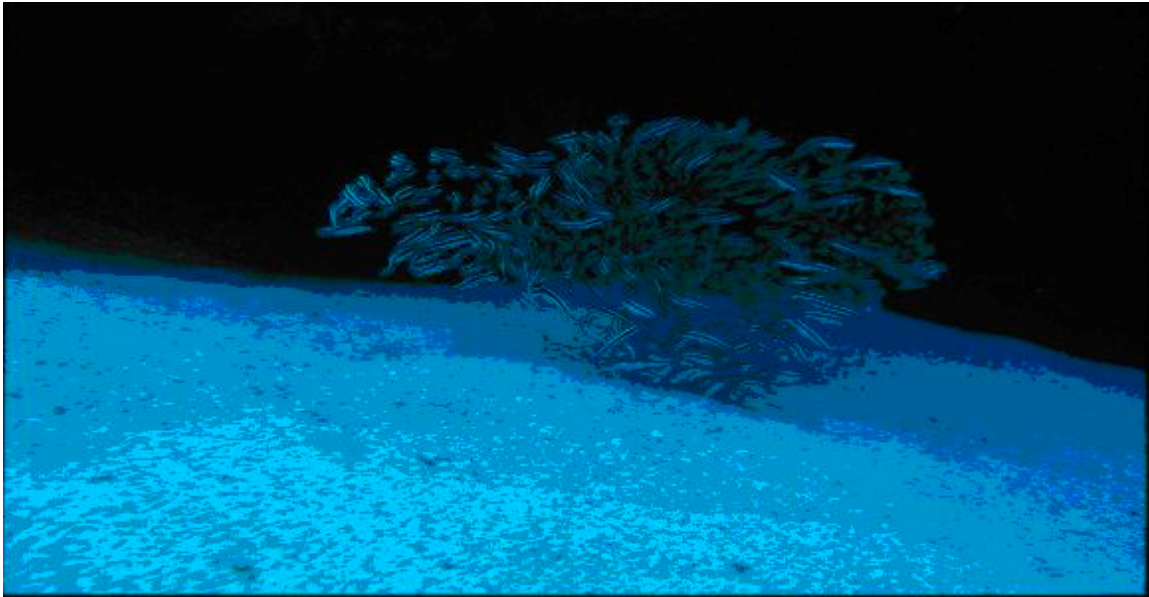


Symbiotic Algorithms

Master Thesis

Kees Pieters



Abstract:

In Nature, many successful strategies have evolved to effectively deal with complexity. Any organism faces the challenge to survive in its, often unpredictable, environment, aided by only those tools that have proved to be effective enough to survive evolutionary selection. Often however, the environment is too complex to deal with alone and so support is needed from other organisms. In its extreme form, this has resulted in *symbiosis*, a form of co-operation between different species that has actually evolved to become instinct.

It is still unclear how two organisms can evolve from independent beings into partners in such a close-knit relationship, but it is a biological fact that this behaviour can be identified in very simple systems.

This essay aims to provide a model that demonstrates such behaviour in a very basic form and investigate how networks of such symbionts behave. This attained knowledge is used to see how symbiosis could be applied to solve a well-known np-complete problem.

© copyright 2003 C. P. Pieters

2003-03-17,
Open University the Netherlands
Technical Sciences
Computer Systems
Version 5

Symbiotic Algorithms

1	Introduction	6
1.1	SYMBIOSIS IN A BIOLOGICAL CONTEXT	7
1.2	A THEORY OF SYMBIOSIS.....	8
1.3	A MODEL OF SYMBIOSIS.....	8
1.4	SOLVING KNOWN PROBLEMS WITH SYMBIOTS	9
2	Evolution Theory	10
	NEO-DARWINISM.....	10
	NEO-DARWINISM AND ENGINEERING.....	10
	EVOLUTION	12
	ENGINEERING SYMBIOSIS	13
	SUMMARY	14
3	Agents, Systems and Environments	15
	INTRODUCTION	15
	RATIONAL AGENTS	16
	SYMBIOTIC RATIONAL AGENTS	16
	AGENTS AND THE ENVIRONMENT	16
	AN EXAMPLE: ENGINEERING A CELL.....	22
	SUMMARY	24
4	Modelling Symbiosis	25
	INTRODUCTION	25
	SYMBIOTS	25
	A BASIC SYMBIOTIC NETWORK	26
	CONVERGENCE	29
	A SIMPLE EXAMPLE: A BAKERY AND ITS CUSTOMERS.....	31
	ANOTHER EXAMPLE: ARTIFICIAL AMOEBA	34
	SUMMARY	37
5	More on Algorithms	38
	INTRODUCTION	38
	THE SOFTWARE ARCHITECTURE	39
	ARCHITECTURE.....	40
	MORE ON ALGORITHMS AND ENVIRONMENTS	41
	THE GOALSEEKER EXAMPLE.....	43
	SUMMARY	48
6	Symbiotic Networks as Problem Solvers.....	49
	PROBLEMS AND SOLUTIONS.....	49
	CATEGORISING SYMBIOTIC NETWORKS	51
	USING SYMBIOSIS FOR SOLVING PROBLEMS	53
	SUMMARY	56
7	The Travelling Salesman Problem	57
	INTRODUCTION	57
	1. THE SOLUTION STRATEGY	58
	AN EXAMPLE RUN.....	60
	2. DEFINING THE SYMBIOTS.....	62
	1: Determining the goal.....	62
	2,3: Defining output and input.....	62
	4: Analysing the complexity of the environment	62
	5: Determining the relationship between out- and input through the environment	62
	6,7: Identifying amplifications and conflicting goals.....	62
	8: Defining the symbiotic algorithms.....	63
	9: Wrapping up.....	63
	A SIMPLE CONVERGING STRATEGY (TSP).....	64
	INCLUDING PROBABILITY (WEIGHTED)	65
	AN OPTIMISING STRATEGY (NEURON)	68
	RESULTS	70
8	Epilogue.....	72
	BACK TO NATURE.....	73
9	Conclusion	74

Front Page, header and footer photos

Striped catfish flock so closely together when threatened, that they appear to become one big organism. (Photo: Kees Pieters, Great Barrier reef 2000)

Summary

This document is the record of the author's master thesis in computer science for the Open University, the Netherlands. The subject is *symbiosis*, and the research aims to investigate if the advantages of symbiosis, which are well recognised in biology, can be applied in Information Technology, especially in solving complex problems. As no conclusive theory has been formed yet that explains which mechanisms drive such an extreme form of co-operation between different species, a formal model is presented that can clarify symbiotic behaviour. Networks of these so-called symbiots are investigated and a convergence proof is derived to show that such networks can converge to certain goal, regardless of the environment in which the symbiots reside. A few limitations imposed on this convergence are also identified and examined. Based on the results, a method is given on how to create an effective symbiotic network in an unknown or complex environment. This method is applied to give a symbiotic solution strategy for the Travelling Salesman Problem. This also aims to demonstrate how neural networks may have evolved from symbiotic networks.

Samenvatting

Dit is het verslag van een afstudeeropdracht in computersystemen bij de Open Universiteit Nederland. Het onderwerp is *symbiose*, met het doel om te onderzoeken of deze extreme vorm van samenwerking, waarvan de voordelen al lang onderkend zijn in de biologie, is toe te passen in de Informatie Technologie, met name bij het oplossen van complexe problemen. Omdat er vooralsnog geen sluitende theorie is die verklaart welke sturende mechanismen schuil gaan achter symbiose, wordt een formeel model gepresenteerd waarmee symbiotisch gedrag verklaard kan worden. Met behulp van dit model wordt vervolgens een netwerk van dergelijke zogenaamde symbiots bestudeerd en wordt een bewijs van convergentie van deze netwerken afgeleid. Ook worden een aantal beperkingen die door deze convergentie opgelegd worden, nader onderzocht. Dit leidt uiteindelijk tot een methode waarmee nagegaan kan worden in hoeverre een symbiotisch netwerk een bepaald doel kan bereiken in een onbekende of complexe omgeving. Deze methode wordt vervolgens toegepast om een symbiotische oplossingstrategie voor het Handelsreizigers Probleem te demonstreren. Daarmee wordt tevens getracht te laten zien hoe neurale netwerken uit symbiotische netwerken kunnen evolueren.

Preface



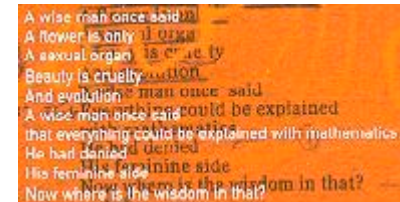
Allard's anemonefish, *Amphiprion allardi*, 10 cm
(E. Africa).

The anemone fish is a classic example of symbiosis. The anemone fish has evolved to have a layer over its skin to protect it from the anemones poisonous tentacles. The anemone provides protection and food, while the anemone fish is a fierce defender against harmful animals [Allen et. al 1999]

If there is any morale of symbiosis, it is that nothing is done in isolation. Even though the research and writing of this thesis was carried out as a one-man mission, it would not have reached its current form if it had not been for the many people around me that gave their support and enthusiasm in the past few years.

1 Introduction

Computing and Information Sciences have always had an interest in research carried out in other disciplines, in order to learn and understand mechanisms that might directly or indirectly apply to their own area of research as well.



"This is the 21st Century", Anoraknophobia, Marillion

Especially the increasing connective structure of computers and networks has led to an understanding that a complex artificial environment is emerging where various similarities with Nature can be identified. Nowadays numerous research areas try to understand "natural" phenomena such as evolution, social structures and behaviour, in order to utilise its "good tricks" [Dennet 1995]. This approach has helped to unravel existing biological structures and mechanisms, such as the nervous system, language and speech. Their findings have been applied in computers and resulted in neural networks, genetic algorithms and numerous strategies to solve specific problems. In turn, computing science has provided many new insights for the research areas that spurred these ideas in the first place. In fact, a number of interdisciplinary research institutes have been founded, where biology, physics, psychology, sociology, philosophy and computer science work side-by-side to understand the mechanisms that Nature has so readily made available. Therefore it is often very difficult to draw clear-cut lines between these disciplines, as they freely borrow each other's work and often find their own research being referred to in, what seemed to be non-related research areas.

Symbiosis is a subject that isn't clearly related to one discipline. In its most elementary form, it is strongly embedded in Biology, where it has been identified and extensively studied. However, when symbiosis is considered as a form of co-operation, it becomes interesting for Sociology, Economics, Philosophy and other social sciences as well.

And not only social sciences, as current research on agent technology, artificial life, distributed autonomous systems and other computer-related areas are based on co-operative structures. In the last decades, these artificial structures have evolved into extremely complex artificial environments and so there is certain logic to look if such a successful evolutionary strategy as symbiosis could be applied in this area as well. However, currently there is no conclusive theory about the mechanisms that enable symbiotic behaviour.

Imagine two organisms trying to survive in a hostile, unpredictable environment. By chance they find each other, become aware of each other's presence and suddenly find out that surviving is becoming easier. In time, this awareness becomes engraved in the organism's genes; it becomes instinct. The organisms converge to a new stable situation in which a symbiotic relationship is established.

The research that is carried out for this thesis tries to find the minimal requirements needed to achieve such a convergence in co-operating software entities. By formalising these requirements, an attempt is made to present a model of symbiosis that could be applied to solve problems in an artificial environment. The lack of a theory of symbiosis however, forces us to present one first, based on what we know about symbiosis in Biology. This theory needs to be verified and be translated in a form that allows the theory to be modelled for use in a software environment. The following step is to see which problems could be solved effectively with this model and to demonstrate this. Summarised, this leads to the following approach:

Symbiotic Algorithms

1. Investigate symbiosis in a known Biological context, identify its function in Nature and define the minimal requirements that are needed to create symbiotic behaviour between organisms
2. Present a theory of symbiotic behaviour and verify this theory against the biological context presented earlier
3. Translate the theory to a model that allows symbiosis to be used in environments other than a Natural one
4. Investigate which classes of problems could be solved using this model and compare its use against other, known solution strategies.
5. Demonstrate the use of the model in solving such problems

Each of these steps will be described in detail in the next sections.

1.1 Symbiosis in a Biological Context

“Symbiosis is a relationship in which two dissimilar organisms live in close association with each other” [Academic Press 1996]

(www.academicpress.com/inscight/05071997/symbios2.htm)

This definition, found on the Internet, is one of many but covers the general view on this subject well. Some definitions allow more than two organisms, other definitions also allow similar organisms to engage in symbiosis. The latter extension tends to a rather broad definition of symbiosis, which may not fully agree with more formally inclined Biologists. However, this broad definition is used in this essay as it allows a very generic view on symbiosis. The special case of symbiosis between dissimilar organisms will be kept in mind at all times.

“Symbiosis is a relationship in which two or more organisms live in close association with each other”

In order to understand how a phenomenon as symbiosis can have manifested itself, it is crucial to embed it in a general theory of Nature. *Evolution Theory* provides such a framework, more specifically the *neo-Darwinian* viewpoint, which takes evolution to its extreme form and allows no force of creation to describe natural phenomena. Although this essay does not aim to fuel the controversy between neo-Darwinists and Creationists, we will take the neo-Darwinian viewpoint here, because it provides a formal approach to investigate symbiosis.

The most important supplements of neo-Darwinian Evolution Theory to symbiosis are:

- ◆ Awareness that every successful mechanism in Nature can be described in terms of *benefit* for at least one of its participants. This will allow identification of the benefits or advantages of symbiosis.
- ◆ Implicit awareness of the intricate relationship between (successful) organisms and the environment they live in.
- ◆ Neo-Darwinian reasoning encourages an *engineering view* [Dennet, 1995] on evolution and therefore on symbiosis.

Symbiotic Algorithms

These additions limit the scope of the available natural resources that *could* be used to model symbiosis. A scrutiny of current literature on Evolution Theory will help to set up a time scale of evolution and pinpoint symbiosis somewhere on this time scale. This time scale can also be regarded as a scale of *increasing complexity*, which will help to discard a great deal of complex natural phenomena, such as genetic reproduction, neurones (brains) and so on as possible explanations for symbiosis.

1.2 A Theory of Symbiosis

Based on the Biological framework, a theory of natural symbiosis will be set up. To achieve this, the *environment* of organisms needs to be examined as closely as the organisms themselves. This approach is very similar to *Phuelian Reactive Systems* [Wooldridge, 2000], a specialisation of Systems Theory that is base for research in so-called *agent-based systems*. Adapting certain aspects of this view will add the following contributions to symbiosis:

- ◆ It provides a generic modelling tool that allows biological systems to be described in a formal fashion
- ◆ It allows a generally accepted redefinition of an environment to a *problem domain* (figure 1)

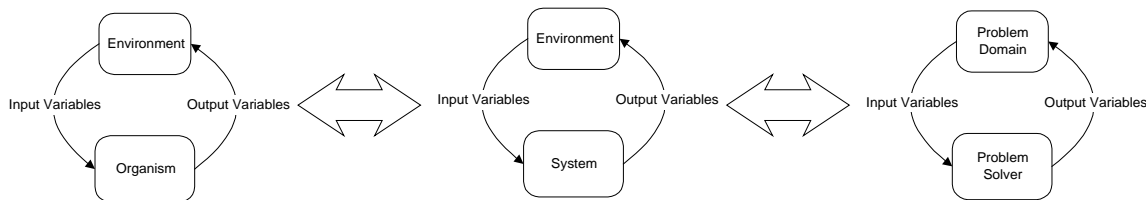


Figure 1 : Redefinition of Biological Systems to a Problem Domain

1.3 A Model of Symbiosis

With the preparation completed, we can now make a model of a symbiotic entity, a formal entity that is able to engage in *symbiotic behaviour*. This means that it should be able to adapt its behaviour for the benefit of others as well as provide information about its own state to others so they can support or help it. In order to come to the simplest symbiotic system imaginable, a basic entity with a *primary transformation process* that transforms an input variable to an output variable is extended to perform symbiotic behaviour. To achieve this, the entity will get a *goal function* that is base for the “benefit” that evolution theory requires.

An informal description of symbiotic behaviour will be given for two of these symbiotic entities (symbiots), to give an intuitive impression of how symbiosis will take place. Then a *convergence proof* for an arbitrary amount of symbiots in a symbiotic network will be derived to show that:

- ◆ The input vector \underline{l} of the combined inputs of each symbiots can converge to a goal vector \underline{g} of the combined goals of each symbiant.
- ◆ This convergence depends on a function that determines how the behaviour of a symbiot is influenced by the information that other symbiots pass on. This function is called a *symbiotic algorithm*.
- ◆ Specific knowledge about the problem domain is not required to reach this convergence, the information that the entities pass should be sufficient

The convergence proof should identify which class of algorithms will actually converge and what the possibilities and limitations are of such a network of symbiots.

Symbiotic Algorithms

In summary, we now have a network of symbiotic entities that each can work together to reach a certain goal in a problem domain or environment. In order to test the theory, a test environment will be programmed in JAVA that is used to test the theoretical results.

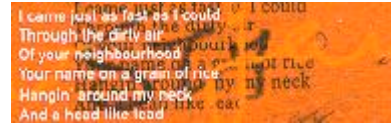
1.4 Solving Known Problems with Symbiots

To really get a grasp of the possibilities and limitations of symbiotic networks, such a network will be used to solve a known problem. This research has taken the Travelling Salesman Problem as the benchmark problem. This part of the research aims to investigate the following:

- ◆ Verification of the claims about symbiotic networks that were made earlier
- ◆ Gain experience with using symbiotic networks as a problem solving mechanism
- ◆ Compare the solution with known solution strategies for a given problem

2 Evolution Theory

Neo-Darwinism



"This is the 21st Century", Anoraknophobia, Marillion

It is evident that symbiosis is strongly embedded in Biology. More specifically, the subject has a strong significance for evolution theory or, more popularly, Darwinism. Symbiosis has been first coined in biological systems and has been extensively studied. More so, it is now generally acknowledged that symbiosis is a driving force behind the evolution of living organisms [Margulis, 1999]. Evolution theory however covers an extensive area and many topics are subject to debate. In order to use evolution theory to come to a formalism of symbiosis, we need to limit the scope dramatically. This essay therefore closely adheres to so-called *neo-Darwinian* theory, and specifically the ideas expressed by Daniel Dennet [Dennet 1995], one of the most influential spokesmen of this branch of evolution theory

So, exactly what *is* neo-Darwinism? Well, putting it shortly, neo-Darwinians uphold an extreme view on Charles Darwin's original ideas and claim that there is no driving force of creation behind evolution except evolution itself. Adapting this view allows us to examine symbiosis within the mechanisms and principles of evolution theory, which allows an *engineering perspective* on this subject. Although this may seem an almost trivial observation, it is subject to serious debate in neo-Darwinian research [Dennet 1995]. An engineering perspective may help bridge the gap between the often *empirical* contributions to the research on Evolution Theory so far, and the formal approach that other disciplines require.

Before investigating symbiosis from a neo-Darwinian perspective, we will therefore first need to identify what the "engineering perspective" on neo-Darwinism implies.

Neo-Darwinism and Engineering

Dennet [1995] has argued that neo-Darwinian research would benefit greatly if it was regarded from an engineering perspective. In order to appreciate this claim, we first need to know what "engineering" is. According to Bruegge & Allen [2000], engineering is a:

- ◆ Modelling activity. Engineers deal with complexity through modelling.
- ◆ Problem-solving activity. Engineering is aimed to provide solutions for a problem
- ◆ Knowledge-acquisition activity. Data about the problem and solution domain is organised into information and formalised into knowledge.
- ◆ Rationale driven activity. The context of decision-making and the rationale behind these decisions are just as important as the knowledge they produce.

Symbiotic Algorithms

The tools that an engineer can use are:

- ◆ Principles and Mechanisms. Principles are rules that are considered “true” within a certain domain. For instance, Physics and Mathematics provide collections of such principles. Mechanisms are a composition of principles aimed to solve a specific problem or to describe occurring phenomena
- ◆ Notations; a graphical or textual set of rules for representing a model
- ◆ Methods; repeatable techniques for solving a specific problem and are closely related to mechanisms
- ◆ Methodologies; a collection of methods for solving a class of problems

The rationale of an engineering activity is captured by the following properties:

- ◆ Goals; high-level principles that guide the engineering activities
- ◆ Requirements; features that should be present in the result of the engineering activities
- ◆ Constraints; limitations that can be identified in the engineering activities

Lastly, the engineering activities itself are subject to planning and control:

- ◆ Activities are a set of tasks that are performed towards a specific purpose
- ◆ Phases are activities termed as units of time
- ◆ Work products are artefacts produced during the engineering activities
- ◆ Deliverables are work products that are presented outside the scope of the engineering activities
- ◆ Milestones are important points in time during the engineering activities. They are usually deliverables termed as units of time

One of the “rules-of-thumb” for any engineer is that one should always opt for *generic* and *simple* solutions for a given problem if this is possible. Both allow a presented solution to the problem to be used for a large set of similar problems, and often improve the robustness of the solution as well. Given a number of solutions to a specific problem, generality and simplicity are often the decisive criteria for the eventual choice and are the cornerstones for good engineering practice.

Although engineering encompasses much more than the features mentioned above, this selection provides a sufficient framework to “engineer” evolution.

The first problem we face when viewing evolution through the eyes of an engineer is that we in fact have to engage in a *reverse engineering* activity. Evolution is a process that has progressed through millions of years and, along the way, has produced an impressive set of deliverables and milestones. The engineer faces the task to reconstruct this process based on the deliverables and milestones that can be identified. This approach limits the solution space that the regular problem solving activity of engineering normally is. For instance, we can not freely choose from available mechanisms, as some of them may not have been available *at some point* of evolution. An anemone fish can not have evolved before anemones!

Another problem is that Evolution has to be regarded as an engineering activity, but as it is the result of a process, we will have to define in which ways evolution fulfils the requirements of engineering:

- ◆ Nature is a massive model of evolution’s traits. Nature therefore is the main source of information about evolution and (therefore) it also is the main source for knowledge-acquisition
- ◆ Although evolution is not a problem-solving activity in the sense that we normally use it, we could consider evolution as a process that delivers

Symbiotic Algorithms

enduring systems. The problem-solving activities of evolution are to overcome instability and decay. According to Dawkins [1984], evolution is a process that delivers enduring information through genes. The main problem-solving activities of evolution would be to optimise storage and transfer of information in the agents (organisms) that carry them. An *agent* in this sense is a carrier of information.

- ◆ The rationale of evolution is another matter that cannot be termed in the usual way. Based on the deliverables of evolution we termed previously, we could add that the rationale of evolution is to deliver enduring systems of *increased complexity*. The definition of Dawkins [1984] can help to overcome lengthy philosophical discussions about exactly *what* complexity is (is a dinosaur more or less complex than a Human being?). Complexity can be easily defined as the *amount of information that is contained in an agent*.

Evolution

Based on the engineering qualities we defined previously, we can now determine the main goal of evolution:

“Evolution is the process of creating enduring repositories of information of increasing complexity through agents”

In biological evolution, agents could be replaced by organisms, but the definition we present here is aimed to cover other kinds of evolution as well.

An effect of this evolution process is, that the tools that are at evolution’s disposal become more complex as well. Agents become more complex as do the principles and mechanisms. This means that the *constraints* and *requirements* of evolution are determined by time and that the process of evolution has to be modelled in time. A time scale of evolutionary principles and mechanisms may help to limit the scope of possible solutions.

Although a time scale of evolution has not been made yet, there is ample literature [Margulis 1999, Dawkins 1984, Dennet 1995, Pinker 1997, Kauffman 2000] that can help to draw such a model. Figure 2 depicts an attempt to draw such a time scale of evolution, based on the referred literature. Although the “invention” of the brain could be considered a Human-centred view on evolution, it is a generally accepted one [Pinker 1997] and therefore used here as well.

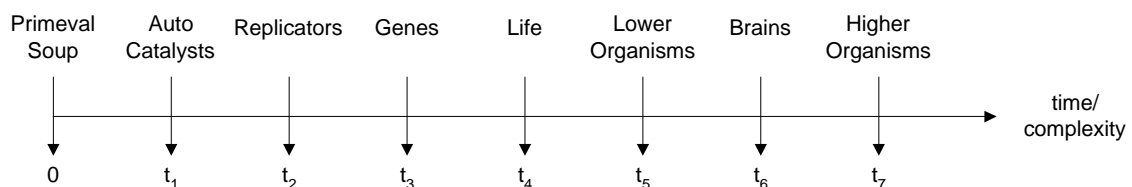


Figure 2: Time Scale of Evolution

Symbiotic Algorithms

Natural evolution started out with the *primeval soup* [Dawkins 1984, Dennet 1995, Pinker 1997] that was reigned by physics and chemistry. In this phase of evolution molecules of increasing complexity evolved until the first *auto catalysts* [Kauffman 2000] emerged, complex molecules that were able to speed up their creation by using catalysts. Catalysts are chemicals that are not involved in a chemical reaction but do help in making this reaction more efficient. The next phase of evolution is demarcated by the creation of *replicators* [Dawkins 1984, Dennet 1995]. Replicators are complex molecules able to create copies of themselves. Replicators evolved into increasingly complex structures based on amino acids and eventually into *genes* [Dawkins 1984, Dennet 1995, Pinker 1997]. Genes were not only able to replicate, but also able to drive (auto-catalyst) reactions in their immediate surroundings. Some of these structures eventually merged into a symbiotic relationship [Margulis 1999], which resulted in *life*. From life, the first one-celled and multi-cell organisms evolved [Margulis 1999, Dawkins 1984, Dennet 1995, and Pinker 1997]. *Neurones* triggered the birth of the *brain* as central steering mechanism of the agents (organisms) that carry genes [Pinker 1997] and this eventually resulted in higher organisms, such as *Homo Sapiens*, that is: us humans.

In neo-Darwinism, every natural phenomenon that is investigated has to be put somewhere in the timeline between the primeval soup and life-as-we-know-it. Evolution Theory however still strongly depends on the contribution of Biology, and so the timeline is often started at replicator level or preferably, at the birth of genes. The Natural phenomenon we call “symbiosis” is no exception to this rule. There are quite a number of scholars that try to come to a “Symbiosis Theory”, based on a genetic approach [Watson & Pollack 1999, Moriarty & Miikulainen 1994], but so far with little success.

To summarise the above, neo-Darwinian evolution theory contributes the following to our understanding of symbiosis:

- ◆ Evolution theory adds a requirement of *benefit* to our investigation, as evolution is all about survival and optimisations. Symbiosis as mechanism will not last in Nature if at least one of the involved organisms would not benefit from it.
- ◆ Symbiosis can be pinpointed on a time scale of evolutionary mechanisms and principles
- ◆ No conclusive theory on symbiosis is available

Engineering Symbiosis

Based on the time scale presented in the previous paragraph we can try to determine where symbiosis emerged as a mechanism that could be used by evolution. Based on the definition coined in chapter one and adapted to the line of reasoning that has been conducted so far, we can state that:

“Symbiosis is a relationship in which two or more agents live in close association with each other”

“Close association” is a term that needs a more precise definition. From neo-Darwinian reasoning, we know that *benefit* should be the cause of this interdependency. To be more specific, at least one agent involved in symbiosis appears to provide some service that benefits the others. Often this works both ways, causing a *mutually* beneficial situation and in such a case, we could speak of *co-operation*. Logically, such a relationship will be stronger, and thus be more enduring in evolutionary terms. Based on this reasoning we redefine symbiosis to:

Symbiotic Algorithms

“Symbiosis is relationship of enduring co-operation between two or more agents”

A subtle implication of this redefinition is that a one-way benefit is no longer regarded as symbiosis. Besides this, the consequence to our evolutionary time scale is that co-operation is *at least* as old as symbiosis is. Margulis [1999] has presented a widely recognised theory that life would not have been possible without symbiosis and so this mechanism *must* have been present in a pre-life stage of evolution. This means that symbiosis should be investigated with the mechanisms and principles that apply at this stage.

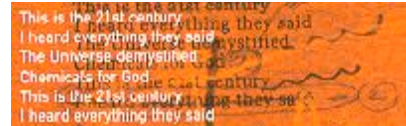
Note that this place in time is also where biological organisms have not evolved yet, which makes our choice to use the term “agents” more plausible. This does however impose some additional demands to the rather informal definition of agents that was used so far. This will be dealt with in the next chapter.

Summary

In this chapter, a framework based on neo-Darwinian evolution theory has been set up and symbiosis has been “re-engineered” into this framework. This led to a definition of symbiosis that can be used outside Biological Systems as well. Besides this, we have argued that complex biological mechanisms as life, organisms or brains, are not prerequisites for symbiosis. Lastly, the term *agent* has been introduced to describe the entities involved in symbiosis.

The following step is to formalise “agents” and to embed this in a generally accepted research area that can provide the tools to model agents that engage in symbiotic relationships.

