Digital Business Ecosystem

Contract n^{o} 507953

Workpackage: 6
Self-Organisation

Deliverable: 6.5
The effect of Distributed Intelligence in Evolutionary Dynamics

Information Society and Media

Project funded by the European Community under the "Information Society Technology" Programme

Contract Number: 507953
Project Acronym: DBE
Title: Digital Business Ecosystem

Deliverable No: D6.5
Due date: 31/10/2006
Delivery date: 31/10/2006

Short Description:
A paper on our simulations of the effect of distributed intelligence in evolutionary dynamics, specifically the effect of the Distributed Intelligence System (DIS) on the Evolutionary Environment (EvE). The evolutionary dynamics was seen to be enhanced significantly with the application of the *distributed intelligence*, and marginally more so when using Support Vector Machines (SVM) than Neural Networks (NNs).

Author: HWU (Gerard Briscoe, Philippe De Wilde)
Partners contributing: HWU
Made available to: DBE Consortium and European Commission

Versioning		
Version	**Date**	**Author, organisation**
1	31/10/06	Gerard Briscoe, HWU

Quality Check:
1st Internal Reviewer: David Mckitterick (INTEL)
2nd Internal Reviewer: Jesús Gabaldon (TechIDEAS)

The Effect of Distributed Intelligence on Evolutionary and Ecological Dynamics

Gerard Briscoe

Intelligent Systems Lab
Department of Computer Science
School of Mathematical and Computer Sciences
Heriot-Watt University

December 2006

Acronyms

ABM	Agent-Based Model
BML	Business Modelling Language
DBE	Digital Business Ecosystem
DIS	Distributed Intelligence System
EvE	Evolutionary Environment
JDJ	Java Developer's Journal
MUH	Migration and Usage History
NN	Neural Network
SIM	Social Interaction Mechanism
SME	Small & Medium Enterprise
SMO	Sequential Minimal Optimisation
SOA	Service-Oriented Architecture
STU	Salzburg Technical University
SUN	Sun Microsystems
SVM	Support Vector Machines

Contents

Executive Summary

A primary motivation for research in digital ecosystems is the desire to exploit our understanding of the self-organising properties of natural ecosystems. Ecosystems are thought to be robust, scalable architectures that can automatically solve complex, dynamic problems. However, the biological processes that contribute to these properties have not been made explicit in digital ecosystems research. We discuss how biological properties contribute to the self-organising features of natural ecosystems. The potential for exploiting these properties in artificial systems is then considered and the EvE architecture is considered in detail, with simulation results implying that the EvE performs better at large scales than a comparable Service-Oriented Architecture (SOA). These results suggest that incorporating ideas from theoretical ecology can contribute to useful self-organising properties in digital ecosystems, and that the EvE has sufficient ecological and evolutionary dynamics to be affected by the *distributed intelligence* model.

Regarding the research objectives of whether 'intelligence can optimise the evolutionary process' [25], 'how does this distributed intelligence interact with the ecosystem dynamics' [25], and whether 'software components that are part of genetic selection can be intelligent in themselves' [25], and as we expected the result was 'a significant change in the evolutionary dynamics with distributed intelligence applied' [25]. The *distributed intelligence* model optimises the Agent-Pool at each habitat to support the evolving populations created to respond to user requests, thereby creating a distributed optimisation to support the local evolutionary optimisation. The *distributed intelligence* has been investigated, and the experimental results indicate that it will optimise the EvE. Based on our theoretical understanding and the experimental results, we would recommend SVM over NNs for the learning functionality of the DIS. We arrived at your new SVM recommendation after it was feasible to include this approach in the task of implementing the DIS, and therefore the DIS implementation was developed on the initial recommendation of NNs.

Regarding the objective of 'the research, design and implementation of a Distributed Intelligence System (DIS) based on intelligence based on self-organisation to optimise and enhance the Evolutionary Environment (EvE)' [25], we have completed the research sufficiently in task S5 to support the architectural specification of the DIS in task C42. The task C42 was used in the implementation process of C53, providing continued support to the task C53 in achieving the objective of an 'implementation of a DIS which optimises the EvE' [25].

Regarding the objectives of 'the dissemination effort to distribute and explain our results to the computation work package contributors in C53 via C42' [25], the 'creation of a high-level design specification for the Distributed Intelligent System which augments the EvE, for use in the implementation of the Distributed Intelligence System' [25], 'finalisation of the EvE Architecture specification' [25], and 'a design of the DIS which is complementary to the EvE' [25], deliverable 6.6 [17] was explicitly provided to meet these objectives and to complement our dialogue with the partners involved.

SOAs, such as the Digital Business Ecosystem (DBE), promise to provide potentially huge numbers of services that programmers can combine via standardised interfaces, to create increasingly sophisticated distributed applications. The EvE optimised by the *distributed intelligence* model extends this concept by automatically combining available and applicable services effectively in a scalable architecture.

1 Introduction

This report is aimed at investigating the effects of a *distributed intelligence* mechanism on the evolutionary and ecological dynamics of the EvE. Given that the *distributed intelligence* effects the evolutionary dynamics indirectly via the ecological dynamics, we were required to investigate, justify and prove both the evolutionary and ecological dynamics within the EvE to ensure the validity of our research into effects of the *distributed intelligence*. The evolutionary dynamics of the EvE has been well established through the evolving populations of composite services, which use evolutionary computing. However, the ecological dynamics are less well defined, with only the simple concepts of distribution and migration being present within the Digital Ecosystem model of the EvE.

A primary motivation for research in digital ecosystems is the desire to exploit our understanding of the self-organising properties of natural ecosystems. Ecosystems are thought to be robust, scalable architectures that can automatically solve complex, dynamic problems. However, the biological processes that contribute to these properties have not been made explicit in digital ecosystems research. In **Section 2**, we discuss how biological properties contribute to the self-organising features of natural ecosystems. These properties include populations of evolving agents, a complex dynamic environment, and spatial distributions which generate local interactions. The potential for exploiting these properties in artificial systems is then considered, including in detail the EvE of the DBE.

Section 3 briefly summarises the *distributed intelligence* model presented in deliverable 6.4 'Intelligence, Learning and Neural Networks in Distributed Agent Systems' and 6.6 'High Level Design Specification of the Distributed Intelligence System' [17], the pattern recognition based Social Interaction Mechanism (SIM) of targeted-migration and how it would optimise the EvE. We then consider NNs and SVM for the pattern-recognition capabilities required.

Section 4 summarises the simulations created for our experimental investigations, and presents the results from these simulations. Finally, the conclusions are presented and discussed in **Section 5**, including the value of the contributions both scientifically and to the DBE, with a brief discussion of our intended future work. Consideration is also given to the relation of this work to the other activities within the project.

Appendix A contains minor updates to the EvE/DIS architecture, as defined in deliverable D6.6 'High Level Design Specification Of The Distributed Intelligence System' [17]. Including the alteration of the SBVR pre-processing algorithm to be applicable to BMLv1. **Appendix B** is a master's thesis, co-supervised by the author, extending our earlier work of Physical Complexity from deliverables 6.1 [14] and 6.2 [15].

2 Evolutionary and Ecological Dynamics

The work in this section, and parts of sections 4 and 5, was done collaboratively with Dr Suzanne Sadedin of the University of Tennessee and Gregory Paperin of Monash University. This work has been accepted in the upcoming IEEE inaugural conference on Digital Ecosystems [13].

Is mimicking nature the future for information systems? A key challenge in modern computing is to develop systems that address complex, dynamic problems in a scalable and efficient way. One approach to this challenge is to develop digital ecosystems, artificial systems that aim to harness the dynamics that underlie the complex and diverse adaptations of living organisms. However, the connections between digital ecosystems and their biological counterparts have not been closely examined. Here, we consider how properties of natural ecosystems influence functions that are relevant to developing digital ecosystems to solve practical problems. This leads us to suggest ways in which concepts from ecology can be used to combine biologically inspired techniques to create an applied Digital Ecosystem, the EvE of the DBE.

The increasing complexity of software [44] makes designing and maintaining efficient and flexible systems a growing challenge. Many authors argue that software development has hit a *complexity wall* which can only be overcome by automating the search for new algorithms [60]. Natural ecosystems possess several properties that may be useful in such automated systems. These properties include self-organisation, self-management, scalability, the ability to provide complex solutions, and automated composition of these complex solutions.

A natural ecosystem consists of a community of interacting organisms in their physical environment. Several fundamental properties influence the structure and function of natural ecosystems. These include agent population dynamics, spatial interactions, evolution, and complex, changing environments. We examine these properties, and the role they may play in an applied digital ecosystem.

The term *Digital Ecosystem* has been used to describe a variety of concepts. Some of these refer to the existing networking infrastructure of the internet [69, 20]. Several companies offer a Digital Ecosystem service, which involves enabling customers to use existing e-business solutions [57, 54, 61]. However, perhaps the most frequent references to Digital Ecosystems arise in Artificial Life research. These Digital Ecosystems are created primarily to investigate aspects of biological and other complex systems [70, 52, 2, 49, 48], rather than to provide a service for human users. The extent to which these disparate systems resemble natural ecosystems varies, and in many cases has not been examined. We will restrict our consideration of digital ecosystems to software systems designed to exploit properties of natural ecosystems for practical purposes. We suggest that several key features of natural ecosystems have not been fully explored in existing digital ecosystems, and discuss how mimicking these feature may assist in developing robust, scalable self-organising architectures.

Arguably the most fundamental differences between biological and digital ecosystems lie in the motivation and approach of researchers. Biological ecosystems are ubiquitous

natural phenomena, whose maintenance is crucial to our survival. The performance of natural ecosystems is often measured in terms of their stability, complexity and diversity. In contrast, digital ecosystems as defined here are technology engineered to serve specific human purposes. The performance of a digital ecosystem, then, can be judged relative to the function it is designed to perform. Frequently, the purpose of a digital ecosystem is to solve dynamic problems in parallel with high efficiency. This criterion may be related only indirectly to complexity, diversity and stability, an issue we shall examine further below.

Simple systems are easier to understand and control than complex ones. So, we begin with the assumption that features of real ecosystems that offer no clear advantage in digital ones will be excluded.

Genetic algorithms are the subset of evolutionary computation that uses natural selection to evolve solutions. Starting with a set of arbitrarily chosen possible solutions, selection, replication, recombination and mutation are applied iteratively. Selection is based on conformation to a fitness function which is determined by the specific problem of interest. Over time, better solutions to the problem can thus evolve. Because they are intended to solve problems by evolving solutions, applied digital ecosystems are likely to incorporate some form of genetic algorithm. However, we suggest that a digital ecosystem should also incorporate additional features which give it a closer resemblance to natural ecosystems. Such features might include complex, dynamic fitness functions, a distributed or network environment, and self-organisation arising from interactions among agents and their environment. These properties are discussed further below.

We will use the idea of the DBE, specifically the EvE, as an example application. The DBE is a proposed methodology for economic and technological innovation. Specifically, the EvE is a Digital Ecosystem software infrastructure for supporting a large number of interacting business users and services. The individuals of a EvE are software agents that represent business entities or services. These agents interact, evolve and adapt to a dynamic digital environment, thereby serving the ever-changing business requirements imposed by the economy.

2.1 Fitness Landscapes and Agents

As described above, an ecosystem comprises both an environment and a set of interacting, reproducing entities (or agents) in that environment. The environment, in biological ecosystems, promotes as a set of physical and chemical constraints on reproduction and survival. These constraints can be considered in abstract using the metaphor of the fitness landscape [68]. Individuals are represented in the fitness landscape as solutions to the problem of survival and reproduction. All possible solutions are distributed in a space whose dimensions are the possible properties of individuals. An additional dimension, height, indicates the relative fitness (in terms of survival and reproduction) of each solution. The fitness landscape is envisaged as a rugged, multidimensional landscape of hills, mountains and valleys, because individuals with certain sets of properties are fitter than others.

In biological ecosystems, fitness landscapes are virtually impossible to identify. This

is both because there are large numbers of possible traits that can influence individual fitness, and because the environment changes over time and space. In contrast, within a digital environment, it is normally possible to specify explicitly the constraints that act on individuals in order to evolve solutions that perform better within these constraints.

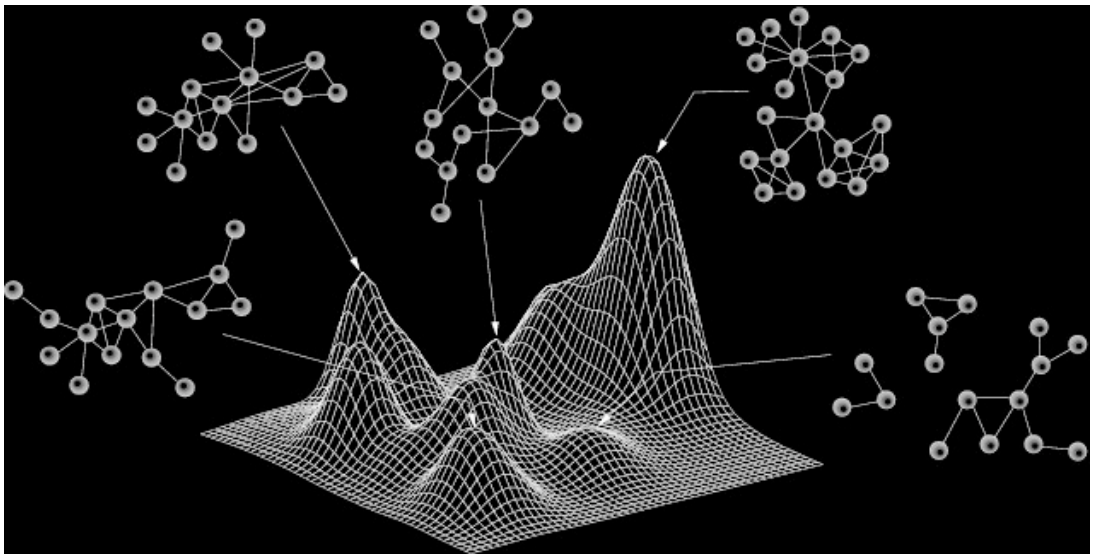


Figure 1: Fitness Landscape Visualisation

Within genetic algorithms, exact specification of a fitness landscape or function is common practice. However, within a digital ecosystem the ideal constraints are those that allow solution populations to evolve to meet user needs with maximum efficiency. User needs will change from place to place and time to time. In this sense the EvE fitness landscape is complex and dynamic, and more like that of a biological ecosystem than like that of a traditional genetic algorithm. The designer of a digital ecosystem therefore faces a double challenge: firstly, to specify rules that govern the shape of the fitness landscape in a way that meaningfully maps landscape dynamics to user requests, and secondly, to evolve solution populations within this space that are diverse enough to solve disparate problems, complex enough to meet user needs, and efficient enough to be preferable to those generated by other means.

The agents within a digital ecosystem are like biological individuals in the sense that they reproduce, vary, interact, move and die. Each of these properties contributes to the dynamics of the ecosystem. However, the way in which these individual properties are encoded may vary substantially depending on the purpose of the system.

2.2 Networks and Spatial Dynamics

A key factor in the maintenance of diversity in natural ecosystems is spatial interactions. Several modelling systems have been used to represent spatial interactions. These include metapopulations, diffusion models, cellular automata and agent-based models (termed individual-based models in ecology) [29]. The broad predictions of these diverse models are in good agreement. At local scales, spatial interactions favor relatively abundant species disproportionately. However, at a wider scale, this effect can preserve diversity,

because different species will be locally abundant in different places. The result is that even in homogeneous environments, population distributions tend to form discrete, long-lasting patches that can resist invasion by superior competitors [29]. Population distributions can also be influenced by environmental variations such as barriers, gradients and patches. The possible behavior of spatially distributed ecosystems is so diverse that scenario-specific modelling is necessary to understand any real system [27]. Nonetheless, certain robust patterns are observed. These include the relative abundance of species (which consistently follows a roughly log-normal relationship) [7] and the relationship between geographic area and the number of species present (which follows a power law) [4]. The reasons for these patterns are disputed: both spatial extensions of simple Lotka-Volterra competition models [35], and more complex ecosystem models [55], are capable of generating them.

