answered. Many interpretations of SDS or useful frameworks for its study were not treated: SDS as a multi-agent system, a self-organising system, an unconventional neural network, a random sampling algorithm, a Bayesian method – all answers to question 3 of the roadmap – did not find a natural place within the thesis. Multi-agent systems (Ferber, 1999; Weiss, 1999) in particular seem to form a useful practical framework for SDS, especially in the context of the recent *multi-agent metaheuristics architecture*, developed by Roli and Milano (2002) in order to unify the study of metaheuristics under the umbrella of multi-agent systems.

The answer to question 6 of the roadmap, while attempting to be rigourous about what can be changed within the algorithmic formulation of SDS, will undoubtedly have missed a great number of potential modifications. Question 8, concerning decomposable functions, has left the issue of how non-summation functions can be decomposed largely unanswered. The measures that were developed in answer to question 9 only provide an informal start to the characterisation of search problems. More work is needed to turn these ideas into a formal protocol for algorithm design.

The remaining questions of the roadmap, regarding analysis, applications and implementations, have not been treated. However, future work in this area will also benefit from the ideas developed here: for instance, on the analysis side, it may be possible to express the existing mathematical models of SDS in function of the measures for characterisation of search problems that were proposed in Chapter 6. This could eventually lead to a formalised design methodology for real-world applications and implementations.

7.3 The Future of SDS

When SDS was introduced in (Bishop, 1989*a*), it was as an "anarchic technique for pattern classification". Since then, it has proven to be more than a pattern classification technique, but it is still as anarchic as ever. In the coming era of grid and quantum computing, a decentralised technique such as SDS may just be what is needed to make these systems operate efficiently.

There is something very elegant about the completely distributed and concurrent way that SDS performs computations. As our understanding grows of how algorithmic properties are best tuned towards problems will its underlying stochastic process become easier to handle. And as it becomes easier to handle, the algorithm will be more frequently adopted as a problem solving method, eventually spreading itself through the scientific and technological community. This diffusive process has already started; hopefully this work will speed up its acceptance, and contribute to a wider dissemination of its principles.

Appendix A

Examples

The following pages present examples of SDS variants on different types of objective functions. They are offered as support for arguments of the main text; each example exhibits multiple salient features that are explained in the accompanying description. A particular feature can often be demonstrated in more than one way, and the decision to report one variant does not exclude the potential usefulness of other variants. Results show "typical runs" of an algorithm; formal analysis or comparison across variants is rarely performed. All problems are based on the two image matching problems of Figure 2.3 (p33); if a particular figure consists of two graphs, then the left corresponds to the left part of Figure 2.3, and the right graph to the right hand problem.

A.1 Context-Sensitive SDS

Principles A.1. *Context-sensitive* SDS *has a more balanced pattern of resource allocation than standard* SDS. *Multiple clusters can coexist in the solution space; these clusters are not necessarily located in neighbouring points.*

Context-sensitive SDS was first proposed by Nasuto (1999). The sole difference with the standard SDS algorithm can be found in the diffusion

phase for *active* agents: each active agent selects one agent at random; if the selected agent is active and supports the same hypothesis, then the selecting agent becomes inactive and picks a new random hypothesis. The context-sensitive mechanism counteracts the formation of large clusters, since the probability that an active agent contacts an agent with the same hypothesis increases with relative cluster size. The mechanism thus adds a component of negative selection or a negative feedback loop from clusters onto themselves.

Figure A.1 shows the search behaviour of context-sensitive SDS for the *Matching on Mars* problem (§2.2.4p31). Context-sensitive SDS divides its resources more evenly than standard SDS, resulting in multiple stable clusters in a number of good hypotheses, even when a perfect solution exists. In this example, the clusters are located in neighbouring points in the solution space, but they do not need to be in each other's vicinity. In the left graph, the fourth best hypothesis manages to attract a small cluster as long as the optimal solution has not been discovered, but cannot maintain it once the optimal solution has been found. In the right graph, all four hypotheses are of comparable quality and maintain clusters of similar size.

A.2 Deterministic Diffusion Search

Principles A.2. *A de-randomised version of standard* SDS *exhibits largely the same behaviour as standard* SDS.

When de-randomising standard SDS, one important condition needs to be fulfilled: for any hypothesis, each component should have the *potential* of being evaluated. Furthermore, a deterministic procedure needs to generate enough variation to exhibit behaviour qualitatively similar to SDS. For the *Matching on Mars* example (§2.2.4p31), DDS is obtained by replacing all randomised steps of standard SDS with the following deterministic steps:

1. Micro-features are numbered 1 to m; each agent is initialised with a micro-feature number f. During the test phase of iteration t, an agent evaluates micro-feature $((f + t) \bmod m)$.

2. Agents are numbered 1 to N; each agent is initialised with an agent number a. In the diffusion phase of iteration t, an inactive agent contacts agent $((a + t) \bmod N)$.

3. Candidate solutions are numbered 1 to M; each agent is initialised with a solution number s. When an inactive agent needs to pick a new hypothesis at the end of the diffusion phase, he chooses hypothesis $((s + t) \bmod M)$, with t the number of the current iteration.

4. Micro-feature numbers f are initialised as follows: the sequence 1 to m is deterministically permuted and the numbers assigned to agents 1 to m; if $m < N$, then this procedure is repeated until all agents have a micro-feature. Agent numbers a are initialised by deterministically permuting the sequence 1 to N, and assigning the numbers of this sequence to agents 1 to N. Solution numbers s are initialised by distributing the agents at regular intervals throughout the solution space.

At first sight, this procedure seems more complicated than standard SDS. However, in some circumstances, de-randomising at least some of the steps may be useful: for instance, distributing agents evenly across the solution space during initialisation or distributing contact agents evenly across all agents can have positive consequences for particular problems (remembering NFL, it can also have negative consequences). Furthermore, de-randomisation can reduce the computational cost of a single iteration for software implementations, since adding numbers is more computationally efficient than generating random numbers. In case of implementation in hardware, de-randomisation reduces circuit complexity. For the current example, Figure A.2 shows that DDS is indistinguishable from a typical run of standard SDS.

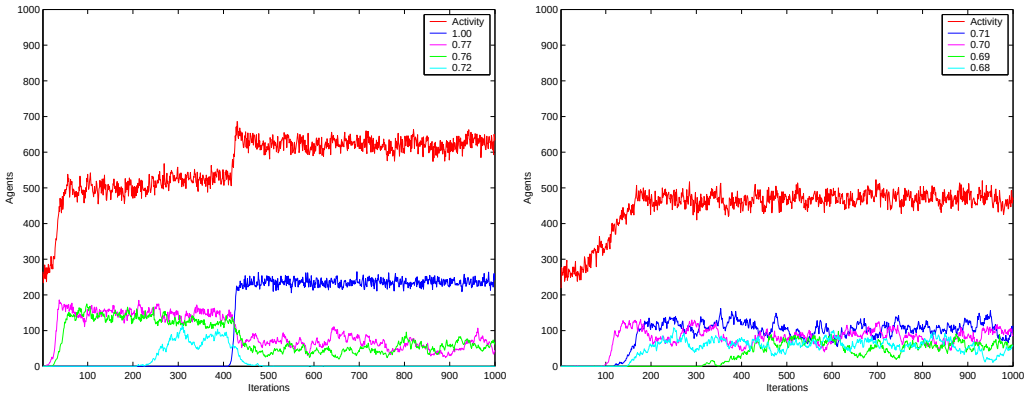

Figure A.1: *Search behaviour of a population of 1000 context-sensitive agents for 1000 iterations on the objective functions of Figure 2.5 (p35). Depicted are the total agent activity and the number of agents supporting the best 4 hypotheses. During each test phase, agents compare a single micro-feature from the target with the corresponding micro-feature from the search space.*

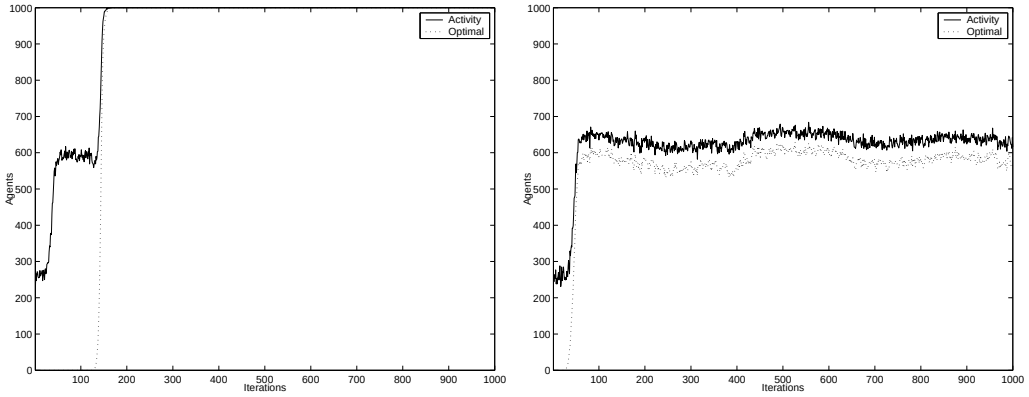

Figure A.2: *Search behaviour of a population of 1000 deterministic agents for 1000 iterations on the objective functions of Figure 2.5 (p35). Depicted are the total agent activity and the number of agents supporting the optimal hypothesis. During each test phase, agents compare a single micro-feature from the target with the corresponding micro-feature from the search space.*

A.3 A Strategy of Errors

Principles A.3. *An* SDS *variant with small-variational mechanism exhibits hill-climbing behaviour, and can therefore converge faster on self-similar objective functions. Clusters are spread out over a few neighbouring locations.*

A small-variational mechanism can be added to standard SDS in several ways; one possibility is to slightly disturb the copying of hypotheses in the diffusion phase. For the *Matching on Mars* example (§2.2.4p31), this can be achieved by adding a random integer-valued offset during copying. The fifth line of the standard SDS diffusion phase in Table 2.3 (p27) is replaced by:

$$\text{agent.hypothesis} = \text{agent2.hypothesis} + \text{offset};$$

For this particular example, the offset is generated as $\left[\frac{r}{s}\right]$, $[\,]$ denotes the operation of rounding to the nearest integer, r is a random 2-tuple sampled from a normal distribution with mean 0 and standard deviation 1, and s is a parameter controlling the accuracy of the copying process.[1] The effect of parameter s on accuracy can be seen for three different values in Figure A.3.