3 Agents, Systems and Environments



"This is the 21st Century", Anoraknophobia, Marillion

Introduction

In the previous chapter, *agents* were informally introduced as “*carriers of enduring information*”. This description was founded on “engineering” neo-Darwinian ideas on evolution. Agents however are a field of research in their own right [Wooldridge 2000], and therefore it is logical to integrate the contributions of this research in the framework that has been set up so far.

There are many views and approaches to agents and as many definitions (and controversies). This is mainly because agents can be equipped with many traits, such as intelligence, mobility, motion and so forth. As we are mainly interested in one trait, the ability to engage in symbiotic behaviour, these additional qualities are not the main point of attention here. One observation from neo-Darwinism, namely that symbiosis is driven by benefit, and that this benefit depends largely on the *environment* of the agent, drives us to so-called *Pnuelian Reactive Systems* [Wooldridge 2000]. This discipline within agent-based systems is named after A. Pneuili, who has extensively studied systems that are intricately embedded in the environment they inhabit.

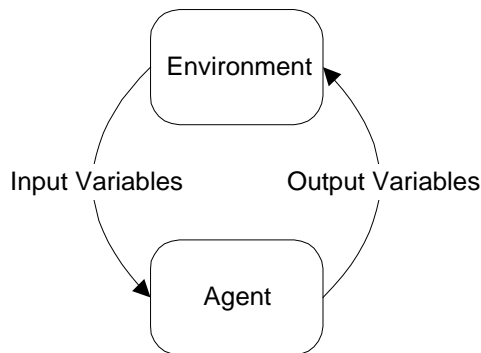


Figure 3: Pnuelian Reactive Systems

As opposed to (classic) Systems Theory [Breedveld & van Dixhoorn 1987, Open University 1992], this branch of Systems Theory and Cybernetics investigates entities (agents) that influence their environment as much as they are influenced by it (figure 3). Therefore, such a system can never be studied in isolation from its environment.

An advantage of Pnuelian Reactive Systems is, that is base for research to formal agents, that have properties that can be described in Mathematical terms.

The next paragraphs will introduce agents according to the views of Wooldridge [2000], which will then be equipped with symbiotic behaviour. It will be argued that these symbiotic agents interact with each other, and therefore belong to each others' environment. This way, a network of agent-environment pairs is created. This approach results that we get two views on the environment, namely one from the perspective of individual agents, and one from the perspective of the system as a whole.

Symbiotic Algorithms

Rational Agents

Wooldridge [2000] defines an *agent* as an:

“Entity that acts upon the environment it inhabits”

An agent is considered to be *rational* if it:

“chooses to perform actions that are in its own best interests, given the beliefs it has about the world”

The *properties* of a rational agent are derived from the fact that a rational agent is:

- ◆ Situated in an environment. This follows from the definition of an agent
- ◆ Autonomous, it can operate independently
- ◆ Proactive, it exhibits goal-directed behaviour
- ◆ Reactive, it responds to changes in its environment
- ◆ Socially able, it is able to cooperate and negotiate with other agents

Symbiotic Rational Agents

Clearly the definition of an agent that Wooldridge provides applies for entities engaged in symbiosis, for each entity acts upon the services provided by the other entities, which can be regarded as belonging to the environment of that agent. The neo-Darwinian directive of benefit implicitly makes a symbiotic agent rational, because this forces it to pursue its “best interests”. Its “beliefs about the world” are not defined, and may even be severely limited to input signals and a given state. We can now define a symbiotic (rational) agent as

“An entity capable of enduring co-operative behaviour with other entities in the environment it inhabits”

The *autonomy* of rational agents can, at first glance be a source of conflict in the case of symbiosis, for symbiosis seems to suggest giving up this autonomy. However, the given definition of symbiosis does *not* imply that an agent is *incapable* of autonomous behaviour. Symbiosis merely underlines the benefit of an agent to engage in such a relationship.

The other properties of a rational agent are more or less implicitly present in case of symbiosis.

Agents and the Environment

As said earlier, an interesting property of an agent and its environment is that the environment influences the agent, but that the behaviour of the agent can also change the environment [Wooldridge 2000].

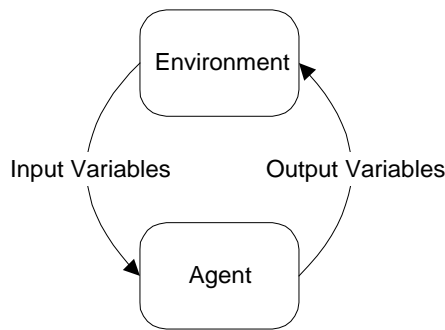


Figure 4: relationship between an agent and its environment

In the case of symbiosis we have to deal with multiple agents that, by definition, live in each other's environments. This means that the agents are somehow connected through their environment.

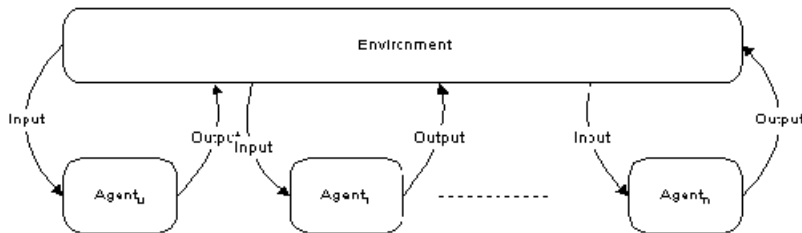


Figure 5: multiple agents

“Environment” however, is a very broad term. In its extreme form, one could consider the entire Universe to belong to an agent's environment. However, such a viewpoint is hardly practical if we want to investigate the influence of, say, a mosquito in a bedroom, or the heating capabilities of a thermostat and a boiler in a house. Although this observation may seem trivial, its consequences for a network of symbiotic agents are not. For all agents, part of the environment influences its behaviour and, in turn, the agent influences only part of the environment. The environment of an agent can therefore be split up in two distinct parts:

- ◆ The part that is within the agent's range of influence
- ◆ The part that falls outside the agent's influence

We will call the sub-environment that is within the agent's sphere a *neighbourhood*.

Even with one agent we can extrapolate this discussion, as the neighbourhood that influences the agent (input) does not necessarily have to be influenced by the neighbourhood that is affected by the agent's output. Therefore, each agent can be considered to “connect” with its environment through two neighbourhoods that may, or may not, be related.

The environment of a network of agents consists of a whole set of such neighbourhoods, many of which overlap. Even more so, this overlap is a minimal requirement for symbiosis, as agents would otherwise be independent entities and not be related in any way!

Symbiotic Algorithms

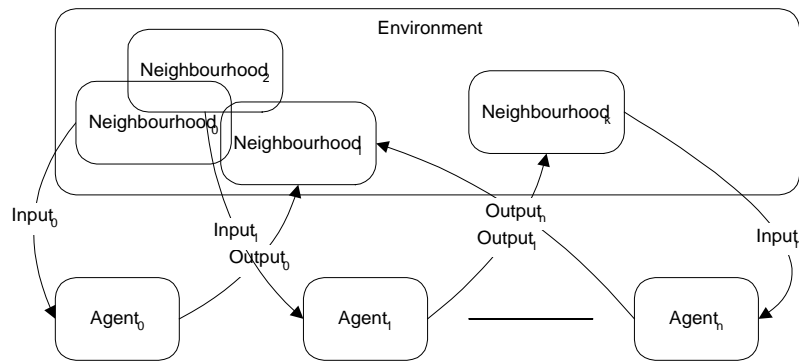
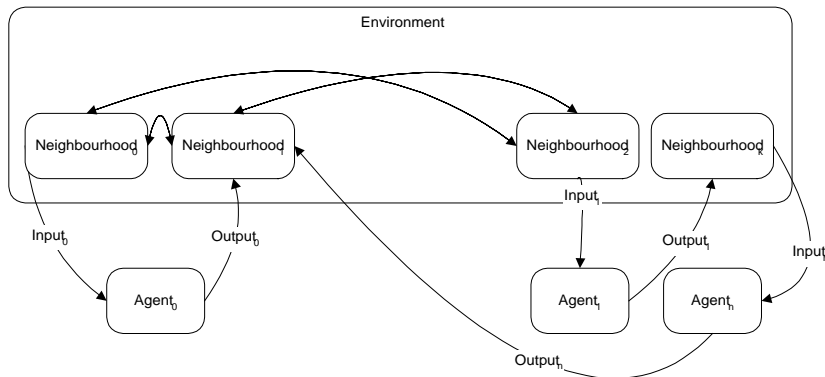


Figure 6: multiple agents

The overlap between neighbourhoods can also be seen as relationships (dotted lines in figure 7), so the environment becomes a graph representation where neighbourhoods form the nodes and the relationships between the neighbourhoods form the edges.



If we demand that every neighbourhood may only connect one in- and output of an agent, then some neighbourhoods will be split up with a relationship of similarity between them. In figure 7 for instance, neighbourhood₁ would be split up in two new ones as in figure 8.

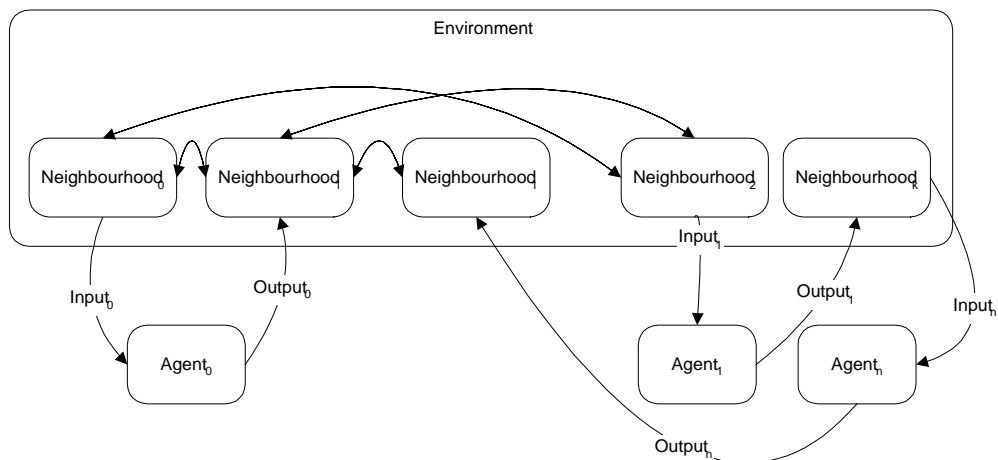


Figure 8: separating Neighbourhoods

Evidently, the neighbourhoods now merely serve as connection points, or *probes*, of the agents in the environment. The environment can be described as a graph with edges $\{E_0, E_1, \dots, E_m\}$ of relationships between the neighbourhoods.

Symbiotic Algorithms

In figure 8, the neighbourhoods may appear to be disjunct by their *location* in the environment but, although this certainly is a valid criterion, a neighbourhood will normally be differentiated from others through its *dimension*. Neighbourhood₁ could be a temperature, while neighbourhood₂ could be described in speed, acceleration or momentum. These dimensions also reflect the behaviour of the agents that connect to the environments and therefore determine the similarity or dissimilarity of those agents in respect to each other. "Dimension" in this respect could also relate to the (type of) data available in the neighbourhood, for instance because the neighbourhood interfaces with a database. Therefore, this term is broader than the pure physical meaning.

How does all of this help in understanding symbiosis? Well, to answer that question we will have to redirect our attention to the agents that interact with the environment. One of the properties that identify a rational agent was *goal directed behaviour*. This means that we have to add a goal to each agent, resulting in a goal vector $[g_0, g_1, \dots, g_n]$ for all the agents. It would be logical to relate this goal to the agent's output, but in biological systems, the goal is more often related to the agent's *input*! Survival, for instance, largely depends on the ability of an organism to acquire energy and resources to maintain and build up its system, i.e. food. The output of the individual organism often only consists of by-products that are formed because of this goal-directed behaviour and are less important for the individual itself. The fact that *other* organisms can utilise these by-products and thus create a network of interdependencies is the essential organising principle for symbiosis. Therefore the classical viewpoint to relate the goal of an agent to its output is not of interest for this research, as we *want* to investigate networks of interdependencies between agents.

Suppose that the goal of agent₁ in figure 8 would be to get a certain input level, say:

$$I_1 = g_1 \quad [3.1]$$

The given environment shows that such a goal could *only* be reached if agent₀ is able to influence the input of agent₁ through the environment, or agent_n manages to do so through neighbourhood₁. The same dependency applies for agent_n with respect to agent₁ and, to a lesser extent, agent₀ as the latter could theoretically influence itself.

If an agent depends on another agent, it would benefit greatly if there was a way to pass information to the other agent about its needs. Ideally, the agent would want to know specifically which agent it depends upon in order to negotiate its requirements. The main point of this essay is that such a *consciousness* about itself and its relationships with other agents is not necessarily available. This follows from the neo-Darwinian time-scale we set up earlier. According to Pinker [1997], complex mechanisms such as a consciousness of a "self" and of "others" can not be present at the point where symbiosis emerges. This means that such an awareness of oneself and others can only be derived indirectly, through the information that is passed through the by-products that each individual creates. Note that this information can be much richer than purely quantities of output that is produced. For instance, the *pattern* of distribution of the output, the *decay* or other qualities of the output also provide information that could be used by other organisms to "sense" others in the network. This way information about themselves can be passed to the others. This information is often not specifically directed from one agent to another, but by the physical limitations of this information. The heat produced in the agent's primary transformation process may be utilised as information by others, but it will be cumulated with other heat sources. Therefore can never result in a manner of direct communication ("I want

Symbiotic Algorithms

you to increase your output to support **me**") and therefore such a request can at best be passed indirectly through these channels.

The main proposition of this essay is that, in a system that consists of a number of interdependent agents, symbiosis is the result of:

"establishing a form of communication between those agents, composed of rules present in the physical world, that is sufficient to result in goal-directed behaviour of the system as a whole"

In other words, if we have a system containing $n+1$ interdependent agents with input vector

$$\mathbf{L} = [l_0, l_1, \dots, l_n] \quad [3.2]$$

And goal vector

$$\mathbf{g} = [g_0, g_1, \dots, g_n] \quad [3.3]$$

then a symbiotic system will possess a means of communication $\mathbf{s}(\mathbf{g}, \mathbf{L})$ between the agents in the system, described with simple rules, that results in a convergence of:

$$\lim_{t \rightarrow t_c} (l_i - g_i) = \varepsilon_i, \text{ for all } l_i \quad \text{[Proposition 1]}$$

In this proposition, ε_i stands for an error signal that is a measure for the success of the applied rules and t_c is the time needed to achieve this convergence. The error ε_i should ideally be zero at convergence ($l_i = g_i$), but as we will show later, this is not always possible. This unwanted offset should however be minimised, i.e. $|\varepsilon_i| \ll |g_i|$.

Note that in the proposition no reference is made to the environment \mathbf{E} as the convergence is proposed to be independent of the environment.

If we consider the communication to be outside the environment (which is more a matter of perspective than of incorrectness), then we can regard the entire system of interdependent agents, or *symbiotic organism*, as one entity (figure 9).

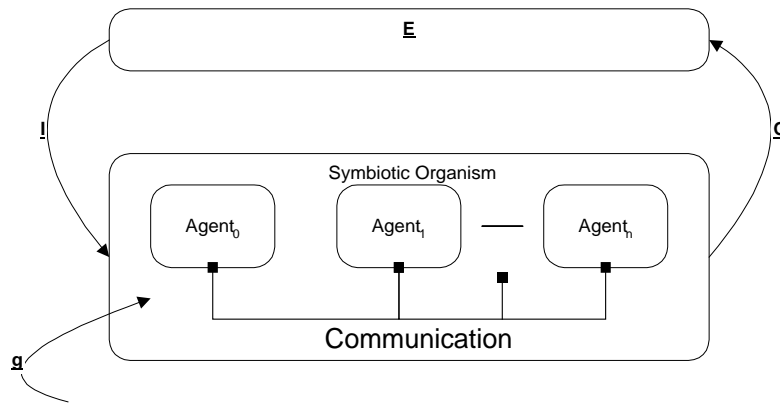


Figure 9: Symbiotic Organism

Of course, the communication is not really set apart from the physical environment, but taking the perspective of dividing the agents primary transformation processes from their mutual means of communication prevents these two forms of information-passing to get mixed up. Besides this, the primary transformation process (related to the *behaviour* of the agents and the symbiotic system) is usually considered from a different perspective than the communication, as the first focuses on interactions with the environment. This view implies that the *dimensions* of the neighbourhoods are an important part of the analysis. However, they play no role when investigating the communication between the symbiots. For this essay, the only thing that matters is the fact that there is response through a neighbourhood, regardless its dimension, and that this determines how the agents in the network are connected.

The remaining matter to resolve is how this communication works. Again, Nature provides clues on how this is set up. We know that the agents try to exchange information about the relation between their input values and their goals. In other words, they output a *stress signal* s that is a function of the input and the goal:

$$s_i(I_i, g_i) \quad [3.4]$$

This signal is dimensionless, in line with the previous argumentation. This means that the stress signal is always normalised in some way or another.

It is clear that calling out in panic doesn't help if no one responds. In Nature, such a cry of help results in a *change of behaviour* of other entities. Therefore, each agent in the symbiotic relationship has a behaviour μ_i that is a function of the stress signals:

$$\mu_i(s_0, s_1, \dots, s_n) \quad [3.5]$$

The function μ_i defines in which way an agent's behaviour is affected by the stress signals of other agents and its own. This function, together with the dimensions of the neighbourhoods, therefore also determines the similarity or dissimilarity of agents in respect to others.

The main proposition of this essay states that a symbiotic organism can converge to a state where, ideally, $\underline{l} = \underline{g}$. If \underline{g} is constant in time, then \underline{l} should be constant and therefore $\underline{\mu} = [\mu_0, \mu_1, \dots, \mu_n]$ should be constant as well. This means that in such a case:

$$\frac{d\mu_i}{dt} = 0 \Rightarrow \frac{d\mu_i(s_0, s_1, \dots, s_n)}{dt} = 0 \quad [3.6]$$

Symbiotic Algorithms

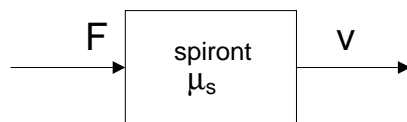
In other words, if an agent's behaviour changes according to a rule that becomes zero if the stress is zero, then the symbiotic organism should have converged. We will call such a rule a *symbiotic algorithm*.

We have introduced agents as entities being able to process input signals from, and providing output to their environment. The relationship between in- and output is determined by their behaviour, just like a conventional entity in a system. However, our agents also output stress-signals, which influences the behaviour and therefore the inputs and outputs. Through this additional mechanism, the system can be optimised. The next section gives an example of how this could work.

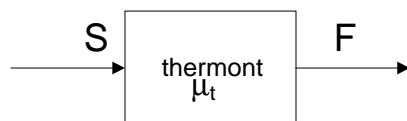
An Example: Engineering a Cell

This example, which is loosely based on Margulis' theory on how cells are formed, is meant to demonstrate how a network of symbiotic entities can be made through the environment they inhabit.

Consider a world where two hypothetical bacteria exist. One of these bacteria, which we will call a *spiront*, lives on some food source F and, because of its eel-like structure, is capable of a spiralling motion. It is able to move around or, in other words, output a velocity v . The primary transformation process of a spiront is modelled as follows:



The *behaviour* μ_s is a factor that denotes a certain relationship between v and F . The other bacterium, called a *thermont*, thrives on a sulphuric substance, which we will call S . The transformation process is similar to that of yeast, and breaks down chemical structures into smaller ones. Amongst others, it produces the food source F of the spiront:



The substance consisting of, amongst others, F and S tends to stick around the thermont as a droplet. By coincidence the spiront penetrates this droplet and discovers the treasure hidden inside. In turn, the thermont now also possesses the gift of motion, and it can start to "graze" the acid environment E_s in which it lives. This means that the speed v of the spiront influences S through the environment:

Symbiotic Algorithms

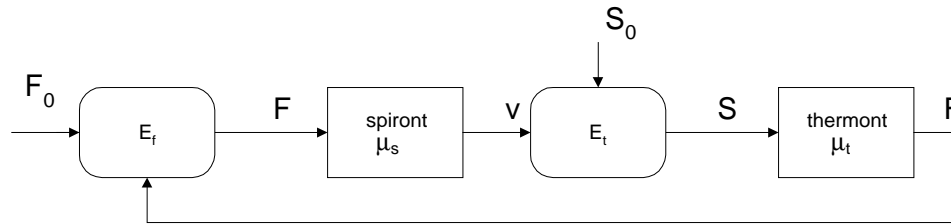
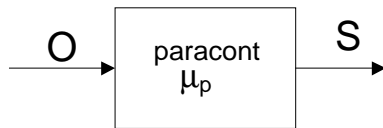


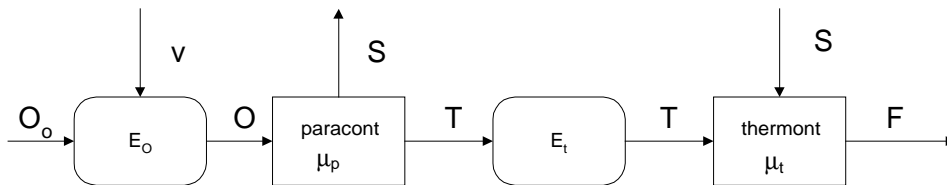
Figure 10: symbiosis between a spiront and a thermont

F_0 and S_0 are the amounts of F and S available in the environment. The amount of F that the thermont makes is not enough to sustain the spiront. Therefore, the external food source F_0 remains a necessity.

A third bacterium, which we will call a paracont, is capable of 'breathing' oxygen and creates S once it finds itself in the droplet around the thermont and the spiront:



This allows the symbiots to leave their acidic environment and penetrate the realm of oxygen O (environment E_o), as the organism can now do without S_0 . However, due to this change, the relationship between v and S is also broken, although the mobility now becomes more important to prevent the organism from smothering. A new relationship between v and O emerges. Besides this, the paracont also produces a substance T that increases the reactivity of the thermont:



This results in the following model of the organism:

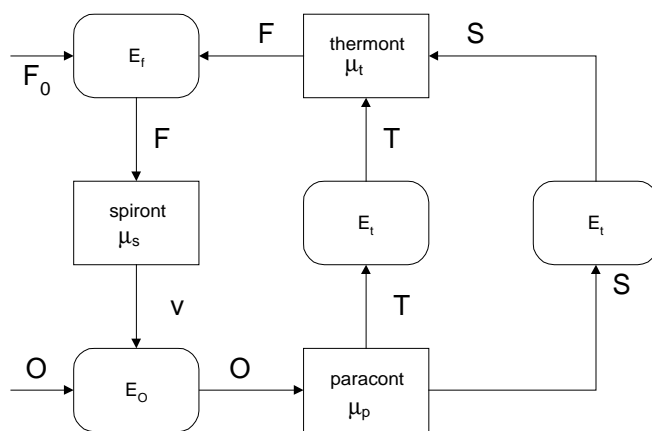


Figure 11: symbiosis between three symbiots

Symbiotic Algorithms

Although the newly formed organism is able to populate a different world than the acid one of its (competing) forefathers, the fact that S now has to be created from F poses a threat. If the spiront moves away from the food source F_0 then too little might be produced in the internal system to maintain the organism. The symbiots require minimum inputs of F, S and O to sustain themselves. If F_0 becomes too low to achieve this, then the organism would not survive. However, evolution teaches the organism a little trick to improve its chances. In such stressful situations, the symbiots learn to change their behaviour μ_s , μ_t and μ_p which allows them, within certain limits, to produce more output, given a constant external supply from the environment. In other words, the system as a *whole* deals more efficiently with its external resources and is able to adapt to a harsher environment. This could be achieved for instance, if the behaviours are dependent of the temperature and the symbiots' temperatures increase if the supply dries up.

The hypothetical example described above demonstrates how a network of symbiots, described by their behaviour $\{\mu_s, \mu_t, \mu_p\}$, can be created. They are connected through their inputs and outputs and through transformations of these in- and outputs in the environment $\{E_f, E_o, E_s, E_t\}$.

The example also aims to show how dependency the system is of external inputs and give an intuitive idea on how this dependency could be decreased by allowing the behaviour of the symbiotic entities to be dynamic as well. This construction will be studied more extensively in the next chapters.

Summary

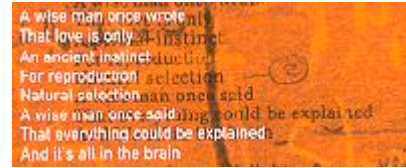
The neo-Darwinian engineering view presented in the previous chapter has been expanded to determine the properties of the entities engaged in symbiosis. This expansion was conducted through the research area of Pnuelian Reactive Systems that accentuate the intricate relationship between an entity and its environment and allows a fluent transition to Rational Agent Theory. Entities engaged in symbiosis were identified as rational agents, which allows adoption of the properties that such agents should have. One of these properties was goal-directed behaviour.

Further scrutiny of the environment which agents inhabit, showed that the environment can be regarded as a graph of *neighbourhoods*, identified by a position and dimension, and their mutual relationships. A system, or organism, consisting of n agents connects to these neighbourhoods, and forms a network this way. This leads to the central proposition of this essay, which states that a successful symbiotic system can more or less converge all its input signals to match its goals.

Lastly, the agents were examined further and their properties extended, based on what we can see in Nature or reason with logic. These properties were identified empirically but depicted in a formal fashion as base for the model of a symbiotic agent we will introduce in the next chapter.

4 Modelling Symbiosis

Introduction



"This is the 21st Century", Anoraknophobia, Marillion

In the previous chapter, a number of properties of a symbiotic agent were derived through empirical investigation. These properties will now be included in a very simple entity, namely an entity that consists solely of a *primary transformation process* [Breedveld & van Dixhoorn 1987] of input variables to output variables. This kind of entity is used extensively in Systems Theory and is base for more complex types. We will use such an entity to present a model of symbiosis. A network of these symbiots should perform the convergence that was proposed earlier, and could provide additional information on how such a network behaves as well. This will be carried out by first examining a network of two, specifically configured, symbiots and expanding the results to a network of *n* generic symbiots.

Symbiots

As mentioned in the introduction, our model starts with a *primary transformation process* [Breedveld & van Dixhoorn 1987] of an input signal to an output signal (figure 12).

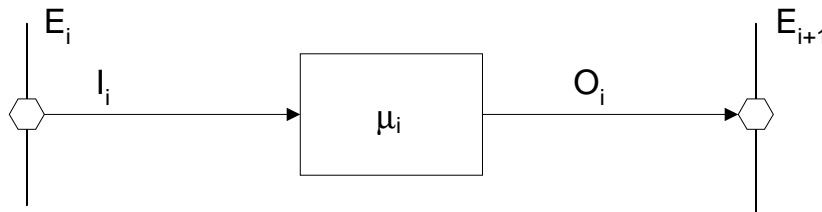


Figure 12: Primary Transformation Process

For reasons of descriptive clarity, a dotted line marked with the token E will be used to denote the environment instead of the rounded square that was used so far.

The response of this process is:

$$O_i = \mu_i \cdot I_i \tag{4.1}$$

Every entity is connected to the environment through a pair of neighbourhoods $\{E_i, E_{i+1}\}$, that may or may not have different dimensions. Entities are connected through these neighbourhoods:

$$I_{i+1} = E_{i+1} \cdot O_i \tag{4.2}$$

It is evident by this formula that neighbourhoods also have response, but as they belong to the environment, this response can not be controlled and may even be unknown.

Symbiotic Algorithms

For each symbiot_i, the following properties of symbiotic agents were derived in the previous chapter:

- ◆ A goal g_i
- ◆ A dimensionless stress function $s_i(l_i, g_i)$ [3.4]
- ◆ Symbiotic behaviour $\mu_i(s_0, s_1, \dots, s_n)$ [3.5]

These properties, added to figure 12, result in the following entity, which we will call a *symbiot*.

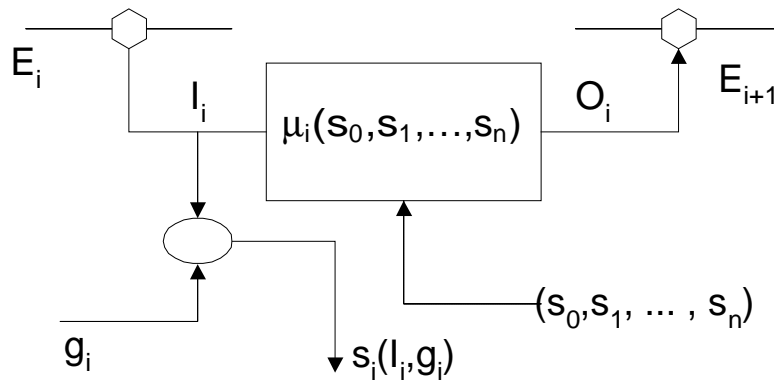


Figure 13: Symbiot

Connecting two of these symbiots together as in figure 13, results in a very simple network that will be investigated in the next paragraph.