Landscape connectivity plays an important part in ecosystems. When the density of habitat within an environment falls below a critical threshold, widespread species may fragment into isolated populations. Fragmentation can have several consequences. Within populations, these effects include loss of genetic diversity and harmful inbreeding [28]. At a broader scale, isolated populations may diverge genetically, leading to speciation. From an information theory perspective, this phase change in landscape connectivity can mediate global and local search strategies [31]. In a well-connected landscape, selection favors the globally superior, and pursuit of different evolutionary paths is discouraged, potentially leading to premature convergence. When the landscape is fragmented, populations may diverge, solving the same problems in different ways. Recently, it has been suggested that the evolution of complexity in nature involves repeated landscape phase changes allowing selection to alternate between local and global search [30].

In a digital context, we can simulate spatial interactions by creating a distributed system that consists of a set of interconnected locations. Agents can migrate between connected locations with low probability. Depending on the how the connections between locations were organised, such digital ecosystems might have dynamics closely parallel to spatially explicit models, diffusion models, or metapopulations [27]. In all these systems the spatial dynamics are relatively simple compared with those seen in real ecosystems, which incorporate barriers, gradients and patchy environments at multiple scales in continuous space. Another alternative in a digital ecosystem is to apply a spatial system that has no geometric equivalent. These could include, for example, small world networks and geographic systems that evolve by Hebbian learning. We will discuss an example of a non-geometric spatial network, and some possible reasons for using this approach.

2.3 Selection and Self-organisation

The major hypothetical advantage of digital ecosystems over other complex organisational models is their potential for dynamic adaptive self-organisation. However, for the solutions evolving in digital ecosystems to be useful, they must not only be efficient in a computational sense, but they must also solve meaningful problems. That is, the fitness of agents must translate in some sense to real-world usefulness as demanded by users.

Constructing a useful digital ecosystem therefore requires a balance between freedom of the system to self-organise, and constraint of the system to generate useful solutions.

These factors must be balanced because the more the system's behavior is dictated by internal dynamics of the system, the less it may respond to fitness criteria imposed by users. At one extreme, when system dynamics are mainly internal, agents may evolve that are good at survival and reproduction within the digital environment, but useless in the real world. At the other extreme, where the users' fitness criteria overwhelmingly dictate function, we suggest that dynamic exploration of solution space and complexity are likely to be limited.

The reasoning behind this argument is as follows. Consider a multidimensional solution space which maps to a rugged fitness landscape (as described above). In this landscape, competing solution lineages will gradually become extinct through chance processes. Consequently, the solution space explored becomes smaller over time as the population adapts and diversity of solutions decreases. Ultimately, all solutions may be confined to a small region of solution space. In a static fitness landscape, this situation is not undesirable because the surviving solution lineages will usually be clustered around an optimum. However, if the fitness landscape is dynamic, the location of optima varies over time. Should lineages become confined to a small area of solution space, subsequent selection will locate only optima that are near this area. This is undesirable if new, higher optima arise that are far from preexisting ones. A related issue is that complex solutions are less likely to be found by chance than simple ones. Complex solutions can be visualised as sharp, isolated peaks on the fitness landscape. Especially for a dynamic landscape, these peaks are most likely to be found when the system explores solution space widely. Consequently, a self-organising mechanism other than the fitness criteria of users is required to maintain diversity among competing solutions in the digital ecosystem.

2.4 Stability and Diversity in Complex Adaptive Systems

Ecosystems are often described as complex adaptive systems (CAS) [40]. Such systems are comprised of diverse, locally interacting components that are subject to selection. Other complex adaptive systems include brains, individuals, economies, and the biosphere. All are characterised by hierarchical organisation, continual adaptation and novelty, and non-equilibrium dynamics [40]. These properties lead to behavior that is non-linear, historically contingent, subject to thresholds, and contains multiple basins of attraction [40].

In the above sections we have advocated digital ecosystems that include agent populations evolving by natural selection in distributed environments. Like real ecosystems, digital systems designed in this way fit the definition of complex adaptive systems. The features of these systems, especially non-linearity and non-equilibrium dynamics, offer both advantages and hazards for adaptive problem-solving. The major hazard is that the dynamics of complex adaptive systems are intrinsically hard to predict because of self-organisation, and other characteristics of complex systems.This observation implies that designing a useful digital ecosystem will be partly a matter of trial and error. The occurrence of multiple basins of attraction in complex adaptive systems suggests that even a system that functions well for a long period may suddenly at some point transition to a less desirable state. For example, in some types of system mass extinction might result inadvertently from interactions among populations, leading to temporary unavailability

of diverse solutions. This would be highly undesirable in digital ecosystems, and this concern may be addressed by incorporating negative feedback mechanisms at the global scale.

The challenges in designing an effective digital ecosystem are mirrored by the system's potential strengths. Nonlinear behavior provides the opportunity for scalable organisation and the evolution of complex hierarchical solutions. Rapid state transitions potentially allow the system to adapt to sudden environmental changes with minimal loss of functionality.

A key question for designers of digital ecosystems is how the stability and diversity properties of natural ecosystems map to performance measures in digital systems. In traditional solution finding algorithms, stability might not be very useful, and diversity may be hardly be taken as a useful general rule for measuring systems performance, but for a digital ecosystem it is critical for the overall system performance. For a digital ecosystem the ultimate performance measure is user satisfaction, a system-specific property. However, assuming the motivation for engineering a digital ecosystem is the development of scalable, adaptive solutions to complex dynamic problems, certain generalisations can be made. Sustained diversity, as discussed above, is a key requirement for dynamic adaptation. In the digital ecosystem, diversity must be balanced against adaptive efficiency because maintaining large numbers of poorly-adapted solutions is costly. The exact form of this tradeoff will be guided by the specific requirements of the system in question. Stability, as discussed above, is similarly a tradeoff: we want the system to respond to environmental change with rapid adaptation, but not to be so responsive that mass extinctions deplete diversity or sudden state changes prevent control.

2.5 EvE Digital Ecosystem

The DBE represents a business interaction concept supported by a software platform that is intended to have the desirable properties of natural ecosystems. The *EvE* was constructed [18, 11, 39, 42] using principles similar to those outlined here [9, 10, 12]. In the following subsections we will briefly introduce the EvE model and then compare some of its dynamics to those of natural ecosystems.

The EvE is a digital ecosystem in which autonomous mobile agents represent various services (or compositions of services) offered by participating businesses. The *abiotic* environment is represented by a network of interconnected habitats nodes. Each connected enterprise has a dedicated habitat. Enterprises search for, and deploy, services in local habitats. Continuous and varying user requests for services provide a dynamic evolutionary pressure on the agents, which have to evolve to better satisfy those requests. The EvE acts as a decentralised multi-agent system where the habitats form agent stations [67, 6].

The architecture of the DBE software platform is built around the idea of a semantically-aware SOA [23, 34]. The system is based on modular, distributed and reusable business services that have clearly defined and standardised interfaces. These interfaces include formal semantic descriptions of the operations offered which provide the basis for

automatic matching of user requests to business services. The services are available at network nodes to be accessed by other services or users on the network. This approach allows for applications to be constructed from loosely joined, highly interoperable components. Because these services interoperate over different development architectures, they are reusable and relocatable. The system also provides many features required for a dynamic and scalable digital ecosystem, for instance, platform independence, location transparency, service discovery, scalability and fault tolerance.

The EvE is a two-level optimisation scheme inspired by natural ecosystems. A decentralised peer-to-peer network forms an underlying tier of distributed services. These services feed a second optimisation level based on a genetic algorithm that operates locally on single peers (habitats) and is aimed at finding solutions satisfying locally relevant constraints. Through this twofold process the local search is accelerated to yield better local optima, as the distributed optimisation provides prior sampling of the search space by making use of computations already performed in other peers with similar constraints.

The users of the architecture (businesses connected to the system) formulate queries to the system by specifying a semantic description of the services they require. These descriptions define the metrics for evaluating the fitness of a service or composition of services as a distance function between the request and the service description. A composition of one or more services constitutes an agent in the system. An agent can migrate from one habitat (peer) to another, getting hosted only in locations where it is most useful in satisfying the requests.

In natural ecosystems geography defines the connectivity between habitats. Nodes in the EvE do not have a default geographical topology to define connectivity. A re-configurable network topology is used instead, which is dynamically adapted on the basis of observed migration paths of the individuals within the EvE habitat network. Following the idea of Hebbian learning, the habitats which exchange individuals more often obtain stronger connections and habitats which do not exchange individuals will become less strongly connected. This way, a network topology is discovered with time, which reflects the structure of the business-sectors within the user base. The resulting network will resemble the connectivity of the businesses within the user base, typically a small-world network in the case of Small & Medium Enterprises (SMEs) [71, 21, 46]. Such a network has many strongly connected clusters called *sub-networks* (quasi complete graphs) and a few connections between these clusters. Graphs with this topology have a very high clustering coefficient and small characteristic path lengths [64, 45], as shown in Figure 2.

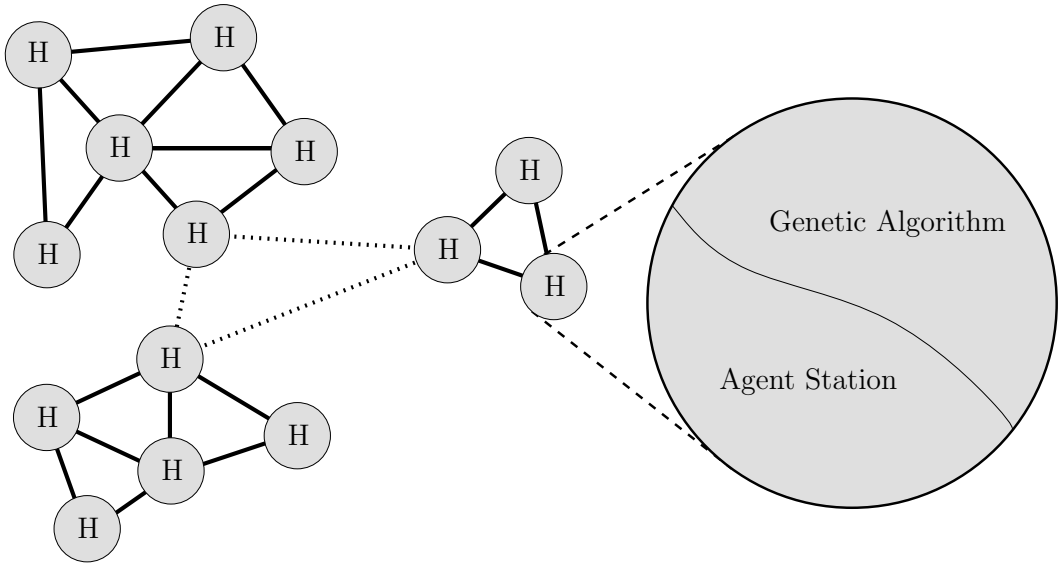


Figure 2: Optimisation architecture: agents travel along the peer-to-peer connections that have the topology of a small-world network; in each node local optimisation is performed through a genetic algorithm, where the search space is defined by the agents contained in the node.

Existing distributed evolutionary computing uses *parallel processing* to solve a single problem (request) [3]. For example, in the relatively natural *Island Model*[26], a distance is set between the sub-populations on each island, together with a probability of migration between one *island* and another. This model has been successfully used to determine investment strategies [63]. The EvE uses a similar strategy to solve multiple *similar* requests simultaneously. In the EvE separate requests are evaluated on separate *islands* within a habitat, with an island being an evolving population of composite services. Adaptation is accelerated by sharing solutions with other islands evolving solutions to similar requests.

We will experimentally investigate the EvE through simulations, with several key variables being recorded to determine whether it displays behaviour typically of natural ecosystems, i.e., ecological dynamics. If we have confidence in the results, we will be able to have confidence in our research of *distributed intelligence*, which is intended to effect the ecological dynamics of the EvE.

3 Distributed Intelligence

The *distributed intelligence* was created with the intention of optimising the evolutionary dynamics within the EvE. The *distributed intelligence* should assist the EvE to provide better solutions than it could alone, especially in the short term, by positively indirectly effecting the evolutionary dynamics via the ecological dynamics. The *targeted-migration* of the *distributed intelligence* will achieve this by targeting migration based on Migration and Usage Historys (MUHs), in response to similarity between agents determined by comparing their descriptions.

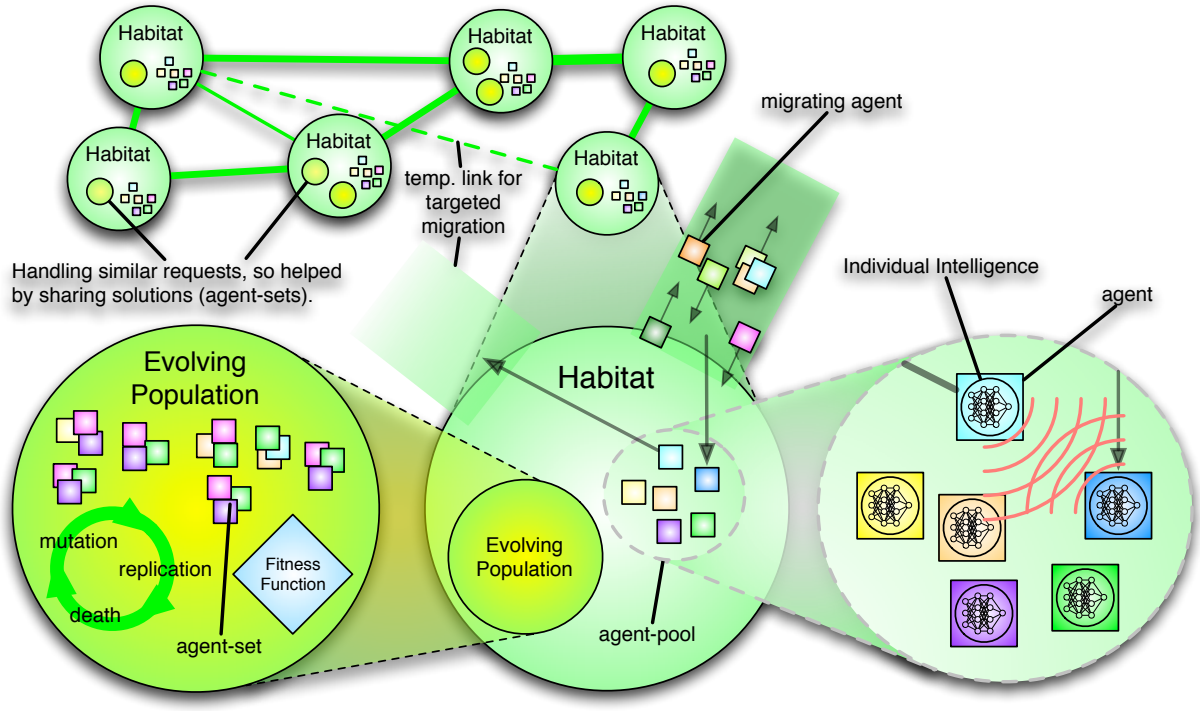


Figure 3: Distributed Intelligence

The *distributed intelligence* will work to complement the evolutionary optimisation indirectly. An agent, when finding similarity with another agent based upon their descriptions, will be able to migrate. It will target its migration to a Habitat where it will have the opportunity to be useful, based upon the MUH of the similar agent. In biological terms, it is equivalent to providing each individual with a relatively sophisticated SIM. It should allow niches to be fulfilled faster, so that the Habitat clusters will reach their 'climax communities' sooner, thereby significantly improving the speed of the 'succession' process. Also, communities will be able to adapt faster to changing conditions.

As the EvE increases in size (number of users), the total number of agents (services) available globally will become increasingly large. For the evolutionary optimisation to continue to work efficiently as the EvE expands, optimal subsets of the power-set of agents available globally will be required at the Habitats. The migration probabilities between the Habitats will work to achieve this in a 'passive' manner. Although the mechanism is effective, the migration of agents will be relatively slow and not strongly directed. It allows the agents, based primarily upon success at their current location, to spread in the correct general direction within the Habitat network. The *distributed*

intelligence will work in a more 'active' manner allowing the agents immediate highly targeted migration to specific Habitats, independent of success at the current location, instead of the generally directed migration only after success at the current location. This will help to optimise the subset of possible agent-sets found at the Habitats, which are then used in the evolutionary optimisation processes to find applications (combinations of agents) to user requests.