Figure A.4 demonstrates the effect of this mechanism on the behaviour of standard SDS. When a solution succeeds in attracting agents, the random-offset mechanism distributes part of the agents over neighbouring locations in the solution space. When one of those neighbouring solutions is better than the first, it attracts *more* agents, while spreading agents evenly around its own location. This effect accelerates the discovery of better solutions in the vicinity of already discovered solutions. The cluster of agents follows an uphill trajectory as time progresses. This hill-climbing behaviour benefits convergence times on self-similar objective functions; more specifically, it

[1] s actually controls the standard deviation of the normal distribution. For instance, for $s = 4$, the standard deviation of the sampled normal distribution is $\frac{1}{4}$.

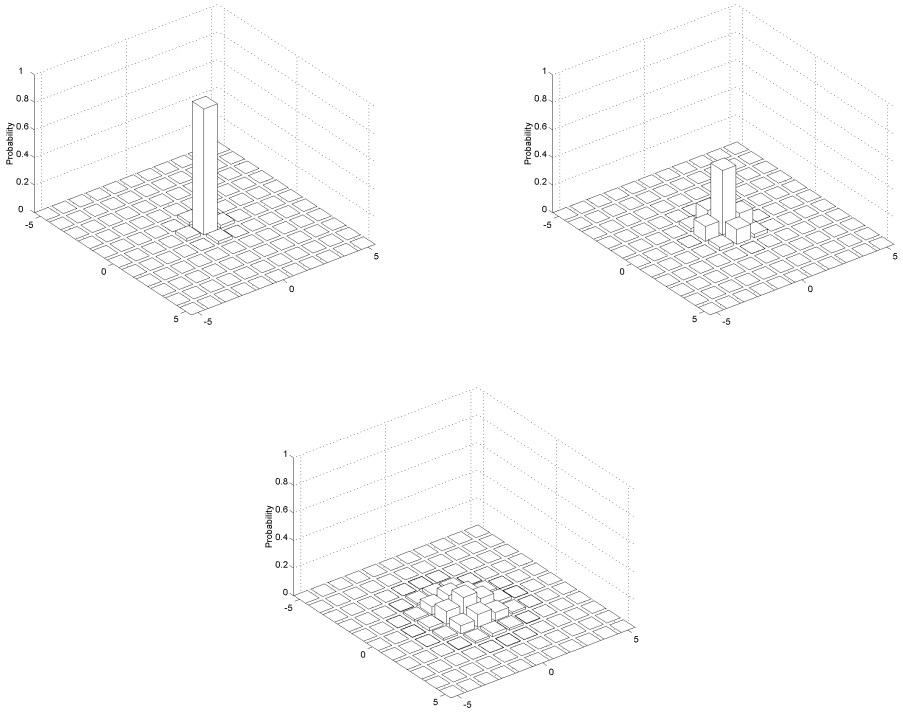

Figure A.3: Relative frequencies of integer-valued offsets generated by $\left[\frac{r}{s}\right]$ for three different s-values: $s = 4$ (right), $s = 2$ (left) and $s = 1$ (bottom). In each figure, point $(0, 0)$ corresponds to faultless copying of the hypothesis. For $s = 4$, approximately 91% of all hypotheses are copied accurately; for $s = 2$ and $s = 1$, the numbers are 47% and 15% respectively. Smaller s-values result on average in larger offset values, and thus less accurate copying.

reduces the long periods of sub-optimal behaviour observed in standard SDS (§2.2.4p36). The large-variational mechanisms of standard SDS meanwhile ensure that the search cannot get permanently stuck in a local optimum.

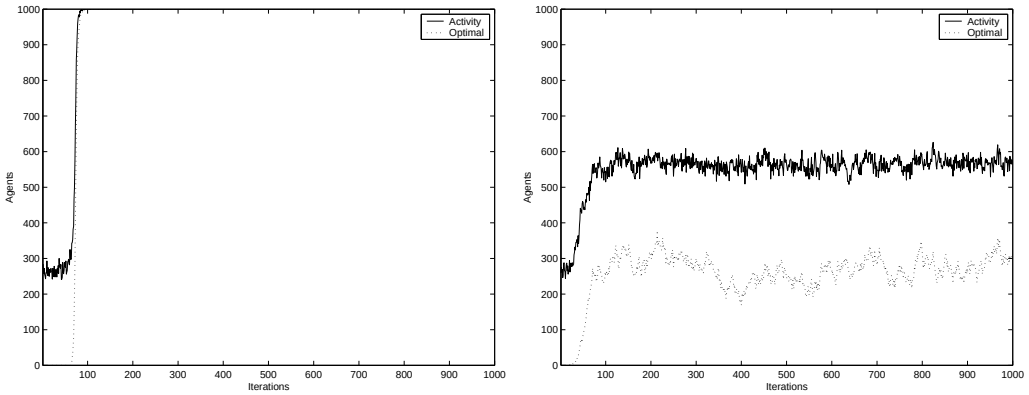

Figure A.4: *Search behaviour of a population of 1000* SDS *agents for 1000 iterations on the left and right objective functions of Figure 2.5 (p35). A random-offset mechanism during copying is used with* $s = 4$*. Depicted are the total agent activity and the number of agents supporting the optimal hypothesis. During each test phase, agents compare a single micro-feature from the target with the corresponding micro-feature from the search space.*

How a strategy of errors affects convergence is demonstrated in Figure A.5. It reports a cumulative distribution of convergence times for standard SDS with and without the random-offset mechanism. Adding random offsets greatly reduces the mean and variance of convergence times: *with* random offsets, all runs converge in less than 200 iterations; *without* offset mechanism, 5% of the runs has even failed to converge after 1000 iterations.

The righthand graph of Figure A.4 shows that the average cluster size for the optimal solution is smaller than without strategy of errors. However, if no perfect solution exists, then the cluster remains spread out over a *region* of the solution space. The principle is demonstrated in Figure A.6. At the end of the search, good solutions that are close to the optimal solution

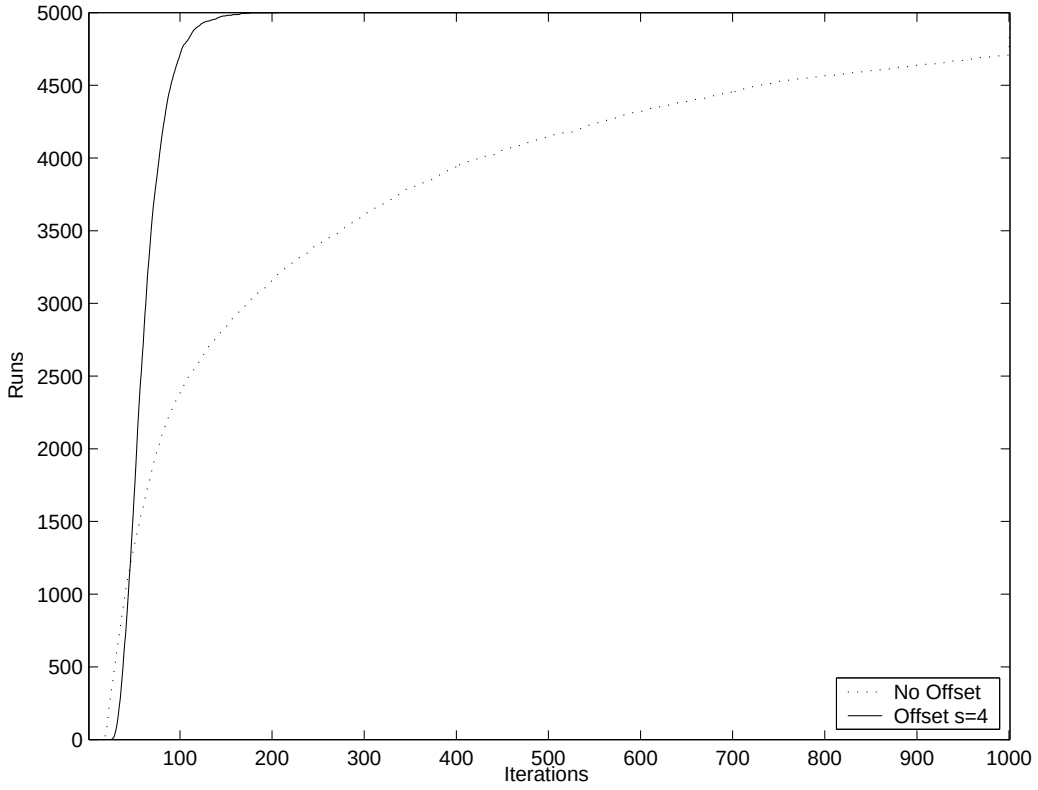

Figure A.5: *Comparison of convergence times for standard* SDS *with and without a strategy of errors. 5000 runs of standard* SDS *without and with offset mechanism ($s = 4$) were performed on the left objective function of Figure 2.5, where a perfect solution exists in the solution space. The convergence time for this example is the number of iterations until* all *agents are active. Depicted are the cumulative distributions of convergence times.*