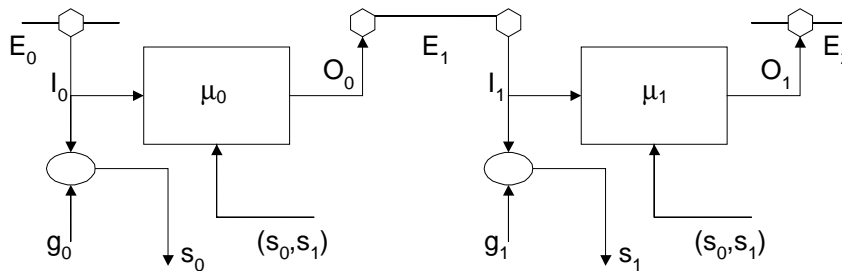


Figure 14: Symbiotic Network

A Basic Symbiotic Network

Following the line of reasoning set up in the previous chapter, the symbiotic network of figure 14 should enable a situation where ideally symbiot₀ is able to support symbiot₁ in reaching its goal $l_1 = g_1$. The offset $\varepsilon_i = 0$ in such an ideal situation.

We can say that symbiot₁ is the *successor* of symbiot₀ through the environment. According to [4.2], the following applies at convergence:

$$l_1 = g_1 = O_0 \cdot E_1 \Rightarrow O_0 = g_1 / E_1 \quad [4.3]$$

Symbiotic Algorithms

According to [4.1], this results in:

$$O_0 = \mu_0 \cdot I_0 \Rightarrow \mu_0 = O_0 / I_0 \Rightarrow \mu_0 = g_1 / (E_1 \cdot I_0) \quad [4.4]$$

The behaviour μ_0 is determined by the stress signals and should converge to a situation where $\underline{L} = \underline{g}$, or $I_0 = g_0$ and $I_1 = g_1$. In such a situation [4.4] becomes

$$\mu_0 = g_1 / (g_0 \cdot E_1) \quad [4.5]$$

It is clear that if this applies for all symbiots in the system, one could say that *the system has mapped its behaviour $\underline{\mu} = [\mu_0, \mu_1]$ against its environment and its goals*. At convergence the behaviour of the network can be described in terms of the environment, it inhabits and the goals it has. This is interesting when the environment is unknown, for a converged system can provide information about the environmental relationships between the various probes that the system has put in the environment. Note that in this situation the inputs should never be allowed to become zero, as this would make convergence to the goals impossible. If the symbiots of figure 14 were connected in such a way that $E_2 = E_0$, we have a situation of *mutual benefit*, because now the output of symbiot₁ supports symbiot₀ as well as the other way round. This results in symbiosis according to the definition we presented earlier.

Convergence of the symbiotic system takes time. According to [3.6] we can state that, when the system has converged:

$$\lim_{t \rightarrow t_c} \Delta \mu_i \approx 0 \quad [4.6]$$

According to [3.5], this will occur when:

$$\lim_{t \rightarrow t_c} \Delta s_i \approx 0 \quad [4.7]$$

Where Δ is the change over a certain amount of time.

If the behaviour μ_i changes according to some algorithm $f_{\mu,i}(s_0, s_1, \dots, s_n)$, then:

$$\Delta \mu_i = f_{\mu,i}(s_0, s_1, \dots, s_n) \Rightarrow \quad [4.8]$$

$$\mu_i^{t+1}(s_0, \dots, s_n) = \mu_i^t(s_0, s_1, \dots, s_n) + f_{\mu,i}(s_0, s_1, \dots, s_n) \quad [4.9]$$

where t stands for an iteration in time. At ideal convergence:

$$f_{\mu,i}(s_0, s_1, \dots, s_n) = 0 \quad [4.10]$$

Symbiotic Algorithms

Adding this so-called *symbiotic algorithm* to the symbiot results in:

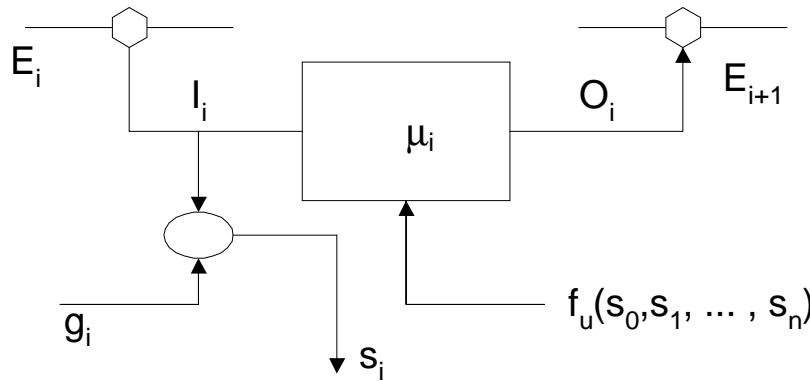


Figure 15: Symbiot

The stress signal $s_i(I_i, g_i)$ reflects the aim that one has when applying the network to a specific problem, and according to [4.6] should be zero once the goals have been achieved. In our model, the network aims to achieve $\underline{L} = \underline{g}$. The following stress function could therefore be used:

$$s_i(I_i, g_i) = \rho(g_i - I_i), \rho > 0 \quad [4.11]$$

The idea behind this choice is that the stress will increase the further the input edges away from the goal. The factor ρ is used to make the stress signal dimensionless. In this case, at convergence we get:

$$\lim_{t \rightarrow t_c} s_i = \lim_{t \rightarrow t_c} \rho(g_i - I_i) = 0 \quad [4.12]$$

The model of a symbiot we started out with has been refined by examining a network of two symbiots and by general observations. We have introduced a process of convergence of inputs to outputs and derived some facts of the model in this situation. There are however some additional remarks to be made about this convergence. Firstly, convergence should ideally never take an infinitesimal time. The *convergence time* t_c poses a restriction to the goals of the symbiot. If these goals are functions of time, then it will be clear that during the convergence time a goal should not change, as the system would otherwise possibly never be able to converge. This also applies for the environment \underline{E} .

Another matter is that, in practical systems, the various signals are limited to minimum and maximum ranges. If the behaviours of the symbiots all run into their extremes, then the system may converge to a state where $\underline{L} \neq \underline{g}$.

Last, but certainly not least, there is the matter that the properties we have derived so far are based on the *assumption* that convergence has taken place. We do *not* know however *if*, and *when*, the system actually converges. We need to prove somehow that the symbiotic network actually does converge to a situation where $\underline{L} \approx \underline{g}$. This will be examined in the next section.

Symbiotic Algorithms

Convergence

Consider a network of $n+1$ symbiots with:

- ♦ an environment $\underline{E} = [E_0, E_1, \dots, E_n]$
- ♦ an input vector $\underline{L} = [l_0, l_1, \dots, l_n]$
- ♦ a goal vector $\underline{g} = [g_0, g_1, \dots, g_n]$
- ♦ an output vector $\underline{O} = [O_0, O_1, \dots, O_n]$
- ♦ a stress vector $\underline{s} = [s_0, s_1, \dots, s_n]$

The goal vector is constant within the convergence time t_c .

For each symbiot i , the following applies at a given time t :

$$O_i^t = \mu_i^t * l_i^t \quad [4.13]$$

$$\mu_i^{t+1} = \mu_i^t + \delta \cdot f(\underline{s}), \delta > 0, f(\underline{s}) \text{ is dimensionless} \quad [4.14]$$

$$s_i^t = \rho (g_i^t - l_i^t), \rho > 0, s_i \text{ is dimensionless} \quad [4.15]$$

δ and ρ are added in [4.14] and [4.15] to scale the dimensionless qualities $f(\underline{s})$ and s_i with the dimensions of l_i and μ_i .

The environment has the following property:

$$l_j = E_j \cdot O_i, E_i > 0 \text{ for all } i, j \quad [4.16]$$

In natural language, [4.16] tells us that the output of every symbiot is connected to the input of another and that this connection causes the input of symbiot $_j$ to be proportional to the output of symbiot $_i$. Note that E_i doesn't have to be a constant value, but could be a function as well. The only restriction that it imposes is the fact that it is larger than zero. It is not known which symbiots are connected this way.

Suppose we have an initial situation where $\underline{L} \neq \underline{g}$ and the following relations apply:

$$\begin{cases} f(\underline{s}) > 0 \text{ if } \sum s_i > 0 \\ f(\underline{s}) < 0 \text{ if } \sum s_i < 0 \end{cases} \quad [4.17]$$

The \sum operator being a summation over n :

$$\sum_{i=0}^n s_i = s_0 + s_1 + \dots + s_n \quad [4.18]$$

Because of [4.14], μ_i will increase or decrease due to [4.17]. The same will also happen to $\sum O_i$ due to [4.13] and [4.16]. But according to [4.15] the opposite will occur to $\sum s_i$. This process will therefore repeat itself until $\underline{s} = 0$.

The time t_c that this convergence takes depends on the values of the elements of \underline{s} , and therefore of the values of \underline{L} ([4.15]), and therefore of \underline{E} as well ([4.16]). Therefore the environment does have an effect on the convergence time of the network.

Symbiotic Algorithms

The convergence proof tells us a number of things:

1. it clearly shows that *any* symbiotic algorithm that complies with the constraints of [4.17] will cause converge of the system
2. Every symbiot in the network should be connected to at least one other through the environment
3. it shows that $\underline{l} = \underline{g}$ is *only one* of the possible solutions. Depending on the symbiotic algorithm, averages of the various $(g_i - l_i)$ functions can also create convergence as $\underline{s} = \underline{0}$ as well in such a case. This will result in *premature convergence* $g_i - l_i = e_i$, where e_i is not zero.
4. The neighbourhood influences the convergence process through its sign. If we would allow a certain E_i to be a negative function in [4.16], then convergence is still possible, provided that the signs of the appropriate s_{i+1} is inversed in the symbiotic algorithm that is used.
5. The convergence time t_c is affected by the neighbourhoods.

The convergence time t_c depends on the number of iterations needed before the convergence criterium has been reached. Depending on the algorithm that is chosen, the rate of increase or decrease per iteration could be faster or slower. The trade off for increasing this rate is that *overshoot* can take place [Breedveld & van Dixhoorn 1987, Open Universiteit 1992], which means that the change becomes so large that the system starts oscillating around the convergence spot.

Point 3 addresses the matter of premature convergence. We will therefore introduce two types of convergence:

$$\textit{ideal:} \quad f(\underline{s}) = 0 \text{ if, and only if, } \underline{s} = \underline{0}; f(\underline{s}) = \langle -f_{\max}, f_{\max} \rangle \quad [4.19]$$

$$\textit{premature:} \quad f(\underline{s}) = 0 \text{ if } \underline{s} \neq \underline{0}; f(\underline{s}) = \langle -f_{\max}, f_{\max} \rangle \quad [4.20]$$

The problem of premature convergence is a well-recognised in neural networks [Hassoun 1995] and the explanation is the same. The n symbiots as a whole form an $n \times n$ matrix A which is dynamically altered by the stress vector \underline{s} . This matrix has a number of eigenvectors. The feedback loop that is constructed through the environment causes the system to converge to one of them. The problem is, that the eigenvector still represents a whole range of solutions and we ideally only want one of them, namely the solution where ideal convergence takes place. The challenge that one faces, is to continue convergence along the eigenvector once the system has reached it. We will return to this topic in the next section.

Up to now, we have used a very rigid goal criterion, namely that $l_i = g_i$. In nature, the survival criterion is often less strict and could be, for instance, $l_i \geq g_i$ (with food!). The stress signals could for instance be as follows:

$$\left\{ \begin{array}{l} s_i(l_i, g_i) = \rho(g_i - l_i), \text{ if } l_i < g_i \\ s_i(l_i, g_i) = 0, \text{ if } l_i \geq g_i \end{array} \right. \quad [4.21]$$

This choice increases the solution space of the network significantly, as a whole range of input vectors \underline{l} lead to a situation where $\underline{s} = \underline{0}$. This allows a much large portion of the solution space to give adequate results, leading to efficient systems that use very simple symbiotic algorithms.

Symbiotic Algorithms

A Simple Example: A bakery and its customers

After the abstract theory of the previous chapters, it may be a welcome change to apply the terms and definitions in a real life example. In this section, the relationship between a bakery and its customers will be described in “symbiotic” terms. This example is mainly intended to get an understanding how the theory can be used and to demonstrate one or two concepts that may be counterintuitive at a first glance. In this case, the theory of symbiosis is used as a *modelling* tool.

A bakery and its customers can be described as a symbiotic system in an environment consisting of two neighbourhoods, with *dimensions* money and bread (figure 16). The symbiot bakery inputs money and outputs bread, the symbiot customer does exactly the opposite. Note that the environment $\underline{E} = [E_{\text{money}}, E_{\text{bread}}] = [1,1]$, as the value of money is not altered as it passes from customer to the baker and a bread remains a bread as well.

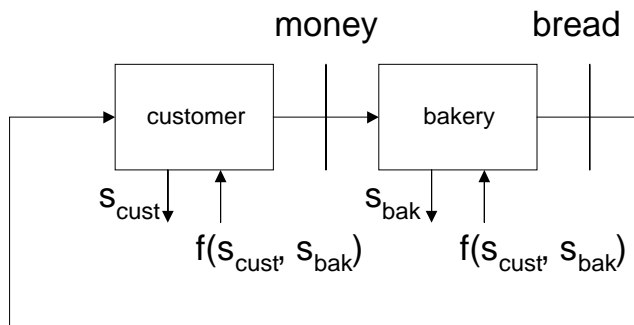


Figure 16: A bakery and a Customer

Now for a counterintuitive observation: because the goal is related to the input of an entity, the goal g_{baker} of the bakery is to make *money*, not bread! Normally the goal of a system is related to the system’s output, but we have argued earlier that this is not necessarily a logical choice. With some reflection, the bakery’s goal of making money doesn’t seem so strange at all! This is even more evident with the customer: is it the *goal* of a customer to spend money? In our two-symbiot organism, the goal g_{customer} of the customer is to eat a certain amount of bread!

The stress of both customer and bakery is based on the observation that each symbiot *at least* needs to achieve their goals, so we can say:

$$\begin{cases} S_{\text{customer}} = (g_{\text{customer}} - I_{\text{customer}}) & \text{if } I_{\text{customer}} < g_{\text{customer}} \\ S_{\text{customer}} = 0 & \text{if } I_{\text{customer}} \geq g_{\text{customer}} \end{cases}$$

$$\begin{cases} S_{\text{baker}} = (g_{\text{baker}} - I_{\text{baker}}) & \text{if } I_{\text{baker}} < g_{\text{baker}} \\ S_{\text{baker}} = 0 & \text{if } I_{\text{baker}} \geq g_{\text{baker}} \end{cases}$$

If the stress signals of the customer and the baker are normalised to the same order of magnitude, it isn’t hard to see that both symbiots can effectively reach their goals with the following symbiotic algorithm:

$$f(S_{\text{customer}}, S_{\text{baker}}) = w_0 S_{\text{customer}} + w_1 S_{\text{baker}} \quad (w_0, w_1 > 0)$$

This symbiotic algorithm controls both the behaviour of the customer as it does the baking process. If the inputs are smaller than the goals, then this will result in a stress signal that increases the value of function $f(S_{\text{customer}}, S_{\text{baker}})$ which represents the change of the behaviour of the system. Therefore, the outputs and inputs increase until the goals have been met.

Symbiotic Algorithms

This system will converge ($f(s_{\text{customer}}, s_{\text{baker}}) = 0$) when both goals have been reached, which means that one of the inputs probably is larger than its goal.

This model is not very realistic, for a baker usually has more than one customer:

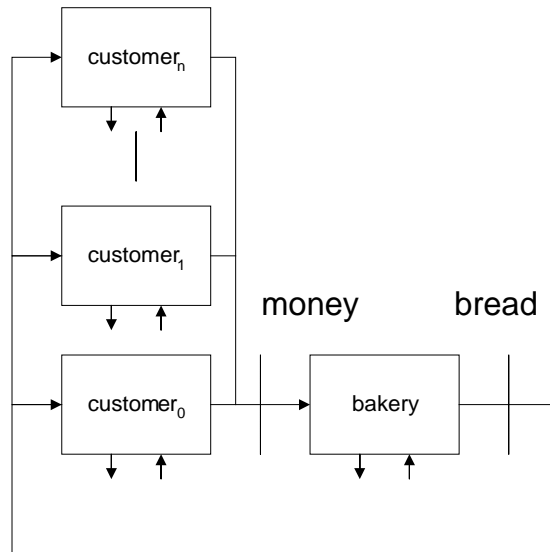


Figure 17: Bakery with multiple customers

The algorithm now becomes:

$$f(s_{\text{customer}}, s_{\text{baker}}) = w_0 \cdot s_{\text{customer}_0} + w_1 \cdot s_{\text{customer}_1} + \dots + w_n \cdot s_{\text{customer}_n} + w_{n+1} \cdot s_{\text{baker}}$$

($w_i > 0$, $0 \leq i \leq n+1$)

With more than one customer, the goal of the baker will be reached easily, because more money is generated. With enough customers, the contribution of the baker's stress signal to the symbiotic algorithm becomes negligible, and the system will converge when all the customer's goals have been reached.

The baker however, always has to make more bread than is required, for all but one customer will have a surplus of bread. It would be better to use an algorithm that converges to the *average* of the (customer's) goals, which can be reached by changing the stress signals of the customers to:

$$s_{\text{customer}} = g_{\text{customer}} - l_{\text{customer}}, \text{ for all customers.}$$

The stress function of the baker remains the same. This modification results in a negative stress signal if a customer's requirement has been fulfilled. This negative stress is subtracted from the stress signals of customers that still require more bread. The system will converge when the sum of all $w_i \cdot s_i$ factors are zero:

$$\sum_{i=0}^n w_i \cdot s_i = 0$$

This occurs when the average of all the customer's goals have been reached. Now the baker will have made just enough bread to fulfil all its customers.

Note that in this situation, the baker differs from the customer not only in the transformation (money \Rightarrow bread), but also in the stress function that is used. This difference in behaviour complies with the strict definition of symbiosis that is normally used.

Symbiotic Algorithms

The model can be expanded by adding the fact that a customer has more options to buy food from than a baker alone, which results in a system of producers and consumers:

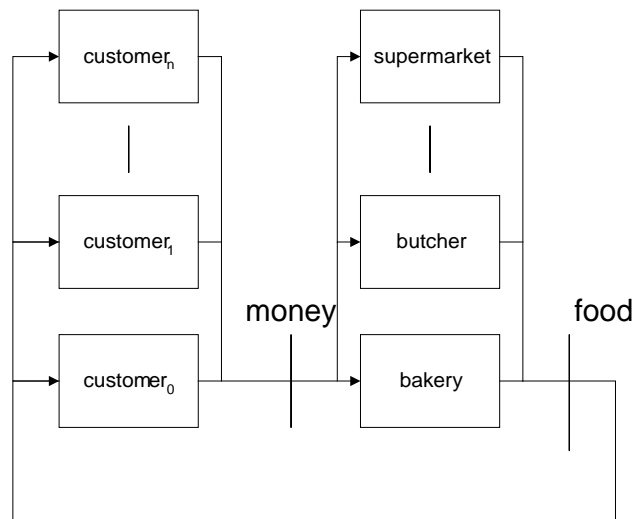


Figure 18: Simple producer – consumer model

When changing the stress of the producers to operate on averages, the system will converge to an optimal situation. Note that consequently, some producers will not reach their goals and end up with having to close their shops. Therefore, a producer will generally not use such an algorithm if she has the choice. On the other hand, if no distribution strategy is defined for the customers, some will end up with a surplus of bread while others go hungry.

In the example described above, we haven't modelled the fact that a customer has the power of *choice* to select a producer. This is the reason that, at the level of individual consumers and producers, the model isn't symbiotic. At a macro-economic level however, the system is definitely symbiotic, as producers and consumers cannot exist without each other.

Adding this choice would however not change the convergence, because dynamically changing the connections between consumers and producers does not impair [4.16].

Symbiotic Algorithms

Another Example: Artificial Amoeba

The Artificial Ant presented by Jefferson et al. [Koza 1992, Jacob 2001] involves an artificial being (an “ant”) that has to learn a strategy to find a winding path of food pellets in a grid. The ant has only limited means to sense its environment, as it can only detect the food directly in front of it. The possible actions that the ant can take is:

- ◆ move in the direction that it is currently facing
- ◆ make a left turn
- ◆ make a right turn

The task of the ant is made more difficult as the path of food shows irregular gaps, where food is not available. These gaps can also exist in corners and they become larger as the ant progresses further along the path. This so-called “Santa Fe trail” is depicted in the following figure.

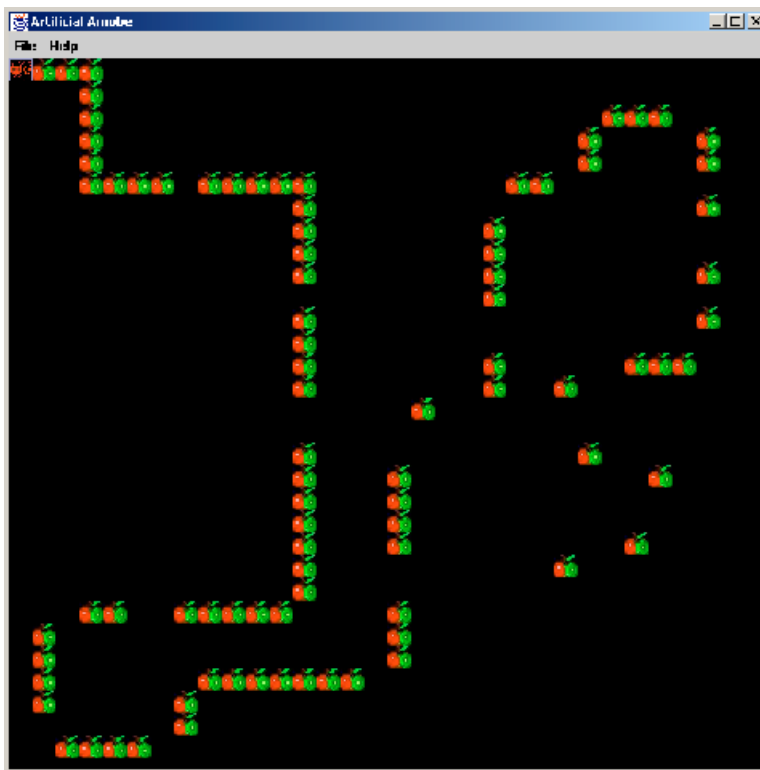


Figure 19: Santa Fe trail

The artificial ant has to learn to find a finite-state automaton within a reasonable time that allows the ant to eat all the food pellets in the trail.

Artificial Ant is considered a benchmark case to test emerging behaviour and learning strategies of artificial life forms. Successful Genetic Algorithms and Neural Networks have been implemented to solve this problem. It may therefore be interesting to see if a symbiotic network can tackle this trail successfully as well.

Symbiotic Algorithms

Koza [1992] has argued that a four-state automaton, such as the one depicted in figure 20, is not sufficient to solve the problem.

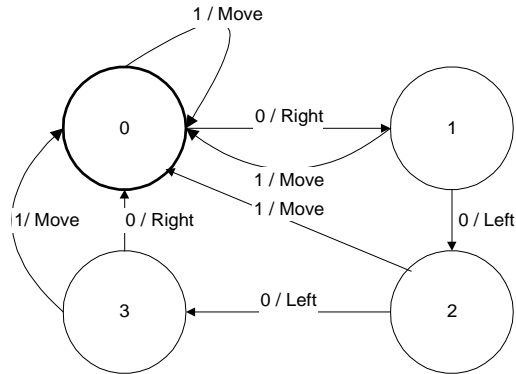


Figure 20: Finite State Automaton

In this automaton, the edges represent input-action pairs, so a “1 / move “ edge, tells the ant to move in the direction it is facing if a food pellet is found on that location. It is easy to see what happens if the Ant reaches a gap. Starting from state 0, the ant will perform a “0 / right” edge to state 1, then a “0 / Left” edge to state 2, another “0 / Left” edge to state 3, only to end up in state 0 again, through the “0 / Right” edge. Then it starts all over again, leaving the ant stuck in its position.

Before presenting a symbiotic solution to this problem, it should be noted that the symbiotic network we present here is *not* a learning algorithm. This is the reason why we present this solution as “Artificial Amoeba” as we try to connect two simple forms of behaviour through their stress signals, using a simple algorithm to control their behaviours. We are *creating* an artificial being, and this is in many ways less of a challenge than a learning ant. A learning scheme can probably teach an ant to traverse a more extensive set of paths than amoeba (or, in general, spatial distribution schemes of the food pellets). Our aim here is to demonstrate that a simple network consisting of simple rules can effectively graze environments with sparse food distribution.

The two symbiots that make up the amoeba are:

- ◆ the head, that can turn to the left, and sense the position it is facing
- ◆ the body, that can move in the direction that head is facing

The organism’s life cycle consists of a simple iteration:

```
WHILE(true) DO
BEGIN
  perform head rule;
  perform body rule;
END;
```

Symbiotic Algorithms

The two rules that make up the behaviour of each symbiot are:

head:

```
stress = false;
IF( food = false ) THEN
BEGIN
  stress = true;
  turn left;
  IF( all four directions are scanned ) THEN
  BEGIN
    face original position;
    stress = false;
  END;
END;
```

body:

```
IF( stress of head is false ) THEN
  move;
```

These two rules together implement the required behaviour for the amoeba to traverse the Santa Fe trail. The head looks to see if there is food in front of it, given a certain location and direction. If not, it outputs a stress signal, which blocks the movement of the body, and starts to look around. If no food is found, then the head faces in the initial direction and releases the stress, as if the idea that it has tried all it could brings a sense of peace. The result is that the body is able to move one position ahead. The automaton is depicted in figure 21.

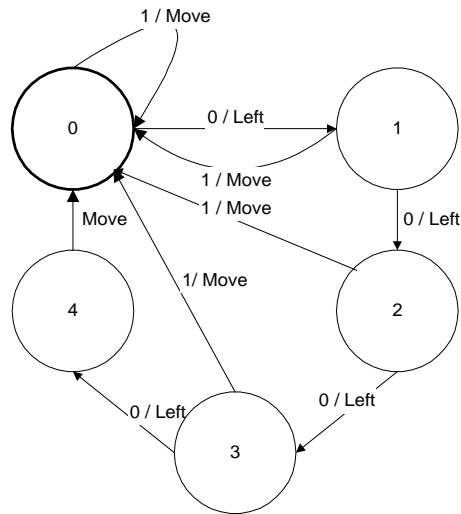


Figure 21: Finite State Automaton of Amoebae

Amoeba is able to traverse the Santa Fe trail, because the trail guarantees that there is always a food pellet further on the track, or just beside it. The Los Altos Hills trail [Koza 1992] does not have this regularity and amoeba will fail there. This problem could be tackled by introducing a continuous stress signal in the body, which is used to make multiple move operations. Without going into the details, it should be possible to create a “spiralling” motion, where Amoeba explores an increasing ring as the stress increases. This should eventually get it back on the track.

Note that Amoeba will also successfully eat all the food pellets if the grid was entirely filled up.

Symbiotic Algorithms

Summary

We have introduced a *symbiot*, a formal entity consisting of a primary transformation function, which is able to perform the symbiotic behaviour that was described in the previous chapter. A simple network of two of these symbiots was investigated, and a number of qualities of this network were derived. Some of these qualities were then expanded to a system, or organism, consisting of n symbiots. A convergence proof was given to prove that the system could converge in time. However, the proof also shows some restrictions and limitations that will have to be addressed when we want to apply the theory in practical situations.

Two simple examples of symbiotic systems demonstrated how the theory could be applied as modelling tool and how symbiosis could be applied to create an artificial life form capable of more or less complex behaviour.

We will look more closely on the limitations that the theory seems to impose on a symbiotic network in the next section. Specific attention will be given to the fact that at convergence, \underline{L} doesn't necessarily have to be equal to \underline{g} .

5 More on Algorithms

Introduction



"This is the 21st Century", Anoraknophobia, Marillion

The previous chapters demonstrated how a network of symbiots is created by connecting their in- and outputs through the environment they inhabit. This network allows each symbiot to help others to approximately achieve their goals, which results in goal-directed behaviour of the system as a whole.

In this section, a beginning is made to see if this trait of symbiotic networks can be used as a generic problem-solving instrument. Instead of looking at the individual symbiots, we will now look at the entire network. The environment is regarded as a *problem-domain*.