The agents can interact with one another, using their embedded *individual intelligence* components, to determine functional similarity based on their Business Modelling Language (BML)v1 descriptions. Circumstances have resulted in us using BMLv1, and an updated pre-processing algorithm can be found in the Appendix A. This involves comparing their BML descriptions to one another for similarity, and so is independent of the current Habitat's context (user requests). If agents are found to be similar through the inter-agent comparison, then they can share their MUHs to determine Habitats where they could be valuable. This interaction would occur outside the evolutionary optimisation, in the Agent-Pool and the consequence of this interaction is in the additional targeted-migration of agents, which can occur between connected and unconnected Habitats.

When targeted-migration occurs between connected Habitats it will speed up the natural migration of agents. When it occurs between unconnected Habitats, it will help the Habitat network to adapt to changing conditions faster than it otherwise would. In effect, during targeted-migration the *distributed intelligence* short-circuits the hierarchical structure of the caveman topology, to avoid being unnecessarily slowed by it. This may seem counter-productive after creating such a topology, but the caveman topology is what allows the Habitat network to specialise solutions to specific users and specific user requests. This ability leads to a weakness, specifically that the Habitat network can be slow to adapt to changing conditions, which the *distributed intelligence* will address. The *distributed intelligence* will help catalyse the formation of caves, and caves with the correct nodes, within the caveman topology. The desired effects of the *distributed intelligence* on the Habitat network will be achieved by integrating the targeted-migration with the existing migration feedback mechanisms.

The *distributed intelligence* will interact with the ecosystem dynamics of the EvE. Embedding the *individual intelligence* components within the agents, not only maintains the consistency of the Agent-Based Model (ABM) model of the EvE, but also strengthens the 'agent' definition by giving the individuals (agents) some intelligence and control over their existence.

3.1 Pattern Recognition Techniques

In this subsection we will discuss the alternative pattern recognition techniques we investigated, continuing the NNs approach introduced in deliverables 6.4 [16] and 6.6 [17], and introducing the SVM approach we are considering.

3.1.1 Neural Networks

NNs were the first technique considered, because it is an established and widely applicable pattern recognition technique. The pattern recognition capabilities of NNs can be leveraged to allow agents to determine similarity based on their descriptions. Using multilayer feed-forward NNs and the backpropagation training technique to specify the required pattern recognition behaviour.

The weights, $w_{i,j}$, between neurones are randomly initialised and then trained to the *real* numbers that provide the desired functionality [33]. We are currently using a sigmoid transfer function, in which the output, y, of neurone is the sum of the weighted input values, x, applied to a sigmoid function. This function is expressed mathematically as:

$$y = \frac{1}{(1 + e^{-x})} \tag{1}$$

This has the effect of smoothly limiting the output within the range of 0 and 1. Our simulation currently use a threshold of 0.90, for the neurone of the *output layer*, to determine similarity. This value may need to be proportioned to the BML vocabulary of the Habitat cluster (opportunity space) and/or the experience of the agents, as it defines how focused the targeted-migration will be. So, the Intelligence component will also require an *experience counter*.

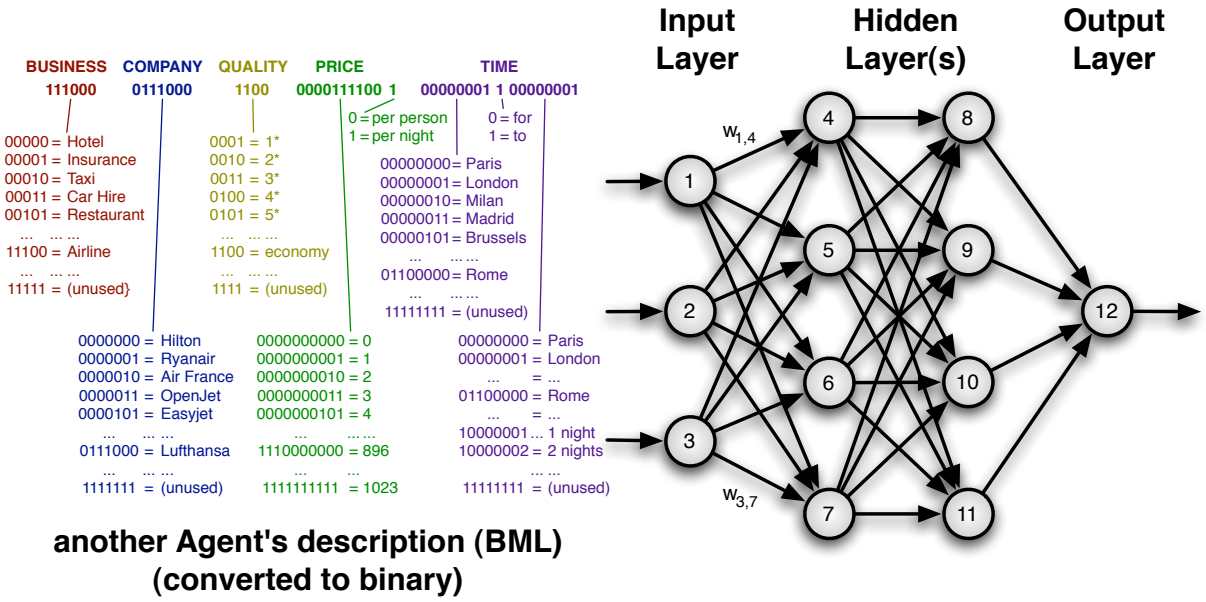


Figure 4: NN Structure for the NN-based Intelligence Component

The size of the *input layer*, the number of neurons, is proportional to the BMLv1 description of the agent in which the NN is embedded. Generally, a single *hidden layer* is sufficient, and depending on the desired functionality, is usually significantly larger than the *input layer* [33]. The *output layer* consists of a single neurone, with values between zero and one, to be used in providing an answer to the question of similarity.

The ideal scenario of having a large training set upon which to train a NN will not be possible in the EvE. The preliminary simulations have been with multilayer feedforward NNs with backpropogation training, in which a continuous training regime is followed. As the agent needs to be trained to recognise similarity to itself, during its deployment to the EvE, the first training set will consist of its own BML description. At the start of an agent's life-cycle its NN's pattern recognition capabilities would be limited, but still functioning. When visiting a Habitat based on an inter-agent interaction, whether successfully or not, it can be added to the training set.

The greatest challenge of the *distributed intelligence* will be the pre-processing of agent descriptions (BML) to be used with the NNs, in the same way that the greatest challenge of the EvE is the processing of the service BML descriptions for fitness determination. The pre-processing will require an ordering of the properties within a BML description.

3.1.2 Support Vector Machines

SVM are a learning system based on recent advances in statistical learning theory, and is becoming established as a tool for machine learning. SVM are a set of related supervised learning methods used for classification and regression. They belong to a family of generalised linear classifiers. They can also be considered a special case of Tikhonov regularisation. A special property of SVM is that they simultaneously minimise the empirical classification error and maximise the geometric margin. Hence, it is also known as the maximum margin classifier [58].

SVM are forcefully competing with NNs as tools for solving pattern recognition problems. They are based on some beautifully simple ideas and a clear intuition of what learning from examples is all about. More importantly they are also showing high performances in practical applications [53]. In very simple terms, SVM correspond to a linear method in a very high dimensional feature space that is non-linearly related to the input space. Although we think of it as a linear algorithm in a high dimensional feature space, in practice, it does not involve any computations in that high dimensional space. Using kernels, all necessary computations are performed directly in the input space.

SVM were developed for binary classification from their inception, making them ideal for our *distributed intelligence* purposes. But, before any of these methods can be applied to the agent description data, it should be pre-processed to be suitable for analysis by computational learning techniques. For that data should be represented as labelled vectors in a high dimensional space. This representation is usually constructed to preserve as much information as possible about features which are responsible for correct classification of samples [53].

A simple way to build a binary classifier is to construct a hyperplane separating class members from non-members in this space. This is the approach taken by perceptrons, also known as single-layer NNs. Unfortunately, most real-world problems involve non-separable data for which there does not exist a hyperplane that successfully separates the class members from non-class members in the training set [22]. One solution to the inseparability problem is to map the data into a higher-dimensional space and define a separating hyperplane there. This higher-dimensional space is called the feature space, as opposed to the input space occupied by the training examples. With an appropriately chosen feature space of sufficient dimensionality, any consistent training set can be made separable [36].

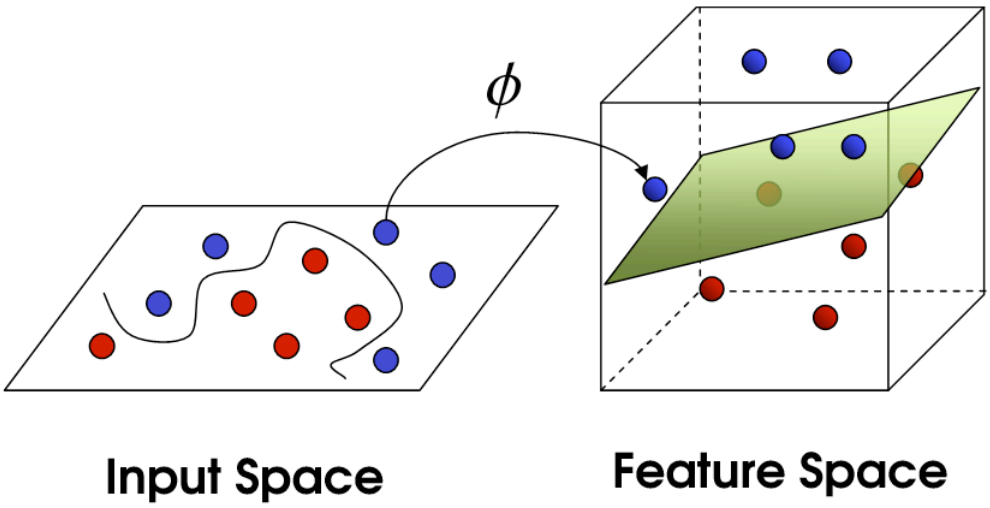


Figure 5: SVM Visualisation

The classifier is then a surface in this high dimensional space that separates it two parts for class and non-class. SVM do not try to construct a classifying surface directly in the given target space, contrary to many standard approaches. First, sample points are projected to a significantly higher dimensional space, where separating surface can be found in a form of hyperplane. Corresponding surface in the original space is then presented as a result of SVM training.

However, translating the training set into a higher-dimensional space incurs both computational and learning-theoretic costs. Representing the feature vectors corresponding to the training set can be extremely expensive in terms of memory and time. Furthermore, artificially separating the data in this way exposes the learning system to the risk of finding trivial solutions that overfit the data.

Support vector machines elegantly sidestep both difficulties. SVMs avoid overfitting by choosing a specific hyperplane among the many that can separate the data in the feature space. SVMs find the maximum margin hyperplane, the hyperplane that maximixes the minimum distance from the hyperplane to the closest training point. The maximum margin hyperplane can be represented as a linear combination of training points. Consequently, the decision function for classifying points with respect to the

hyperplane only involves dot products between points. Furthermore, the algorithm that finds a separating hyperplane in the feature space can be stated entirely in terms of vectors in the input space and dot products in the feature space. Thus, a support vector machine can locate a separating hyperplane in the feature space and classify points in that space without ever representing the space explicitly, simply by defining a function, called a kernel function that plays the role of the dot product in the feature space. This technique avoids the computational burden of explicitly representing the feature vectors [37].

The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set examples will be classified. As long as the kernel function is legitimate, an SVM will operate correctly even if the designer does not know exactly what features of the training data are being used in the kernel-induced feature space. The definition of a legitimate kernel function is given by Mercer's theorem: the function must be continuous and positive definite [36]. Human experts often find it easier to specify a kernel function than to specify explicitly the training set features that should be used by the classifier. The kernel expresses prior knowledge about the phenomenon being modeled, encoded as a similarity measure between two vectors.

Aside from counteracting overfitting, SVM use of the maximum margin hyperplane leads to an uncomplicated learning algorithm that can be reduced to a convex optimisation problem. To train the system the SVM must find the unique minimum of a convex function. Unlike the backpropagation learning algorithm for artificial neural networks, a given SVM will always deterministically converge to the same solution for a given data set, despite the initial conditions. For training sets containing less than approximately 5000 points, gradient descent provides an efficient solution to this optimisation problem.

Another appealing feature of SVM classification is the sparseness of its representation of the decision boundary. The location of the separating hyperplane in the feature space is specified via real-valued weights on the training set examples. Those training examples that are far from the hyperplane do not participate in its specification and therefore receive weights of zero. Only the training examples that are close to the decision boundary between the two classes receive non-zero weights. These training examples are called the support vectors, since removing them would change the location of the separating hyperplane [62].

Training an SVM requires the solution of a very large quadratic programming problem (finding lagrange multipliers for each data point) which is often intractable. Sequential Minimal Optimisation (SMO) approaches the solution by solving for two lagrange multiplier's (keeping the others fixed) at each step, and doing hill-climbing. Because there are only two variables, the quadratic problem can be solved analytically, making the inner loop of the training program very fast. SMO is competitive with other SVM training methods such as Projected Conjugate Gradient 'chunking' and in addition is easier to implement [50].

The SVM learning algorithm is defined so that, in a typical case, the number of support vectors is small compared to the aggregate number of training examples. This property allows the SVM to classify new examples efficiently, since most of the training examples

can be safely ignored. In essence, the SVM focuses upon the small subset of examples that are critical to differentiating between class members and non-class members, throwing out the remaining examples. This is a crucial property when analysing large data sets containing many uninformative patterns, as happens in many data mining problems. SVMs effectively remove the uniformative patterns from the data set by assigning them weights of zero [22].

Uniquely to the *distributed intelligence*, there is only one piece of input data and it has a very high dimension (relative to typical input). The high dimension of the data albeit untypical, still conforms to the requirements for SMO training of SVM. The greater concern is the insufficient number of training samples, a minimum of two samples (one positive and one negative) are required. Ideally, more would be much better. The best way to manage this requirement is to create variants from the one piece of data we have which is a 100% positive match for a similarity function. If the variant is less than 10% different of the original then it should be processed as a positive input, and if the variant is more than 10% different it should be processed as a negative input. This approach is possible because the hyperplane determination in the feature space, values the points near to the hyperplane far more significantly over those far from the hyperplane.

4 Simulation and Results

Throughout the EvE simulation we assumed 100 users, which means that at any time the number of users joining the network equalled the number of users leaving. The users' habitats were initially randomly connected. At the start, users deployed five services each. Services could migrate from one habitat to another thereby affecting the connections between the habitats. Users would submit requests for services. Each user then deployed a new service after the submission of five requests. The order in which users submitted requests was random. A more detailed description of the simulation can be found here [10].

4.1 Evolutionary and Ecological Dynamics

The EvE was simulated with several key variables being recorded to determine whether it displayed behaviour typically of natural ecosystems, i.e., ecological dynamics.

4.1.1 Species Abundance

Relative abundance is the proportion of all organisms in a community belonging to a particular species. Relative abundance distributions provide a measure of inequalities in population size within an ecosystem. In most ecosystem, this distribution takes a log-normal form [7].

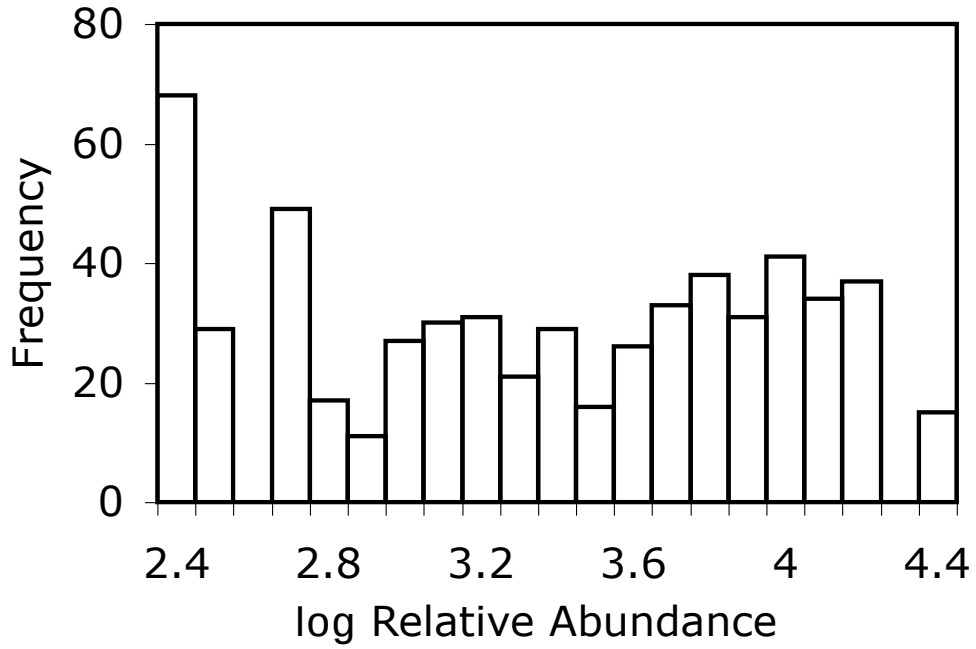


Figure 6: Relative Abundance Distribution of the EvE Digital Ecosystem

A snapshot of the digital organisms within the EvE was taken after a thousand user requests. A species was defined by the grouping of genetically similar digital organisms (based on their service descriptions), with no more than 10% variation within the species group. Relative abundance was calculated for each species, and is shown grouped by

frequency for Figure 6. In contrast to expectations from natural ecosystems, relative abundance in the EvE does not conform to the lognormal. We speculate that the high frequency for the lowest relative abundance was caused by the dynamically re-configurable landscape, which allowed species of small abundance to survive as their respective habitats were clustered by the system. It is possible that this effect also skewed other frequencies for relative abundance measures.