have also attracted sizeable clusters. These are not isolated, as in the case of context-sensitive SDS, but mutually support each other: the offset mechanism diverts agents from one cluster to a neighbouring cluster, and does so in both directions. The spread-out population of agents can therefore be regarded as a *multiple-hypothesis cluster* consisting of several single-hypothesis clusters.

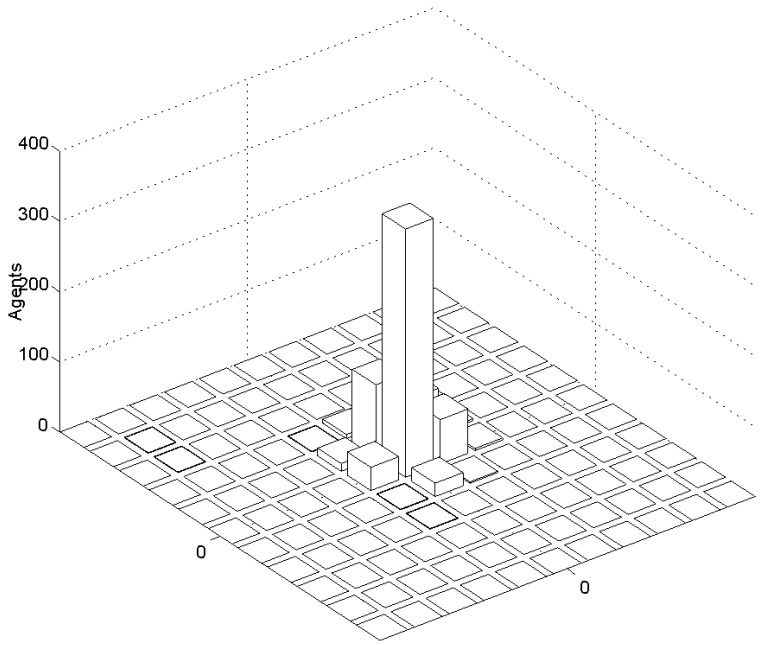

Figure A.6: The distribution of agents around the location of the optimal solution at the end of the experiment depicted in the right graph of Figure A.4. Point $(0, 0)$ contains the optimal solution, $s = 4$.

A.4 Error Minimisation

The next two examples demonstrate how minimisation of an error function can be achieved through partial evaluation only. The problem is the *Matching on Mars* example of §2.2.4p31, but with a different objective function: instead of maximising the number of corresponding micro-features, the aim is to find the (x, y) location in the solution space with the minimal *Manhattan distance* between target and underlying part of the image:

$$\min_{x,y} \sum_{i,j} |T(i,j) - I(x+i, y+j)| \tag{A.1}$$

where $T(\cdot)$ and $I(\cdot)$ are pixel grey scale values of template and image respectively. The objective function is decomposable: each term of the summation can be evaluated independently. All terms of the summation function are expressed on a common scale: the absolute value of the difference of two grey scale values varies between 0 and 255. For a perfect solution, each individual term of summation A.1 equals 0; deviation from 0 for an individual term thus forms a partial indication of the quality of a match. Figure A.7 shows the *average* manhattan distance for each (x, y) location in the solution space, i.e., the summation A.1 divided by the number of pixels in the template.

A.4.1 Error Minimisation with Standard SDS

Principles A.4. *Mapping an integer-valued objective function to* $\{0, 1\}$ *activity, using a non-linear mapping to* $[0, 1]$ *and probabilistic test functions.*

Standard SDS can be used to perform optimisation of a general summation function by modifying the test procedure so that it maps values of the individual terms of the summation to $\{0, 1\}$ activity levels. To achieve this, the test procedure calculates, for hypothesis (x, y) and a randomly-selected pixel of the template (i, j):

$$u_{ij}(x, y) = 1 - \frac{|T(i,j) - I(x+i, y+j)|}{C} \tag{A.2}$$

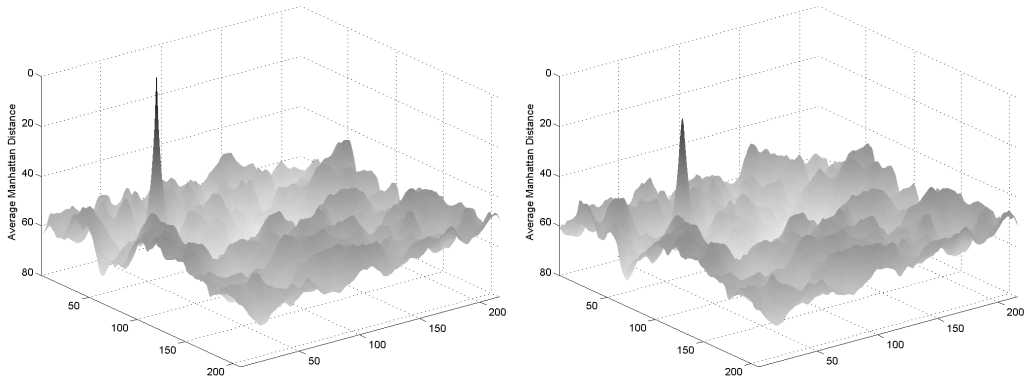

Figure A.7: Average Manhattan distance for the left and right image matching problems of Figure 2.3 (p33). Note that the z-axis of the graphs has been reversed. The minima of the objective functions correspond to the single well-defined peaks; these indicate the location of the correct solution.

with C the cutoff value; in this example, $C = 50$. Activity is then determined as follows: for $u_{ij}(x, y) \leq 0$, activity of the agent is set to 0; for $u_{ij}(x, y) > 0$, activity of the agent is 1 with probability $u_{ij}(x, y)$. This can be achieved by drawing a uniform random random number $r \in [0, 1]$; if $r < u_{ij}(x, y)$, then activity is set to 1, otherwise it is set to 0. The resulting test score averaged over all pixels – the expected proportion of times an agent's activity level equals 1 after testing hypothesis (x, y) – can be seen in Figure A.8.

The search behaviour of a population of standard SDS agents using this test procedure is reported in Figure A.9. Since the test scores are quite like the test scores from Figure 2.5 (p35), and the diffusion phase is exactly the same, the convergence and quasi-equilibrium behaviour of this experiment are very similar to the results of Figure 2.6 (p35).

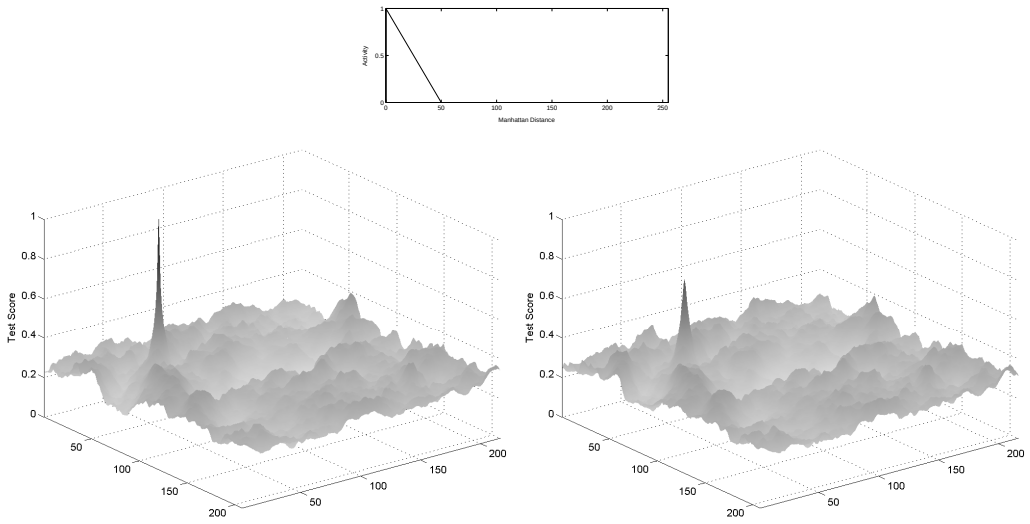

Figure A.8: *Test scores derived from the objective function in Figure A.7. The values of the individual terms of Equation A.1 are mapped to the interval* $[0, 1]$ *via the scaling specified by the small top-level graph. For each hypothesis, these values are then averaged over all terms of the summation. Note that this non-linear mapping is not strictly order-preserving.*