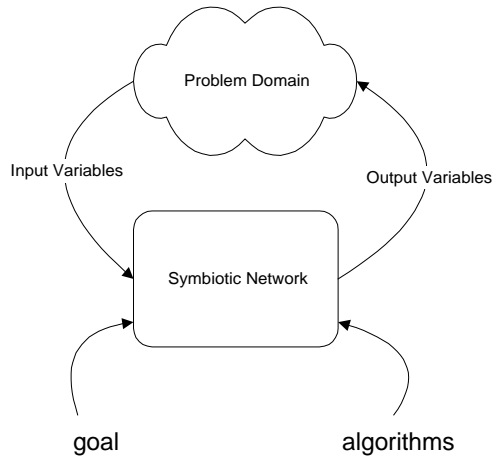


Figure 22: Symbiotic Network

The goal describes what the network should attempt to achieve, and therefore determines the solution space of the system. The algorithm, or library of algorithms, should contribute to the effectiveness of the problem solving process.

The figure depicted above has been implemented in software (JAVA) [Holzner 2000, Open University 1999] as a means of testing the theory. It also has been used to implement a few benchmark problems to see how effectively the system can be used as a practical problem-solver. This section will therefore pay some attention to the architecture of this software.

The software architecture is also used to further investigate the limitations that were encountered in the previous chapter, especially the problem of premature convergence. These problems will be examined with the following symbiotic algorithm:

$$f(\underline{s}) = w_0 s_0 + w_1 s_1 + \dots + w_n s_n \quad [5.1]$$

In this function, that was briefly introduced in the bakery example, the *weight vector* $\underline{w} = [w_0, w_1, \dots, w_n]$ determines how much a stress signal s_i of a symbiot_{*i*} contributes to the algorithm as a whole.

The Software Architecture

The software is based on the model-view-controller paradigm [Open University, 1999], [Gamma *et. al.*,2000].

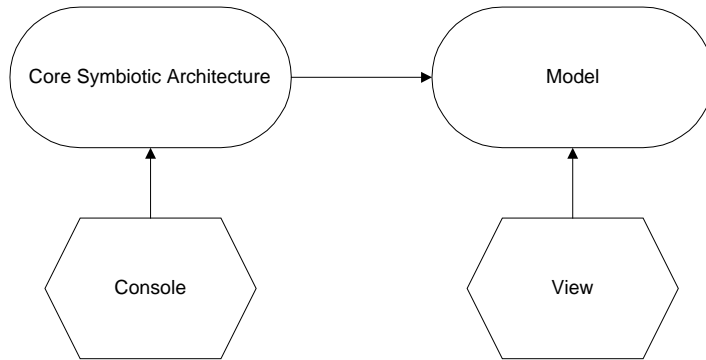


Figure 23: The experimental model

The *model* consists of the actual problem that has to be solved, while the core architecture implements the controller. Both have their own views, where the *console* is a view on the controller that provides a monitor on the process. The view on the model, of course, depends on the problem that is being implemented. The figures below give examples of the two views.

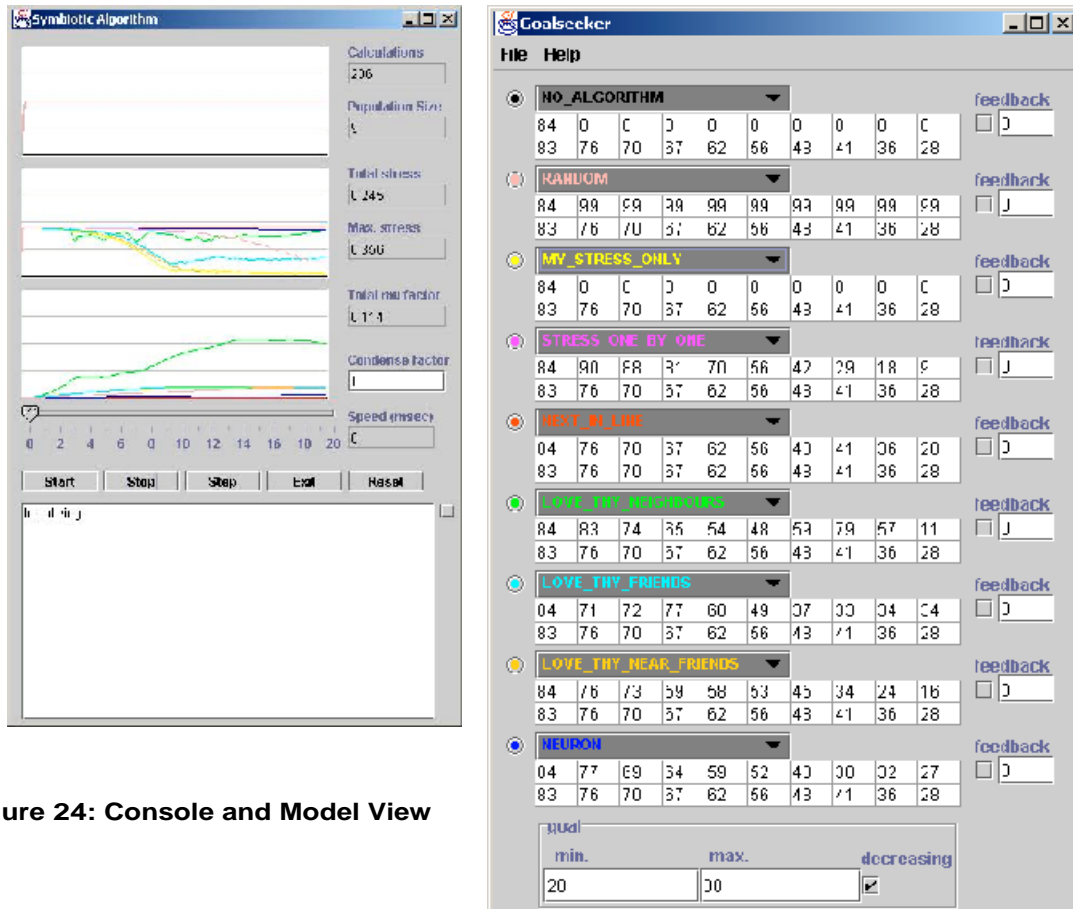


Figure 24: Console and Model View

Symbiotic Algorithms

The console shows accumulated data of the symbiots. In particular, the following three screens are displayed:

- ◆ Population size
- ◆ Accumulated stress $\sum abs(s_i)$
- ◆ Accumulated behaviour $\sum abs(\mu_i)$

The accumulated stress value is a measure of the quality of a certain symbiotic algorithm. If this value is zero, it means that all the goals of the symbiots have been met (ideal convergence). The higher the accumulated stress, the poorer the performance of the symbiotic network.

The accumulated behaviour is a measure for the energy (consumption) of the symbiot. Although this value is of little importance in a purely computational environment, it can become important when behaviour consumes energy, such as an electronic amplifier or motor control unit. A large value for the accumulated behaviour could signify that the symbiot consumes a lot of energy to reach its goals. This value, of course, should always be set against its stress value, and compared with other symbiotic algorithms and their stress values.

The console further contains a number of buttons to start, stop and step through the life cycle of the symbiotic organism.

Architecture

The basic layout of the controller is visualised in the following figure:

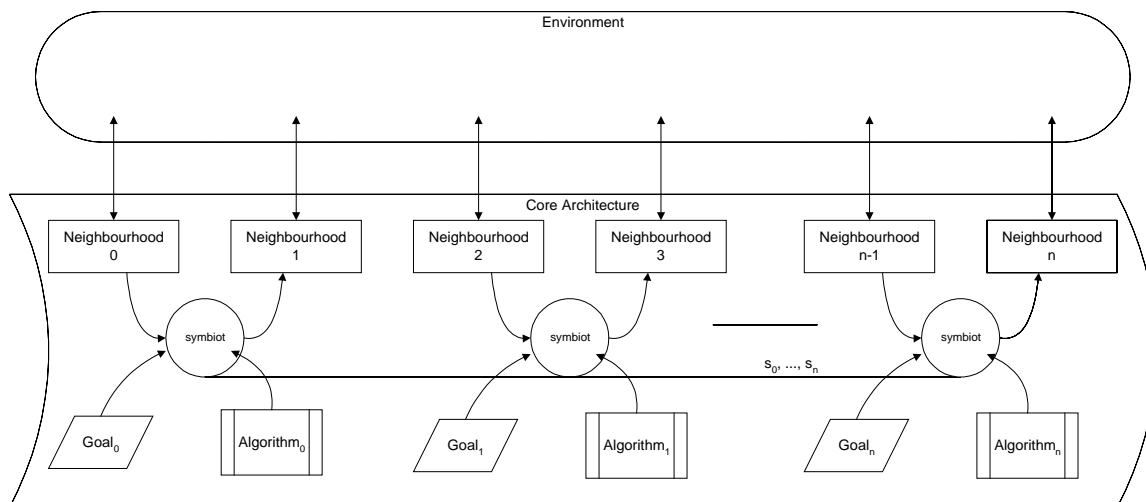


Figure 25: Global Architecture of the Controller

The controller provides classes that implement symbiotic entities (symbiots) and their neighbourhoods. The following rules apply for these items:

- ◆ A symbiot always has one input and one output neighbourhood
- ◆ A neighbourhood can connect multiple symbiots

The neighbourhoods form the interface with the actual problem domain, and therefore the values used in the environment will need to be scaled to match the internal representation of the controller, which is normalised between $<-1, 1>$. This normalisation is done by dividing values of the problem domain by their maximum values.

Symbiotic Algorithms

A symbiot i is identified by:

- ◆ an input I_i
- ◆ an output O_i
- ◆ a dimensionless stress signal s_i
- ◆ a goal function g_i
- ◆ a type of behaviour μ_i , which changes according to a dimensionless symbiotic algorithm $f(\underline{s})$.

According to the previous chapters, the state of a symbiot at a given iteration t , is determined by:

$$O_i^t = \mu_i^t * I_i^t \quad [4.13]$$

$$\mu_i^t = \mu_i^{t-1} + \delta \cdot f(\underline{s}^{t-1}), \delta > 0 \text{ is a scale factor} \quad [4.14]$$

$$s_i^t (g_i^t, I_i^t) \quad [3.4]$$

A *library* class contains a number of predefined symbiotic algorithms, but they can be easily expanded with custom-made algorithms.

The architecture has been used to investigate the limitations that were identified in the theory. This is the point of attention of the next sections.

More on Algorithms and Environments

A symbiotic network is intrinsically embedded in the environment it inhabits. Therefore, the environment partially determines the network structure, and the neat division between environment and network as depicted in figure 22, does not really exist. The following figure is a more realistic representation:

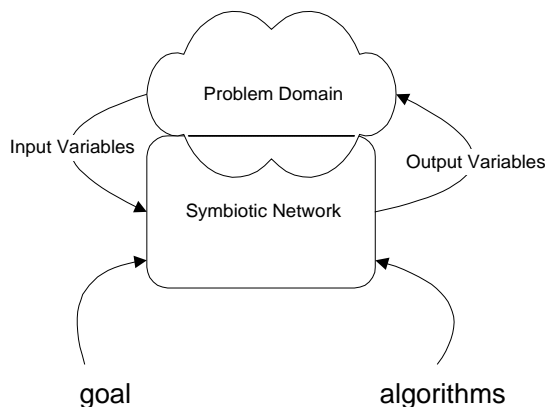


Figure 26: Symbiotic Network

The environment is the unpredictable element in this scheme, and influences the network in two ways:

- ◆ By the transformation of an output signal of one symbiot to the input of another. This is determined by the vector $\underline{E} = [E_0, \dots, E_n]$.
- ◆ By the connection scheme, or structure of the network.

The convergence proof showed that the first issue mainly concentrates on the sign of each transformation, and furthermore that the values of \underline{E} affect the convergence time t_c of the entire network.

The connection scheme, which deals with the question which and how many *successors* a symbiot has, deserves some further scrutiny, as it influences premature convergence where $\underline{l} \neq \underline{g}$ when $\underline{s} = \underline{0}$. A successor of a certain

Symbiotic Algorithms

symbiot_i is a symbiot_j of which the input is either connected to the output of symbiot_i through the environment (immediate successor) or through a number of symbiot-environment pairs:

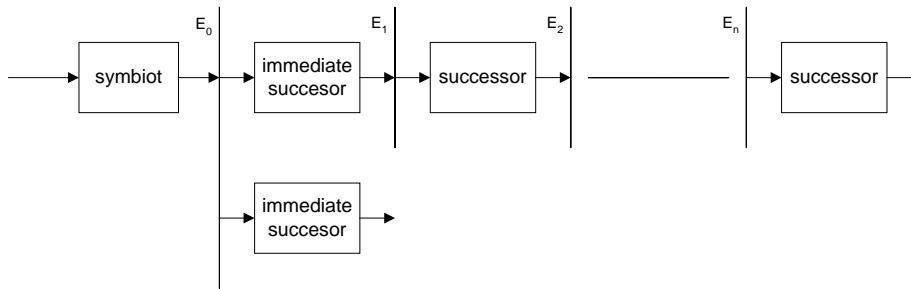


Figure 27: Successors of a symbiot

Suppose we take a symbiotic algorithm of the following type:

$$f(\underline{s}) = w_0 s_0 + w_1 s_1 + \dots + w_n s_n = \sum_{i=0}^n w_i \cdot s_i \quad [5.1]$$

It is clear that a major point of discussion is to decide which stress signals should get which weights. A symbiot can only serve its immediate successors and therefore the symbiotic algorithm should ideally be constructed to discard the stress signals of other symbiots. These contributions may possibly only pollute the useful information of the symbiot. It is as if trying to hear your children's call on a busy playground. The dilemma that a symbiot faces is that its successors are determined by the environment and are therefore not necessarily known.

A network consisting of n entities can be constructed somewhere between two extremes, where every entity is an immediate successor of:

- ◆ All the other entities (a fully coupled network)
- ◆ At most one other entity. This results in a path or a cycle of symbiots

Every existing network is somewhere between these two extremes and can be regarded as a graph where one or more paths can be identified. These paths start from an external input, which are inputs to entities that do not have a predecessor. They end in:

- ◆ External outputs, outputs of entities that do not have a successor
- ◆ In cycles

From these paths new paths, or *branches*, can be formed, creating a tree-like structure. They can also form parallel connections over one or more other entities.

Parallel connections of symbiots have been treated in the bakery example and, along the lines, demonstrated that they can impair ideal convergence. It is clear that if a symbiot outputs to multiple successors, convergence to all their goals is only possible if there is overlap between the goals of these successors. The symbiot will never be able to serve its successors if they have contradicting goals. For instance, if in figure 27, the goal of one immediate successor is $I_i < g$ and the other has a goal $I_i > g$, then the symbiot will not be able to satisfy both of them. It will opt for an average, the upper, or the lower bounds of their goals, depending on the symbiotic algorithm that is chosen. These limitations are *intrinsically* determined by the way the symbiots are connected and the goals they have. There is no way that a symbiotic algorithm can work around these limitations.

Symbiotic Algorithms

If a symbiot knows what its immediate successors are, and there is overlap in their goals, then it can concentrate on optimising these goals. This results in a network where ideal convergence has taken place. Therefore, a perfect symbiotic algorithm will be able:

- ◆ to identify the immediate successors
- ◆ have their corresponding weights to be unequal to zero (usually larger than zero)

The weights of other stress signals should ideally be zero, as their sum totals zero at convergence anyway, and should be so that their total influence does not affect the successor's operation. Say, for instance, that symbiots j , k and l belong to the immediate successors of symbiot i and there is overlap in their goals, then the following algorithm would behave optimally ($w_j, w_k, w_l > 0$):

$$f_i(\underline{s}) = 0 \cdot s_0 + 0 \cdot s_1 + \dots + w_j \cdot s_j + w_k \cdot s_k + w_l \cdot s_l + 0 \cdot s_m + \dots + 0 \cdot s_n \quad [5.2]$$

If the other weights are unequal to zero, then they will contribute to the sum and could cause premature convergence.

Branches in a network can also cause premature convergence, namely through *propagation*. In a branch, a change in the behaviour of one symbiot results in changes to the input of its immediate successors, and their immediate successors and so forth. If such sequence of successors contains amplifying elements, then a small adjustment at the start of the branch can cause enormous fluctuations just behind the amplification. This amplification is transferred to the stress signals and result in a situation where the contributions of symbiots further down the chain are much higher than that of the immediate successors. They “out-shout” not only the immediate successors, but also other stress signals in the network. Therefore, they impair correct functioning of the network. Note that amplification has to occur when a goal value of a symbiot is lower than that of a successor. This situation will be investigated in the following paragraph.

According to [4.19], ideal convergence takes place with algorithms that comply to the following rule:

$$f(\underline{s}) = \sum_{i=0}^n (w_i \cdot s_i) = 0 \Rightarrow \underline{l} = \underline{g}, w_i \cdot s_i > 0 \quad [5.4]$$

It would be ideal if the weight vector \underline{w} is not a fixed value, as in the bakery example, but can adjust itself somehow to reach this situation. This would mean that \underline{w} could be a function of \underline{s} as well, and would have to learn to reflect the connection scheme of the network, identifying which successors belong to which symbiot, despite the previous limitations. The challenge for such a network is to “detect” successors dynamically.

During this research, we have not succeeded to find algorithms that effectively faced this challenge and converge under all circumstances. Various algorithms were tested in a branch of symbiots, to see which ones converged best. The results are depicted in the next paragraph.

The GoalSeeker Example

The GoalSeeker example has been implemented to simulate a branch and is used to investigate the limitations that were described in the previous section. The

Symbiotic Algorithms

example consists of a sequence of symbiots, with primary transformation function:

$$O_i = \mu_i \cdot I_i \quad [5.5]$$

Each symbiot i is connected with one immediate successor, through a neighbourhood that does not transform the output of the symbiots:

$$E_i = 1 \quad [5.6]$$

The output of the symbiot is therefore equal to the input of its successor. This results in the following connection scheme:

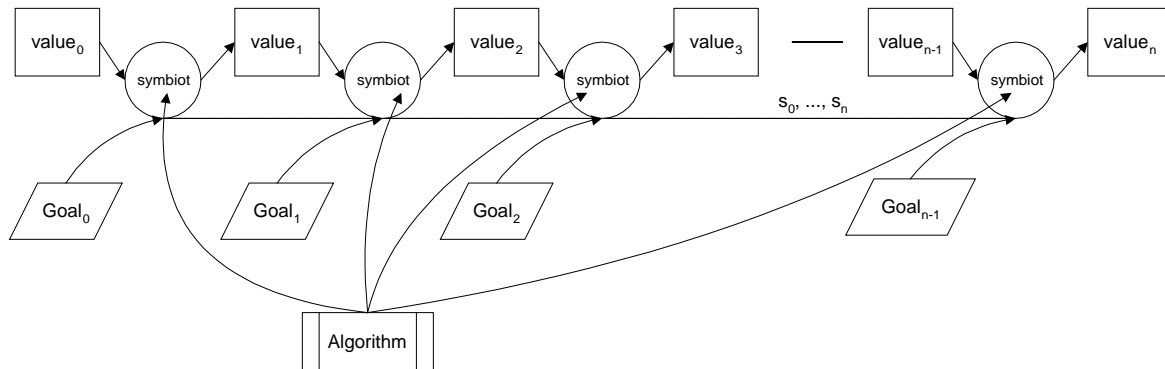


Figure 28: The GoalSeeker Example

Each neighbourhood maintains a value, which is produced by a predecessor symbiot, and has to match the goal of the successor. This goal is a randomly generated number, excluding zero. The first symbiot gets a fixed value for I_0 that is equal to its goal:

$$I_0 = g_0 \quad [5.7]$$

The other symbiots all start with $I_0 = 0$, but gradually change their output values depending on the algorithms that are used. This means that all symbiots initially produce a lot of stress, which may become less if a correct algorithm is used.

The symbiots in a branch all use one kind of algorithm. In the test environment, nine of these branches are created, all using a different algorithm. This way, the behaviour of the algorithms can be compared to each other. The symbiots have no stop criteria, but keep on running until the “stop” button on the console is pressed. The values of the goals and inputs in the model have been converted to integer values ranging from 0 – 100, but internally all signals are normalised between $<-1, 1>$ by dividing all values from the environment by their maximum values.

According to the theory, the following algorithm should perform very well:

$$f_i(\underline{s}) = w \cdot s_{i+1}, w > 0, 0 \leq i < n \quad [5.9]$$

This algorithm, called NEXT_IN_LINE, takes only the stress signal of the successor to converge on, causing ideal convergence. The constant weight factor w influences how fast the network converges.

Symbiotic Algorithms

As said before, the test is performed with nine algorithms, eight of which are depicted in the following table:

Name	$\mu_i^t = \mu_i^{t-1} + \delta \cdot f(\underline{s})$, with $f(\underline{s})$:
NO_ALGORITHM	0
RANDOM	$w^t \cdot s_i$, w^t is a random number $\langle -1, 1 \rangle$, which changes with each iteration.
MY_STRESS_ONLY	$w \cdot s_i$; $w > 0$ is a constant value
STRESS_ONE_BY_ONE	$\sum w \cdot s_j$; $w > 0$ is a constant value
NEXT_IN_LINE	$w \cdot s_{i+1}$, or 0 if $i+1 > n$; $w > 0$ is a constant value
LOVE_THY_NEIGHBOURS	$0.5 \cdot w \cdot (s_{i+1} + s_{i-1})$, or 0 if $i+1 > n$ or $i-1 < 0$; $w > 0$ is a constant value
LOVE_THY_FRIENDS	$w \cdot \sum s_j / (i - j)$ IF $i \neq j$, otherwise 0; $w > 0$ is a constant value
LOVE_THY_NEAR_FRIENDS	$w \cdot \sum s_j / (i - j)^2$ IF $i \neq j$, otherwise 0; $w > 0$ is a constant value

Table 1 : Algorithms used in GoalSeeker

Unless noted otherwise, index i is used for the considered symbiot. Index j is used to designate other symbiots with respect to the considered symbiot.

NO_ALGORITHM does not use any stress signals, and therefore is not symbiotic. MY_STRESS_ONLY and RANDOM are “selfish” algorithms, which only use the stress of the symbiot itself.

STRESS_ONE_BY_ONE is the benchmark averaging symbiotic algorithm, which takes equal contributions of all stress signals.

The last four algorithms all have a basic form of selection, based on proximity or their position in the branch. They implement a concept of near / far, as the indexes in the branch increase from left to right. LOVE_THY_NEIGHBOURS uses the stress of the predecessor and the successor equally, while the others gradually decrease the contributions of stress signals the “further” the symbiots are located in the branch. It is interesting to see these algorithms in comparison to STRESS_ONE_BY_ONE, to see if selection can outperform a basic averaging algorithm.

Symbiotic Algorithms

Also an algorithm called NEURON is included that, based on [5.1], uses weight adjustment ($\underline{w} = \underline{w}(\underline{s})$) for the symbiotic algorithm $f_j^t(\underline{s}) = \sum w_j^t \cdot s_j^t$ of symbiot_i. This algorithm consists of two rules, namely a *Hebbian Learning Rule* [Hassoun 1995]:

$$w_j^t = w_j^{t-1} + \rho s_j, \quad 0 < \rho \ll 1 \quad [5.10]$$

and a *forgetting rule* ($0 < \varepsilon \ll 1$) :

```
IF(  $s_j^t - s_j^{t-1} = 0$  ) THEN
BEGIN
  IF(  $w_j > 0$  ) THEN
     $w_j = w_j - \varepsilon$ ;
  ELSE
     $w_j = w_j + \varepsilon$ ;
END;
```

The forgetting rule starts working when a symbiot has converged ($\Delta s_j = 0$) and tries to decrease the weights that have been learned by the learning rule. Eventually this causes a new stress signal ($\Delta s_j \neq 0$), which stops the forgetting rule. The learning rule is always active. The idea behind this strategy is, that “false” learning in the initial situation is slowly filtered out once the system has converged. For instance, it is quite logical that many symbiots initially behave the same, as all the stress signals are decreasing towards average values. The result is that false patterns are learned, which are no longer appropriate once the system has stabilised on its main eigenvector.

This algorithm adjusts the weights dynamically and closely resembles a perceptron, one of the first artificial neural entities that were ever examined (Hassoun 1995, Kasabov 1998), hence its name.

The GoalSeeker test is meant to check the following:

- ◆ see if the expected behaviour of the symbiots comply with the actual results
- ◆ investigate what the effect is of propagation through the branch

Propagation is tested by running the network in a setting where the goals have decreasing values in the branch, starting from left to right:

$$g_i \geq g_j, \quad \forall i < j \quad [5.10]$$

and in a situation where this constraint is not imposed on the network.

The effect of decreasing goal values is that the branch as a whole does not amplify the input- and output signals, thereby dampening the propagation of changes due to the change of stress signals. The results of a typical run are depicted in Appendix A, which is the situation after a number of iterations.

In the appendix, a typical run is displayed which allows amplification and one that only has decreasing goal values (no amplification). Especially the cumulated stress signals are of interest (see the following figures). The “condense factor” means the number of iterations after which the screen is updated. Therefore, the first screen has made twice the number of iterations as the second. This has been done to show where the orange line converges to, as it otherwise would not fit on the screen.

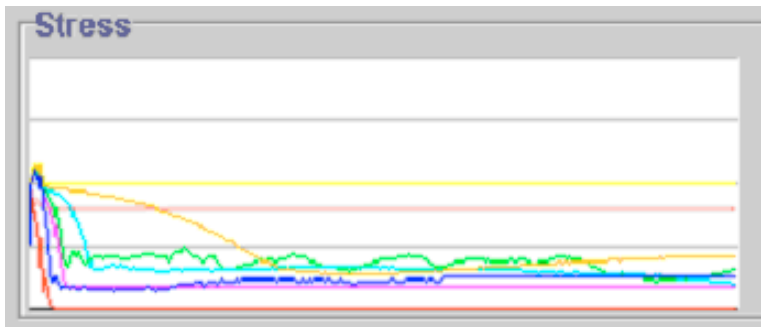


Figure 29: Cumulated Stress with random goals (condense factor = 2)

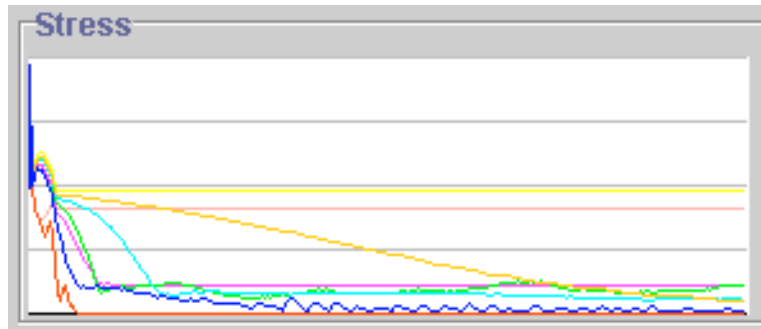


Figure 30: Cumulated Stress with decreasing goals (condense factor = 1)

In the figures, the red line is NEXT_IN_LINE which, as expected, converges to zero stress (ideal convergence). STRESS_ONE_BY_ONE is coloured magenta and is the true averaging algorithm. In the theory of the previous chapter, it was clear that this tendency to converge to average values is more or less “default” behaviour. It is clear that NO_ALGORITHM (black, hidden behind yellow), RANDOM (pink) and MY_STRESS_ONLY (yellow) behave poorly.

All the other algorithms perform a bit worse than STRESS_ONE_BY_ONE when the goals are not in decreasing order, but become significantly better in the second test with decreasing goals. NEURON eventually converges almost ideally.

The three LOVE_THY_... algorithms all are based on the well-known fact in nature that the influence of a certain property normally decreases the further they get from the source that creates them (think of heat). All the algorithms are functions of r , where r is the distance with respect to symbiot_{*i*}. This is the best that a “physical” algorithm can do, to identify its successors. Note that, when using a function like [5.1], the effect of converging to an average value has a side effect that it doesn't really matter how many symbiots partake in the network. This is why LOVE_THY_NEIGHBOURS (green) tends to converge around the same values as STRESS_ONE_BY_ONE. The oscillation is a result of the fact that, with two symbiots, overshoot occurs more easily. This observation also explains why LOVE_THY_FRIENDS (cyan) performs above average. This algorithm uses a $1/r$ function, which causes the average values to become much lower, because the function dampens the stress signals that are “far” from the source. LOVE_THY_NEAR_FRIENDS (orange) ($1/r^2$) uses this effect more dramatically, which results in same overshoot problems as LOVE_THY_NEIGHBOURS. Even premature convergence becomes difficult in this situation.

Symbiotic Algorithms

The main results of the test runs are:

- ◆ symbiotic networks with “neural” capabilities perform significantly better in networks without amplification
- ◆ The best converging strategy is to identify successors in one way or the other

There are a few ways to detect successors. One way is to initially activate the symbiots one by one, and checking which symbiots start converge. These must be the successors of the active symbiot. This approach will fail if the relationship between symbiots can change or when there is a delay (buffers) in the environment. Another method could be to introduce a sudden change in an output signal and monitoring the changes in stress signals. This will also cause problems if the environment has delays or filters that affect the signal.