4.1.2 Species-Area Relationship

The species-area relationship measures diversity in relation to spatial scale. In the EvE, this relationship represents how similar solutions are to one another at different habitat scales. For this experiment, we assume each habitat to have an area of one unit.

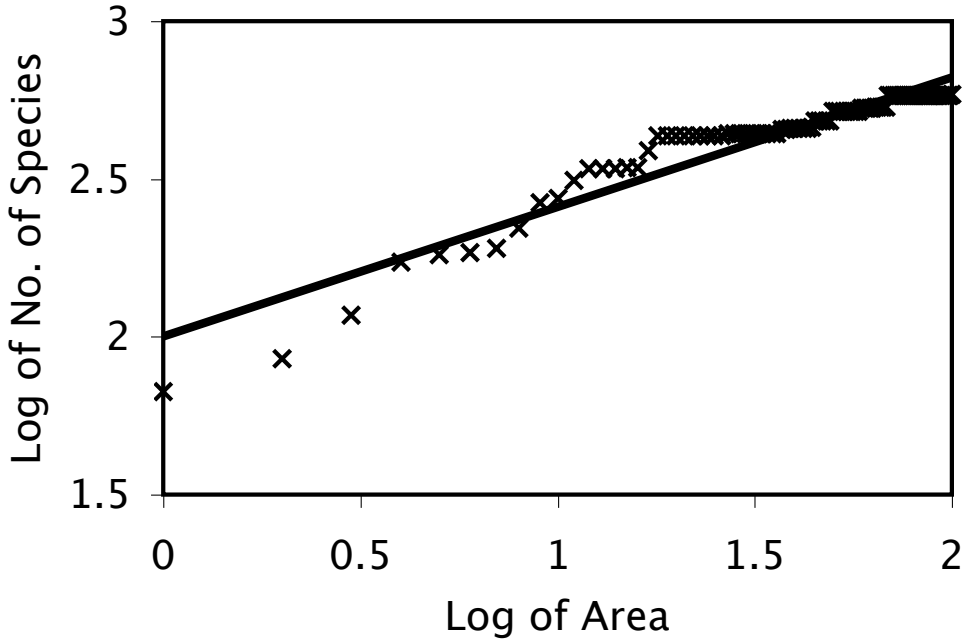


Figure 7: Log_{10} of No. of Species against $Log_{10}s$ of Area for the EvE Digital Ecosystem

Again, a snapshot of the digital organisms within the EvE was taken after a thousand user requests. The number of species at n randomly chosen habitats was measured, where n ranged between 1 and 100. For each n, 10 sets of measurements were taken at different random sets of habitats to calculate averaged results. The log_10 values of the results are depicted in Figure 7. The species-area relationship is commonly found to follow a power law in natural ecosystems [4]. The distribution of species diversity over a spatial scale in the EvE appears to demonstrate similar behaviour to our expectations from natural ecosystems. However, diversity at fine spatial scales appears to be marginally lower than predicted by the line of best fit. Again, this result may be explained by the dynamically reconfiguring habitat network.

4.1.3 Summary

The experimental results indicate that under simulation conditions the EvE behaves in some ways like a natural ecosystem, providing us with sufficient confidence of the ecological

dynamics within the EvE. We can therefore proceed with evaluating the effect of the *distributed intelligence* on the evolutionary dynamics of the EvE, which will occur via the ecological dynamics (as previously discussed).

4.2 Adaptive Efficiency of The EvE

We also tested the efficiency of the system in responding to user requests, a measure of how well the system serves the purpose for which it is ultimately intended, and to be able to see where the *distributed intelligence* could be most effective. The primary goal of the EvE is to handle and satisfy user requests for services. Therefore, an important measure for determining the success of the EvE is its performance relative to a traditional SOA based system.

We simulated a simple system with a distributed service registry based on a traditional SOA approach. We compared a typical simulation run of the EvE under the above conditions with a typical run of the traditional SOA based system. We limited the time available to the SOA to search the distributed registry for a user request, to the time required by the EvE to respond to the same user request.

In Figure 8 we graphed the percentage match to the user requests for typical runs, as determined by a distance function between the request and the service descriptions.

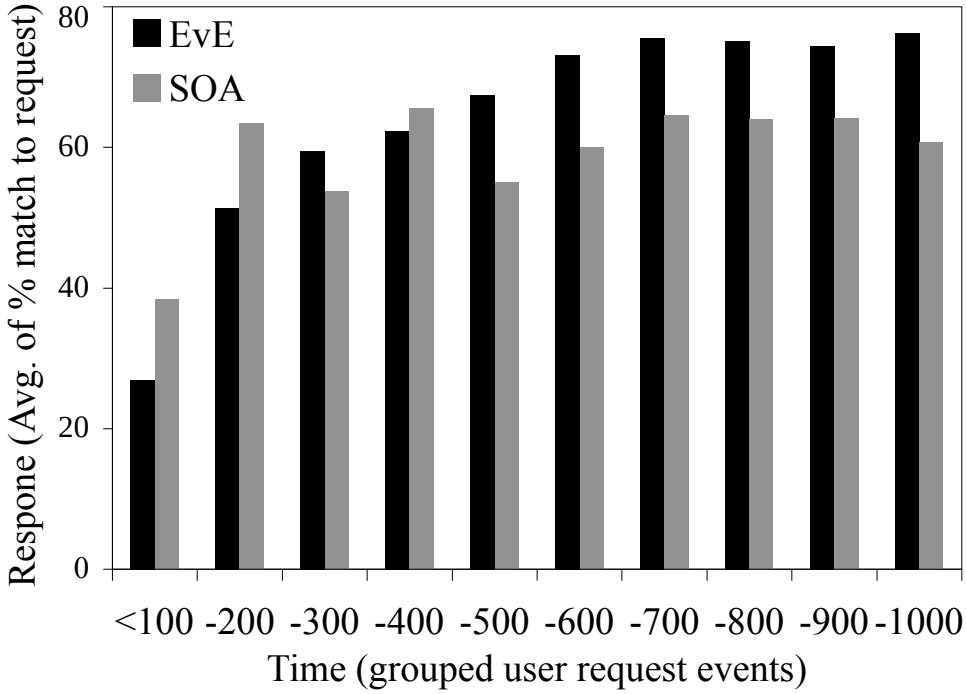


Figure 8: Graph: Performance of the EvE against a traditional SOA based system

Both the EvE and the traditional SOA based system performed as expected, providing better responses to user requests as more services became available. The traditionally designed reference system initially performed better than the EvE, but with the increasing number of services the EvE began to outperform the reference system. Even at large

scales, the responses provided by the reference system were more effective than those given by the EvE ($t(230) = 5.63, p < 0.001$). However, the incidence of non-response in the SOA increased to over 50% at large scales, whereas in the DBE, non-response frequency remained under 4%. This result indicates that the DBE offers a dramatic improvement in response frequency at large scales. This behaviour was expected, given that a key feature of the EvE is its scalability and robustness[12, 10].

4.3 Distributed Intelligence

There are several scenarios in which the *distributed intelligence* can optimise the EvE. Including changing conditions in the DBE, and the bootstrapping scenario of the EvE. The bootstrapping of the EvE involves activating many Habitats simultaneously, and allowing them to optimise their connectivity and Agent-Pools, through the ecological dynamics of succession.

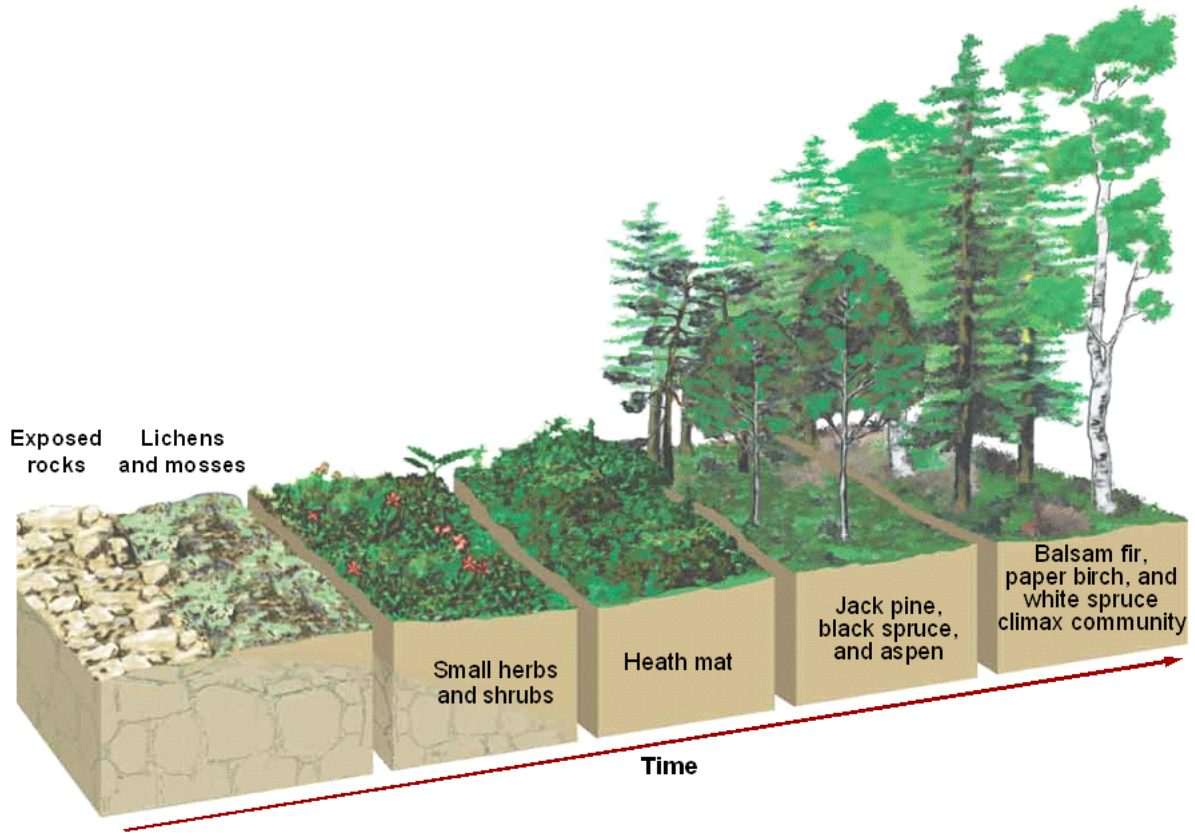


Figure 9: Ecological Succession

We considered existing theories of complexity in biological ecosystems and how it would apply to the EvE Digital Ecosystem. We looked for a high-level understanding that would apply equally to both biological and digital ecosystems. The complexity has to increase initially or there would not be an ecosystem, and presumably this increase eventually stops, because there is a limit to how many species can be supported. The period in between is more complicated.

If we consider the neutral biodiversity theory, which basically says network aspects of

ecosystems are negligible, we would probably get a relatively smooth progression, because although you'd get occasional extinctions, they would be randomly isolated events whose frequency would eventually balance arrivals, not self-organised crashes like in systems theory. In systems theory, when a new species arrives in an ecological network, it can create a positive feedback loop that destabilises part of the network and drives some species to extinction. Ecosystems are constantly being perturbed, so it is reasonable to think that a species that persists is most likely to be involved in some type of stabilising interaction with other species. So, the whole ecological network evolves to resist invasion. That would lead to a spiky succession process, perhaps getting less spiky over time.

So, which theory is more applicable to a digital ecosystem depends on the extent that species in the ecosystem act as independent, competing entities (smooth succession) versus tightly co-adapted ecological partners (spiky succession). Our Digital Ecosystem despite its relative complexity, is quite simplistic compared to a real biological ecosystem. We have the essential and fundamental processes, but no sophisticated social mechanisms. The *distributed intelligence* will be the first such mechanism. Therefore, currently the neutral biodiversity theory with a smooth succession is more probable. The effectiveness of responses to user requests will be used as an estimate of complexity, from all habitats within the Digital Ecosystem. The *distributed intelligence* can optimise the EvE by helping the agents to find their niches within the ecosystem. Specifically, the *distributed intelligence* should help the EvE to adapt faster. This has all been summarised graphically in Figure 10.

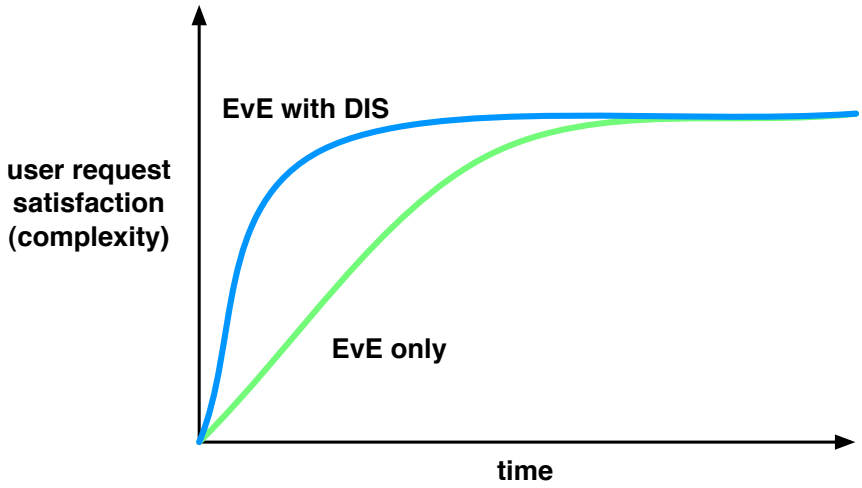


Figure 10: EvE Digital Ecosystem Complexity Over Time

The global behaviour predicted will also be visible within a single Habitat cluster. A simulation was constructed with 100 SMEs and their Habitats. The SMEs produce services and requests, and cannot satisfy their own requests. The range and diversity of requests and agents were constrained to average at 60%, so that the full range of results could be seen. The programming proved to be a significant task, as simulating the DIS required much of the existing EvE simulation to be rewritten, primarily because it had to be changed from a single Habitat simulation to a multiple Habitat simulation. Code from the 'Programming Neural Networks in Java'[32], from the Java Developer's Journal (JDJ) was extended to create the NNs based variant of the *individual intelligence* component. The 'BioJava' library [8] was used to create the SVM based variant of the *individual intelligence* component.

4.3.1 Neural Networks

The EvE performed as expected, adapting and improving over time to reach the 'climax community' (in biological terminology), as can be seen by the graph in Figure 11 in which a typical run of the EvE is shown. In biological terms, each species has only evolved over 10 generations on average and already the EvE is reaching consistently near 70% responsiveness to user requests.