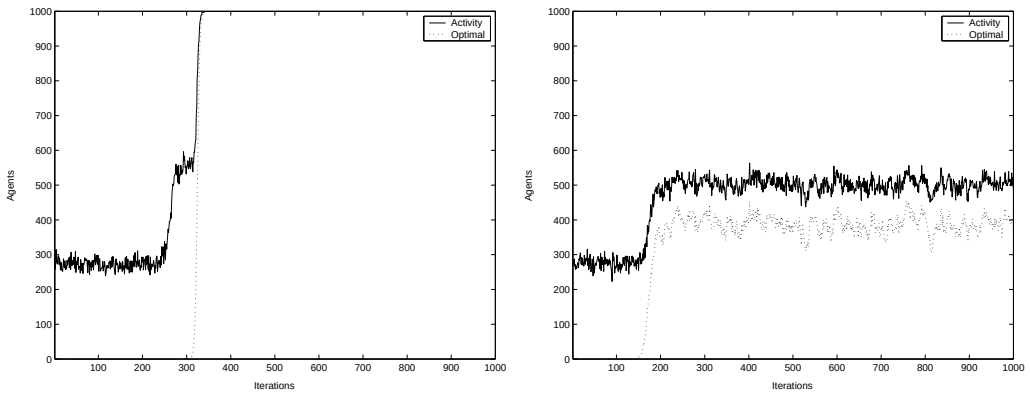

Figure A.9: *Search behaviour of 1000 standard* SDS *agents for 1000 iterations for the left and right test scores of Figure A.8. Depicted are the total agent activity and the number of agents supporting the optimal hypothesis. During each test phase, agents calculate Equation A.2, with* $C = 50$.

A.4.2 Activity as Belief

Principles A.5. *Using* $[0, 1]$ *activity levels; a modified selection mechanism in the diffusion phase radically alters resource allocation. Convergence behaviour on self-similar objective functions is improved by a small-variational mechanism. Clusters are spread out over regions in the solution space.*

The SDS variant described in this section is quite unlike any other variant used before. It does not rely on binary activity levels, but on activity levels in the interval $[0, 1]$; activity can thus be interpreted as the *belief* of an agent in the correctness of its hypothesis. The more fine-grained type of activity implies a different diffusion phase: an agent compares its activity with the activity from another agent, and based on the magnitude of the difference, decides whether to retain its own hypothesis, copy the other agent's hypothesis, or randomly adopt a new one. This mechanism radically alters the resource allocation: its behaviour is not dependent on a fixed critical response; since the process consists of *comparing* activity levels, it is the relative difference between the peak values and the background level that determine the resource allocation pattern. Conversely, the mechanism requires a good component similarity; if component similarity is low, then additional measures need to be taken to ensure proper convergence to the optima.

This mechanism is applied to the *Matching on Mars* example. In the test procedure, agents calculate Equation A.2; if the value of $u_{ij}(x, y) < 0$, then activity is set to 0, otherwise activity is set to the value of $u_{ij}(x, y)$. In this diffusion procedure, *all* agents select a random agent for communication. If the selected agent has a lower activity, then the selecting agent does nothing. If the selected agent has a higher activity, and the difference between the two activity levels is larger than a constant d, then the hypothesis is copied with addition of a random offset. If the difference in activity levels is smaller than d, then the selecting agent picks a new random hypothesis. The diffusion procedure is outlined in pseudo-code in Table A.1.

```
for agent = 1 to All Agents
   agent2 = Pick-Random-Agent(Agents);
   if (agent2.activity > agent.activity)
      if (agent2.activity − agent.activity > d)
         agent.hypothesis = agent2.hypothesis + offset;
      else
         agent.hypothesis = Pick-Random-Hyp();
      end
   end
end
```

Table A.1: *Activity-as-belief diffusion phase.*

In each diffusion phase, roughly half of all agents will either copy a hypothesis or adopt a new one at random (unless a perfect solution exists). The proportion of these two actions is controlled by diffusion parameter d: the smaller d, the lower the proportion of newly generated hypotheses. For not-too-high values of d, the diffusion procedure somewhat resembles a context-free diffusion: when all agents have low activity levels, the proportion of newly generated hypotheses will be relatively high; if an agent has a hypothesis that results in activity levels that are on average more than d units higher than the activity levels of the surrounding agents, then this hypothesis will be copied by other agents and spread through the population. As the proportion of agents with this better hypothesis grows, the average difference levels in activity will gradually drop below d, stabilising the cluster size in a context-free manner. This diffusion mechanism is in theory not strictly order-preserving. However, the circumstances in which it does not preserve order are rarely encountered in practice. By not using a fixed parameter d, but by making the decision to retain, copy or randomly select hypotheses probabilistically dependent on the magnitude of the difference of the two activity levels, the diffusion procedure can be made strictly order-preserving.

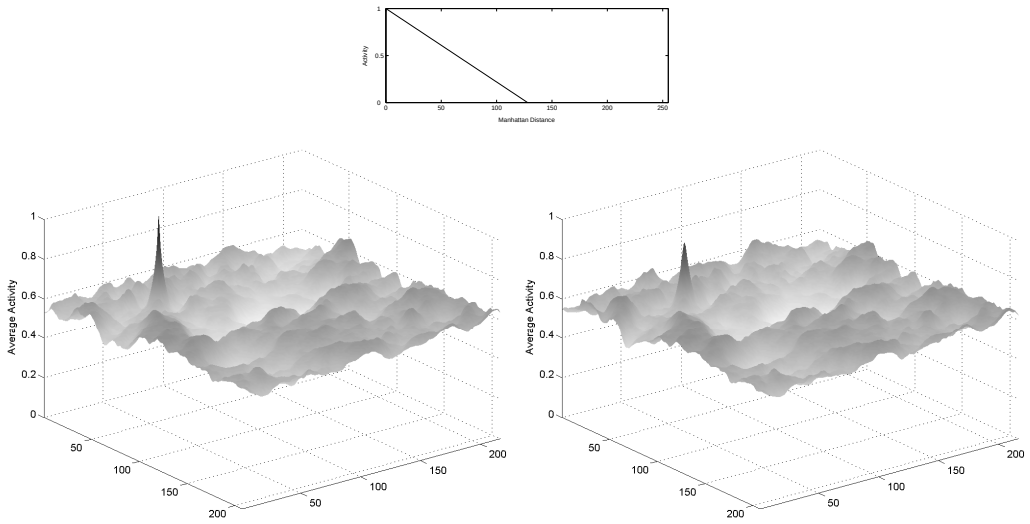

Figure A.10: *Average activity derived from the objective function in Figure A.7. The values of the individual terms of Equation A.1 are mapped to the interval* $[0, 1]$ *via the scaling specified by the small top-level graph. For each hypothesis, these values are then averaged over all terms of the summation.*

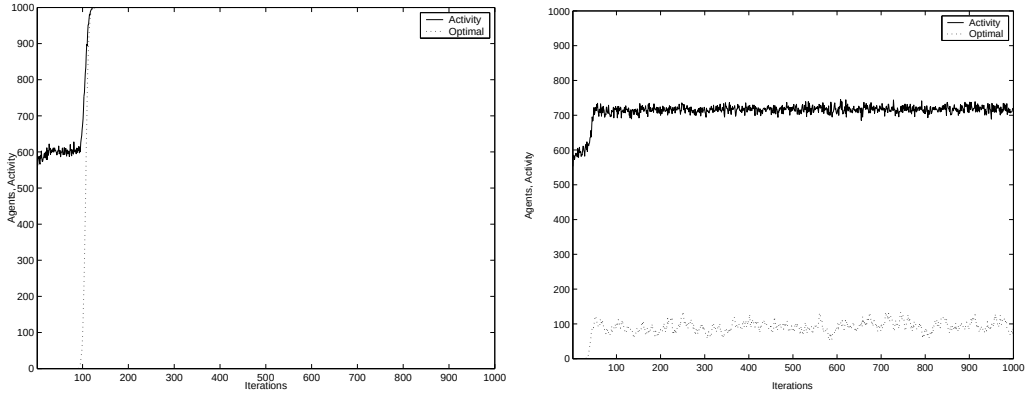

Figure A.11: *Search behaviour of a population of 1000 agents for 1000 iterations for the left and right average activity levels of Figure A.8. Depicted are the overall agent activity and the number of agents supporting the optimal hypothesis. During each test phase, agents calculate Equation A.2 with* $C = 128$*. Diffusion parameter* $d = 0.1$*, and random-offset parameter* $s = 2$*.*

Figure A.10 reports the average activity levels calculated in the test procedure. Figure A.11 shows the search behaviour of a population of agents, using the above outlined test and diffusion procedure, and a strategy-of-errors. The final multiple-hypothesis cluster is shown in Figure A.12.