Many algorithms were tried and tested for this research to dynamically find successors in a symbiotic network. Such an algorithm would pave the way for a universal problem-solver. However, currently such an algorithm has not been found and the best way to get a converging symbiotic network is by explicitly identifying them in the symbiotic algorithm that is used.

Summary

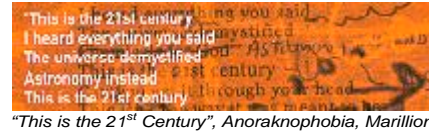
In this chapter, we have further investigated the limitations that were identified in the previous chapter and tried to find ways to overcome the limitations that the unpredictability of the environment imposes on such a network. Two major points of attention were investigated that seem to impair ideal convergence:

- ◆ goal conflicts between *successors* of a symbiot
- ◆ the propagation of the effects of an amplification in a branch

In order to test the latter problem, a test environment, programmed in JAVA, has been introduced and a test, the GoalSeeker example, has been used to examine the behaviour of a symbiotic network in a branch. Nine algorithms were tried out, including one called NEURON that dynamically can change its weights as an optimisation strategy. This algorithm resembles a *perceptron*, one of the first artificial neurones ever examined. The examples made clear that a good symbiotic algorithm is able to identify the successors of each symbiot. Although NEURON seems to be a step in the right direction, no algorithms have been found that can overcome the limitations that have been identified. Such an algorithm could result in a universal problem solver that can be used regardless of the problem domain.

In the next chapter, we will further examine symbiotic networks as a potential problem solving mechanism, and try to see how we can deal with the limitations that were identified here. One important issue will be whether a symbiotic network is the right way to utilise symbiosis to solve problems.

6 Symbiotic Networks as Problem Solvers



Problems and Solutions

According to Kasabov [1998], a solution to a defined problem is a mapping of n solutions from n independent input variables from the domain space.

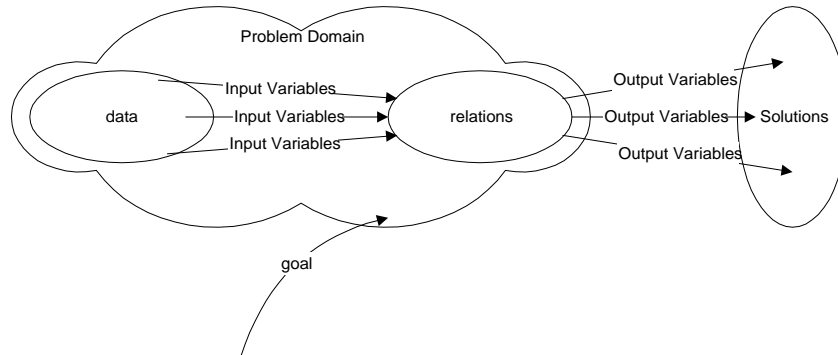


Figure 31: Problems and solutions

In the above figure, the problem domain consists of data and relations. Relations give meaning and structure to the data of the problem domain. This structure can be used to identify patterns in the possible solutions. This can never be considered without a goal, a rule, or set of rules, that designate the purpose of the problem-solving activities. The goal is used to filter out data and relations that do not contribute to the expected solutions, and often serves as a reference for the success of the problem solving activities. For instance, if one wants to find the maximum values of the data of the problem domain, then the goal will be different to when the minimum values or all values within a certain range are searched.

The problem solving process is usually impaired by the fact that data of the problem domain may not be complete, is ambiguous, or that the relations are unknown. The following table classifies problem characteristics according to such qualities:

Data		Relations	Solutions
Exact	Vague	Known	Known
Unambiguous	Ambiguous	Partially unknown	Too many
Complete	Incomplete	Unknown	Unknown
Static	Dynamic	Static / Dynamic	Static / Dynamic

Table 2: Classification of Problem Characteristics

Strictly speaking, the goals could be categorised in a similar manner, but ambiguous, impossible or vague goals can only be solved by further analysis of the purpose of the problem solving activities and do not influence the problem solving itself.

Symbiotic Algorithms

Any combination of the qualities in the above table can occur, and will require a different approach to finding a solution. A few examples of problem classes are depicted below:

Data		Relations	Solutions	Approach	
Exact	Ambiguous	Incomplete	Unknown	Known	Modelling, Pattern Recognition
Exact		Complete	Unknown	Unknown	Classification Optimisation
Exact		Complete	Known	Too many	Optimisation
Inexact	Ambiguous		Known	Unknown	Diagnosis

Table 3: Examples of Classes of Problems and their Approach

In the above table, the dynamic features have not been taken in account, as this does not concern the required approach of the solution. It is, however, a very important issue for the problem solver.

For some of these approaches, automated solutions exist, which assist in, or take over, the problem solving process. These so-called *engines* are the focus of knowledge systems.

Bearing in mind that the solution space is a subset of the problem domain, problem solving engines fall apart in two distinct categories:

- ◆ Open Loop Systems
- ◆ Feedback Systems

The first category “feeds” the engine with data from the problem domain only, while the other uses the results from the solution space to further limit the scope of the problem domain. In other words, the results become part of the input as well. Although there are no clear-cut lines, the first category of engines are often used for pattern recognition, modelling and diagnosis, while the second category is used for classification and especially optimisation.

Another distinct feature of the engine is the way the goal is used. This can be *intrinsically defined*, such as with modelling and classification, or *extrinsically defined*, such as with pattern recognition, optimisation and diagnosis.

In the latter case, the goal is externally determined by rules (fitness, criteria) or examples (teaching sets), while in the first situation the engine is specifically designed for its objectives.

The third trait of a problem-solving engine is its ability to deal with dynamic aspects of data, relations and solutions. This is partially a matter of speed. In the time that the engine needs to infer its solution, data, relations and goals should be static, as the eventual solution could be meaningless if the used data or goals

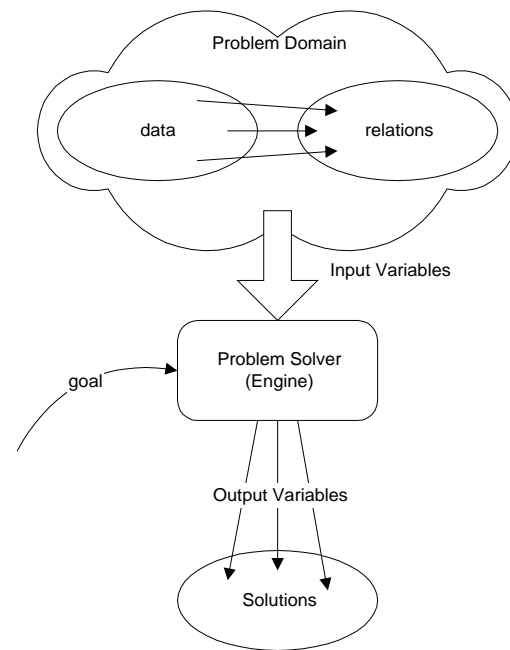


Figure 32: Problem Solvers

Symbiotic Algorithms

have become obsolete. The dynamic aspects however also become important if the system's adaptive or learning abilities become an important objective.

Kasabov [1998] identifies the following incentives for knowledge systems:

1. Representation: the ability of the system to structure the problem domain to known schemes or formats, so that it becomes meaningful.
2. Inference: the ability of the system to derive new facts by matching acquired knowledge to existing facts.
3. Learning: the ability of the system to improve its behaviour by effectively utilising the knowledge it has acquired.
4. Generalisation: the ability of the system to match new, unknown input data with its knowledge in order to obtain the best possible solution.
5. Interaction: the ability of the system to adapt to a new situation, or to improve itself by communicating with its environment or a user.
6. Explanation: the ability of the system to retrace the path that led to the eventual solution in a contextually comprehensible way.
7. Validation: the ability to test the quality of the solutions produced by the system.
8. Adaptation: the ability of the system to change when the environment or problem domain changes.

These objectives can help in identifying strengths and weaknesses of symbiotic networks.

Categorising Symbiotic Networks

If the environment of a symbiotic network is considered similar to a problem domain, the network can be regarded as a problem solver, as it is able to optimise its behaviour against a certain (extrinsic) goal. As the system can only optimise to one solution at a time, its abilities for classification is limited to one class. A neural network, for instance, is able to classify multiple classes, depending on the training sets that it has learned.

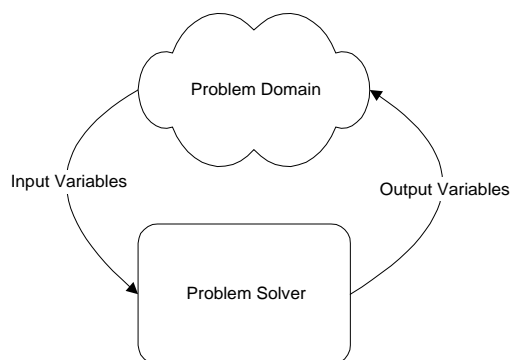


Figure 33: Symbiotic Network

The theory of symbiosis also showed that a converged symbiotic network also has mapped its environment through its behaviour. In other words, the behaviours of the symbiots show the relationships between the input data. Therefore, a symbiotic network can also be used for modelling (as was demonstrated in the example of the bakery) and pattern recognition, although the latter is limited to one pattern at a time for the same reasons as with classification.

The following table compares a symbiotic network with two other engines, using the objectives of Kasabov for a knowledge system:

Symbiotic Algorithms

Characteristic	Neural Network	Genetic Algorithm	Symbiotic Network
Representation	Unstructured	Unstructured	Unstructured
Inference	Approximate	Inexact	Depends on algorithm
Learning	Very good	Good	Reasonable / good
Generalisation	Very good	Reasonable	Poor
Interaction	Good	Very good	Very good
Explanation	Weak	Good	Good
Validation	Modest	Good	Very good
Adaptation	Good	Very good	Very good

Table 4: Characteristics of Knowledge Systems

A symbiotic network equals the others in its ability to deal with unstructured data. Its inference depends on the symbiotic algorithm that is used and can be exact if the algorithm is exact or approximate if optimising algorithms are used.

Learning is reasonable to good, depending on how the limitations of the network affect the goals and the solution strategy of the network.

Generalisation is poor, as the system is always optimised for one goal and only input variables that are not related to a goal can be generalised without a problem.

The explanatory capabilities of the network are good, as the result can be compared with the goal. This also counts for validation, as the stress signals can be easily monitored, which give a detailed account of the process that leads to the solution. Adaptation is also very good, as a change in the environment after convergence immediately triggers a stress signal.

According to the table, the ability of generalisation is the only poor characteristic of a symbiotic network. This, for a large part, is directly related to the problems that were investigated earlier, and give reason to wonder if a “symbiotic network” is the right way to use symbiosis as a problem-solving mechanism.

Symbiosis in nature is an *additional* trait of different entities and our choice of model reflects this. A simple entity with a primary transformation process has been *extended* with a stress signal and a means to deal with the stress signals of other entities by changing its behaviour. Any entity, no matter how complex, could be extended this way, by defining a stress signal and a set of behaviours that are controlled by the stress. However, this extension only is beneficial in parts of a network that contribute to the goals that have been defined. This by no means excludes non-symbiotic entities to partake in the network. These entities, which one could consider to have don't-care goals, could contribute significantly to the ability of the network to generalisation. By restraining the number of goals or stress signals, premature convergence can also become less of a problem.

When applying symbiotic entities to solve actual problems, a ‘hybrid’ approach may also be the most practical, as some entities may not require help at all to perform certain sub-tasks, while still contributing to the system as a whole. For instance, it could be very plausible to embed a neural network between symbiotic input entities and symbiotic output entities, thus creating a multi-layer network that could deal with higher order complexity of the environment. The neurones do not have to be symbiotic, as they are already sufficiently equipped to deal with this task.

The point that is being made here, is that symbiotic entities, as in Nature, are not unique species, but instead are normal entities that have acquired an additional trait of beneficial co-operation. By forcing all entities in a network to be symbiotic, we may likely introduce more complexity than is required to fulfil the task of the network. Instead, symbiosis should be restrained to those parts of the

Symbiotic Algorithms

network where it has a positive advantage, namely where it can assist in converging to the goal of the entire network. We will demonstrate this “hybrid” view on a network in the Travelling Salesman Problem in the next chapter.

Using Symbiosis for Solving Problems

The model of symbiosis that was presented earlier has been expanded to form networks that are able to perform goal-directed behaviour. These networks are tightly embedded in their environment, which largely determines the connection structure of the network.

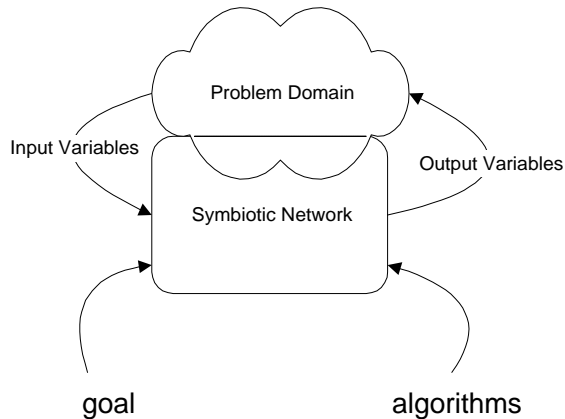


Figure 34: Symbiotic Network

Based on the theory we have presented so far, a systematic approach can be derived to determine whether it is possible to include symbiosis in a network that is able to perform goal directed behaviour in an unknown environment. This approach consists of the following steps:

1. Determine the goal vector $\mathbf{g} = [g_0, \dots, g_n]$. In other words, what is the intention of the network?
2. Select a number of entities whose combined input range covers the goal. The input vector is the organism's *probe* in the environment.
3. Select a number of entities whose combined outputs $\mathbf{Q} = [O_0, \dots, O_n]$ should be able to control the input vector through the environment.

Note that the in- and outputs may be fixed. A construction company could for instance, carry out a project to drill a tunnel under a city and aim to minimise the damage for the buildings. In such a case, the input of the network consists of the sensors that measure ground displacement and the vibrations on the surface. The output entities are the positioning actors of the boring process and the velocity of the drill. This would clearly be a situation where existing entities can be extended with symbiotic behaviour to assist in the drilling process.

Now the basic entities have been identified, the complexity of the environment needs to be further investigated. Recalling that $\mathbf{L} = \mathbf{E} \cdot \mathbf{Q}$:

4. If the environment \mathbf{E} is likely to be complex (many local minima and maximums), then additional (non-symbiotic) entities need to be created to connect the input entities with the output entities. This results in a multi-layered network structure between the probes and the output. A single layer of entities can be used if the environment is fairly straightforward, or when the solution space for \mathbf{g} is large (for instance when the goals are termed as $l_i \leq g_i$ or $l_i > g_i$, instead of $l_i = g_i$). This means that the input entities and the output entities are the same.

Symbiotic Algorithms

The additional layer or layers most likely will contain entities with neural, evolutionary, or other learning capabilities.

The next step is to find out if the environment has qualities that might cause the network to converge prematurely:

5. Find out if, for all values of O_j , a value E_a can be found so that:

$$\sum_{i=0}^n l_i = E_a \cdot \sum_{i=0}^n O_i, E_a > 0, \text{ not necessarily constant} \quad [6.1]$$

This rule means that L becomes larger if O increases and smaller if O decreases. If there are inverse relationships between certain O_i and l_{i+1} of O and L respectively, then these inputs or outputs should be negated. This will have to be based on assumptions or analysis. In the example of the drilling project, for instance, one may assume that the vibrations and ground displacement become larger if the speed of the drill increases, or if the distance of the drill to a sensor becomes smaller. Such heuristics could be sufficient to estimate the effectiveness of a symbiotic network. Detailed analysis however, might prove that the vibrations decrease if the drill's velocity passes a certain value. Such additional complexity may require neural capabilities to be added to the system.

6. Identify branches in the environment. This is also a matter of assumption and/ or analysis. For instance, our drill is likely to contain a branch between a vibration sensor (say East of the drill), the location actor (in Eastern direction), the ground displacement sensor (East) and the velocity of the drill. Note that a branch always contains at least three entities.
7. Identify amplifications in the branches. If rule [6.1] can be restricted to $0 \leq E_a \leq 1$, then knowledge of the connection structure of the network is not required. This is also true if the goals in the branch do not require amplification. For example, the goal of the drill is to minimise vibrations and ground displacement on the surface. The fact that these goals both are zero have the result that the branch identified in step 6 is not going to cause a problem (see GoalSeeker example).
If there are suspected amplifications, then branches containing the amplifying units should be prevented. As branches may not be identified beforehand, amplification in branches may only reveal themselves through premature convergence when the system is running. If so, the goals or the likely relationships between the probes may have to be re-examined. It is also possible to identify the amplifying elements and filter their stress signals from the symbiotic algorithms that are used, save those of their predecessors. This does not limit the possible problems that could be solved with a symbiotic algorithm, but does require additional effort while designing the network, as they impair a generic solution for the given problem and result in "tailor-made" algorithms.
8. The symbiotic algorithms that are used in the network can be anything in between a simple averaging function, such as the previously mentioned STRESS_ONE_BY_ONE, or a complex learning algorithm. A simple averaging function is sufficient when the goals are termed as "larger-than" or "smaller-than" a certain reference. An algorithm with a weight function will be the most versatile, while algorithms that specifically "know" successors will work best. If a system converges prematurely, and the resulting cumulated stress signal

Symbiotic Algorithms

is unacceptable, then the only solution is to identify the symbiots that cause the stress and then find their predecessors. This can only be done by analysing the environment or experiments. If the predecessor is found, then it should get an algorithm that is “tailor-made” for its successor, like NEXT_IN_LINE in the GoalSeeker example.

The best approach is to take each source of stress signals one by one, starting with the symbiot that causes the largest stress. Chances are that one or two specific symbiots are sufficient to “break” the premature convergence and cause the other symbiots to converge properly as well.

Premature convergence in a symbiotic network can easily be monitored by looking at the individual stress signals, as the following will apply:

$$\sum_{i=0}^n |s_i| > 0 \quad [5.2]$$

Improving the network means examining the contributions of the individual stress signals. Every $s_i \leq 0$ means that it converges with a $s_j \geq 0$. This may mean that:

- ◆ an amplification in a branch was overlooked
- ◆ There may be a conflict in the goals of multiple successors of a symbiot

The network can be used this way to teach us things about its environment.

Extremely slow convergence may be tackled by introducing amplifications to the inputs of the entities whose stress signals decrease too slow. This may result in premature convergence, due to propagation of the amplification, so this needs to be done with care.

Symbiotic Algorithms

Summary

In this chapter, a framework was presented that allows us to examine the problem-solving capabilities of a symbiotic network, by comparing it with a neural network and a genetic algorithm. It showed that, provided the limitation of premature convergence can be controlled, applying symbiosis as a problem-solving mechanism can be promising.

Lastly, a structured approach has been given to design networks that could effectively utilise symbiosis in an unknown or complex environment.

7 The Travelling Salesman Problem

Introduction

The Travelling Salesman Problem (TSP) is a classic benchmark problem to test a solution strategy.

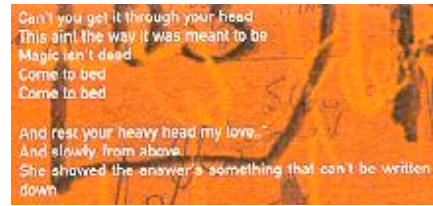
The problem concerns a sales person who has to visit n cities and wants to find the shortest route that connects all these cities (and brings him back home in the end) [Michalewicz 1996, Koza 1992]. The problem is so important, because it is *np-complete*, which means that adding one element (city) to the problem increases the problem space (number of possible paths) drastically. This feature can easily be demonstrated, for if all the n cities are connected to each other, then there are $(n-1)!$ possible paths that can be traversed. Adding one city will increase the number of paths to $n!$, which is unquestionably much more than one extra city deserves.

This trait of *np-complete* problems poses a potential problem for a computer, as it gradually becomes impossible to calculate all the possible solutions when the elements increase. This class of problems is therefore often used to test new strategies that lead to acceptable solutions with considerable less effort. The aim of a good strategy is to do this in *polynomial time*, preferably within n^2 or n^3 calculations. "Good" strategy does not necessarily mean finding the best solution within this time, but acceptable (sub-optimal) solutions are allowed as well. As a "bonus", a strategy for solving one class of *np-complete* problems also means that it can be used for others as well.

In this example, the TSP will be used to create a complex problem space in which a symbiotic network will have to find an optimal solution. The solution strategy is aimed to verify that local knowledge at the level of individual nodes, about the distances to the other nodes, can be used to solve a problem that exceeds the capabilities this local knowledge (a shortest path).

The example aims, within reason, to put as much "intelligence" in the symbiotic algorithms, but it will also address some ways to pre-process

the neighbourhoods so that the network will converge better. This example pushes the theory further from its biological roots and that from Systems Theory. The proposed solution uses the structured approach that was presented in the previous chapter.



"This is the 21st Century", Anoraknophobia, Marillion

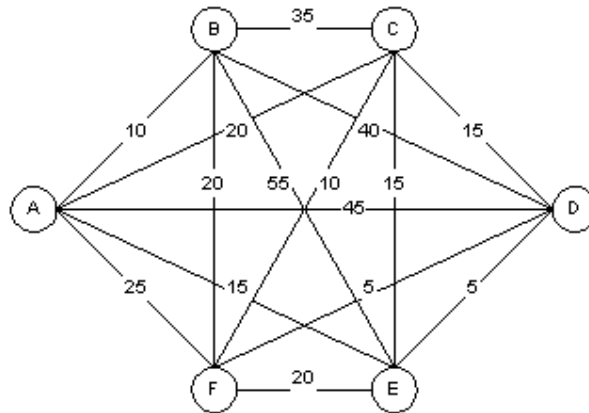


Figure 35 : Graph with 6 nodes (cities)

Symbiotic Algorithms

1. The Solution Strategy

In the strategy that will be used in this example, the sales person travels from town to town and in each town asks a local person for advice to which town should be travelled next. The local persons share stress signals, which are used to adjust their advice to the sales person. The salesperson collects the advised directions and tries to create a valid route from them. This route is then communicated back to the locals (input), who are then able to adjust their advice. This adjustment is determined by the symbiotic algorithm that is used by the "locals".

A run can be described as follows:

```
path = randomly created;
WHILE (termination condition is not reached) DO
BEGIN
  FOR (all nodes) DO
  BEGIN
    index position = local rule(node, path);
  END;
  path = driver rule(index positions);
END
```

Each local makes a suggestion for the next place the sales person can take (local rule), but because he already might have been there, he may decide to take an alternative route (driver rule). The sales person does this by selecting another town on his list, which belongs to the symbiot with the lowest stress signal:

```
add a random node to path;
WHILE(there are still nodes to add)
BEGIN
  select proposed destination from the node;
  IF(destination is available) THEN
  BEGIN
    add destination to path;
    node = destination;
  END;
  ELSE
  BEGIN
    node = node of an available symbiot with lowest stress;
    add node to path;
  END;
END;
```

In this particular example, the driver entity is symbiotic, as it uses the stress signals of others. Other strategies have been used as well during the experiments, including a number that do not use any stress signals at all (the network then uses hybrid entities). All these strategies behave similar, although some have less variance in the convergence process. The reason for this is, that the main task for the sales person entity is to limit the search space from $O(n^n)$ complexity to $O(n!)$ as it filters illegal paths. This task is straightforward and does not have a significant influence for the convergence process.

Symbiotic Algorithms

Each symbiot is assigned to a node of the graph and it only contains knowledge about the distances between its “home town” to the other places. This approach makes the problem *homogeneous*, as each local symbiot will face the same task and so only one strategy (symbiotic algorithm) should be sufficient for the representatives of the nodes.

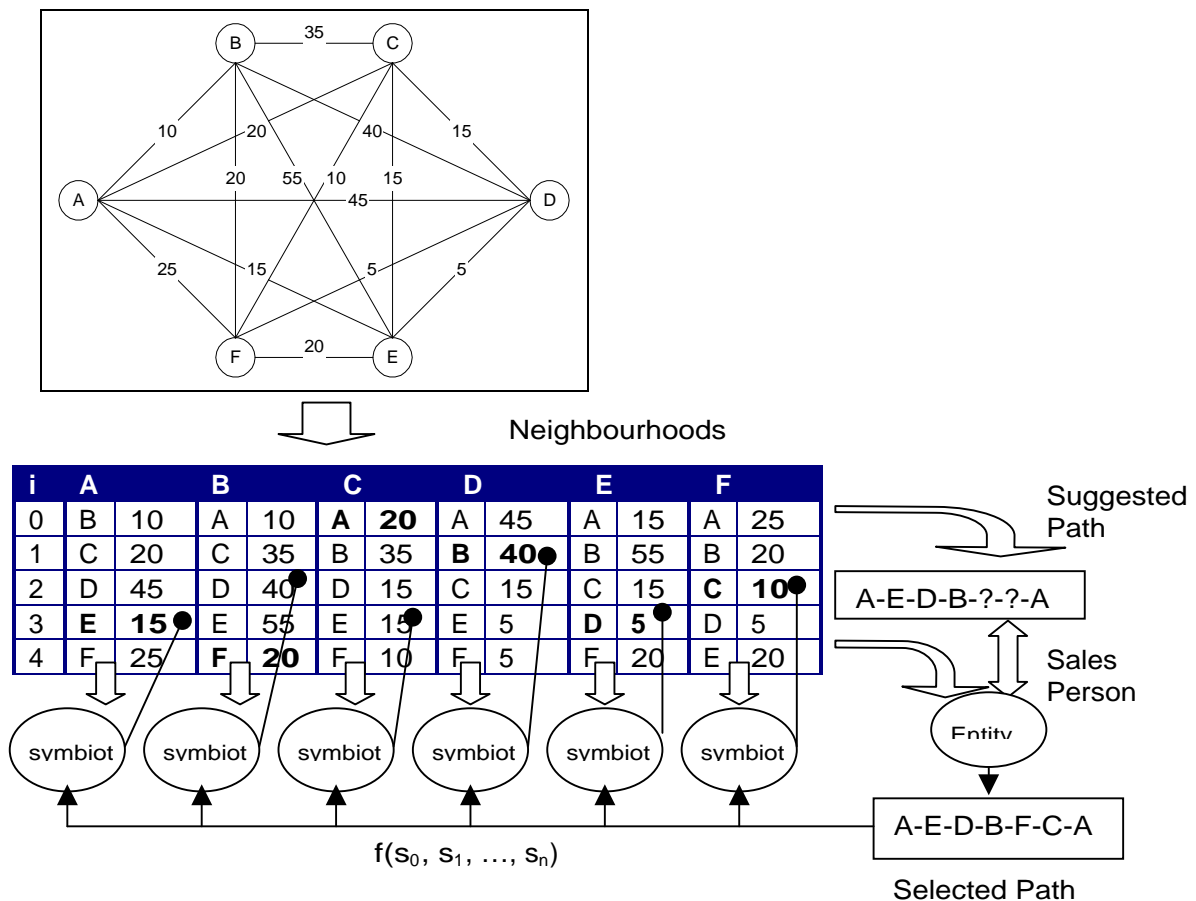


Figure 36: TSP Strategy

The “local” knowledge that is used in this strategy uses the distances of all the cities with respect to one city as base for solving the TSP. This is similar to an edge list representation of a graph [Goodrich and Tamassia 1998] and means that we can use the “raw” data of the graph as base for our solution.

The neighbourhoods are linked to a node (city) of the graph. They contain a list of the line segments of the edges to the other nodes. Each symbiot is connected to one of these neighbourhoods and they can select one of the line segments depending on the value of their behaviour (which depends on the stress signals).

The input therefore consists of a pointer, or index (i), to a line segment of a currently selected path (the boldface line segments in the figure), and the output consists of an index to a proposed new line segment. The proposed line segments are represented by lines with a ball-shaped end in the previous figure. In the figure, the symbiots propose the line segments AE, BD, CE, DB, ED and FC.

During a run, the nodes will point to different indexes, depending on the way the symbiotic algorithm readjusts the advice and on the actual path that is presented as input. The selection is not circular (although this will probably work as well), merely because a linear motion is the easiest to implement. This choice means in figure 36 that when a certain symbiot is pointing to index four, and the advice is to increase, the index will remain on index four. Likewise, an index position on zero with an advice to decrease will keep the index to zero.

Symbiotic Algorithms

An Example Run

In order to give an impression of the way the strategy works, this section will describe an example run, using figure 36. Every iteration of the run will cause the index positions to change (except in the outer boundaries of the range), which results in a very rigid strategy which is not likely to work very well. However, it does make clear how a run will behave. The strategies that have been implemented behave in a similar fashion, although the changes are more gradual. Starting from a random node, a path is created based on the proposed line segments. In the figure, the driver starts to create one from node A: AE-ED-DB-BD....

At node B, a conflict arises and the driver has to choose an alternative from the remaining two nodes C and F.