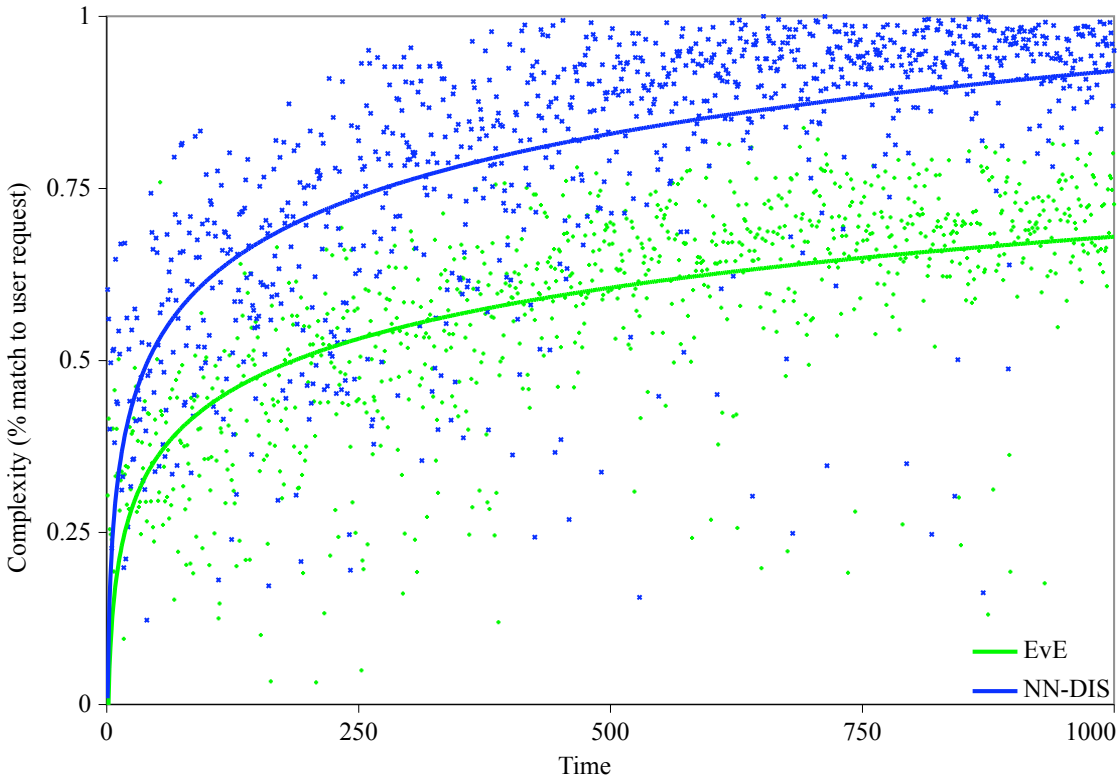


Figure 11: Graph: EvE and NN-based DIS over Time

The *distributed intelligence* using the NNs was then activated within the simulation, and the same measurements of the EvE were taken. So, also in Figure 11 is a typical run of the DIS powered by NNs (NN-DIS), which affected and optimised the EvE almost immediately. The EvE with the NN-DIS, reached the same performance on the EvE in less than a fifth of the time, and within the same timeframe reaching consistently over 90% in response to user requests. Although the effect is impressive, it is not as significant as the *preliminary* results of this simulation shown in Appendix A of deliverable 6.6 [17]. In deliverable 6.6 the simulation 'run' shown was atypical, and not typical as previously thought, but then the results were preliminary.

4.3.2 Support Vector Machines

Figure 11 shows a typical run of the DIS using SVM (SVM-DIS) relative to the NN-DIS and the EvE. The effect of the SVM-DIS on the EvE was more significant than the NN-DIS, but not profoundly so. Given the very significant effect of the NN-DIS on the EvE, it was expected that even the superior SVM matching would not be able to improve significantly upon the NN-DIS.

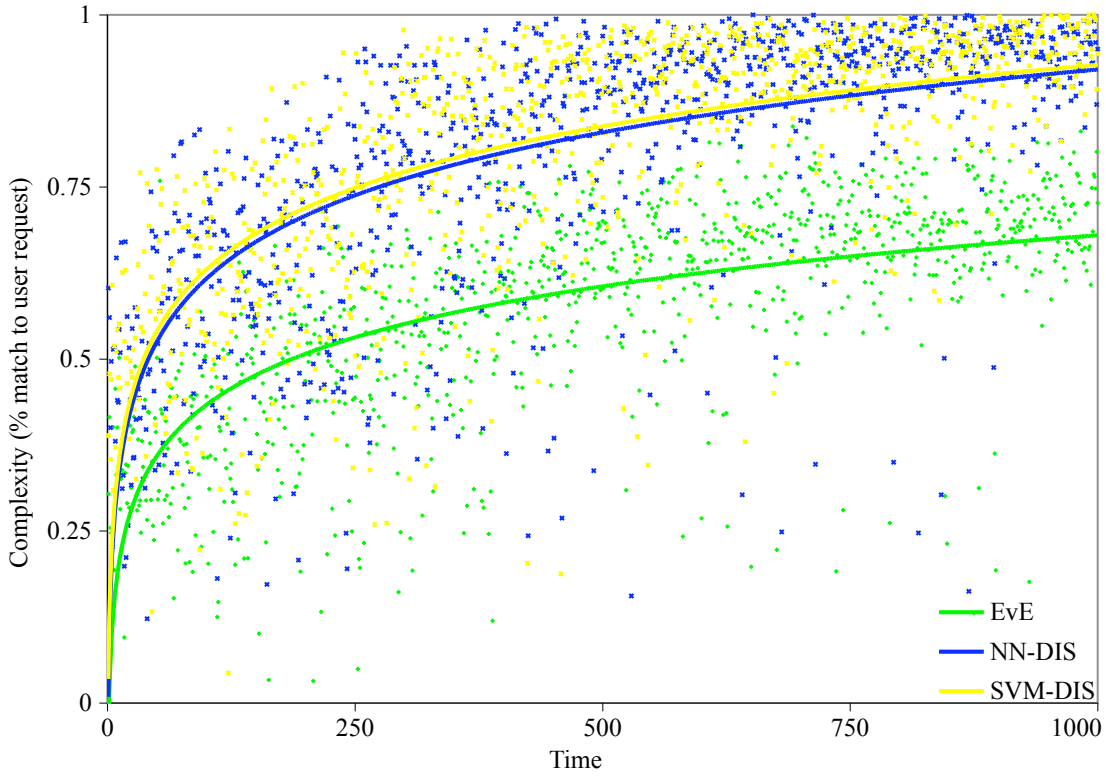


Figure 12: Graph: EvE, NN-based DIS and SVM-based DIS over Time

The plot and best fit curves of the graph in Figure 12 do not clearly show that the SVM-DIS reduces the frequency of poor matches (<50%) to nil by the 750th time unit, unlike the NN-DIS and the EvE. Figure 13 shows this more clearly.

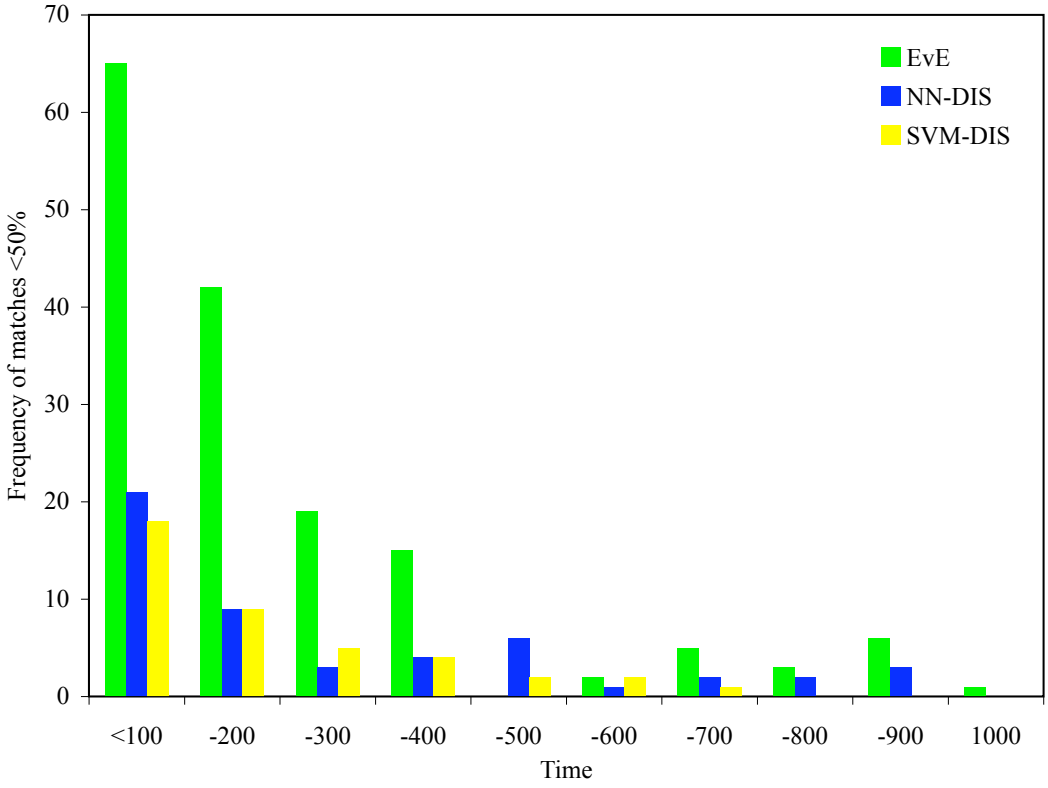


Figure 13: Graph: EvE, NN-DIS vs SVM-DIS

4.3.3 Distance Based DIS (control)

A simple *distance measure* based DIS was used to act as an experimental control, made possible by the simulated agent descriptions (BML) being numerical. This allowed us to determine the contribution of the NNs or the SVM in the effect of the *distributed intelligence* on the ecological and then evolutionary dynamics of the EvE.

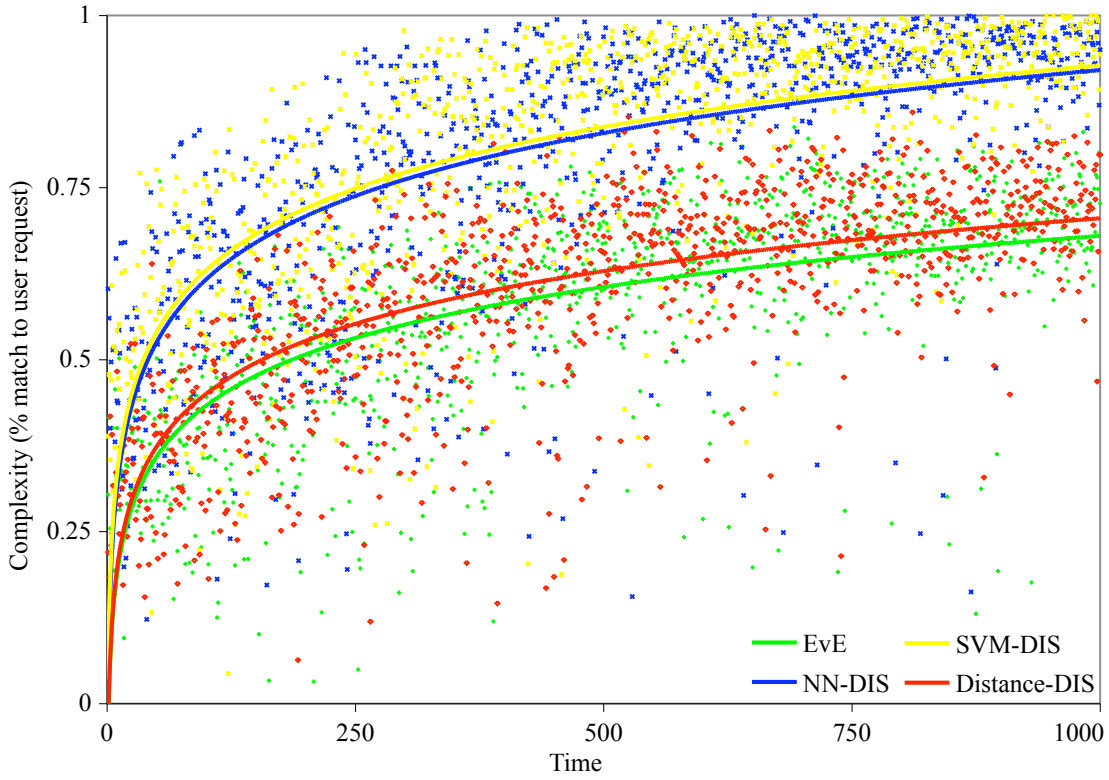


Figure 14: Graph: EvE, NN-DIS, SVM-DIS and Distance-DIS over Time

If we consider Figure 14, the Distance-DIS performed only marginally better than the EvE alone. Clearly, the technique used for the similarity matching function has a very significant impact on the effectiveness of the *distributed intelligence*. Both NNs and SVM are a suitable choice.

4.3.4 Random Migration DIS

The optimisation of the EvE is dependent on agent migration, and the *distributed intelligence* does lead to more migration, although targeted migration. It was still possible the improvement was more from the migration than the intelligent targeting of the migration. So we created a scenario with random migration comparable to the migration of the NN and SVM powered *distributed intelligence*.

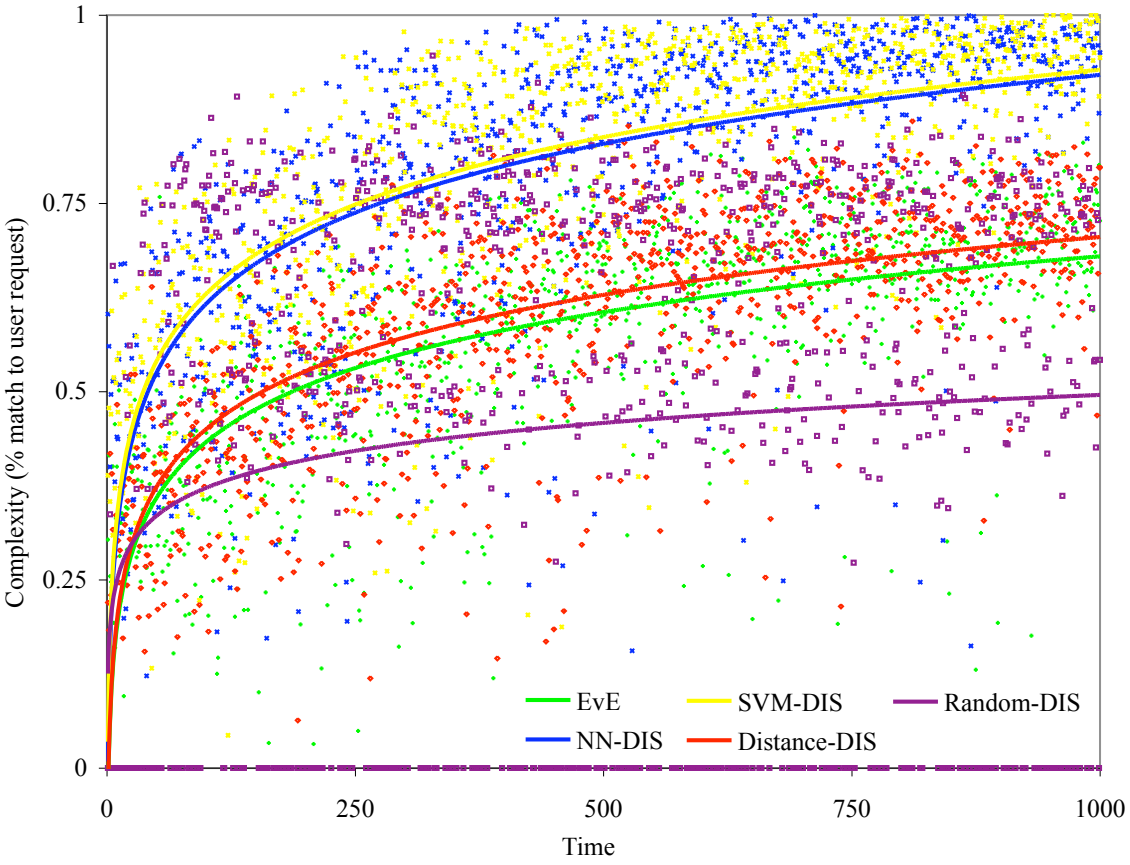


Figure 15: Graph: EvE, NN-DIS, SVM-DIS, Distance-DIS and Random-DIS over Time

The additional random migration, albeit helpful in the very early stages, ultimately decreased the efficiency of the EvE. We therefore feel confident in concluding that the improvement from the *distributed intelligence* is primarily from the intelligent targeting of migration, and not from the additional migration created.

5 Conclusion

The experimental results indicate that under simulation conditions the EvE behaves in some ways like a natural ecosystem, and at large scales outperforms the comparison system based on a traditional SOA. These findings support the general idea that the self-organising properties of digital ecosystems can assist in generating scalable solutions to complex dynamic problems.