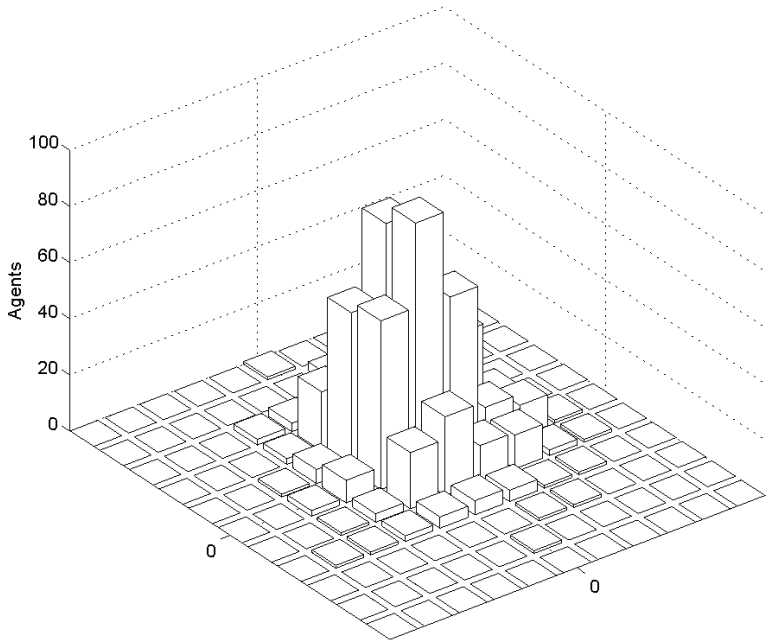

Figure A.12: The distribution of agents around the location of the optimal solution at the end of the experiment depicted in the right graph of Figure A.11. Point $(0, 0)$ contains the optimal solution, $s = 2$.

A.5 Lattice Stochastic Diffusion Search

Small-World Effects in Lattice Stochastic Diffusion Search

K. De Meyer, J.M. Bishop, and S.J. Nasuto

Department of Cybernetics, University of Reading, Whiteknights,
PO Box 225, Reading, RG6 6AY, United Kingdom
k.demeyer@rdg.ac.uk

Abstract. Stochastic Diffusion Search is an efficient probabilistic best-fit search technique, capable of transformation invariant pattern matching. Although inherently parallel in operation it is difficult to implement efficiently in hardware as it requires full inter-agent connectivity. This paper describes a lattice implementation, which, while qualitatively retaining the properties of the original algorithm, restricts connectivity, enabling simpler implementation on parallel hardware. Diffusion times are examined for different network topologies, ranging from ordered lattices, over small-world networks to random graphs.

1 Introduction

Stochastic Diffusion Search (SDS), first introduced in [1], is a population-based best-fit pattern matching algorithm. It shares many similarities with e.g. Evolutionary Algorithms, Memetic Algorithms and Ant Algorithms [2]. During operation, simple computational units or *agents* collectively construct a solution by performing independent searches followed by diffusion through the population of potentially relevant information. Positive feedback promotes better solutions by allocating more agents for their investigation. Limited resources induce strong competition from which a large population of agents corresponding to the best-fit solution rapidly emerges.

SDS has been successfully applied to a variety of real-world problems: locating eyes in images of human faces [3]; lip tracking in video films [4]; self-localisation of an autonomous wheelchair [5]. Furthermore, a neural network model of SDS using Spiking Neurons has been proposed [6]. Emergent synchronisation across a large population of neurons in this network can be interpreted as a mechanism of *attentional amplification* [7]; the formation of dynamic clusters can be interpreted as a mode of *dynamic knowledge representation* [8].

The analysis of SDS includes the proven convergence to the globally optimal solution [9] and linear time complexity [10]. Recently it has been extended to the characterisation of its steady state resource allocation [11].

As search is applied to ever more complex problems with larger search spaces, even the most efficient algorithms begin to require some form of dedicated hardware to meet real-world performance demands. The standard formulation of

SDS is parallel in nature and thus implementing it on parallel hardware seems straightforward. However, the requirement for efficient communication links between all agents means that it is difficult to implement efficiently, both on dedicated hardware (e.g. FPGA's) or general purpose parallel computers.

This paper describes the effects of restricting communication between agents. In particular, the effects of the number of connections and the topology of the underlying connection graph on search performance are investigated empirically. It will be shown that, even for a modest number of connections, the performance of randomly connected networks of agents is close to the performance of standard SDS, and much better than performance of ordered lattices with the same average number of connections. However, small-world networks [12], based on regular lattices with a few long-range connections, perform almost as good as random networks. Two important conclusions can be drawn from the results:

1. Inter-agent communication in SDS can be significantly restricted without decreasing the performance of the algorithm too much, given that either a random or small-world network topology is used. However, the limited number of long-range connections in small-world networks facilitates the layout of the connections, making them the preferred network topology for hardware implementation.
2. Independent from the actual search process of SDS, the paper seems to confirm results in several epidemiological models using the small-world network topology, e.g. [13, 14]: namely that information or disease spreads much easier on small-world networks and random graphs than on ordered lattices.

2 Stochastic Diffusion Search

SDS utilises a population of *agents* to process information from the *search space* in order to find the best fit to a specified target pattern, the *model*. Both the search space and model are composed of *micro-features* from a pre-defined set. For instance, in a string matching problem, both the search space and model are composed of a one-dimensional list of characters.

In operation each agent maintains a hypothesis about the location and possible transformations (the *mapping*) of the model in the search space. It evaluates this hypothesis by testing how a randomly selected micro-feature of the model, when mapped into the search space, compares to the corresponding micro-feature of the search space. This part of the algorithm is called the *testing phase*. Based on the outcome of this test, agents are divided into two modes of operation: *active* and *inactive*. An active agent has successfully located a micro-feature from the model in the search space; an inactive agent has not.

During the *diffusion phase* the information about potential solutions may spread through the entire population. This is because each inactive agent chooses at random another agent for communication. If the selected agent is active, the selecting agent copies its hypothesis: *diffusion* of information. Conversely, if the selected agent is also inactive, then there is no information flow between agents; instead, the selecting agent adopts a new random hypothesis.

By iterating through test and diffusion phases agents will stochastically explore the whole search space. However, since tests will succeed more often in regions having a large overlap with the model than in regions with irrelevant information, an individual agent will spend more time examining 'good' regions, at the same time attracting other agents, which in turn attract even more agents. Potential matches to the model are thus identified by concentrations of a substantial population of agents.

Two important performance criteria for SDS are *convergence time* and *steady-state resource allocation*. Convergence time can in general be defined as the number of iterations until a stable population of active agents is formed and is very clearly defined when a single, perfect match of the model is present in the search space: it is then simply the number of iterations until all agents become active. Resource allocation is a measure for robustness in the case of imperfect matches and presence of noise: it is defined as the average number of active agents during steady-state behaviour, and is dependent on the quality of the match.

Examples of search behaviour, resource allocation and a more detailed description of the algorithm can be found in [11, 15].

3 Lattice Stochastic Diffusion Search

SDS gains its power from the emergent behaviour of a population of communicating agents and as such is inherently a parallel algorithm - notionally each agent is independent and its behaviour can be computed by an independent processor. However, a fundamental difficulty in implementing standard SDS efficiently on either a parallel computer or dedicated hardware is its requirement that each agent is able to directly communicate with all others. An obvious alteration to the algorithm is thus to restrict agent communication to a smaller number of agents. In the resulting algorithm, Lattice Stochastic Diffusion Search (LSDS), agents are assigned to spatial locations (e.g. on a 2D square grid) and connections between agents are specified. During the diffusion phase, agents will only communicate with agents they are connected to. Regular, local connections lead to an ordered lattice; or connections can be specified at random, thus effectively constituting a random graph.

An important question is how the performance and robustness of LSDS compares to standard SDS. The performance of standard SDS has previously been extensively analysed using Markov chain theory [9–11]. However, in LSDS the probability distribution determining communication between agents defines a neighbourhood structure over the entire set of agents. Analysis of this kind of process as a Markov chain is extremely complex: the process is not characterised by a simple integer denoting the number of active agents, but by the exact topological location of both active and inactive agents. These types of Markov processes are also known as Markov random fields. Work on a mathematical model incorporating the effects of restricted connectivity is ongoing, but at present performance measures for LSDS are investigated through simulations.

	$k=4$		$k=8$		$k=12$		$k=24$		$k=N$
	random	lattice	random	lattice	random	lattice	random	lattice	
$N=64$	15.5	15.7	11.5	13.4	10.9	11.8	10.3	10.4	10.2
$N=256$	21.3	29.5	15.0	23.8	13.9	20.1	13.1	16.3	12.5
$N=1024$	25.8	55.5	18.1	44.1	16.7	36.0	15.6	27.3	14.8
$N=4096$	32.3	106.9	21.1	83.4	19.5	66.9	18.1	49.3	17.1

Table 1. T_d in iterations for 4 different populations sizes N. Results are reported for random graphs with a mean number of connections per agent k; and for regular 2-dimensional square lattices with k-nearest neighbours connections and periodic boundary conditions. The case where $k = N$ corresponds to standard SDS. All results are averaged over 1000 runs, and for random graphs over 10 different graphs each.