The node with the lowest stress signal is selected by the driver rule (stress C \equiv line segment CE = 15, stress F \equiv FC = 10), so F is selected:

$$\text{AEDBFCA} \Rightarrow \text{length} = 15+5+40+20+10+20 = 110$$

This path is represented in the table of figure 36, using boldface font.

The chosen path is entered as input to the "locals" and, partially based whether the sales person has or has not followed their suggestions, they will change their advice. This is determined by the symbiotic algorithm that is used. Because the algorithms actually use the index position (i), the result for instance could be that all the index positions of the new path are increased (if this is possible). As the selected path consists of line segments AE, BF, CA, DB, ED and FC, this results in the following proposals: AF(25), BF(20), CB(35), DC(15), EF(20) and FD(5). The lengths of the line segments have been included in brackets.

The driver selects a random start position, for instance D, and creates a new path, for instance:

$$\text{DCBFED} = \text{ADCBFEA} \Rightarrow \text{length} = 45+15+35+20+20+15 = 150$$

FE is chosen because of the two remaining nodes E and A, of the suggested segments EF(20) has a lower stress than AF(25), so the driver rule selects this option.

The proposed path has become significantly larger, which should increase the stress signals. According to the theory, this should mean that a proper algorithm should (eventually) turn back from the chosen direction and go in the direction of the original path. From the line segments of the path: AD, BF, CB, DC, EA and FE the following proposal would be made: AC, BE, CA, DB, EA and FD. Again, the driver creates a valid path, this time for instance starting from E:

$$\text{EACFDBE} = \text{ACFDBEA} \Rightarrow \text{length} = 20+10+5+40+55+15 = 145$$

Because the path has become a bit shorter than the previous one, there is no incentive to correct the direction and the system continues in its downward direction if this is possible:

{AC, BE, CF, DB, EA, FD} \rightarrow {AB, BD, CE, DA, EA, FC}.

Symbiotic Algorithms

Starting from, say F, this results in the following possible iterations:

Start	Line	Path Length
F:	FCEABDF = ABDFCEA ⇒	10+40+5+10+15+15 = 95
B:	BCFDEAB = ABCFDEA ⇒	10+35+10+5+5+15 = 80
C:	CABFDEC = ABFDECA ⇒	10+20+5+5+15+20 = 75
A:		ABEDCFA ⇒ 10+55+5+15+10+25 = 120

As the selected path suddenly has become larger, the direction of the search should change:

{AB, BE, CF, DC, ED, FA} → {AC, BF, CF, DE, EF, FB}.

Starting from, say E, the next path could for instance become:

E:	EFBDCAE	AEFBDCA ⇒ 15+20+20+40+15+20 = 130
----	---------	-----------------------------------

The system now moves away from the best path that was found so far. Clearly, the algorithm used in this example doesn't converge very well, and the intelligence of the driver is not sufficient to converge to a good solution. Apparently, a finer mechanism is needed for the local advisors.

An alternative strategy could define symbiots that output a valid path. Its behaviour could consist of a number of possible operations that can be performed on the path, while the stress signals determine which operation is carried out on such a path. People well versed in Genetic Algorithms (GA) [Koza 1992, Michalewicz 1996] might recognise the cycle that results with this strategy. The stress signal becomes similar to the "fitness" defined in such algorithms, while a symbiot represents an individual in the population that is created. In many ways, a GA can be considered a symbiotic network where connections are made to previous populations. The symbiotic algorithms that are used, are the so-called genetic operators, which are partially based on probabilistic qualities. The combination of the mapping of chromosomes to fitness, combined with a selection process where poor results are discarded, create an environment that complies with the criteria for convergence. Because GAs have been extensively studied, we will not pursue GA-like solutions further.

The three algorithms that have been implemented and tested will be a bit more sophisticated than the one described previously. Instead of turning the direction around based on the total path length (stress), the algorithms will rather operate on individual stress signals, which allows the system to move around local minima and maximums more easily. The general idea however, remains the same.

2. Defining the Symbiots

1: Determining the goal.

The *goal* of the organism is to find the shortest path possible. In the examples used, the goal values of all the symbiots are set to zero. This means that the stress signals will be proportional to the lengths of a line segment in the selected path. The shorter the length, the smaller the stress. This solution strategy has to take in account that there will *always* be a stress signal, even at convergence, as the stress can not become zero. Note that the stress signals can not become smaller than zero as well. This means that the algorithm will have to add some offset that allows the values of the behaviour to become either smaller or larger at a given iteration.

An alternative approach can be to adjust the goal value with a line segment belonging to the shortest path that is encountered at a given iteration. This would result in stress signals becoming zero, when the shortest path is found, but also increase the risk of premature convergence as the stress may become zero at *any* line segment. Another option could be to define a stress value relative to the shortest line segment. Although this would decrease the stress signals, it would still result in stress at convergence, as the shortest line segment is no guarantee for the shortest path.

2,3: Defining output and input

The combined output range of the chosen symbiots should cover the solution space of the TSP. This is equal to the set of all possible paths that can be created. From the strategy discussed in the previous chapter, both the inputs and the outputs consist of an index position to a line segment. The inputs are indexes to line segments of the selected path, while the outputs are indexes to suggested line segments of a proposed path.

4: Analysing the complexity of the environment

Although the graph is likely to contain quite a lot of local minima and maximums, our solution will be based on a one-layer network. The inputs and outputs are therefore directly connected to the same symbiots. The reason for this is partially that we are interested in purely symbiotic networks and not in hybrid networks, and partially because the search is a “better-than” (a previous) path and not an “is-equal-to”. The solution space is therefore larger than one (the best).

5: Determining the relationship between out- and input through the environment

The changes between in- and output are based on *selection* (of the salesperson entity) and not of amplifications or reduction. Therefore the environment complies with rule [6.1], as $\rho = 1$.

6,7: Identifying amplifications and conflicting goals

The input and the output of symbiots are all indexes to line segments. Every iteration the connection structure of the network (which is equal to the selected path) is changed, and this always results in a branch. However, it is not expected that the algorithms will suffer from premature convergence due to the limitations of the network, such as amplification, as the connection scheme of the network can change dynamically. The change causes the effects of amplification to be distributed evenly over all the symbiots over time. The environment does not change the values of in- and output signals ($E_i = 1$).

Symbiotic Algorithms

If one was to further optimise the solution strategy, an improvement could be made to the neighbourhoods that are used by sorting the list of line segments in increasing or decreasing lengths. This *pre-processing* would take $n \log(n)$ extra calculations, but would result in less local minima and maximums in the problem domain. In a way, it can be considered as increasing the local knowledge contained in a neighbourhood.

8: Defining the symbiotic algorithms

Each symbiot outputs a (preferred) index position of a line segment, based on the stress signals. This algorithm determines the ability and success of the networks means to search for the best path. The algorithms that are used are based on:

$$f(\underline{s}) = w_0 s_0 + w_1 s_1 + \dots + w_n s_n \quad [5.1]$$

In this example, we wouldn't be able to use most other algorithms that were presented in the GoalSeeker example, mainly because successors change dynamically. This would mean that the algorithms based on vicinity, like NEXT_IN_LINE and the LOVE_THY... algorithms would not be very useful. STRESS_ONE_BY_ONE is likely to perform reasonably, with a risk for premature convergence. However, for the TSP we will use algorithms that adjust their weights, in a similar fashion as NEURON in the GoalSeeker example. This is done because:

- ◆ this strategy proved to be the most robust
- ◆ we have seen in step 6 that amplifications in branches are unlikely to influence the behaviour
- ◆ of the complexity of the environment

The weights are adjusted to some criterion, and can be a function of the stress.

For the TSP, three of these symbiotic algorithms, which are all similar to NEURON, have been implemented for the "local" symbiots. They are tested against an algorithm that has random behaviour. A random algorithm should be able to generate the best path in approximately n^n iterations.

9: Wrapping up.

In the coming sections, the index i will normally designate that it belongs to symbiot $_i$, while the index j is reserved for the contribution of symbiot $_j$ in the symbiotic algorithm of symbiot $_i$. Index j is therefore mainly used for the stress signals in a symbiotic algorithm.

All the functions are normalised between -1 and 1 , unless otherwise denoted. This means that the values of the problem domain (index positions) and the symbiots (inputs, outputs and stress) are scaled to values within $\langle -1, 1 \rangle$. This is done by dividing the actual values by their maximum value. For instance, the stress signals are proportional to line segments, and scaled by dividing these line segments by their maximum value:

$$s_i = (\text{length of line segment}) / (\text{max length of line segment}) \quad [7.1]$$

The inputs and outputs of the symbiots are index positions, but internally they are also scaled between $\langle -1, 1 \rangle$.

Due to the choice of [5.1], the complexity of the algorithm will always be in order of $O(t.n^2)$, with t being the number of iterations needed to scan the n symbiots. This fact makes the presented solution poorer than known algorithms, such as Kruskal's algorithm, which is known to have a complexity of $O(n.\log(e))$, with e being the number of edges in a graph [Goodrich and Tamassia 1998].

Symbiotic Algorithms

A simple converging strategy (TSP)

The input of each symbiot consists of an index position of a line segment in the path that was chosen by the sales person. In this algorithm, called TSP, the index position belonging to a line segment of the best path known at a given iteration is stored. This path can be found due to the choice of stress signals, as their sum is proportional to the total length of a path:

$$\min(s_{tot}^t); s_{tot} = \sum_{i=0}^n s_i^t, s^t \text{ is stress at iteration } t \quad [7.2]$$

The index position is determined by the index position i_i of a symbiot $_i$. The index i_i^* that belongs to a best path at a given iteration is stored as well, and used to adjust the weights:

$$\Delta w_j = \rho \cdot (i_i - i_i^*); 0 < \rho \leq 1 \quad [7.3]$$

$$i_i \equiv w_i \cdot \sum_{i=0}^n s_i \quad [7.4]$$

Formula [7.2] results in a tendency of the symbiot to move towards the best solution that was found so far. Due to [7.3], the behaviour (index) also has the property of changing less when the currently selected path is small. Therefore, line segments belonging to shorter paths are reselected more often, which allows more paths to be created with them, which *probably* are short as well.

The network converges when $\Delta w_j = 0$, because of the fact that $s_i > 0$. Therefore, this algorithm has converged when the output values of all symbiots are advising the index of a line segment of the best path that was found.

The approach of this algorithm is comparable to the one described in the previous section, with the difference that the changing of the index locations is slower and differs per symbiot. The result is that the generated paths “curl” through the local minima of the problem space like a snake, resting longer in short line segments and quickly moving away from large ones. For instance, we return to our previous example and, start with AE(15), BF(20), CA(20), DB(40), ED(5) and FC(10), the numbers between brackets being the line segments. The first path that is created automatically becomes the “best” path found so far. The line segments belonging to this path (which is similar to the example run) will therefore determine the direction of the index positions for the next iteration(s), due to [7.2]. Of course, this will only start with the next iteration.

$$\text{AEDBFCA} \Rightarrow \text{length} = 15+5+40+20+10+20 = 110$$

AE(15), BF(20), CA(20), DB(40), ED(5) and FC(10) become the target directions. From these line segments, DB causes the most stress, so this index position is likely to change first, because of [7.3], for instance in:

AE(15), BF(20), CA(20), DC(15), ED(5) and FC(10).

Starting in D, like in the example run, we could get the following path:

$$\text{DCAEFBD} \Rightarrow \text{length} = 15+20+15+20+20+40 = 130$$

This path, consisting of AE(15), BD(40), CA(20), DC(15), EF(20) and FB(20), is worse than the previous, so the line segment of the previous path will remain the targets:

Symbiotic Algorithms

Best Path	AE(15)	BF(20)	CA(20)	DB(40)	ED(5)	FC(10)
Current Path	AE(15)	BD(40)	CA(20)	DC(15)	EF(20)	FB(20)

From these line segments, the index of node B has “strayed” the most, namely from F to D, so this one is likely to change first due to [7.3]. BD will also change due to [7.4], as this is the largest line segment. The change, of course, is in the direction of the line segments of the best path, for instance AE, BE, CA, DC, EF, FB. Starting from E, as in the example, we could now get:

EFBADCE ⇒	length = 20+20+10+45+15+15 = 125					
Best Path	AE(15)	BF(20)	CA(20)	DB(40)	ED(5)	FC(10)
Current Path	AD(45)	BA(10)	CE(15)	DC(15)	EF(20)	FB(20)

BA is now likely to change to BC, because of [7.3], while AD(45) is now likely to change to AE because of [7.4], resulting in AE, BC, CE, DC, EF, FB. Starting from F, as in the example:

FBCEADF ⇒	length = 20+35+15+15+45+5 = 135					
Best Path	AE(15)	BF(20)	CA(20)	DB(40)	ED(5)	FC(10)
current Path	AD(45)	BC(35)	CE(15)	DF(5)	EA(15)	FB(20)

Although the example seems to show otherwise, this continuous reshuffling of long line segments and far index positions causes the system to converge more and more towards the optimum path. Note that the more the system converges, the less it matters with what node is started, as the suggested line segments move towards a valid path. Therefore, at convergence the driver logic is no longer needed.

Including probability (Weighted)

The second algorithm that is used on the TSP has additional random behaviour that increases the variance of the paths that are tried out. The basic algorithm uses a normalised total stress. Bearing in mind that the range of $s_i = <0,1>$ because of [7.1], this means:

$$s_{\text{norm}} = 1/n \cdot \sum_{i=1}^n s_i \quad [7.5]$$

In Formula [7.5], n is the number of nodes – 1 in the graph. The formula has the effect that the range of s_{norm} is $<0,1>$ as well.

For each node (symbiot), only the weights of line segments belonging to a path that is selected are adjusted, according to:

$$\Delta w_j = \rho(1 - s_{\text{norm}}); \quad 0 < \rho \leq 1 \quad [7.6]$$

This formula only applies for the index position j of symbiot _{i} , belonging to a line segment of the path that is chosen. Formula [7.6] has the effect that the weights of line segments of better paths that have been selected increase more than those of poor ones. The idea is that the success of the path as a whole is reflected in the weights of the line segments of that path. The weights of line segments that are not selected do not change. Therefore:

1. short paths increase weights relatively more than large paths

Symbiotic Algorithms

2. The weight of a line segment increases more if the line segment is selected often

The selection of a new line segment is done by generating a random number r between zero and the total of the weights for each symbiot:

$$w_{\text{tot}} = \sum_{i=1}^n w_i; 0 \leq r \leq w_{\text{tot}} \quad [7.7]$$

The weights of each index position are then added one by one (starting from zero) until the index m is found where:

$$\sum_{i=1}^m w_i \geq r \quad [7.8]$$

This index, which will always be equal to or smaller than n , is then used as output. This determines the behaviour of the system, and therefore this algorithm does not comply with [4.14]. The result is that line segments with the largest weight have the biggest probability of being selected.

Consider the previous example and take an initial situation where the weight of each line segment is equal to 0.1 and that:

- ◆ $s_i = (\text{length of line segment}) / (\text{max length of line segment})$ [7.1]
- ◆ The maximum length of a line segment is 100
- ◆ $\rho = 0.2$

The maximum length of a path with six nodes is 600. If we now take the initial path of the example AE(15), BF(20), CA(20), DB(40), ED(5) and FC(10), then total length of the selected path is:

$$\text{AEDBFCA} \quad \Rightarrow \text{length} = 15+20+20+40+5+10 = 110$$

Dividing the path length by the maximum length of the path is another way of calculating s_{norm} in [7.5], so formula [7.6] now becomes:

$$\Delta w_j = 0.2 (1 - 110/600) \approx 0.16$$

This weight is added to the line segments of the path, which were initially 0.1:

line segment:	AE	ED	DB	BF	FC	CA
weight:	0.26	0.26	0.26	0.26	0.26	0.26

When we concentrate on symbiot A, the weights of its line segments become:

line segment:	AB	AC	AD	AE	AF	
weight:	0.1	0.1	0.1	0.26	0.1	$\Rightarrow \text{total} = 0.66$

The other symbiots have a similar pattern, where one line segment has increased its weight.

A random number between one and the total weight is generated for each symbiot. For node A this is a value between 0 and 0.66. Evidently, every line

Symbiotic Algorithms

segment has $1/6.6 \approx 15\%$ probability of being selected, except for AE, which has a probability of $2.6/6.6 \approx 39\%$.

If the next path is DCAEFBD, as in the example, then following line segments are updated: AE(15), BD(40), CA(20), DC(15), EF(20) and FB(20). As the length is 130, the weight change of each line segment of the path becomes:

$$\Delta w_j = 0.2 (1 - 130/600) \approx 0.16$$

The weights of AE and CA are 0.26 and the others are 0.1, so:

line segment:	AE	EF	FB	BD	DC	CA
weight:	0.42	0.26	0.26	0.26	0.26	0.42

Looking at node A, we now get:

line segment:	AB	AC	AD	AE	AF	
weight:	0.1	0.1	0.1	0.42	0.1	$\Rightarrow \text{total} = 0.82$

The random number for node A now has a range between 0 and 0.82, and the chance that AE is selected has become $42/82 \approx 51\%$, instead of 39% at the previous iteration. This means that AE is still increasing its chances of being reselected.

Of course, this effect has been strongly exaggerated in the example described above. The actual algorithm uses a maximum path length of 1000, and $\rho = 0.01$. With these values, it takes a few iterations before the weight of a line segment changes significantly enough to have an effect on being reselected. This causes all line segments to have an equal chance of being selected initially.

If we look at path AEDBFCA again and take $\rho = 0.01$, the change of weight in [7.6] would be:

$$\Delta w_j = 0.01 (1 - 110/600) \approx 0.008$$

The effects on node A would be:

line segment:	AB	AC	AD	AE	AF	
weight:	0.1	0.1	0.1	0.108	0.1	$\Rightarrow \text{total} = 0.508$

In this case, the probability that AE is reselected becomes $108/508 \approx 21.26\%$, against 19.69% for the other line segments. This means that all line segments have an almost equal chance of being selected, although the lengths of the paths they belong to can influence this. If path ABFDECA with length 75 would be selected initially, the result would be:

$$\Delta w_j = 0.01 (1 - 75/600) \approx 0.009$$

For node A:

line segment:	AB	AC	AD	AE	AF	
weight:	0.109	0.1	0.1	0.1	0.1	$\Rightarrow \text{total} = 0.509$

Which leads to a probability of 21.42% that AB is reselected.

Symbiotic Algorithms

A disadvantage of this algorithm is that the initial values of the weights affect the selection process. If the weights are initially quite large, then it takes a long time before the changes between the line segments become significant. On the other hand, a very low value would benefit the first selected line segments tremendously (initial zero values would even give the first selected line segment a probability of 100% of being reselected!). Experimentally it proved that an initial weight value of $10 \cdot p$ gives a good result.

Line segments of successful paths increase their weights much more than poor ones. Therefore, they will be selected more often. Because they are selected more often, they increase their weights more as well! This means that eventually they out-compete other line segments, so that the system converges. Note that, just like the previous algorithm, the system will converge to a valid path

This symbiotic algorithm uses *probability* to favour successful paths. This approach uses a cycle similar to that of a genetic algorithm [Michalewicz 1996, Koza 1992]. Each cycle generates a population of one path, whose fitness value is calculated through the total stress signal (which is proportional to the total length of the path). The operations that are carried out result in either the selection of a current line segment for the next population (relatively large weights) or a new one (mutation). This implementation however, does not use a crossover operation. The algorithm is called "weighted".

An Optimising Strategy (Neuron)

The third algorithm that has been tested also calculates the total stress as in [7.5], but now the normalised index i^* of the line segments belonging to the best path found so far is stored, just like TSP. The index is normalised by dividing the index position by the maximum index position. The stress is normalised through [7.1].

The weights are adjusted according to two rules, namely a *neighbourhood rule*:

$$\Phi(i, i^*) = 1 - \varepsilon|i - i^*|, \quad 0 < \varepsilon \leq 1 \quad [7.8]$$

Symbiotic Algorithms

and a *Hebbian learning rule*:

$$\Delta w_i = \Phi(i, i^*) \cdot (\rho \cdot s_i - w_i), \quad 0 < \rho \leq 1 \quad [7.9]$$

This algorithm operates in a similar fashion as the simple converging strategy (TSP) described earlier. The neighbourhood rule is similar to [7.3], and benefits line segments that are near the best line segment found at a given iteration. The Hebbian learning rule behaves like [7.4], slowing the change of weights down for low stress signals. The main difference with TSP is that the two rules are more intrinsically connected. An example run would be similar to the example of TSP. For instance, consider AEDBFCA again (the maximum line segment, as in the previous examples is 100, the weights are initially zero and the maximum index position is 4):

AEDBFCA \Rightarrow	length = 15+5+40+20+10+20 = 110					
line segment:	AE	BF	CA	DB	ED	FC
index:	3	4	0	1	3	2
normalised index:	0.75	1	0	0.25	0.75	0.5
$\Phi(i, i^*)$ ($\varepsilon = 0.1$):	1	1	1	1	1	1
length of line segment:	15	20	20	40	5	10
s_i (max length = 100):	0.15	0.20	0.20	0.40	0.05	0.10
Δw_i ($\rho = 0.1$):	0.015	0.020	0.020	0.040	0.005	0.010
w_i:	0.015	0.020	0.020	0.040	0.005	0.010

$\Phi(i, i^*)$ is calculated by subtracting the normalised index position of the current path by that of the best path found so far. In this case, these paths are equal, so $i = i^*$ and therefore $\Phi(i, i^*) = 1$. The weights are calculated fairly straightforward, bearing in mind that the stress is calculated by dividing the lengths of the line segments by their maximum length.

The next path that is selected results in:

ADCBFEA \Rightarrow	length = 45+15+35+20+20+15 = 150					
line segment:	AD	BF	CB	DC	EA	FE
index:	2	4	1	2	0	4
normalised index:	0.5	1	0.25	0.5	0	1
$\Phi(i, i^*)$ ($\varepsilon = 0.1$):	0.975	1	0.975	0.975	0.925	0.95
length of line segment:	45	20	35	15	15	20
s_i (max length = 100):	0.45	0.2	0.25	0.15	0.15	0.2
$\Delta w_i, \rho = 0.1$:	0.029	0	0.001	-0.024	0.009	0.01
w_i:	0.044	0.02	0.021	0.016	0.014	0.02

For node A, for instance, $i^* = 0.75$, because the previous path with line segment AE is shortest. $\Phi(i, i^*)$ of AD in the This path therefore is:

$$\Phi_A(i, i^*) = 1 - 0.1|0.5 - 0.75| = 1 - 0.025 = 0.975$$

The stress s_i of 45/100 = 0.45 according to [7.1] and therefore the weight change is:

$$\Delta w_i = 0.975 (0.1 \cdot 0.45 - 0.015) = 0.029$$

Symbiotic Algorithms

0.015 being the weight that was calculated in the first iteration, which results in a new weight of $0.029 + 0.015 = 0.044$.

BF was selected previously and therefore has a weight change of:

$$\Delta w_i = 1 \cdot (0.1 \cdot 0.2 - 0.02) = 0$$

Note that another line segment with the same length will also give this result, as this alternative is considered just as good.

It is a bit difficult to visualise the translation of weights to a new proposed index position, but essentially the weight *change* is used to create an offset from the currently selected line segment. Line segment AD has a positive value of Δw_i , which would make the proposed index increase, eventually resulting in a proposal to select AE. This is logical, as AE belongs to the best line segment found so far. In the same manner, we can see that DC has a negative value of Δw_i , resulting in an eventual proposal of DB. Of course, BF remains at its current position.

The neighbourhood function has the effect of stopping the weight from changing, the closer an index position gets to its preferred value. Node E is a good example of this. Initially ED(5) is selected, to be followed by EA, which is a quite large jump when looking the index positions. Because the new path is not an improvement, the neighbourhood function punishes this choice by calculating a low weight change. This means the proposed line segment will tend to stay in the area of ED. Nodes A and D behave in an entirely different fashion. The change from AE→AD and DB→DC are minimal, and so the neighbourhood function calculates a value close to one, leaving the learning rule to mainly determine the change of weight.

Based on the weight changes, the proposed line segments will probably be something like: AE, BF, CB, DB, EA and FE, where nodes A and D are the only ones to have changed their advice.

People well versed in Neural Networks will probably recognise the similarity of the rules with those of a Kohonen Self Organising Feature Map [Hassoun 1995]. However, the implementation of the neighbourhood function that is used here is simpler than the one usually used in a Kohonen SOFM, which uses an exponential function.

The neighbourhood rule is often described as acting like a fish-trap, initially allowing a large area of the problem domain to be scanned, but gradually decreasing the range to zero. The algorithm is called "Neuron".

Results

The algorithms have been tested in JAVA, using a graph with nine nodes. Nine nodes result in $(9-1)! = 40320$ possible paths, which are calculated in order to give a "score" to the algorithms. This score is based on all the paths with a dissimilar path length, and scaled between 0 and 10000. A score of 10000 denotes a path with maximum length, score zero a path with minimum length.

Symbiotic Algorithms

The test, of which appendix B and C show the results, consists of a *run* where:

- ◆ a graph is created
- ◆ a symbiotic network tries to find the shortest path in a number of *iterations*.
- ◆ Each iteration, a total of n symbiots carry out a symbiotic algorithm that tests n stress signals, so an iteration consists of n^2 *calculations*.

A total of 100 runs have been carried out, consisting of 100 iterations. With nine nodes, this means that $9 \cdot 9 \cdot 100 = 8100$ calculations have been carried out with each run. During the run, every path that is better than the previous is logged. This is displayed as “dots” in the graph and as calculation-score pairs in the raw data (appendix C). This means that at the beginning of a run (number of calculations is low), many paths are logged. This gradually becomes less as the number of calculations increase.

All four algorithms described previously have been tested. They are represented with different colours in the chart of appendix B and with different columns in the table of appendix C. The calculations are displayed in the x-axis, and the score in the y-axis. The dots in the chart are connected, but this has no meaning. It has been done to visualise the range of the data per symbiotic algorithm, more or less to shade the area.

The constants ρ and ε which were described in the theory of the previous sections both are 0.01 in the test that was carried out. These values were determined experimentally and resulted in optimal convergence. Larger values tend to slow convergence down, while lower values result in a larger deviation of the values.

The graphs show that at the start of a run, there is an enormous fluctuation in the scores. This slowly becomes less, and the scores converge to their eventual values. For RANDOM this is the average score (5000), while the other algorithms converge to zero. RANDOM and Weighted however also have a large variance, around these values, while TSP and Neuron gradually become less active and have a smaller variance as well.

To demonstrate the effect of pre-processing the neighbourhoods in increasing length of line segments, the same run is repeated with this modification. The results are graphically depicted in appendix D. In order to give the random algorithm the benefit of this pre-processing, this algorithm was modified to always select a random line segment *lower* than the one used in the current path. This results in a tendency to select lower line segments and, as the results will show, dramatically improves the behaviour of RANDOM.

The charts in the appendices show that all algorithms, including Random, display some convergence, but that the three algorithms that utilise the stress signals actually converge to the best paths. TSP and Neuron are the best algorithms, typically finding the best 10% of paths well within 500 calculations, which suggests a complexity of $O(n^2)$. Optimising the neighbourhood appears to mainly benefit Random, which proves that the TSP and NEURON are sufficiently robust to deal with “raw” data and therefore do not really this pre-processing.

Weighted does not benefit from pre-processing either, which seems logical, as the probabilistic qualities of this algorithm will not be able to sufficiently utilise this feature. Weighted overall behaves poorly. This can be explained by the fact that its operation is similar to a malformed genetic algorithm that uses population of only one individual (path) per cycle and does not have a proper crossover operator. The algorithm does however manage to converge eventually.

8 Epilogue

Symbiosis is a “loose” relationship between two or more entities.



This is the 21st century
Flash to crash and burn
Nobody's gonna give you anything
For nothing in return
There's a man up in a mirrored building
And he just bought the world

"This is the 21st Century", Anoraknophobia, Marillion

‘Loose’ in this sense means that a beneficial interdependency can be identified between entities that are capable of independent behaviour, but choose to stick together because of some mutual advantage. It is logical that when expanding the results to a network of n of such loosely coupled entities, that the resulting network becomes loosely coupled, or *thin*, as well.

This essay started to examine symbiosis and ended up with networks that were ‘somehow’ connected, through an unknown environment that influences the in- and outputs of the entities in a ‘certain’ way.

The most surprising outcome of the research probably is that, given such vague premises, some very interesting properties can still be deduced from a thin network. For one, it isn’t difficult to get the inputs of the network to converge to an average value of a given goal function if certain criteria are met. If this is sufficient, the entities that are needed can be kept quite simple, and the network structure straightforward.

The price one pays for the loose coupling of entities becomes evident when the performance that is required exceeds average values, or when the goal criteria are more specific. However, even then the additional requirements remain rather transparent.

When the requirements become highly specific and maximum control is needed to achieve a certain goal, the additional requirements result in networks that are similar to control systems or neural networks.