The experimental results also indicate that a *distributed intelligence* can positively effect the evolutionary dynamics of the EvE, via the ecological dynamics. An effective similarity matching technique is required for the *distributed intelligence* to operate effectively, and both NNs and SVM proved to be effective, with SVM proving to be marginally more effective than NN. The results also indicated that the 'intelligence' created the improvement in the EvE, and not the additional migration created to enable the *distributed intelligence* to operate.

5.1 Achievements

By comparing and contrasting theoretical ecology with the anticipated requirements of digital ecosystems, we have examined how ecological features may emerge in some systems designed for adaptive problem solving. Specifically, we suggested that a digital ecosystem, like a real ecosystem, will usually consist of self-replicating agents that interact both with one another and with an external environment. Agent population dynamics and evolution, spatial and network interactions, and complex dynamic fitness landscapes will all influence the behaviour of these systems. Many of these properties can be understood via well-known ecological models [41, 35].

A further body of theory treats ecosystems as complex adaptive systems [40]. These models provide a theoretical basis for the occurrence of self-organisation in both digital and real ecosystems. Self-organisation results when interactions among agents and their environment giving rise to complex non-linear behavior. It is this property that provides the underlying potential for scalable problem-solving in a digital environment.

We have created and experimentally investigated the *targeted-migration* approach for a *distributed intelligence* of the EvE Digital Ecosystem. It operates like a SIM for the agents, and constructively interacts with the ecological and evolutionary dynamics. We have considered using NNs and SVM for the pattern recognition needs of the *distributed intelligence*, and found SVM to be more effective.

The value of these contributions to the DBE is that it provides a completed, and now optimised, Digital Ecosystem model for the EvE platform, which uniquely differentiates us from the competition [5].

5.2 Limitations

The only techniques considered to provide a learning capability for similarity recognition were NNs and SVM. The EvE is not a traditional straight-forward genetic algorithm, so there are many potentially effective techniques, but they are all untried. We chose NNs to add the learning capability because it is a well-established technique. We then considered SVM, which is a newer technique and theoretically more suitable for the *distributed intelligence*. Considerable effort is required to enable the agent descriptions to be processed by machine learning techniques, and in the available timeframe it was not possible to integrate any other techniques.

The traditionally designed SOA reference system initially performed better than the EvE in our simulations. The results indicate that the *distributed intelligence* could sufficiently optimise the EvE to outperform the traditional SOA. However, this has not yet been experimentally verified.

It can be argued that ecologists know almost nothing about the self-organising mechanisms of ecosystems. All the theoretical studies that describe the arising of order in natural communities or ecosystems are essentially mathematical algorithms that 'mimic' the biological processes. But there is no direct causal relationship between these kind of mathematical models and nature. Some mathematicians and theoretical biologists think that their models describe the fundamental underlying ecological processes and dynamics, but there is not a single argument that prove such a claim. We consider argument compatible with our work, as it does not actually invalidate the theoretical biology, but it does place it in the correct context.

5.3 Relation to Project Activities, WPs and Tasks

This work has provided scientific support for the engineering decisions of C42, which led to the creation of the DIS in deliverables 6.4 [16] and 6.6 [17]. The task C42 and deliverable 6.6 have provided the fundamental objects and desired behaviour of the DIS, and communicated them to INTEL for DIS's implementation in C53. We have therefore interacted indirectly with task C53, Implementation of the Distributed Intelligence System in deliverable 6.7 [43], which will be integrated with the EvE implementation and DBE core architecture, collaboratively with STU and SUN [24].

We have also continued with task C42 to ensure that the Digital Ecosystem model of the EvE is integrated optimally and faithfully into the DBE Core Architecture. For example, our involvement in developer meetings, including when the switch from SBVR [59] to BMLv1 was made for both the EvE and the DIS [51].

5.4 Progress and Future Work

Regarding the research objectives of whether 'intelligence can optimise the evolutionary process' [25], 'how does this distributed intelligence interact with the ecosystem dynamics' [25], and whether 'software components that are part of genetic selection can be intelligent

in themselves' [25], and as we expected the result was 'a significant change in the evolutionary dynamics with distributed intelligence applied' [25]. The *distributed intelligence* model optimises the Agent-Pool at each habitat to support the evolving populations created to respond to user requests, thereby creating a distributed optimisation to support the local evolutionary optimisation.

Regarding the objective of 'the research, design and implementation of a Distributed Intelligence System (DIS) based on intelligence based on self-organisation to optimise and enhance the Evolutionary Environment (EvE)' [25], we have completed the research sufficiently in task S5 to support the architectural specification of the DIS in task C42. The task C42 was used in the implementation process of C53, providing continued support to the task C53 in achieving the objective of an 'implementation of a DIS which optimises the EvE' [25].

Regarding the objectives of 'the dissemination effort to distribute and explain our results to the computation work package contributors in C53 via C42' [25], the 'creation of a high-level design specification for the Distributed Intelligent System which augments the EvE, for use in the implementation of the Distributed Intelligence System' [25], 'finalisation of the EvE Architecture specification' [25], and 'a design of the DIS which is complementary to the EvE' [25], deliverable 6.6 [17] was explicitly provided to meet these objectives and to complement our dialogue with the partners involved.

We will continue to investigate the EvE and the DIS beyond the end of the DBE project, in the work remaining for the author's PhD and as part of the OPAALS project [47]. Firstly, we would like to consider the optimality of the distribution of the agents within the Habitat network of the EvE, including when the DIS is active. Secondly, if we could make the logical units within the EvE smaller, i.e. make the base unit for evolution smaller than the agent, the system would potentially be more adaptable. Similar to the way in which the 'reduced' instruction set of RISC processors proved more powerful than the 'complex' instruction set of CISC processors.

5.5 Summary

The *distributed intelligence* model has been investigated, and the experimental results indicate that it will optimise the EvE. Based on our theoretical understanding and the experimental results we would recommend SVM over NN for the learning based pattern recognition functionality of the DIS. We arrived at your new SVM recommendation after it was feasible to include this approach in the task of implementing the DIS, and therefore the DIS implementation was developed on the initial recommendation of NNs. Our research will continue to study the complex system that the EvE with *distributed intelligence* represents, to better understand and optimise it. STU has created a dedicated open-source simulation framework to assist further research into the EvE concept [38, 56]. The EvE is currently being implemented [42] and the next major stage in this research will be to collect real world data to investigate whether the EvE can perform usefully in a natural setting. SOAs, such as the DBE, promise to provide potentially huge numbers of services that programmers can combine via standardised interfaces, to create increasingly sophisticated distributed applications. The EvE optimised by the *distributed intelligence* model extends

this concept by the automatic combining of available and applicable services in a scalable architecture.

References

[1] S Abe. *Support Vector Machines For Pattern Classification*. Springer, 2005.

[2] C Adami and C T Brown. Evolutionary learning in the 2d artificial life system "avida". In R Brooks & P. Maes, editor, *Artificial Life IV: Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages pp.377–381, Cambridge, MA, 1994. MIT Press.

[3] E Alba and J M Troya. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4):31–52, 1999.

[4] A P Allen and E P White. Effects of range size on species-area relationships. *Evolutionary Ecology Research*, 5(493-499), 2003.

[5] J Aparicio. D2.3 Software Roadmap. *Digital Business Ecosystem*, Contract no 507953, 2006.

[6] Mike Begon, John Harper, and Colin Townsend. *Ecology: Individuals, Populations and Communities. Third Edition*. Blackwell Publishing, 1996.

[7] G Bell. The distribution of abundance in neutral communities. *American Naturalist*, 396(155):606–617, 2000.

[8] biojava.org. Biojava cookbook. *http://biojava.org/wiki/BioJava:Cookbook*.

[9] G Briscoe. Research blog. *http://www.iis.ee.ic.ac.uk/ g.briscoe/ICL/Blog/Blog.html*.

[10] G. Briscoe. Digital ecosystems: Evolving service-orientated architectures. In *IEEE First International Conference on Bio Inspired mOdels of NETwork, Information and Computing Systems (BIONETICS)*, 2006.

[11] G Briscoe. Evolutionary Environment Architecture Requirements. *Internal*, April 2005.

[12] G Briscoe, M Chli, and M. Vidal. BOF-0759 Creating a Digital Ecosystem: Service Orientated Architectures with Distributed Evolutionary Computing. In *JavaOne Conference*, 2006.

[13] G. Briscoe, S Sadedin, and G Paperin. Biology of applied digital ecosystems. In *IEEE First International Conference on Digital Ecosystems and Technologies*, 2007.

[14] G Briscoe and P De Wilde. D6.1 self-organisation in multi-agent systems. *Digital Business Ecosystem*, Contract no 507953, 2004.

[15] G Briscoe and P De Wilde. D6.2 control of self-organisation and a performance measure. *Digital Business Ecosystem*, Contract no 507953, 2005.

[16] G Briscoe and P De Wilde. D6.4 intelligence, learning and neural networks in distributed agent systems. *Digital Business Ecosystem*, Contract no 507953, 2005.

[17] G Briscoe and P De Wilde. D6.6 high-level design specification of the distributed intelligence system. *Digital Business Ecosystem*, Contract no 507953, 2006.

[18] G Brisoce, J Rowe, and P Dini. Evolutionary Environment Discussion Paper. *Internal*, 2004.

[19] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[20] Hewlett Packard CARLY FIORINA. The digital ecosystem. *http://www.hp.com/hpinfo/execteam/speeches/fiorina/ceo_worldres_00.html*.

[21] Censis. Deliverable 27.1: Territorial social capital and driver smes. *Digital Business Ecosystems Project*, 2005.

[22] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[23] Digital Business Ecosystem Project. Execution Environment. *http://swallow.sourceforge.net/*.

[24] P Ferronato. D21.2 architecture scope document. *Internal*, 2005.

[25] M Giorgetti, P Dini, and A Nicolai. Deliverable D1.2, Detailed Work-Plan for the second phase. *Internal*, WP6 description, 2005.

[26] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[27] D Green and S Sadedin. Interactions matter- complexity in landscapes & ecosystems. *Ecological Complexity*, 2:117–130, 2005.

[28] D G Green and M G Kirley. Adaptation, diversity and spatial patterns. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):184–190, 2000.

[29] D G Green, N S Klomp, G Rimmington, and S Sadedin. *Complexity in Landscape Ecology*. Springer, Toulouse, 2006.

[30] D G Green, T Leishman, and S Sadedin. Dual phase evolution: a mechanism for self-organization in complex systems. In *International Conference on Complex Systems*, Boston, MA, 2006.

[31] D G Green, D Newth, and M Kirley. Connectivity and catastrophe - towards a general theory of evolution. In M A Bedau et al., editor, *Artificial Life VII: Proceedings of the Seventh International Conference*, pages 153–161. MIT Press, 2000.

[32] J T Heaton. Programming neural networks in java. *Java Developer's Journal*, 7(5), 2002.

[33] J T Heaton. *Introduction to Neural Networks with Java*. Lightning Source UK Ltd, 2005.

[34] T Heistracher, T Kurz, C Masuch, P Ferronato, M Vidal, A Corallo, P Dini, and G Briscoe. Pervasive service architecture for a digital business ecosystem. In *ECOOP First International Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT04)*, June 2004. http://wcat04.unex.es/.

[35] S Hubbell. *The Unified Neutral Theory of Biodiversity and Biogeography*. Princeton Univ. Press, Princeton, 2001.

[36] T Joachims. *Learning to Classify Text Using Support Vector Machines: : Methods, Theory, and Algorithms*. Springer, 2002.

[37] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[38] Thomas Kurz. Evesimulator. *http://evesim.sourceforge.net/*.

[39] Thomas Kurz. Evolutionary environment simulator. *http://evesim.sourceforge.net/*.

[40] S Levin. Ecosystems and the biosphere as complex adaptivesystems. *Ecosystems*, 1:431–436, 1998.

[41] R MacArthur and E O Wilson. *The Theory of Island Biogeography*. Princeton University Press, 1967.

[42] Claudius Masuch. Evolutionary environment network. *http://evenet.sourceforge.net/*.

[43] D McKitterick. D6.7 implementation of distributed intelligence system. *Digital Business Ecosystem*, Contract no 507953, 2006.

[44] S K Miller. Aspect-oriented programming takes aim at software complexity. *Technologuy*, 2001.

[45] M E J Newman and D J Watts. Scaling and percolation in the small-world network model. *Physical Review E 60*, 6:7332–7342, 1999.

[46] Observatory of Europeans SMEs. Smes and cooperation. *European Commission*, 2003.

[47] FP6 Network of Excellence. Open philosophies for associative autopoietic digital ecosystems project. *http://www.digital-ecosystems.org/cluster/opaals/opaals.html*.

[48] C Ofria and C O Wilke. Avida: a software platform for researchin computational evolutionary biology. *Artificial Life*, 10:191–229, 2004.

[49] A N Pargellis. Digital life behavior in the amoeba world. *Artificial Life*, 7:63–75, 2001.

[50] John C. Platt. *Fast training of support vector machines using sequential minimal optimization*. MIT Press, Cambridge, MA, USA, 1999.

[51] DBE Project. Developer meeting 12-14/09/2006. *dbestudio.sourceforge.net/wiki/*, 2006.

[52] T S Ray. *An approach to the synthesis of life*, volume XI of *Artificial Life II*, chapter Santa Fe Institute Studies in the Sciences of Complexity, pages 371–408. Addison-Wesley, Redwood City, CA, 1991.

[53] S. Ripley. Pattern recognition and neural networks, 1996.

[54] SILICOM. Digital ecosystem. *http://www.silicom.com/de1.shtml*.

[55] R V Sole, D Alonso, and A McKane. Self-organized instabilityin complex ecosystems. *Philos. Trans. R. Soc. Lond.*, B(357):667–681, 2002.

[56] STU. Report on evolutionary and eve simulator implementation. *DBE*, 2006.

[57] SYNTEL. Building your own digital ecosystem: a holistic approach to enterprise integration. *http://www.syntelinc.com/uploadedFiles/Syntel_DigitalEcosystem.pdf*.

[58] Y Tang, B Jin, Y Sun, and Y Zhang;. Granular support vector machines for medical binary classification problems. In *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2004.

[59] M De Tommasi. D15.3 BML framework 2nd release. *Digital Business Ecosystem*, 2005.

[60] C Tschudin. Fraglets - a metabolistic execution model for communication protocol. In *2nd Annual Symposium on Autonomous Intelligent Networks and Systems*, 2003.

[61] Michel Vandenberghe. The business factory - digital ecosystem solution. *http://themaddesigner.free.fr/XeWOW%20White%20Paper.pdf*.

[62] Lipo Wang. *Support Vector Machines: Theory and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[63] M Ward. Life offers lessons for business. *galapagos*, 2004.

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

[65] Wikipedia. Key word in context. *http://en.wikipedia.org/wiki/KWIC*.

[66] Wikipedia. Stop words. *http://en.wikipedia.org/wiki/Stopwords*.

[67] Michael Wooldridge. *Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[68] S Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Sixth International Congress on Genetics*, pages 355–366, 1932.

[69] XIMBIOTIX. About the digital ecosystem. *http://www.ximbiotix.com/desktop/the-digital-ecosystem/about-the-digital-ecosystem.cfm*.

[70] L Yaeger. Computational genetics, physiology, metabolism, neural systems, learning, vision and behavior or polyworld: Life in a new context. In Langton, editor, *Artificial Life III, SFI Studies in the Sciences of Complexity*, pages 263–298. Addison-Wesley, 1993.

[71] X Yang. Chaos in small-world networks. *PHYSICAL REVIEW E*, 63(046206), 2001.