4 Experimental Results

4.1 Convergence Time

[16] introduced the terms 'time to hit' (T_h) and 'diffusion time' (T_d) in the analysis of convergence time (T_c) of standard SDS. T_h is the number of iterations before at least one agent of the entire population 'guesses' the correct mapping and becomes active. T_d is the time it takes for this mapping to spread across the population of agents. It is clear that T_h is independent of the connectivity of the population and only depends on search space size M and number of agents N. T_d, on the other hand, is very much dependent on the connectivity within the population and on population size N, but independent of M. To focus attention on the effect of connectivity, experimental results for T_d are reported. It could be argued that for complex, high-dimensional problems $T_h \gg T_d$, and thus that the effect of T_d on T_c can be neglected with respect to T_h. However, T_d should not just be regarded as a measure for rate of convergence, but more as a measure for 'ease of information spread'. As such, it is also indirectly a measure for robustness: experiments indicate that the more freely information spreads through the network, the more robust the algorithm is in the case of imperfect matches or noise [15].
T_d is studied by initialising one randomly chosen agent with the correct mapping and recording the number of iterations until this mapping has spread to all other agents. Results are reported in Table 1. T_d for regular lattices does not scale very well with population size for a fixed number of connections k. For random graphs, T_d scales much better with population size and performance remains close to performance of standard SDS, even for a small number of connections and large population sizes.

4.2 Small-Worlds: Between Order and Randomness

Regular lattices have poorer T_d than random graphs, but are easier implemented in hardware, since connections are local and thus shorter, and regular. However,

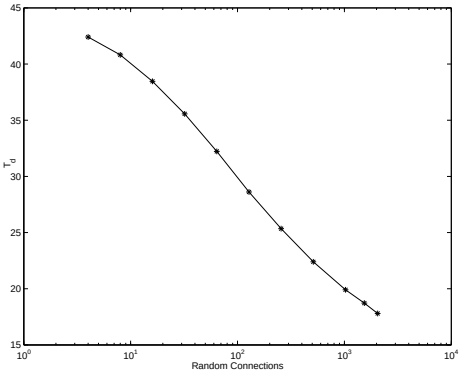

Fig. 1. T_d in iterations for $N = 1024$ and variable number of random extra connections x. Small-world networks are constructed starting from an ordered lattice with $k = 8$ and with x extra connections added at random. Note that for $x = 2048$, the last measurement, the mean connectivity in the network is $k = 12$. All results are averaged over 1000 runs, and over 10 different networks each.

diffusion of information in a population of searching agents shows an obvious correspondence with epidemiological models. Such models of disease or information spread have recently received much attention, due to the interest in the so called 'small-world effect'. It was shown in e.g. [12, 17] that only a limited amount of long-range connections is necessary to turn a lattice with k-nearest neighbour connections into a small-world network, in which spreading of disease or information behaves much more like spreading on random graphs. To test whether the same is true for LSDS, small-world networks were generated as described in [18]: a number of random links is added to an ordered lattice, and no connections are removed. T_d is recorded for various numbers of random connections; the results can be seen in Fig. 1. Randomly adding connections decreases T_d more or less exponential for a wide interval of parameter x, leading to an almost linear curve in the semilog plot. The benefits of adding relatively few long range connections seem obvious: a small-world network with only 256 extra connections (mean connectivity $k = 8.5$) outperforms a regular lattice with $k = 24$; a small-world network with 512 extra connections (thus $k = 9$) diffuses information twice as fast as the underlying regular lattice with $k = 8$, and is only 1.5 times slower in diffusing than fully connected SDS. Note that, even when adding much more connections, T_d will never become less than the value for standard SDS, in this case 14.8 (see Table 1).

5 Conclusions

The effect of mean number of connections and connection topology on diffusion time T_d was investigated empirically. T_d is an important performance parameter,

not just because of its effect on T_c, but more importantly because it is also an indicator for resource allocation stability [15].

The good performance of 'small-world' LSDS has wider implications than just implementation in hardware. It has been suggested (e.g. in [12]) that biological neural structures can show small-world connectivity. The neural network architecture implementing standard SDS [6] uses biologically inspired neurons operating as filters on the information encoded in the temporal structure of the spike trains. Relaxing the requirements of full connectivity in these networks leads to a more plausible architecture, while still allowing for self-synchronisation across a large population of neurons [7] to occur.

References

1. Bishop, J.M.: Stochastic Searching Networks. Proc. 1st IEE Conf. ANNs, London (1989) 329–331
2. Corne, D., Dorigo, M., Glover, F.: New Ideas in Optimisation. McGraw-Hill (1999)
3. Bishop, J.M., Torr, P.: The Stochastic Search Network. In Lingard, R., Myers, D.J., Nightingale, C.: Neural Networks for Images, Speech and Natural Language. Chapman & Hall, New York (1992) 370–387
4. Grech-Cini, E.: Locating Facial Features. PhD Thesis, University of Reading (1995)
5. Beattie, P.D., Bishop, J.M.: Self-Localisation in the SENARIO Autonomous Wheelchair. Journal of Intellingent and Robotic Systems **22** (1998) 255–267
6. Nasuto, S.J., Dautenhahn, K., Bishop, J.M.: Communication as an Emergent Methaphor for Neuronal Operation. Lect. Notes Art. Int. **1562** (1999) 365–380
7. De Meyer, K., Bishop, J.M., Nasuto S.J.: Attention through Self-Synchronisation in the Spiking Neuron Stochastic Diffusion Network. Consc. and Cogn. **9(2)** (2000)
8. Bishop, J.M., Nasuto, S.J., De Meyer, K.: Dynamic Knowledge Representation in Connectionist Systems. ICANN2002, Madrid, Spain (2002)
9. Nasuto, S.J., Bishop, J.M.: Convergence Analysis of Stochastic Diffusion Search. Parallel Algorithms and Applications **14:2** (1999) 89–107
10. Nasuto, S.J., Bishop, J.M., Lauria, S.: Time Complexity of Stochastic Diffusion Search. Neural Computation (NC'98), Vienna, Austria (1998)
11. Nasuto, S.J., Bishop, J.M.: Steady State Resource Allocation Analysis of the Stochastic Diffusion Search. Submitted (2002) `cs.AI/0202007`
12. Watts, D.J., Strogatz, S.H.: Collective Dynamics of 'Small-World' Networks. Nature **393** (1998) 440–442
13. Zanette, D. H.: Critical Behavior of Propagation on Small-World Networks. Physical Review E **64:5** (2001) 901–905
14. Kuperman, M., Abramson, G.: Small-World Effect in an Epidemiological Model. Physical Review Letters **86:13** (2001) 2909–2912
15. De Meyer, K.: Explorations in Stochastic Diffusion Search. Technical Report KDM/JMB/2000-1, University of Reading (2000)
16. Bishop, J.M.: Anarchic Techniques for Pattern Classification, Chapter 5. PhD Thesis, University of Reading (1989)
17. Moukarzel, C. F.: Spreading and Shortest Paths in Systems with Sparse Long-Range Connections. Physical Review E **60:6** (1999) R6263–R6266
18. Newman, M.E.J., Watts, D.J.: Scaling and Percolation in the Small-World Network Model. Physical Review E **60:6** (1999) 7332–7342

Bibliography

Aleksander, I. and Stonham, T. (1979), 'Guide to pattern recognition using random access memories', *Computers and Digital Techniques* **2**(1), 29–40.

Alonso, E., d'Inverno, M., Kudenko, D., Luck, M. and Noble, J. (2001), 'Learning in multi-agent systems', *Knowledge Engineering Review* **16**(3), 277–284.

Andrew, A. (1983), *Artificial Intelligence*, Abacus Press.

Ashby, R. (1956), *An Introduction to Cybernetics*, Chapman & Hall. Internet edition (1999): http://pcp.vub.ac.be/books/IntroCyb.pdf.

Barthélemy-Madaule, M. (1979), *Lamarck ou le Mythe du Précurseur*, Editions du Seuil, Paris.

Beattie, P. (2000), The Design and Implementation of a Focused Stochastic Diffusion Network to Solve the Self-Localisation Problem on an Autonomous Wheelchair, Phd, University of Reading.

Beattie, P. and Bishop, J. (1998), 'Self-localisation in the SENARIO autonomous wheelchair', *Journal of Intelligent and Robotic Systems* **22**, 255–267.

Beveridge, J. (1993), Local Search Algorithms for Geometric Object Recognition: Optimal Correspondence and Pose, Phd, University of Massachusetts.

Bishop, J. (1989*a*), Anarchic Techniques for Pattern Classification, Phd, University of Reading.

Bishop, J. (1989*b*), Stochastic searching networks, *in* '1st IEE Conf. ANNs', London, United Kingdom.

Bishop, J., Minchinton, P. and Mitchell, R. (1990), The Minchinton cell - analogue input to the n-tuple net, *in* 'INNC90', Paris, France.