The results can be expanded to networks of infinite size, or to networks that dynamically change the amount of the its entities or the connection structure between them.

Symbiotic networks, like neural networks, utilise a natural phenomenon that, according to Alan Turing, “global order can arise from local interactions” (Hassoun 1995). In neural networks this has been utilised where models of brain- and nerve functions have been simulated in an artificial environment. Symbiotic networks expand this with networks that control behaviour of all sorts, in stead of the normally homogenous neural nets (networks that use one type of entity). Another notable supplement of symbiotic networks is the fact that the environment of entities is part of the network’s structure, instead of being a rigidly discernible element. Although this clearly comes with a price, it has proven not to stand in the way of Alan Turing’s “global order” and, in the case of biological organisms, has been the *only* way to achieve this. Studying symbiotic networks forced us to examine the environment of these networks, not as an external source of influence, but as an integral part of the system. This view may help in giving artificial systems a common ground with biological ones, even though it may mean giving up a bit of control on it’s behaviour.

The third supplement of symbiotic networks is that it demonstrates that “loose” networks can to a certain extent deal with complexity. The requirement of two- or three layered networks, hidden layers, back propagation, etc., was never required, although probably at the expense of poorer performance than a neural net. If a neural net can be compared to the brain and the nervous system and a control system with our motorial and sensory system, then a symbiotic network can be compared with tissues of cells, or maybe even bodily organs. Nature seems to have concluded that a central control is one of the best ways to overcome complexity of a higher degree, and the limitations we have encountered seem to agree with such a strategy. It will be a very interesting next step to see how combinations of neural nets and symbiotic nets can operate, connected in similar ways our tissues are connected to our nervous system. It

Symbiotic Algorithms

may also give clues how different specialised neural nets could be connected to utilise the joint benefits of their specialisation. An interesting aspect in this light, is the promise it may hold for distributed intelligence.

Back to Nature

Although it is not a requirement for this essay, it is interesting to see if the way this research was conducted has provided new insights to evolution theory and natural organisms. It is the author's personal feeling that the attempt to take a rigorous engineering approach to Evolution Theory does certainly provide new insights to this discipline, especially in formally understanding the dynamic aspects of the evolution process. Evolution theory is currently often used to explain static observations in Nature. Many natural phenomena are explained in terms of benefit of a current state of affairs. The rabbit has a white tail because flashing the tail takes the fox off guard, thereby giving the rabbit valuable time to run away. Humans evolved to maximise their brainpower in order to effectively cope with changes in their environment.

It is the author's firm belief that the challenge in evolution theory is to explain evolutionary processes in logical steps, starting with an ancestor and ending with the traits of a known organism. Darwinian selection is the guiding principle for such a convergence. What steps were needed for a proto-rabbit to develop a white tail? Why didn't the fox evolve to compensate this feature?

In the specific case of symbiosis, it is the author's belief that such an approach has helped to fill many gaps in the time scale between complex chemical structures and (neural) networks. An engineering perspective could probably help to fill the other gaps between the milestones on the evolutionary time scale as well.

9 Conclusion

A quote by Marillion: "Would you want To have kids Growing up Into what's left of this?"

"This is the 21st Century", Anoraknophobia, Marillion

The research that forms the base of this thesis has concentrated around symbiosis. Using an engineering perspective on evolution theory a framework was set up that could identify the properties required for symbiotic behaviour, which was then used to set up a model that displayed symbiotic behaviour. The goal of this exercise was to see if, and how, symbiosis can be applied in an artificial environment such as a computer system.

The model resulted in the following:

- ◆ it identifies a symbiotic entity as a rational agent
- ◆ it presents a view on the environment of a system that can be described as graph of *neighbourhoods*, units of local influence on the entities of a system
- ◆ it demonstrates that a network of *symbiots*, symbiotic entities with a primary transformation process, can display converging behaviour, where the system as a whole is able to reach a certain goal

This converging behaviour does come with a few restrictions and limitations. The most striking being that:

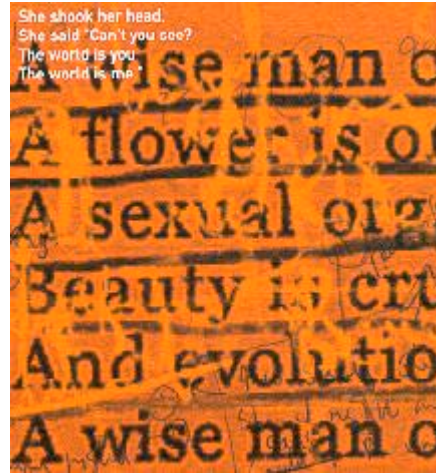
- ◆ the system can converge to other values than the goal values, namely to averages of the individual goals
- ◆ it is very difficult to improve this average behaviour if the connection structure of symbiots and neighbourhoods is unknown, or if there are amplifications in *branches* in the network, which are sequences of symbiots.

A number of examples were conceived to demonstrate aspects of the theory and demonstrate that the symbiotic network can be used to solve a computational problem. These examples added the following insights:

- ◆ A symbiotic network allows distribution of intelligence due to the inherent parallel nature of the network and the relatively loose coupling between the entities
- ◆ A symbiotic network shows rudiments of social behaviour, which allows a *view* on solving certain problems that comply with how social animals, like humans, regard our environment
- ◆ A structured approach is possible to examine a complex environment to see if a symbiotic system can reach its objectives in it. Monitoring its success or failure can help in understanding its environment better.
- ◆ A symbiotic network can provide a good solution strategy to solve a complex problem like the TSP
- ◆ increases our knowledge on how global order can emerge from local interactions

Especially combined systems consisting of symbiotic- and neural nets are an interesting area for further investigation.

Bibliography



1. Allen, Gerald R. & Steene, Roger, "*Indo-Pacific Coral reef Field guide*", Tropical Reef Research 1999
2. Bruegge, Bernd & Dutoit, Allen, "*Object Oriented Software Engineering*", Prentice Hall 2000
3. Breedveld & van Dixhoorn, "*Technische Systemleer*", Twente University 1988
4. Dawkins, Richard; "*the Selfish Gene*", Oxford Press 1984
5. Dennet, Daniel; "*Darwin's Dangerous Idea*", Touchstone 1995
6. Gamma, Erich *et al.*; "*Design Patterns*", Addison-Wesley 2000
7. Goodrich & Tamassia, "*Data Structures and Algorithms*", Wiley & Sons 1998
8. Hassoun, Mohamad H., "*Fundamentals of Artificial Neural Networks*", MIT Press 1995
9. Holzner, Steven, "*JAVA black Book*", Coriolis Group 2000
10. Kasabov, Nikola K.; "*Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*", MIT Press, 1998
11. Kauffman, Stuart, "*Investigations*", Oxford University Press 2000
12. Koza, John R., "*Genetic Programming*", MIT Press, 1992
13. Margulis, Lynn; "*the Symbiotic Planet*", Contact 1999
14. Michalewicz, Zbigniew, "*Genetic Algorithms + Data Structures = Evolution Programs*", Springer Verlag 1996
15. Moriarty, David E. & Miikulainen, Risto, "*Efficient Reinforcement Learning through Symbiotic Evolution*", University of Texas 1994
16. Open University NL; "*Systemen en hun Besturing*", Open University 1992
17. Open University NL, "*Object-Georiendeerd Programmeren met JAVA*", Open University 1999
18. Open University NL; "*Regeltechniek*", Open University 1990
19. Pinker, Steven; "*How the Mind Works*", W.W. Norton 1997
20. Wooldridge, Michael; "*Reasoning about Rational Agents*", MIT Press 2000
21. Watson & Pollack, "*How Symbiosis Can Guide Evolution*", Brandeis University 1999

Symbiotic Algorithms

List of Terms Used in this Document

Agent	Entity that acts on the environment it inhabits
Behaviour:	A causal relationship between the input variables and the output variables of a system or of an entity
Black box Approach:	A description of the behaviour of a system or entity, purely in terms of its input- and output variables, without regard of the structure of the system or entity
Branch	A sequence (chain) of connected entities in a network
Closed System:	System that does not allow influences from outside the system boundary
Convergence	A stable situation in which a system does not change
Convergence Time	Time needed for a system to reach a stable situation
Dynamic Behaviour:	A description of the behaviour as a function of time
Entity, Element:	Any object that can be identified as an individual unit of behaviour
Environment:	Anything that does not belong to the system
Evolution theory:	A scientific discipline that aims to understand how (biological) systems developed based on Darwinian principles
External:	Used to identify a property that only has connection to the environment
Feedback:	An organisation of a system in which (part of) an output signal is reversed to influence the input
Ideal Convergence:	Convergence where $\underline{L} = \underline{g}$
Input Variables:	An external source that affects the state of a system or that of an entity
Model:	A description of a system based on the application of known theories
neo-Darwinism:	Branch of evolution theory that allows no force of creation to explain evolution
neighbourhood:	Part of the environment that is influenced by, or can influence an agent
Open System:	System that allows influences from outside the system boundary
Organism:	1. a living biological system 2. a symbiotic system
Output Variables:	An effect that a system or an entity has on its environment
Path	A sequence of connected elements in a network that start from an external input and end at an external output
Pnuelian Reactive Systems	Systems that are influenced by the environment and, in turn, influence it as well.
Predecessor:	An entity whose output is coupled to the input of another one
Premature Convergence:	Convergence where $\underline{L} \neq \underline{g}$
Primary Transformation	The main transformation of a system's output as function of a system's output
Process:	
Propagation	Used mainly for the effect that a change in the system causes changes in other parts of a network due to the connections
Probe:	The input vector of a system
Process:	A description of apart of a system
Rational Agent	Agent able to perform actions that are in its own best interests

Symbiotic Algorithms

Relationship:	Any clearly identifiable means of influence between two entities
Secondary Process	A process that supports the primary transformation process
State:	A description of the behaviour of a system or entity at a given moment in time
Static Behaviour:	A description of the behaviour that is not influenced by time
Structure:	An coherence between entities and their relationships
Successor:	An entity whose input is coupled to the output of another one
Symbiosis:	A relationship where two entities display mutually adaptive behaviour
Symbiot:	An entity with a primary transformation process, capable of symbiotic behaviour
Symbiotic Algorithm:	A rule that determines the nature of symbiosis of an entity
System:	A collection of entities with a structure of mutual relationships
System Boundary:	Demarcation between a system and it's environment
Travelling Salesman Problem:	A benchmark problem to verify solution strategies with
White Box Approach:	A description of the behaviour of a system, using knowledge about the structure of the system.

Symbiotic Algorithms

List of Symbols

$[v_0, \dots, v_n]$	Vector with $n + 1$ elements
$ v_i $	absolute value of an element v_i
$[a,b>$	Range of a function: $a < v \leq b$
$f(v_0, \dots, v_n)$	f is a function of elements v_0 to v_n
\cdot	multiplication
$v \rightarrow v_i$	the value of v approaches that of v_i
v^*	Best element found so far
Δ	differential operator
Σ	sum of a number of elements
\in	"is an element of"
\notin	"is not an element of"
\approx	"is nearly equal to"
$\underline{\mu}$	Behaviour vector of a network or system
μ_i	Behaviour of symbiot _{i}
e_i	Error offset of symbiot _{i} at convergence
\underline{E}	Environment vector of a symbiotic system
E_i	Neighbourhood _{i} of an environment
\underline{g}	Goal vector of a symbiotic system
i	Usually reserved as index to identify an entity in a network
\underline{I}	Input vector of a network or system
I_i	input of entity _{i}
j	Usually reserved as index to identify a property of an entity with respect to another one
lim	limit of a function
m	subset of entities in a system
n	Number of entities in a system
\underline{O}	Output vector of a network or system
O_i	Output of entity _{i}
r	Normally used for distance
s_i	stress signal of symbiot _{i}
t_c	Convergence time of a system
v^t	Value of element v at iteration t
v_i	element with index i in a vector
\underline{V}	General notation of a vector $[v_0, \dots, v_n]$
\underline{w}	Vector of weight values
w_i	Weight value of element i

“This is the 21st Century”, Copyright Marillion 2001

This is the 21st Century

Lyrics: Hogarth

Music: Hogarth/Kelly/Musley/Rubbery/Trevisanos

(*It may not be only see what he sees, with your eyes*)

A wise man once said
A ticwar is only
A sexual organ
Beauty is cruelty
And evolution
A wise man once said
that everything could be explained with
mathematics
He had denied
His terminology
Now where is the wisdom in that?

I came just as fast as I could
Through the dirty air
Of your neighbourhood
Your name or a grain of rice
I larkin' around my neck
And a head like lead

This is the 21st century
I heard everything they said
The Universe can't feed
Chemicals for God
This is the 21st century
I heard everything they said

A wise man once wrote
That love is only
An ancient insect
For reproduction
Natural selection
A wise man once said
That everything could be explained
And it's all in the brain
We lay on a velvet rug
by the open fire
She blew air on my eyelids
I cried "What's that about?"
As she kissed my hair
One said "There, there."

"This is the 21st century
I heard everything you said
The universe demystified
Astronomy instead
This is the 21st century
Can't you get it through your head
This ain't the way it was meant to be
Magic isn't dead
Come to see
Come to see

And rest your holy head my love.."
And slowly, from above,
One showed the answer's something that
can't be written down

This is the 21st century
Flesh to crash and burn
Nobody's gonna give you anything
For nothing in return
There's a man up in a mirrored building
And he just bought the world

Would you ever
To have kids
Growing up
Into what's left of this?

She shook her head,
One said "Can't you see?
The world is you
The world is me."

Appendix A GoalSeeker Example

Console and model (next page) without decreasing goal values.



Figure 37: Console of the GoalSeeker

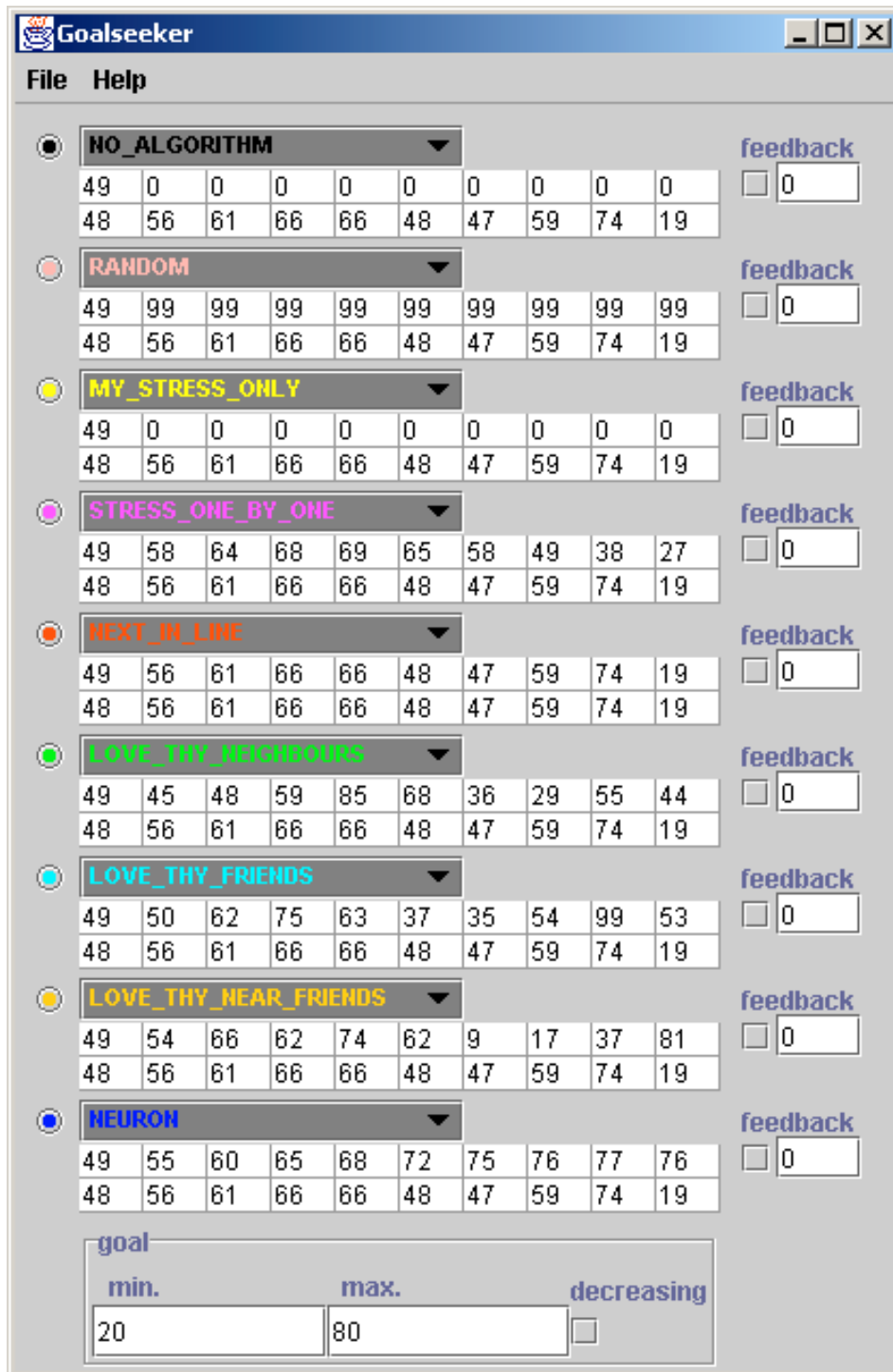


Figure 38: Model of GoalSeeker

Symbiotic Algorithms

Console and model (next page) with decreasing goal values.

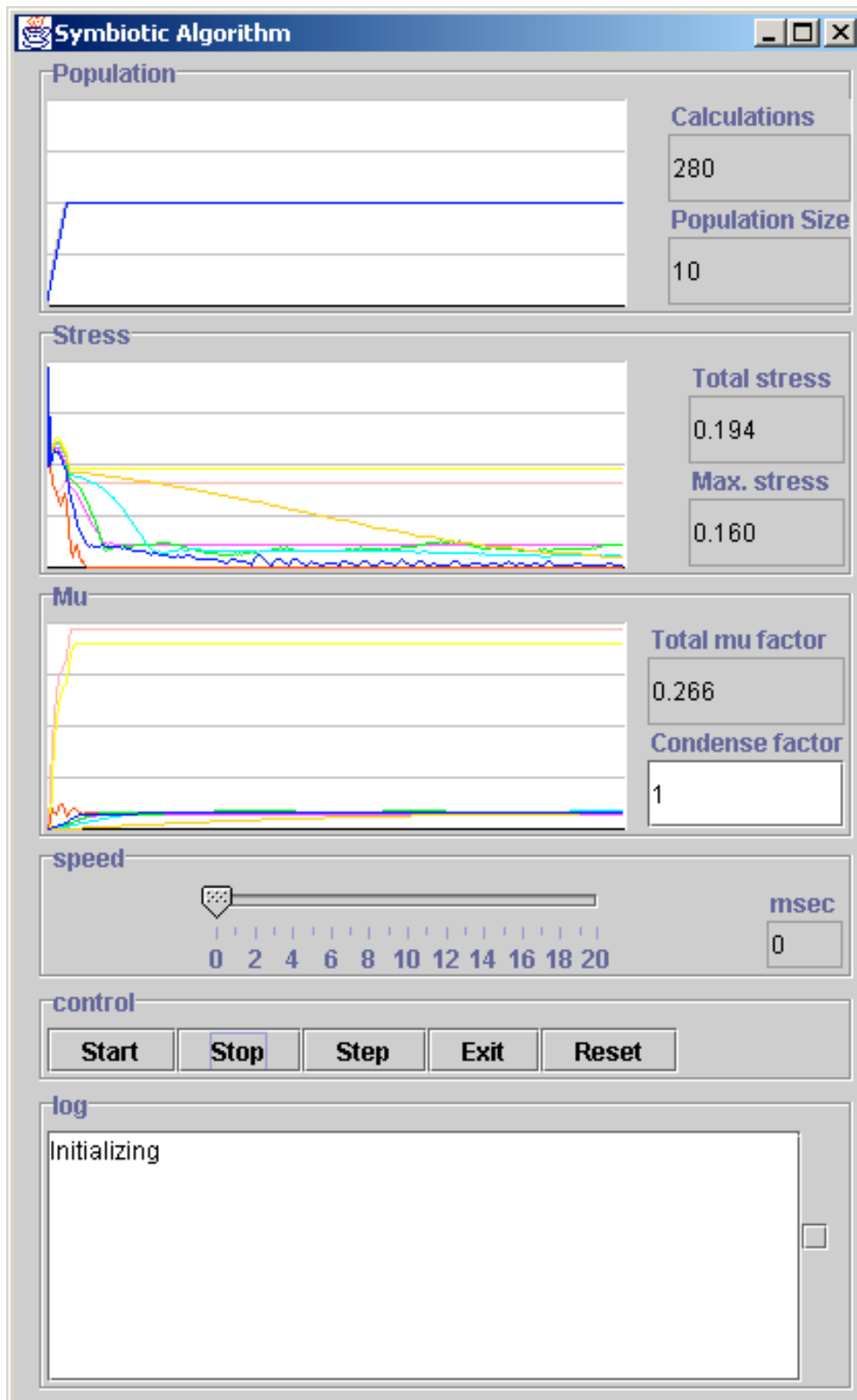


Figure 39: Console of Goal Seeker with decreasing goal values

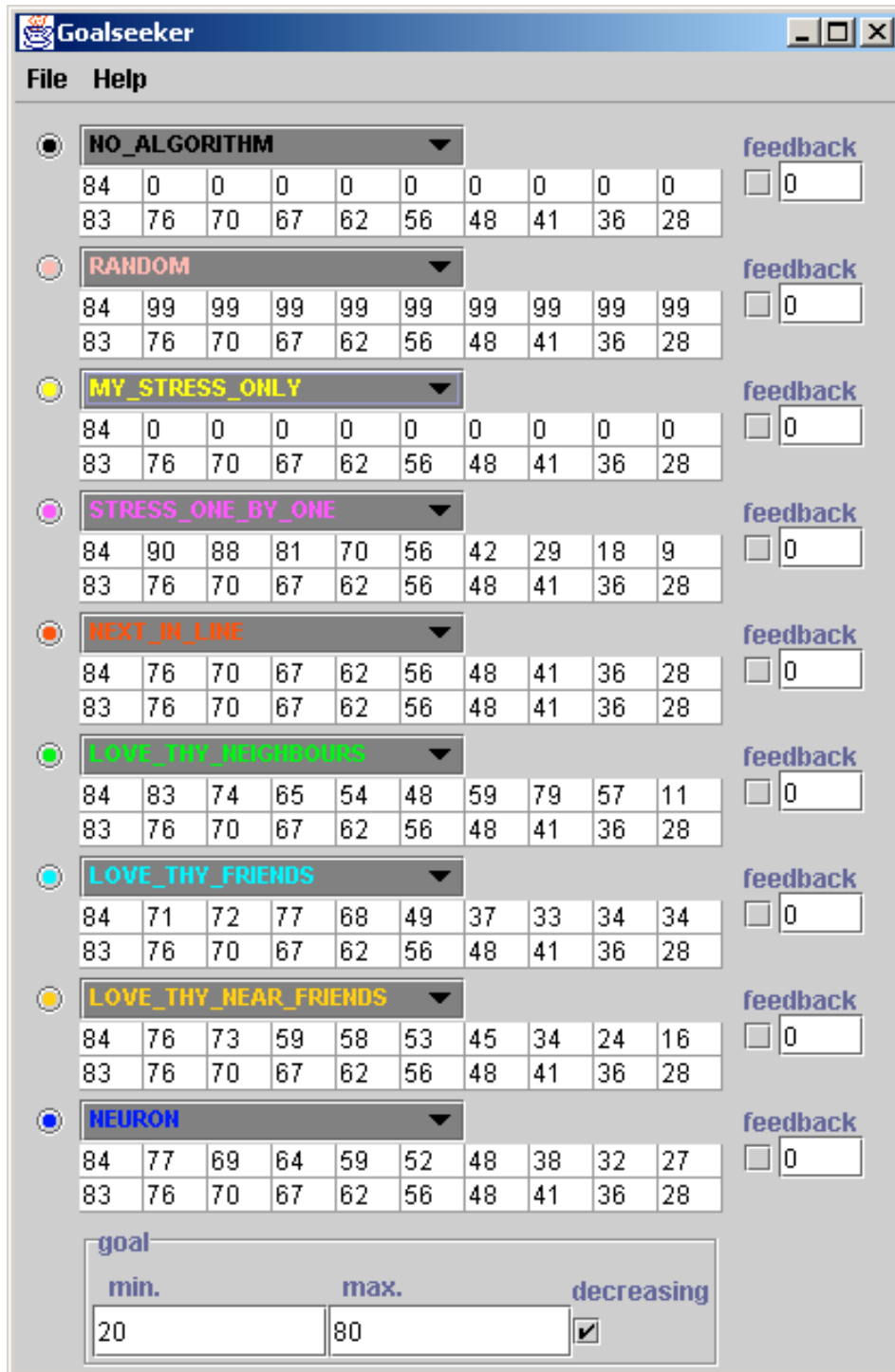
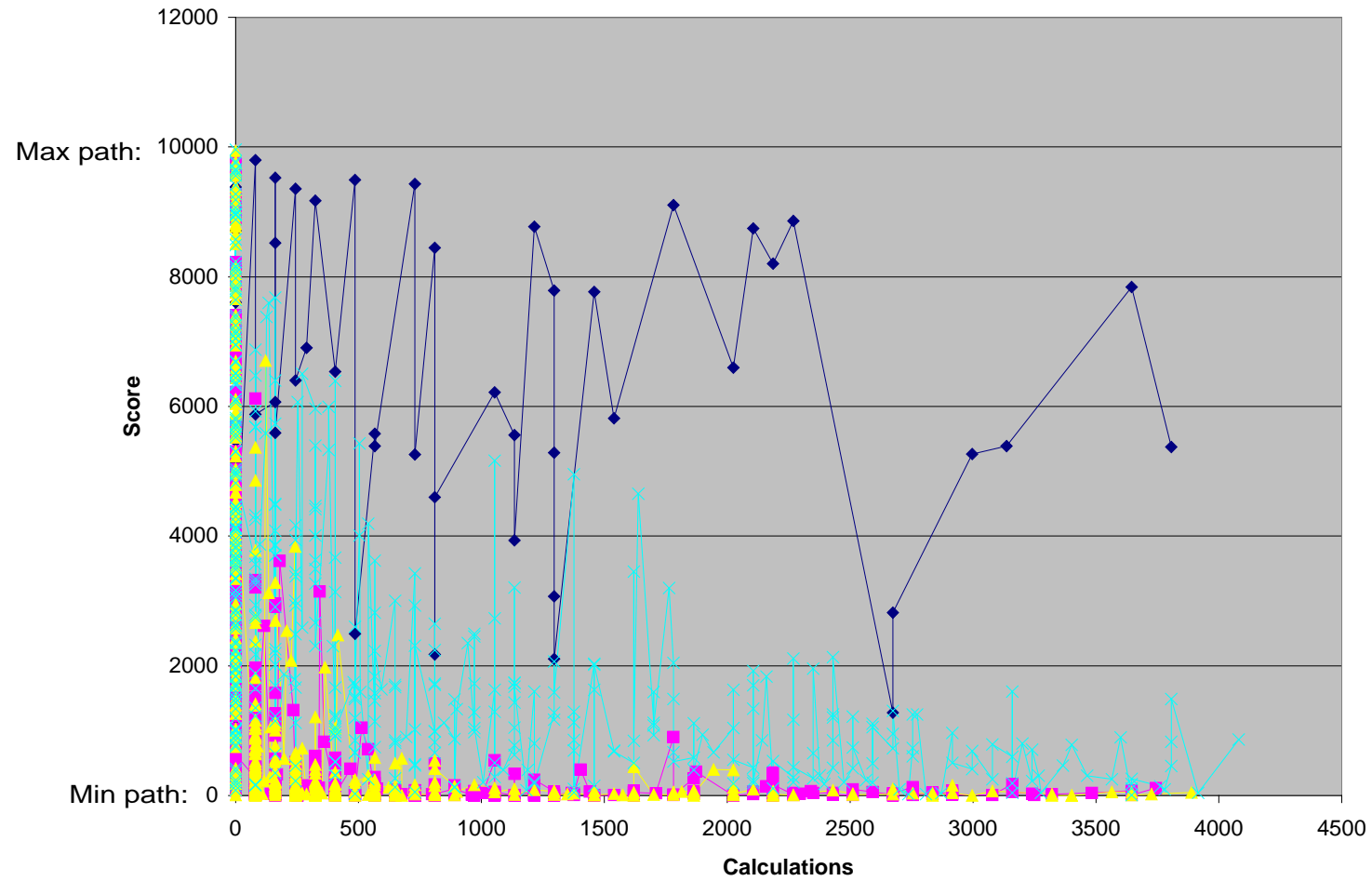


Figure 40: Model of GoalSeeker, with decreasing goal values

Appendix B: Example Run of the Traveling Salesman Problem without preprocessing:

Traveling Salesman Problem



Raw Data:

The following table displays a run of 100 iterations, using four algorithms and a graph with $n = 9$ nodes. With every run, the calculation is shown when a better path is found and the "score" of this path, between 0 for a minimum path and 10000 for a maximum path), compared to an exhaustive search. This score gives an impression of a percentage. Note that the calculation is always a multiple of n^2 (81), as an update is only done after recalculating a path. The iteration therefore is calculation / n^2 . In the data, a score is often calculated at "0". This means that improvements have been made in the first n^2 calculations (the zero iteration). This is because the log only updates the results after an iteration is finished. A calculation is therefore usually a multiple of n^2 , although an occasional "in-between" is found due to the multi-threaded character of the application.