A Updates to the EvE/DIS Architecture

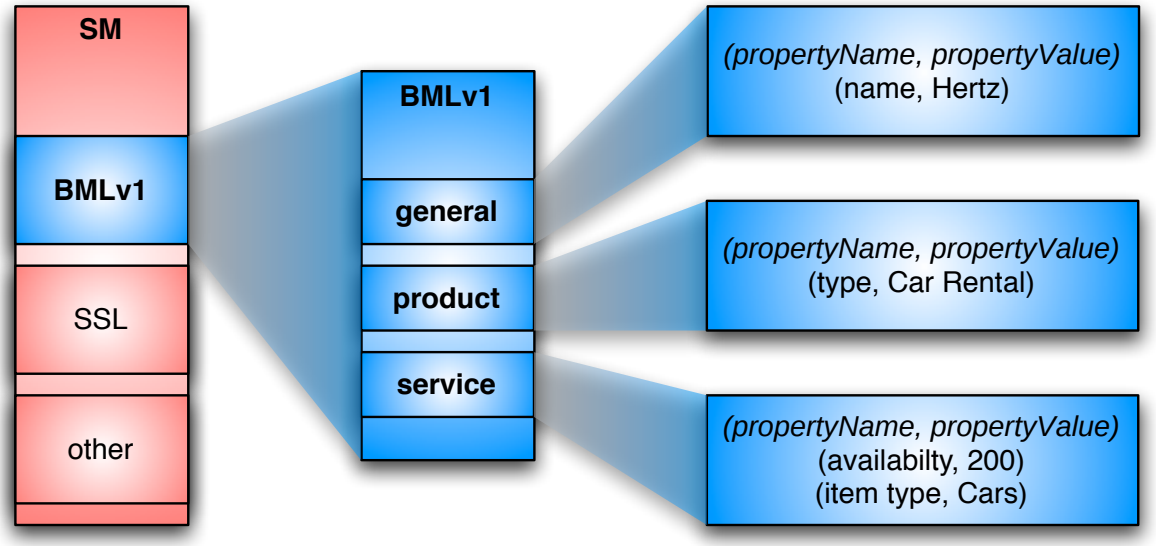
The following section provides minor updates to the EvE/DIS architecture since deliverable 6.6 [17].

A.1 BMLv1 Pre-Processing For DIS Similarity Matching

Given that BMLv1 will now be used for the semantic description of EveServices, an alternative pre-processing algorithm will be required for the Neural Networks (NNs) to act upon, so that the Distributed Intelligence System (DIS) can operate. So, there is still the problem of creating a suitable encoding strategy for the BMLv1 service description, that is compatible with the NNs and scalable in the architecture. This is addressed as follows. Firstly, a very brief summary of the relevant BMLv1 structure is provided. We then provide the algorithm for the pre-processing of the BMLv1 service descriptions, and an approach for combining the results from two NNs (which as will be explained is necessary).

A.1.1 BMLv1 Syntax

The BMLv1 of Service Manifest (SM) is structured as follows:



The BMLv1 is an XML/XMI description consisting of distinct sections, the three of which are concern us are shown in the figure above. The general section will contain information which is either not useful for determining similarity in the context of the DIS (such as company name), or information which is subjectively useful (such as company location). We will therefore focus on the product and service descriptions.

A.1.2 Algorithm: convert BMLv1 service descriptions to ASCII

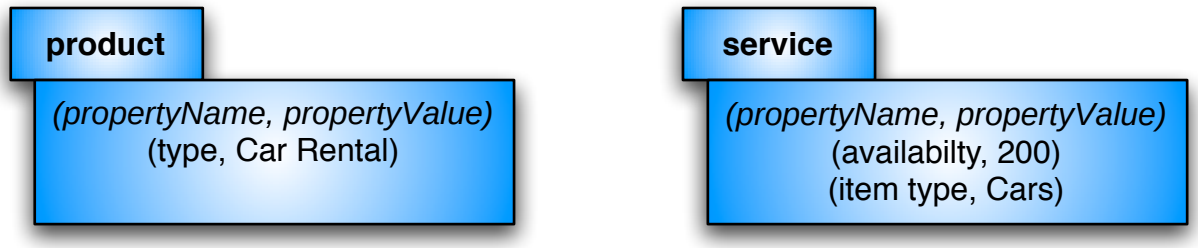
The algorithm firstly reduces the BML to its simplest form, then extracts the relevant terms, then applies a median term length, and then finally convert to ASCII binary. This can be defined in the following steps:

1. Extract the BMLv1 service and product descriptions.
2. Extract the business descriptive terms.
3. Create an order set of the extracted terms.
4. Convert the set of terms to ASCII.

The pre-processing would now be complete and the output can be passed to the NN. The actual steps of the algorithm are specified below in pseudo-code with explanations and examples.

A.1.3 Extract the BMLv1 service and product descriptions.

Th BMLv1 descriptions within SMs are hierarchical XML/XMI. There are defined product and service tags in the markup, and their respective property name and property value tuples are arranged below them in the hierarchy.



A.1.4 Extract the business descriptive terms.

This is simply extracting the property values, and storing them in separate sets for product and service.

OUTPUT: product-set = {Car Rental}, service-set = {200, Cars}

A.1.5 Create an order set of the extracted terms.

The sets of extracted terms now need to be ordered alphabetically. It is suggested to convert all uppercase letters to lowercase where possible to simplify the ordering. For our example:

OUTPUT: product-set = {**c**ar **r**ental}, service-set = {**c**ars, 200}

In the statement above the capital letters changed to lowercase are emboldened to show clearly the change in case, and will remain emboldened in the following steps to show the characters path through the remainder of the algorithm.

A.1.6 Convert set of terms to ASCII.

This involves standardising the length of the terms and then encoding the terms into standard ASCII. The standardising of the term length, requires either padding or shortening the terms. Although it is conceptually easy to shorten or pad terms, the difficulty is in defining an optimal term length to be consistent throughout the EvE/DIS. Also, which is the best choice of the available encoding strategies, or should we create a custom one.

Median Term Length

We need to calculate accurately the median length (number of characters) of key terms within the BML service/product descriptions of the DBE. Unfortunately, there are a small number of examples for BMLv1. It could be estimated using figures from existing analysis of similar communication. Unfortunately, such figures are usual highly proprietary. However, we will outline the correct approach to be taken.

The most similar type of communication style would be advertising brochures for services or products/goods, with written descriptions of business products and services. Advertising style communication, not short catch-phrases, but lengthy descriptions as one would find in advertising brochures. The business type cannot be limited, because it is intended to be applicable to all businesses. Communication can vary highly within the business sector which could also be problematic. We could however make a case to focus on business processes of software services, specifically for the IT sector.

The terms most relevant in communication for describing the service/product, are known as the'key terms' [65]. Not 'stopwords' [66], such as 'the', 'has', 'of', etc. A term may be a number of words, no more than three. I require the median length of syllables and the median number of syllables in the Key Terms, as defined above, for the Type of Communication as defined above. With that I could determine the median number of characters in the Key Terms which will be extracted from the BML service/product descriptions.

There are currently no statistics available on the median word length in BMLv1, or more specifically the key terms we will extract within BMLv1 service descriptions. Also, I have so far found limited information on the average word length in business English, which is for all words and again not just for important keywords. For English, the average word length is just under nine characters, and is only six characters for 'business English'. Most importantly, we are dealing with the median term length and not the median word length. Also, the more padding characters used the more inefficient the NN matching can potentially become, because it can lead to false positives when descriptions with many identical padding characters are compared. This could potentially be trained out, but it would be computationally more expensive. I am therefore going to make the following reasonable assumptions:

- median term length (MTL) in BML will be similar to the MTL in business English
- the majority of BMLv1 terms will consist of one word
- terms deviating from these assumptions will be rare

So, the median term length will be 6 characters and we will remove any space characters before padding or shortening the property values of the BMLv1 in the ordered set from the previous step.

Encoding Strategies

The obvious choices are ASCII, Unicode, or the creation of a custom code. The last would seem unwise and unnecessary. There are surely other encoding strategies, maybe already in use within the DBE core architecture, but we shall focus on the widely used ASCII and Unicode. Unicode is based on ASCII, but has been extended to provide multilingual support, which also makes it more sophisticated (16-bit or 32-bit encoding per character). ASCII is simpler (8-bit encoding per character), but only supports English. The DBE and BMLv1 currently only support English, but promise to support other languages later. ASCII would be sufficient for the current objectives, and it would surely be more computationally efficient. Alternatively, Unicode would additionally meet the future long term objectives, but would be less computationally efficient (more significantly so in the short term than the long term). We believe the choice is ultimately an implementation choice, which must be aware of issues of compatibility, interoperability, and other issues of practicality. For our simulations, which obviously has minimal compatibility or interoperability issues, we have chosen ASCII.

Continuing our example from the previous step:

INPUT: product-set = {**c**ar **r**ental}, service-set = {**c**ars, 200}

As this step is more complicated than the previous ones, we will show its intermediate steps. First, the removal of any space characters in terms with more than one word:

product-set = {**c**ar**r**ental}, service-set = {**c**ars, 200}

Then the standardisation of the term length. This requires the addition of the ASCII null padding character for terms shorter than six characters, and the shortening of terms that have more than six characters.

product-set = {**c**ar**r**en}, service-set = {**c**ars - -, 200 - - -}

The next step is to remove any formatting characters.

car**r**en **c**ars 200

The next step is to convert each character to binary following the ASCII format. The ASCII hexadecimal is initially shown, as it is easier to read and commonly used.

63617**272**656E **63**6172730000 323030000000

The actual output would not have the spaces between the encoded characters, which are present to show the individual character encoding.

The potential loss of information when shortening terms will not overly affect the DIS, because it does not require perfect matching to operate effectively.

A.1.7 Evaluation Function for Two Neural Networks

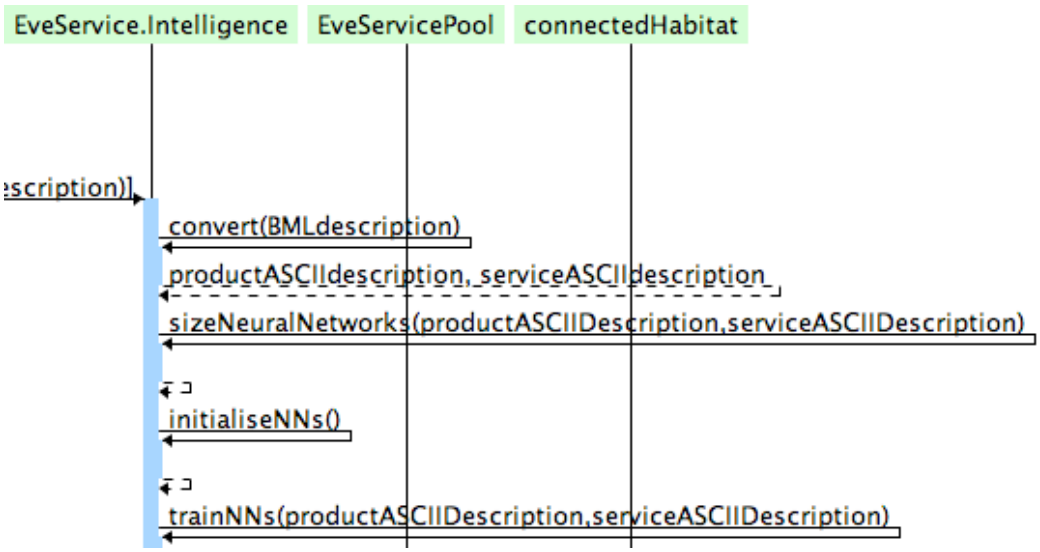
Having two Neural Networks, one for the product description and one for the service description requires that the final matching combines the output from both. We propose that the significance of each on the output is proportional to the size of the input for the Neural Network:

$$Match = \frac{|product - set| * productNN.output() + |service - set| * serviceNN.output()}{|product - set| + |service - set|}$$

This function will be required in the Activate DIS sequence diagram when a match is attempted between interacting EveServices, via the isSimilar() function.

A.1.8 Sequence Diagram

The Deploy Service sequence diagram has been updated and now also includes the method BMLpreprocessing(). The pre-processing of an BML service description is necessary when a DBE service is deployed, specifically when the EveService is created in its home Habitat. We suggest that the ASCII versions of a BML product and service descriptions are stored within the EveService at deployment. This will avoid repeated pre-processing of the service description when the EveService interacts with other EveServices during the periods in which the DIS is activated. The relevant version part of the Deploy Service sequence diagram is shown below. 22



A.2 Updated Sequence Diagrams

The Deploy Service sequence diagram has been updated, and the most relevant part of the Deploy Service sequence diagram is shown below. An updated 'Activate DIS' sequence diagram is also shown on the following page. Full versions of the sequence diagrams can be found at www.iis.ee.ic.ac.uk/ g.briscoe/D6.5/appendix/.

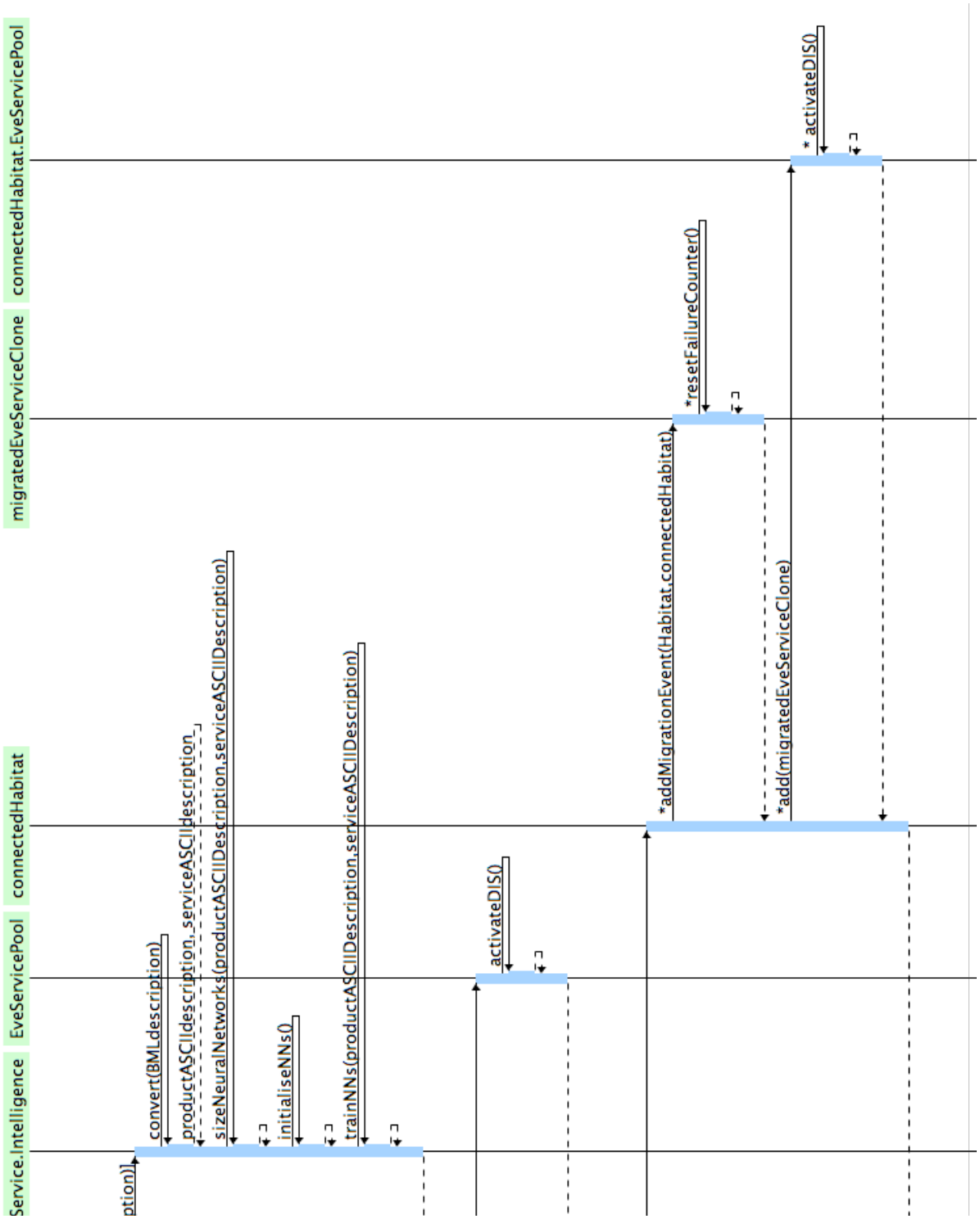


Figure A.1: Section of Updated Deploy Service Sequence Diagram

Figure A.2: Section of Updated Activate DIS Sequence Diagram

A.3 Bootstrapping the EvE

A more biologically biased definition of fitness is required, at least during the bootstrapping phase, and potentially throughout. Otherwise, the currently fittest (sub-optimal) solutions will not be able to migrate and evolve through the Habitat network, making the Habitat network entirely dependent on the fail-safe mechanism for bootstrapping, which may not be sufficient to compensate on such a large scale. The sharing of sub-optimal solutions will allow the EvE Digital Ecosystem to bootstrap naturally, like a biological ecosystem. In our simulations, we achieved this by each Habitat having a running average of the percentage fitness of the solutions created at that Habitat. If a new solution were sub-optimal in percentage terms (less than a 90% match and not to be migrated), but greater than the running average it would still be migrated. We are reiterating this point from deliverable 6.6 [17] because of its importance.