Bishop, J. and Nasuto, S. (1999), Communicating neurons - an alternative connectionism., *in* 'WNNW99', York, United Kingdom.

Bishop, J., Nasuto, S. and De Meyer, K. (2002), 'Dynamic knowledge representation in connectionist systems', *Lecture Notes in Computer Science* **2415**, 308–313.

Bishop, J. and Torr, P. (1992), The stochastic search network, *in* R. Linggard, D. Myers and C. Nightingale, eds, 'Neural Networks for Images, Speech and Natural Language', Chapman & Hall.

Blackmore, S. (1998), *The Meme Machine*, Oxford University Press.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999), *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press.

Bonabeau, E., Theraulaz, G. and Deneuborg, J. (1998), 'Group and mass recruitment in ant colonies: the influence of contact rates', *Journal of Theoretical Biology* **195**, 157–166.

Burns, A. and Davies, G. (1993), *Concurrent Programming*, Addison-Wesley.

Campbell, D. (1974), Evolutionary epistemology, *in* P. Schilpp, ed., 'The Philosophy of Karl Popper', Open Court, pp. 413–463.

Campbell, D. (1997), 'From evolutionary epistemology over selection theory to a sociology of scientific validity', *Evolution and Cognition* **3**, 5–38.

Campbell, N. and Reece, J. (2002), *Biology*, 6th edn, Addison-Wesley.

Chadab, R. and Rettenmeyer, C. (1975), 'Mass recruitment by army ants', *Science* **188**, 1124–1125.

Christensen, S. and Oppacher, F. (2001), What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function, *in* L. Spector et al., ed., 'Genetic and Evolutionary Computation Conference', Morgan Kaufmann, San Fransisco, pp. 1219–1226.

Cowan, G. (1994), Conference opening remarks, *in* G. Cowan, D. Pines and D. Meltzer, eds, 'Complexity: Metaphors, Models, and Reality', Addison-Wesley.

Cruzan, M. (2001), Adaptive landscapes, *in* S. Brenner and J. Miller, eds, 'Encyclopedia of Genetics', Vol. 1, Academic Press.

Dawkins, R. (1989), *The Selfish Gene*, 2nd edn, Oxford University Press.

de Castro, L. and Timmis, J. (2002), Artificial immune systems: a novel paradigm to pattern recognition, *in* J. Corchado, L. Alonso and C. Fyfe, eds, 'Artificial Neural Networks in Pattern Recognition', University of Paisley, pp. 67–84.

de Castro, L. and Timmis, J. (2003), 'Artificial immune systems as a novel soft computing paradigm', *Soft Computing Journal* **7**(7).

De Meyer, K. (2000), Explorations in Stochastic Diffusion Search: Soft- and hardware implementations of biologically inspired spiking neuron stochastic diffusion networks, Technical Report KDM/JMB/2000-1, Department of Cybernetics, University of Reading.

De Meyer, K., Bishop, J. and Nasuto, S. (2000), 'Attention through self-synchronisation in the spiking neuron stochastic diffusion network', *Consciousness and Cognition* **9**(2), S81.

De Meyer, K., Bishop, J. and Nasuto, S. (2002), 'Small-world effects in lattice stochastic diffusion search', *Lecture Notes in Computer Science* **2415**, 147–152.

Deneuborg, J., Pasteels, J. and Verhaeghe, J. (1983), 'Probabilistic behaviour in ants: a strategy of errors ?', *Journal of Theoretical Biology* **105**, 259–271.

Dennett, D. (1995), *Darwin's Dangerous Idea: Evolution and the Meanings of Life*, Simon & Schuster.

Dreyfus, H. and Dreyfus, S. (1990), Making a mind versus modelling the brain: artificial intelligence back at a branch-point, *in* M. Boden, ed., 'The Philosophy of Artificial Intelligence', Oxford University Press, pp. 309–333.

Edelman, G. (1987), *Neural Darwinism: the Theory of Neuronal Group Selection*, Basic Books.

Eksioglu, D., Pardalos, P. and Resende, M. (2002), Parallel metaheuristics for combinatorial optimization, *in* R. Correa, I. Dutra, M. Fiallos and F. Gomes, eds, 'Models for Parallel and Distributed Computation', Kluwer Academic Publishers.

Ferber, J. (1999), *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley.

Fogel, L., Owens, A. and Walsh, M. (1966), *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons.

Giesecke, J. (2001), *Modern Infectious Disease Epidemiology*, Arnold.

Glover, F., Laguna, M. and Marti, R. (2003), Scatter search and path relinking: Advances and applications, *in* F. Glover and G. Kochenberger, eds, 'Handbook of Metaheuristics', Kluwer Academic Publishers.

Grech-Cini, H. (1995), Locating Facial Features, Phd, University of Reading.

Grech-Cini, H. and McKee, G. (1993), Locating the mouth region in images of human faces, *in* P. Schenker, ed., 'SPIE - The International Society for Optical Engineering, Sensor Fusion VI 2059', Massachusetts.

Hahn, S. (1997), Tuning monte carlo generator parameters to measured data by using genetic algorithms, *in* T. Baeck, D. Fogel and Z. Michalewicz, eds, 'Handbook of Evolutionary Computation', Oxford University Press.

Haykin, S. (1999), *Neural Networks: a Comprehensive Foundation*, second edn, Prentice Hall International.

Heims, S. (1991), *The Cybernetics Group*, MIT Press.

Heylighen, F. and Joslyn, C. (2001), Cybernetics and second-order cybernetics, *in* R. Meyers, ed., 'Encyclopedia of Physical Science & Technology', 3rd edn, Academic Press.

Hinton, G. (1981), A parallel computation that assigns object-based frames of reference, *in* '7th Int. Joint Conf. on Artificial Intelligence', Vancouver, Canada, pp. 683–685.

Holland, J. (1992), *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, 2nd edn, MIT Press.

Hölldobler, B. and Wilson, E. (1990), *The Ants*, Springer-Verlag.

Hoos, H. (1999), On the run-time behaviour of stochastic local search algorithms for sat, *in* 'AAAI-99', MIT Press, pp. 661–666.

Hull, D. (1980), 'Individuality and selection', *Annual Review of Ecology and Systematics* **11**, 311–332.

Hull, D. (1988), *Science as a Process: An Evolutionary Account of the Social and Conceptual Development of Science*, University of Chicago Press.

Hurley, S. and Whitaker, R. (2002), An agent based approach to site selection for wireless networks, *in* '2002 ACM symposium on Applied Computing', ACM Press, Madrid, Spain.

Jablonka, E. (2002), Between development and evolution: How to model cultural change, *in* M. Wheeler, J. Ziman and M. Boden, eds, 'The Evolution of Cultural Entities', Oxford University Press, pp. 27–43.

Jones, T. (1995), Evolutionary Algorithms, Fitness Landscapes and Search, Phd, University of New Mexico.

Jones, T. and Forrest, S. (1995), Fitness distance correlation as a measure of problem difficulty for genetic algorithms, *in* L. Eshelman, ed., 'Sixth International Conference on Genetic Algorithms', Morgan Kaufmann, pp. 184–192.

Kearns, M. and Seung, H. S. (1995), 'Learning from a population of hypotheses', *Machine Learning* **18**(2-3), 255–276.

Kennedy, J., Eberhart, R. and Shi, Y. (2001), *Swarm Intelligence*, Morgan Kaufmann.

Kirkpatrick, S., Gelatt, C. and Vecchi, M. (1983), 'Optimization by simulated annealing', *Science* **220**(4598), 671–680.

Knuth, D. (1973), *The Art of Computer Programming. Volume 3: Searching and Sorting*, Addison-Wesley.

Lewin, R. (1993), *Complexity: Life at the Edge of Chaos*, Macmillan.

Looijen, R. (1999), *Holism and Reductionism in Biology and Ecology: the Mutual Dependence of Higher and Lower Level Research Programmes*, Episteme, Kluwer Academic Publishers.

Luger, G. and Stubblefield, W. (1997), *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison Wesley Longman.

Macready, W. and Wolpert, D. (1996), 'What makes an optimization problem hard?', *Complexity* **5**, 40–46.

Magee, B. (1973), *Popper*, Fontana Press.

Mallon, E., Pratt, S. and Franks, N. (2001), 'Individual and collective decision-making during nest site selection by the ant leptothorax albipennis', *Behavioral Ecology and Sociobiology* **50**, 352–359.

Michelsen, A., Andersen, B., Kirchner, W. and Lindauer, M. (1991), The dance language of honeybees - new findings, new perspectives, *in* L. Goodman and R. Fischer, eds, 'The Behavior and Physiology of Bees', C.A.B. International.

Midgley, M. (2002), Choosing the selectors, *in* M. Wheeler, J. Ziman and M. Boden, eds, 'The Evolution of Cultural Entities', Oxford University Press, pp. 119–135.

Mitchell, M. (1996), *An Introduction to Genetic Algorithms*, MIT Press.

Morey, T., De Meyer, K., Nasuto, S. and Bishop, J. (2000), 'Implementation of the spiking neuron stochastic diffusion network on parallel hardware', *Consciousness and Cognition* **9**(2), S97.