Run	Random		TSP		Neuron		Weighted					
	calculation	score	calculation	score	calculation	score	calculation	score				
1	162	6070	0	4464	81	117	0	1966				
			81	272	1053	115			972	981		
			405	69	1296	60			3159	600		
			972	67	3564	54						
			1215	60								
			1296	58								
			2430	38								
2	0	3345	0	3345	0	3345	0	3345				
			81	50	81	1029			405	921		
			486	25	243	30						
					567	4						
3	0	9562	0	9562	0	9562	0	9562				
			0	3157	0	417			0	3484		
			81	282	81	9			486	1197		
			324	108					891	868		
			891	101					2106	428		
			1377	9								
4	0	1841	0	1841	0	1841	0	1841				
			0	881	81	940			0	858		
			405	280	162	517			1539	696		
			567	1	2025	396			3807	453		
					2916	44						
5	0	5728	0	5728	0	5728	0	5728				
			0	4309	0	5031			0	4460		
			252	0	108	30			270	2590		
									405	314		
									540	254		
6	0	9269	0	1221	0	9335	0	5825				
			2268	8862	81	79			648	3003		
					243	39			405	39	2430	430
					1215	25						
					2106	21						
7	0	5575	0	5575	0	5575	0	5575				
			1134	5555	81	217			81	1813	81	3570
					162	100			162	1128	243	1118
					729	2			243	623	405	526
									324	174		
									648	139		
									810	14		

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
8	0	6515	0	6515	0	6515	0	6515
	0	1721	0	5251	81	1154	81	3305
			81	209	162	209	162	2257
			162	10	324	56	1053	1634
					1458	10	2106	188
9	0	1723	0	1723	0	1723	0	1723
			81	1710	81	325	486	665
			162	1123	567	240		
			243	61	810	45		
			2268	15				
10	0	9065	0	2708	0	8771	0	7646
	162	8523	117	2618	81	409	126	7382
	810	8449	234	1320	243	384	252	6069
	3645	7842	468	414	405	80	378	5324
			1404	402	2187	65	504	4010
			1872	369			1008	203
			3159	177			3276	44
			3744	115				
11	0	8161	0	8161	0	8161	0	8161
	0	3095	81	6	225	2080	0	3837
					675	572	81	1224
					900	62	162	1222
					1575	5	810	995
						3807	826	
12	0	7252	0	7252	0	7252	0	7252
	0	4933	0	74	90	759	0	4992
			81	21	270	36	81	3254
			3078	9			1053	1294
						1296	1280	
13	0	1498	0	1498	0	1498	0	1498
			162	1102	405	1031	405	981
			1053	543	810	540	2592	196
			1863	263	1134	101	3645	7
				2025	62			
14	0	7975	0	7975	0	7975	0	7975
	0	4998	0	2166	81	919	162	6402
			81	382	162	382	243	4164
			486	26	486	63	405	1665
					648	47	972	1284
							1053	283
						2268	280	
15	0	9816	0	645	0	9944	0	9412
	81	9797	162	14	81	14	81	6475
	324	9170	1377	11	2268	4	405	6395
	1215	8774					1053	5161
	1296	7787					1377	4953
	1458	7767					1782	2042

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
16	0	752	0	752	0	752	0	752
			81	21	81	210	2511	419
			648	0	405	186		
					729	164		
					1458	75		
					1863	69		
					2997	0		
17	0	6947	0	6947	0	6947	0	6947
	2025	6598	0	3042	81	2739	0	4123
			81	76	162	766	162	3700
			162	62	324	479	243	2932
			1620	0	567	81	405	1053
					729	69		
					2673	19		
18	0	6445	0	6445	0	6445	0	6445
	0	744	81	64	0	5	0	3108
			891				162	2908
						648	1673	
19	0	2646	0	1004	0	8045	0	934
			2295	24	135	3125	2565	235
					270	727	2736	19
					540	309		
				1350	20			
20	0	7167	0	7167	0	7167	0	7167
	0	5127	0	2288	0	751	0	7129
			81	123	81	384	324	5963
			648	89	324	18	405	3672
			729	13			729	3422
							972	2491
							1377	694
						1944	665	
						2187	111	
21	0	993	0	993	0	993	0	993
			81	871	81	54	1134	778
			324	226	324	41		
			405	19				
		1215	7					
22	0	8947	0	8947	0	8947	0	8947
	0	7100	0	1394	0	6528	0	7781
	81	5875	81	364	81	14	81	5681
	3807	5375	243	262			162	344
			405	107				
			2835	33				
			2916	30				
			3240	24				
			3321	17				
23	0	9545	0	3247	0	887	0	9961
	162	9525	81	471	162	189	81	5922
	486	9490	567	187	810	149	162	4084
	729	9432	729	1	1539	26	324	4010
	1782	9105					405	1236
	2106	8744					2754	732
	2187	8203						

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
24	0	9078	0	7154	0	3827	0	4072
	1296	5289	81	1193	81	425	198	1870
			243	46	162	216	1782	1489
			972	35	243	26	2475	203
			1053	33			2871	35
25	0	1228	0	2039	0	319	0	3158
			81	83	675	2	81	2789
			162	5			1053	2729
			2025	0			1458	2031
							2268	203
26	0	7407	0	7407	0	7407	0	7407
	0	6587	0	3361	0	266	0	5786
	567	5582	81	16	324	131	243	457
			162	3	1782	11		
			1215	1	1863	3		
27	0	6104	0	6104	0	6104	0	6104
			81	1113	81	327	162	1693
			243	349	243	126	405	1206
			405	54	3321	19	1701	1120
			810	37			1863	584
		2511	7					
28	0	8880	0	8880	0	8880	0	8880
	0	6863	81	305	0	8658	81	6875
			162	40	81	448	162	3867
			2673	14	162	40	486	1489
					324	4	567	1139
						2187	533	
29	0	4435	0	4435	0	4435	0	4435
			81	26	81	118	324	4416
			324	5	405	4	567	1578
			729	1			648	850
							1377	100
30	0	6703	0	6703	0	6703	0	6703
	0	3830	0	4332	81	65	0	3058
			81	172	2511	30	405	918
			1296	8				
31	0	9748	0	9748	0	9748	0	9748
	0	2786	0	4453	0	6597	0	782
			180	3612	81	194	729	57
			360	830				
			540	713				
			2160	142				
		2340	69					
32	0	3732	0	3732	0	3732	0	3732
			81	1376	81	2665	81	1883
			162	272	162	65	810	659
			243	209	405	34	1053	506
			324	94	729	17	1458	149
			567	5				

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
33	0	1929	0	1929	0	1929	0	1929
			0	915	0	982	81	1651
			81	434	81	434	1134	626
			324	432	324	218		
			405	253	567	68		
			567	66	810	55		
			729	29	3645	52		
			891	8				
34	0	1029	0	1029	0	1029	0	1029
	0	916	162	141	81	39	1701	929
			243	82	3402	3		
			729	11				
			2268	9				
35	0	2218	0	2218	0	2218	0	2218
			81	50	0	781	81	2174
			648	7	81	279	1863	1116
					162	7	2025	1041
36	0	9064	0	9064	0	9064	0	9064
	0	4328	0	8221	0	5914	0	6332
			81	306	81	83	81	167
			162	278				
			243	7				
		2187	1					
37	0	5505	0	1221	0	9388	0	8576
			81	900	162	200	162	7677
			1620	75	324	195	324	4462
			2430	15			1701	1599
38	0	3953	0	7033	0	3937	0	2342
			81	928	81	44	486	660
			324	323	2835	34	3645	174
			648	28				
			2025	4				
39	0	4387	0	4387	0	4387	0	4387
			81	39	405	122	324	3492
			648	1	540	13	1215	1603
							1377	855
							1782	664
						2997	408	
40	0	3998	0	5249	0	4530	0	1678
			81	480	81	878	1458	1633
			405	236	243	298	2511	1219
			648	72	891	144	2754	605
					1296	60		
					1620	1		
41	0	417	0	417	0	417	0	417
			162	49	972	160		
42	0	5692	0	5692	0	5692	0	5692
			0	1506	0	2427	0	5301
			81	16	81	638	162	4486
					243	5	729	1016

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
43	0	4262	81	300	0	5233	81	2933
			243	24	81	140	2106	1699
			648	7	324	111		
					648	24		
					810	7		
44	0	8113	0	8113	0	8113	0	8113
	0	1827	0	325	0	4014	0	1020
			81	239	81	3781	1539	680
			405	70	162	70	2349	266
			1863	58				
			2592	55				
			3483	41				
45	0	5648	0	5648	0	5648	0	5648
			0	589	81	4861	396	2301
			81	250	162	2695	594	1636
					243	659	2772	1252
					567	587	3366	464
					2025	119	3564	250
					2673	105		
46	0	7605	0	6483	0	3892	0	4935
	1539	5814	342	3145	81	23	162	2196
			513	1044			243	1668
			1368	39			1215	800
			1710	34			2673	728
			3249	4				
47	0	6951	0	6951	0	6951	0	6951
	0	4729	0	1707	81	37	0	4117
			81	16	243	4	486	2600
			567	3			729	464
			1134	1				
48	0	6433	0	1450	0	4789	0	6337
			1008	40	117	225	162	4498
			5472	6	351	32	729	2931
							810	2240
							2106	1923
							2268	1168
							2592	1110
49	0	4114	0	4114	0	4114	0	4114
	0	2660	81	446	81	18	324	2667
	486	2491	243	245			891	136
	810	2177	972	69				
	2673	1276	1377	36				
50	0	1486	0	1486	0	1486	0	1486
			162	74	162	1004	2106	1337
			1458	10	486	244		
					648	23		
51	0	8976	0	8976	0	8976	0	8976
	0	2345	0	7209	0	8507	0	1610
			81	3204	208	2543	567	1458
			162	1267	416	2473		
			1782	907	624	126		
			2187	351				

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
52	0	2349	0	2349	0	2349	0	2349
			0	2320	81	700	0	1556
			81	58	405	172	3807	1490
			162	6	1215	87		
					3888	45		
53	0	5691	0	5691	0	5691	0	5691
	0	4410	0	4607	0	2788	121	5585
			1296	31	81	356	242	3002
					243	102	847	1124
					2106	96	1089	391
					2835	35		
54	0	8156	0	8156	0	8156	0	8156
	0	2655	0	7192	0	2955	0	6376
			81	6120	81	491	162	5736
			162	2960	162	229	324	5394
			324	613	810	205	405	3139
			405	583	2916	156	2268	2111
			810	491	3078	71		
			1134	334	3726	17		
			2754	131				
			2835	29				
55	0	3809	0	3809	0	3809	0	3809
			0	3226	0	960	2160	1835
			162	66	243	331	2808	23
			324	31	405	96		
					1863	72		
56	0	4389	0	4389	0	4389	0	4389
	1296	3073	81	1513	81	314	0	2841
			162	8	243	171	405	311
			810	1	324	162		
			972	0	1620	49		
					1863	24		
57	0	1699	0	1699	0	1699	0	1699
			81	765	81	98	1134	1049
			243	516	324	14	2349	654
			324	234				
			486	13				
58	0	3902	0	3902	0	3902	0	3902
	0	116	81	443	0	1264	0	2129
			162	264	81	443	1134	1739
			243	59	243	26	2916	964
			324	49				
			405	26				
59	0	7130	0	7130	0	7130	0	7130
	0	2477	0	3989	0	3415	0	200
			162	2910	81	2380	486	25
			243	251	162	1083		
				324	42			

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
60	0	9273	0	9273	0	9273	0	9273
	0	3334	0	1855	0	2174	0	8966
			81	5	81	795	378	5995
					243	567	504	5426
					405	361	1638	4652
					1053	55	1764	3203
							2142	850
							2394	180
61	0	3658	0	3658	0	3658	0	3658
	0	1368	0	1974	81	1411	567	2819
			162	173	243	263	810	1705
			243	42	567	15	891	1478
			1215	13				
62	0	7368	0	7368	0	7368	0	7368
	567	5387	0	720	0	5885	0	1935
	2673	2817	81	89	81	119	648	265
			324	86	2916	44		
		3645	74					
63	0	8060	0	8060	0	8060	0	8060
	0	2823	0	6809	0	5552	0	3570
			81	401	81	1087	81	1355
			162	274	162	208	567	748
			324	226	1296	65	1863	394
			729	65			2106	230
64	0	3956	0	3956	0	3956	0	3956
	0	2746	81	1167	81	662	0	1194
			162	196	243	482	405	518
			810	170	324	341		
			891	155	810	47		
			1620	47				
			1782	11				
65	0	9657	0	9657	0	9657	0	9657
	0	9384	0	7058	0	4676	0	8488
	243	9354	81	10	81	726	81	3685
	405	6536			243	63	567	3620
							1458	2021
						1620	502	
						3159	188	
66	0	5035	0	5035	0	5035	0	5035
			81	411	81	894	99	3874
			324	57	243	19	693	901
			2673	0	648	0	1188	391
							2376	313
							3267	305
							3465	303
						3762	144	

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
67	0	3673	0	3673	0	3673	0	3673
			81	189	0	2581	324	3633
			243	116	81	378	1620	3453
			729	84	243	250	2430	1251
			1053	57	324	222	2754	1247
			2349	41	567	108		
			2673	37	2754	37		
					3402	1		
68	0	7049	0	7049	0	7049	0	7049
	2997	5265	81	1359	0	2166	0	149
			162	926	81	718		
			243	142	162	157		
			324	104	243	104		
					324	3		
69	0	4301	0	4301	0	4301	0	4301
	0	4183	0	2832	81	218	0	4143
			81	1907	243	21	243	3941
			243	374	1296	12	972	2453
			486	0			2025	1628
							2916	501
						3240	224	
70	0	2615	0	2615	0	2615	0	2615
			81	249	0	186	648	1705
			162	6	243	11	729	480
			1458	2	2835	2	2268	437
71	0	81	0	81	0	81	0	81
			81	30	324	5		
72	0	6062	0	6062	0	6062	0	6062
	0	1000	0	5527	0	4291	81	3340
			81	513	81	196	162	1149
			243	72	405	1	405	1017
			324	45				
73	0	5527	0	5527	0	5527	0	5527
			81	345	0	4468	243	3446
			162	2	81	247	810	820
					162	121		
					2187	14		
					3321	2		
74	0	2382	0	2382	0	2382	0	2382
	0	920	0	1937	0	1835	567	1838
			81	823	162	100	810	1729
			810	61	1782	57	1377	1294
			2835	49			1620	845
							1782	521
						2025	182	

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
75	0	7827	0	3454	0	8810	0	9020
	243	6400	81	836	81	113	135	7600
			162	138	162	39	270	6505
			324	115			540	4192
			891	97			810	2652
							945	2352
							3645	107
							3780	97
76	0	3527	0	3527	0	3527	0	3527
	0	3381	0	412	144	1055	0	1433
			144	25	288	17	900	1331
			4896	1			1900	941
							3200	802
77	0	239	0	239	0	239	0	239
			0	181				
			81	114				
			162	47				
			648	11				
			1782	6				
78	0	9605	0	9605	0	9605	0	9605
	0	8124	0	4505	0	231	0	2785
	1053	6219	81	23	162	135	243	223
					648	96		
					1701	14		
					2025	8		
79	0	2633	0	2633	0	2633	0	2633
	0	452	81	471	0	1341	648	130
			2187	170	405	1038		
					648	498		
					1377	32		
					1863	29		
80	0	9779	0	9779	0	9779	0	9779
	0	1065	0	6553	0	4096	0	4212
			81	654	81	10	1134	1735
			162	319			1296	1171
			567	287			1701	1082
			1053	10				
81	0	4820	0	4820	0	4820	0	4820
	1134	3937	0	2958	0	367	324	3266
			81	186	81	301	729	2313
			162	90	162	149		
			1053	56	243	134		
			2268	43	2430	83		
			2754	20	3645	5		
82	0	6679	0	6679	0	6679	0	6679
	0	2037	81	3326	0	6096	0	3049
			162	1580	81	5365	81	2321
			243	47	162	541	1134	1432
			486	14	243	28	1377	1126
							2430	846
							3645	261

Symbiotic Algorithms

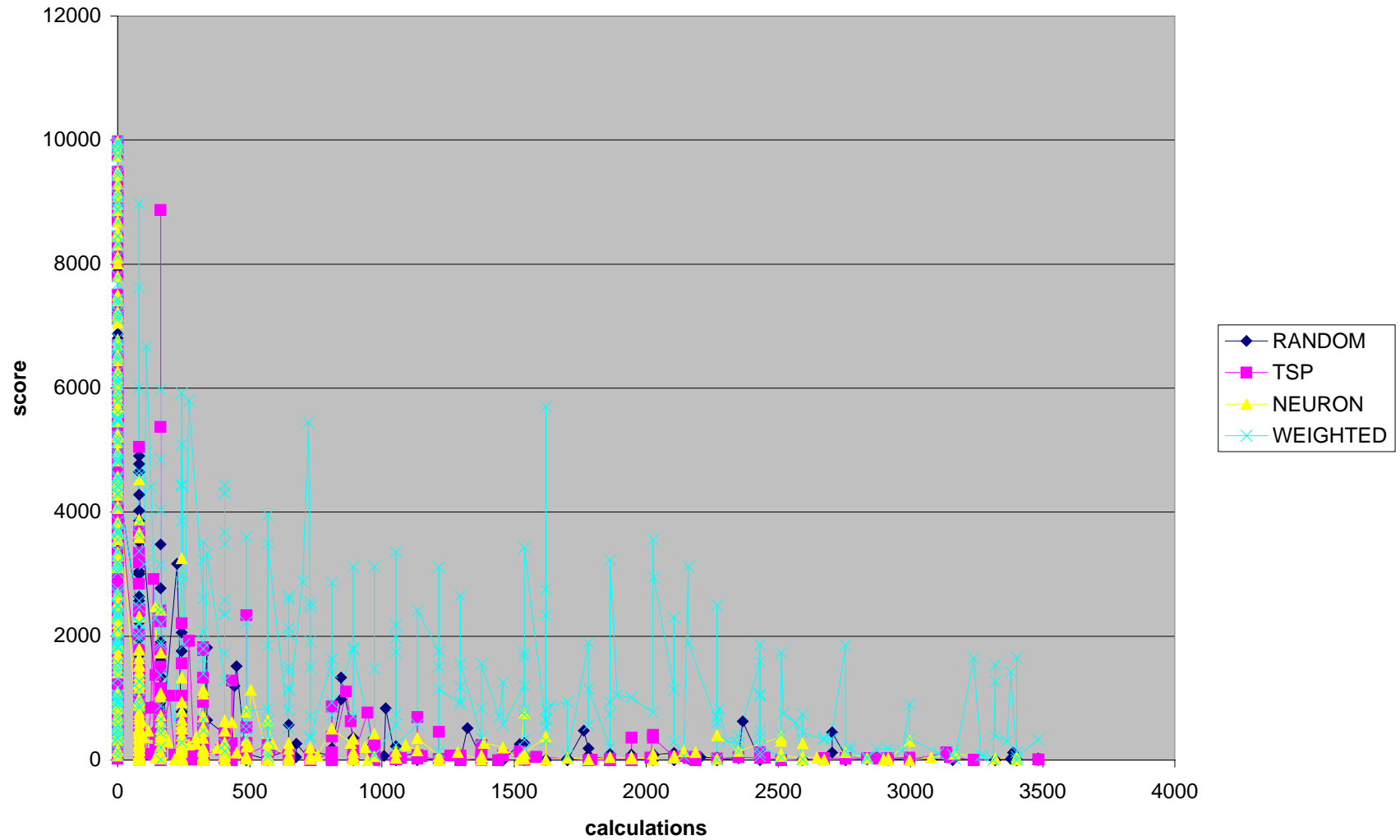
Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
83	0	7857	0	7857	0	7857	0	7857
	0	7234	0	6225	0	7758	243	2486
	3136	5386	81	1296	81	529	1296	2068
			162	529	243	380	2349	1959
			243	70	324	193	3159	1605
					486	70		
84	0	3160	81	1970	0	3842	0	3160
			243	7	81	130	81	2744
					243	0	972	1724
							1215	211
85	0	9642	0	9642	0	9642	0	9642
	0	8711	0	798	0	7259	0	3344
	288	6906	162	80	121	6710	240	1795
	1296	2104	3645	9	242	3839	480	1746
					363	1976	3600	895
					726	80	4050	860
					1815	59		
86	0	2869	0	2869	0	2869	0	2869
			81	1009	81	104	0	2683
			162	568	2187	23	162	2018
			405	495			3240	708
			486	71				
			2106	63				
			3159	54				
87	0	375	0	375	0	375	0	375
			162	149	405	65	1215	182
					567	43		
88	0	2140	0	2140	0	2140	0	2140
	0	395	162	4	0	2024	2025	548
			405	2	81	208		
					324	15		
					1458	4		
89	0	1838	0	1838	0	1838	0	1838
			81	794	0	1395	486	1523
			162	247	243	80	648	816
			486	42	405	18		
			1134	5	891	6		
90	0	5310	0	5310	0	5310	0	5310
			0	201	0	2214	81	4324
			2511	96	81	1043	162	3856
			2916	75	162	570	972	1037
					324	398	2592	494
					567	201		
					648	84		
91	0	9190	0	9190	0	9190	0	9190
	0	332	0	4666	0	6041	0	8226
			81	152	81	1385	81	4251
			162	35	162	105	324	2302
			810	26	486	60	1296	1587
							2430	1203
							2592	1056
						3078	254	

Symbiotic Algorithms

Run	Random		TSP		Neuron		Weighted	
	calculation	score	calculation	score	calculation	score	calculation	score
92	0	6108	0	6108	0	6108	0	6108
			0	4791	81	487	162	3701
			81	186	243	4	567	2231
			243	48	567	3	1377	49
			810	3				
			1296	0				
93	0	6504	0	6504	0	6504	0	6504
	729	5262	81	17	0	4024	0	3355
	810	4601	486	15	81	1161	1134	3210
			729	0	162	221	2430	2137
					324	6	3402	779
94	0	5464	0	269	0	7656	0	7816
			81	88	81	183	81	5685
			405	19	567	54	243	3378
			2916	13			486	849
							891	123
95	0	3622	0	3622	0	3622	0	3622
			81	995	81	173	810	986
			162	56	243	56	2511	742
			810	29	324	21		
			2268	21	486	2		
96	0	7053	0	7053	0	7053	0	7053
	162	5595	81	134	162	3280	0	2245
			162	123	324	1208	3159	42
			243	55	1620	439		
					1944	403		
					2025	29		
97	0	2022	0	2022	0	2022	0	2022
			81	655	81	1044	1134	1675
			243	38	243	434	2673	952
					810	412		
					2511	16		
98	0	904	0	904	0	904	0	904
			81	74	81	352	162	640
			1296	12	243	147		
			1539	9	324	6		
99	0	1340	0	1340	0	1340	0	1340
			0	839	198	573	81	158
			288	159				
			576	102				
			1440	62				
100	0	4966	0	4966	0	4966	0	4966
	0	3446	0	624	81	18	507	1598
			169	324			1183	81
			338	122				
			676	18				

Appendix C : Optimised Neighbourhoods

TSP Optimised



Appendix C: Used Software Architecture

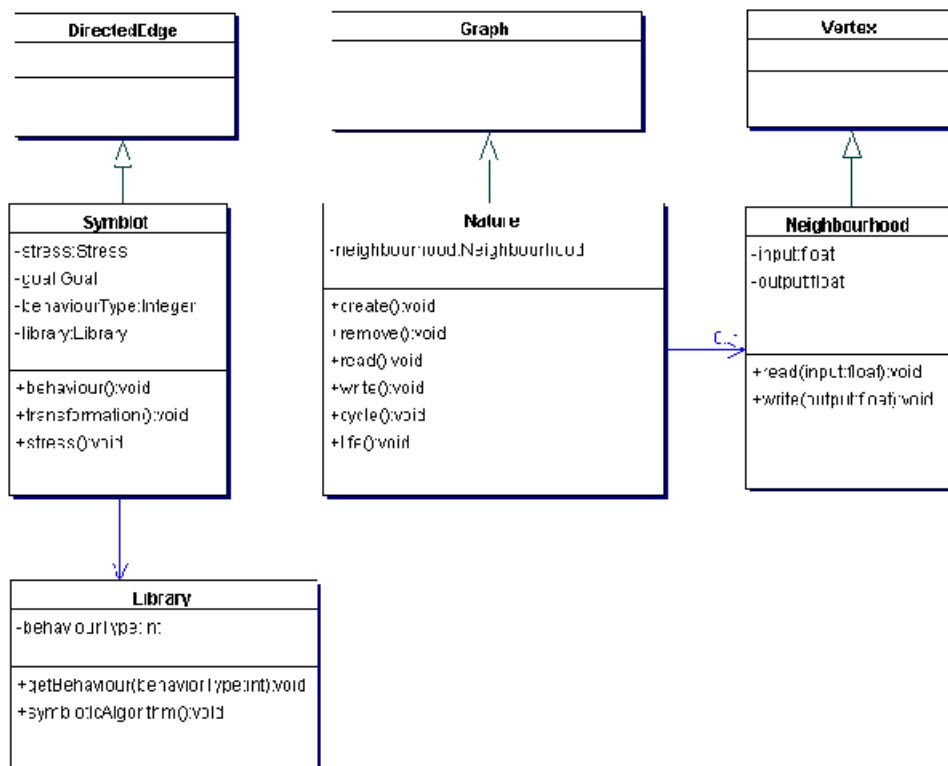


Figure 41: Base UML Diagram of package symbiot

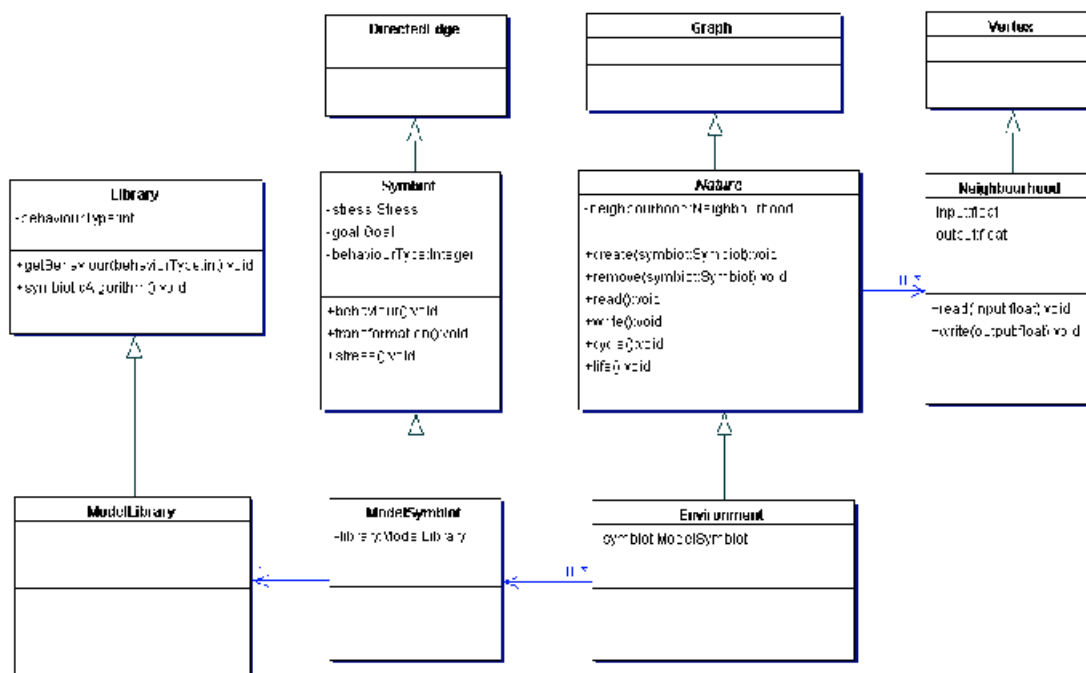


Figure 42: Base UML Diagram of package symbiot and the model

This document was created with Win2PDF available at <http://www.daneprairie.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.