Moscato, P. (1989), On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Technical C3P Report 826, CalTech.

Nair, D. and Wenzel, L. (1999), Image processing and low-discrepancy sequences, *in* 'Advanced Signal Processing Algorithms, Architectures, and Implementations IX', Proceedings of SPIE, Vol. 3807, pp. 102–111.

Nakamura, A., Takeuchi, J. and Abe, N. (1998), 'Efficient distribution-free population learning of simple concepts', *Annals of Mathematics and Artificial Intelligence* **23**(1-2), 53–82.

Nasuto, S. (1999), Resource Allocation Analysis of the Stochastic Diffusion Search, Phd, University of Reading.

Nasuto, S. and Bishop, J. (1998), Neural stochastic diffusion search network - a theoretical solution to the binding problem, *in* 'ASSC2', Bremen, Germany, p. 19.

Nasuto, S. and Bishop, J. (1999), 'Convergence analysis of stochastic diffusion search', *Parallel Algorithms and Applications* **14**(2), 89–107.

Nasuto, S. and Bishop, J. (2002), 'Steady state resource allocation analysis of the stochastic diffusion search'. (submitted) (cs.AI/0202007).

Nasuto, S., Bishop, J. and Lauria, S. (1998), Time complexity of stochastic diffusion search, *in* 'International ICSC/IFAC Symposium on Neural Computation (NC'98)', Vienna, Austria.

Nasuto, S., Dautenhahn, K. and Bishop, J. (1999), 'Communication as an emergent methaphor for neuronal operation', *Lecture Notes in Artificial Intelligence* **1562**, 365–380.

Nelson, R. (2002), Evolutionary theorising in economics, *in* M. Wheeler, J. Ziman and M. Boden, eds, 'The Evolution of Cultural Entities', Oxford University Press, pp. 135–144.

Newell, A. and Simon, H. (1976), 'Computer science as empirical enquiry: Symbols and search', *Communications of the ACM* **19**(3), 113–126.

Nilsson, N. (1998), *Artificial Intelligence: a New Synthesis*, Morgan Kaufmann.

Ochoa-Rodriguez, A. (1997), Partial evaluation of genetic algorithms, *in* '1st Symposium on Artificial Intelligence, CIMAF'97', Havana, Cuba, pp. 29–36.

Papadimitriou, C. and Steiglitz, K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall.

Pask, G. (1961), *An Approach to Cybernetics*, Hutchinson.

Perelson, A. and Oster, G. (1979), 'Theoretical studies on clonal selection: Minimal antibody repertoire size and reliability of self-nonself discrimination', *Journal of Theoretical Biology* **81**, 645–670.

Plotkin, H. (1994), *The Nature of Knowledge: Concerning Adaptations, Instinct and the Evolution of Intelligence*, Penguin Press.

Plotkin, H. (2002), Learning from culture, *in* M. Wheeler, J. Ziman and M. Boden, eds, 'The Evolution of Cultural Entities', Oxford University Press, pp. 103–118.

Popper, K. (1966), *Of Clouds and Clocks: an Approach to the Problem of Rationality and the Freedom of Man*, Washington University. Reprinted in "Objective Knowledge: an Evolutionary Approach". Oxford University Press (1972).

Popper, K. (1972), Evolution and the tree of knowledge, *in* 'Objective Knowledge: an Evolutionary Approach', Oxford University Press.

Prasse, M. and Rittgen, P. (1998), 'Why Church's thesis still holds. some notes on Peter Wegner's tracts on interaction and computability', *Computer Journal* **41**(6), 357–362.

Pratt, S., Mallon, E., Sumpter, D. and Franks, N. (2002), 'Quorum sensing, recruitment and collective decision-making during colony emigration by the ant leptothorax albipennis', *Behavioral Ecology and Sociobiology* **52**, 117–127.

Resende, M. and Ribeiro, C. (2003), Greedy randomized adaptive search procedures, *in* F. Glover and G. Kochenberger, eds, 'Handbook of Metaheuristics', Kluwer Academic Publishers.

Ridley, M. (1994), *The Red Queen: Sex & the Evolution of Human Nature*, Penguin.

Roitt, I., Brostoff, J. and Male, D., eds (2001), *Immunology*, 6th edn, Mosby.

Roli, A. and Milano, M. (2002), Magma: a multiagent architecture for metaheuristics, Technical Report DEIS-LIA-02-007, Universita degli Studi di Bologna.

Rosenblueth, A., Wiener, N. and Bigelow, J. (1943), 'Behavior, purpose and teleology', *Philosophy of Science* **10**(1), 18–24.

Rosenfeld, A. and Vanderburg, G. J. (1977), 'Coarse-fine template matching', *IEEE Transactions on Systems, Man and Cybernetics* **7**, 104–107.

Rumelhart, D. and McClelland, J. (1987), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition Vol. 1: Foundations*, MIT Press.

Runciman, W. (2002), Heritable variation and competitive selection as the mechanism of sociocultural evolution, *in* M. Wheeler, J. Ziman and M. Boden, eds, 'The Evolution of Cultural Entities', Oxford University Press, pp. 9–26.

Searle, J. (1990), Minds, brains, and programs, *in* M. Boden, ed., 'The Philosophy of Artificial Intelligence', Oxford University Press, pp. 67–87.

Seeley, T. (1995), *The Wisdom of the Hive - The Social Physiology of Honeybee Colonies*, Harvard University Press.

Seeley, T. and Buhrman, S. (1999), 'Group decision making in swarms of honeybees', *Behavioral Ecology and Sociobiology* **45**, 19–31.

Seeley, T. and Buhrman, S. (2001), 'Nest-site selection in honeybees: How well do swarms implement the "best-of-n" decision rule ?', *Behavioral Ecology and Sociobiology* **49**, 416–427.

Sharpe, O. (1999), Continuing beyond nfl: Dissecting real-world problems, *in* L. e. al., ed., 'Genetic and Evolutionary Computation Conference', Morgan Kaufmann, Orlando.

Shweder, R. (2001), A polytheistic conception of the sciences and the virtues of deep variety, *in* A. Damasio, A. Harrington, J. Kagan, B. McEwen, H. Moss and R. Shaikh, eds, 'Unity of Knowledge: The Convergence of Natural and Human Science', The New York Academy of Sciences.

Stein, L. A. (1999), 'Challenging the computational metaphor: Implications for how we think', *Cybernetics and Systems* **30**(6), 473–507.

Stolley, P. and Lasky, T. (1995), *Investigating Disease Patterns: the Science of Epidemiology*, Scientific American Library, New York.

Summers, R. (1998), Stochastic Diffusion Search: A Basis for a Model of Visual Attention?, Msc, Keele University.

Taillard, E., Gambardella, L., Gendreau, M. and Potvin, J. (2001), 'Adaptive memory programming: a unified view of metaheuristics', *European Journal of Operational Research* **135**, 1–16.

Turchin, V. (1977), *The Phenomenon of Science*, Columbia University Press. Internet edition (2000): http://pespmc1.vub.ac.be/POSBOOK.html.

Turing, A. (1936), 'On computable numbers, with an application to the entscheidungsproblem', *Proceedings of the London Mathematical Society* **2**(42), 128–157.

Visscher, P. and Camazine, S. (1999), 'Collective decisions and cognition in bees', *Nature* **397**, 400.

von Bertalanffy, L. (1950), 'An outline of general system theory', *The British Journal for the Philosophy of Science* **1**(2), 134–165.

von Foerster, H. (1995), 'Ethics and second-order cybernetics', *Stanford Humanities Review* **4**(2).

von Neumann, J. (1945), First draft of a report on the EDVAC, Technical report, University of Pennsylvania. Online at http://www.wps.com/projects/EDVAC/.

Waldrop, M. (1992), *Complexity: The Emerging Science at the Edge of Order and Chaos*, Simon & Schuster.

Watts, D. J. and Strogatz, S. H. (1998), 'Collective dynamics of 'small-world' networks', *Nature* **393**(6684), 440–442.

Weiss, G., ed. (1999), *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*, The MIT Press.

Wheeler, M., Ziman, J. and Boden, M., eds (2002), *The Evolution of Cultural Entities*, Proceedings of the British Academy - 112, Oxford University Press.

Wiener, N. (1961), *Cybernetics or Control and Communication in the Animal and the Machine*, 2nd edn, MIT Press.

Wilson, E. (1971), *The Insect Societies*, Harvard University Press.

Wilson, E. (1998), *Consilience: the Unity of Knowledge*, Knopf.

Wilson, E. (2001), How to unify knowledge, *in* A. Damasio, A. Harrington, J. Kagan, B. McEwen, H. Moss and R. Shaikh, eds, 'Unity of Knowledge: The Convergence of Natural and Human Science', The New York Academy of Sciences.

Winston, P. (1992), *Artificial Intelligence*, 3rd ed edn, Addison-Wesley.

Wolpert, D. and Macready, W. (1997), 'No free lunch theorems for optimization.', *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Zimmer, C. (2002), *Evolution: the Triumph of an Idea*, Heinemann.