B Master's Thesis: Dynamics of a Digital Ecosystem

This Appendix is a master's thesis, co-supervised by the author, extending our earlier work on Physical Complexity from deliverables 6.1 [14] and 6.2 [15].

Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2005

Project Title:	**Dynamics of a Digital Ecosystem**
Student:	**Fabian Hauser**
Course:	**Occasional**
Project Supervisor:	**Dr P. De Wilde**
Second Marker:	**Dr J. Barria**

Abstract

Evolutionary computing is an optimisation technique mimicking the behaviour of evolution. Therefore it uses the knowledge gained from biology and transfers it to computer science to solve complex optimisation problems. Although it is a field of intense research since almost twenty years, no widely accepted measures for the complexity and efficiency of such evolving systems have been proposed.

The European research project trying to establish a Digital Business Ecosystem, uses evolutionary computing to find optimal service chains. In that scope, this report proposes a new complexity measure for evolving populations and clustering mechanisms for multiple solutions of different populations. To test the measure on the evolving population a simulation is adapted to the newly suggested measure. In this simulation the suggested measure is compared to an existing complexity measure.

This report also highlights the fundamental differences between biology and the evolving agent system used for the Digital Business Ecosystem. It provides solutions to adapt algorithms from biology to this changed requirements.

Acknowledgment

Firstly, I would like to thank my supervisor, Dr. Philippe De Wilde and his PhD student G. Briscoe, for their input, explanations and reading suggestions. They always welcomed my suggestions and supported me in my decisions by giving me additional sources for my research or by highlighting possible difficulties.

I would also like to thank Linda Kovacs, Nick Tulip and Karin Hauser for their feedback on the draft versions of this report. They were a great help for improving this report.

Further, I would like to thank my family and my girlfriend for their support during this busy time.

Contents

Acronyms

Below is a list of the acronyms used within this document.

BML	=	Business Modelling Language
DBE	=	Digital Business Ecosystem
DNA	=	deoxyribose nucleic acid (genetic information)
EC	=	Evolutionary Computing
SME	=	Small & Medium (sized) Enterprises
ICT	=	Information Communications Technology

Chapter 1

Introduction

My final year project is part of a joint European research project called Digital Business Ecosystem. The DBE project is a collaboration between different companies and academic institutions throughout Europe. The DBE project itself is divided into several subprojects concerned with different aspects of the DBE. There are different groups working on business, computing or scientific aspects of the DBE. My final year project is located in the intersection of computing and scientific aspects.

In Figure 1.1 we see that the scientific part focuses on the testing of the evolution in the system, and the computing part is implementing the system according to business needs and the results from science group testing.

1.1 Goals of the Digital Business Ecosystem

The DBE project is a Europe wide project and its goal is to facilitate the adaptation of European Small & Medium sized Enterprises (SMEs) to the ICT.

Currently there are two digital divides within the European Union. The first one is between Nordic/Western and the Southern European Member States and the second one is between larger enterprises and SMEs. Since the SMEs are the innovative backbone of the European economy this is a threatening development. For SMEs, ICT investments are not very popular because it does not concern their core business. Furthermore, the costs of the introduction of ICT usually exceeds the resources of an SME. They cannot finance it autonomously and it is quite difficult for an SME to obtain funding because their main sources are commercial banks which are the most conservative

investors.

To overcome this widening divide, the DBE project will provide an open source infrastructure allowing SMEs to become part of the digital business ecosystem. In this infrastructure, SMEs will not have to provide whole solutions, they can concentrate on their core business and the DBE will link their services together which will form a fully competitive product.

As you can see in Figure 1.2 the path towards this digital business ecosystem follows several stages. SMEs are often one step behind due to a lack of capital or skilled workforce. The DBE project will lower the cost for SMEs to enter into the DBE and provide them with a powerful tool to stay competitive in the digitally connected environment. Thus SMEs are assisted by the DBE project to climb to higher stages on the ICT ladder of adaptation. Additionally, other initiatives have to be taken to achieve a widespread use of ICT. The SMEs problem with a shortage of ICT skilled workforce and knowledge in internet technologies, has to be addressed by local competence

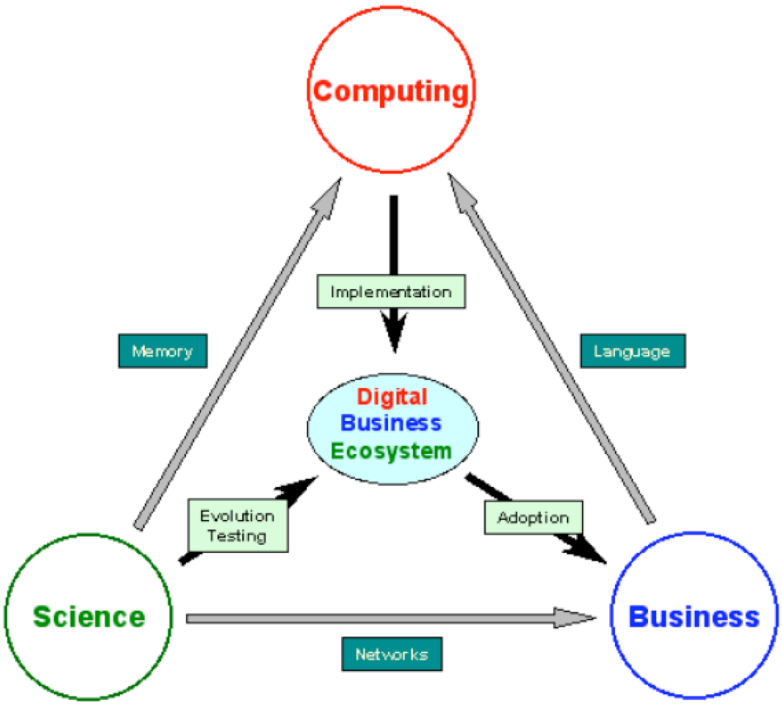


Figure 1.1: DBE Knowledge Flows [?]

centres or virtual learning communities.

All these initiatives together will bring SMEs back to the same level of ICT usage as larger enterprises. Furthermore, because the first rollout is planned to be in Southern Europe it will help to reduce the regional technological gap.

1.2 DBE in Action

The model for the organisation of the DBE comes from biology. It is based on an evolutionary computing environment. For every industry an adapted version of the DBE will be available. The travel industry has different needs than, for example, the construction industry. This industry adapted version will then be instantiated on several servers, one for each region. Each of these servers is a habitat in the evolutionary environment. Some core services will already be available in the base version such as payment services. SMEs can now feed their simple services described in BML as agents into the matching DBE habitat. Those simple agents are the building blocks of the DBEs solutions.

If there is a request for a complex service, the DBE will initialize a population

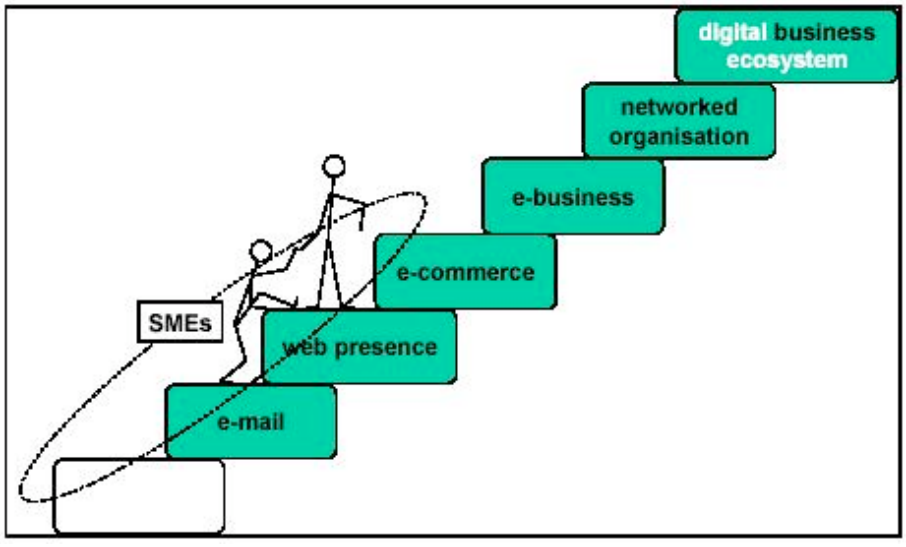


Figure 1.2: Ladder of adaptation of internet technologies [?]

of possible solutions and will then run evolutionary computing algorithms to improve those existing solutions. In this EC, single agents are combined to form an agent-chain and the information about this chain can be compared to the DNA in biology. As in biology this population of agent-chains will undergo mutation, recombination, insertion, deletion and death. The progress of this EC process is closely monitored and as soon as it converges towards an optimal solution, the population is stopped and the best solution found is returned.

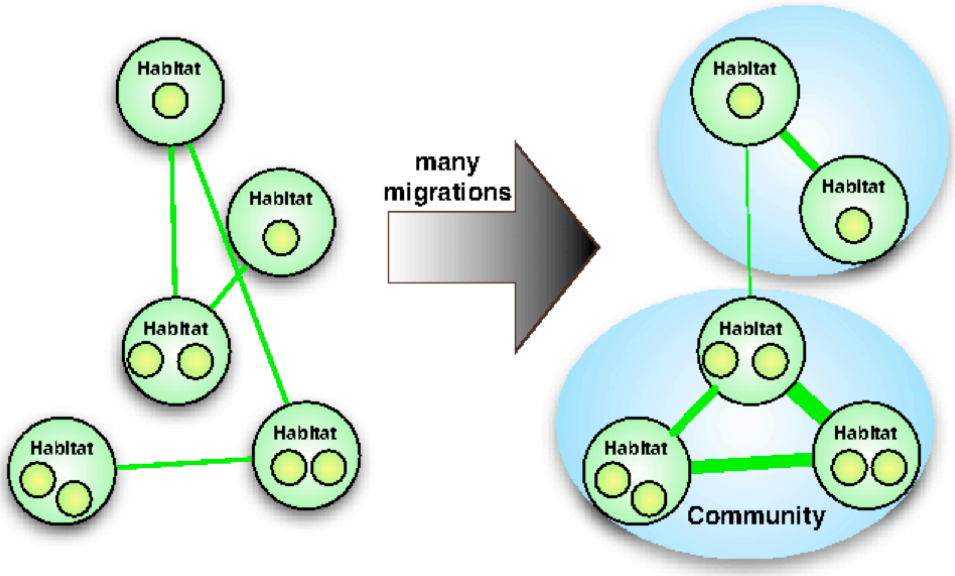


Figure 1.3: Community Formation - Habitat Clustering by Service Migration[?]

The habitats are not closed systems. They are linked with other habitats and they can exchange simple agents and complete agent chains. The exchange intensity will be increased if the received agents are useful or reduced otherwise. According to their exchange intensity multiple habitats will form communities. See Figure 1.3. A community is a cluster of similar habitats.

1.3 Objectives

The objectives of my project are; to improve and extend the given complexity and efficiency measure. Further to create a diversity based, inter habitat

complexity measure for a digital ecosystem; and investigate this in a simulation, modelled as a mobile agent system with distributed evolutionary computing. For this the single-agent station has to be adapted to multiple agent stations with multiple populations.

Furthermore, a model for inter habitats exchange mechanism has to be added. On this simulation different measures will be tested. The entropy-based physical complexity measure proposed by Briscoe[**?**] used for single population can be used as a basis.

Chapter 2

Background

2.1 Evolutionary Computing

Evolutionary computing is a part of bio-inspired optimisation techniques used in a variety of fields. There were three independent approaches towards simulating Darwin's evolution to optimise complex problems. In the 1970s J. Holland proposed genetic algorithms. Rechenberg and Schwefel independently suggested evolution strategies and Fogel evolutionary programming. Later, Koza added another sub-branch called genetic programming.

	Genetic Algorithms	Genetic Programming	Evolution Strategies	Evolutionary Programming
Focus	genotype	genotype	phenotype	phenotype
Representation	bit vector	trees	real vector	real vector (automata)
Mating selection	roulette wheel	roulette wheel	uniform	deterministic
Variation	mutation recombination	mutation recombination	mutation (recombination)	mutation (recombination)
Environmental selection	replacement: M'' = M	replacement: M'' = M	deterministic: μ best	randomized: μ best
Population size	constant	constant	μ parents λ children	constant
Misc			adaptive mutation rates	

*Figure 2.1: Differences between different EC Variants [**?**]*

Implementation Stages
representation of individuals
fitness function
mating selection
environmental selection scheme
mutation operator
recombination operator
parameter setting

Table 2.1: Implementation Stages of EC

Although all were based on the bio-inspired evolution, all four were different and were designed for their special field, see Figure 2.1. Today all these different variants together form the field of evolutionary computing. With increasing computer power it became a rapidly growing field since the mid 1980s. Nowadays, there is still considerable research going on in evolutionary computing.

EC algorithms are population based heuristic algorithms using bio-inspired techniques to solve a problem. A general structure of a genetic algorithm is given by the Figure 2.2.

This description of the algorithm is very generic. For an implementation, several stages of EC need to be adapted to the problem. In general the stages in Table 2.1 can be identified.

For each of these stages decisions have to be made on how it is implemented. There is not one right solution for a stage. The solution for a stage is strongly problem dependent and may differ considerably from one EC algorithm to another. Now I will explain the different stages in detail, highlighting the decisions that have to be taken.

2.1.1 Solution Representation

The chosen representation of individuals can strongly influence the progress EC makes. The search space is given by the representation and an unsuitable representation can slow down EC or it can even get trapped in a local optima. Therefore, the choice of the representation is crucial for successful EC.

Common encodings are bit vectors, integer vectors or real vectors. These encodings are popular because they can be handled similar to genome sequences. Other encodings include trees or graphs.

One might ask the question of why we bother about representation because on a computer everything is represented as bit vectors. The problem with bit vectors is their neighbourhood. For example the number 191 is 10111111 in binary representation. If we now add 1, we get 192 which is 11000000. While in integer representation we just made the smallest possible change

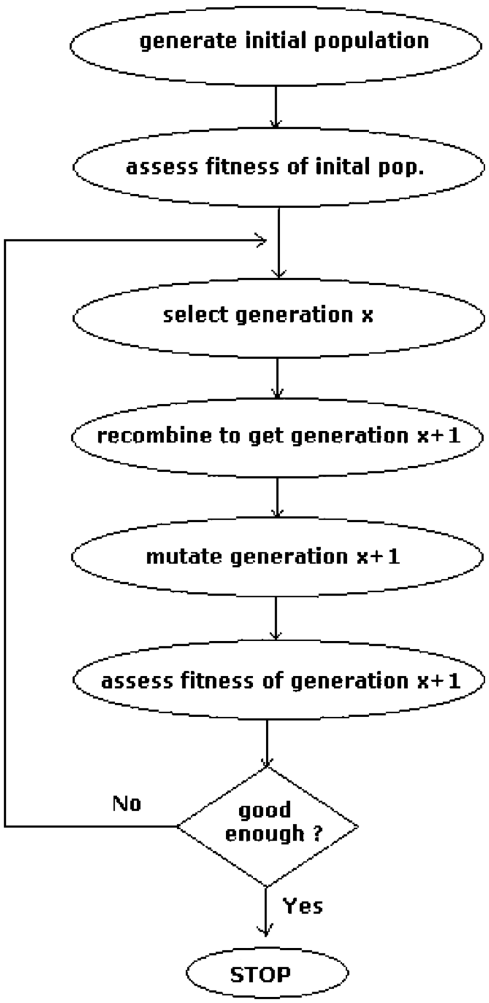


Figure 2.2: Flow Diagram of EC Algorithm [?]

from 191 to 192 in bit vector representation we flipped 7 of 8 bits. So in binary representation there is a hamming distance of 7 between 191 and 192. This might prevent the EC algorithm to find the optima. In Figure 2.3 it is shown that with bit vector representation the population may tend towards the non optimal solution of 191 instead of 192. Therefore instead of using binary number representation one can use Gray code. There two consecutive numbers always have a hamming distance of one. Or one can directly use integers or real numbers.

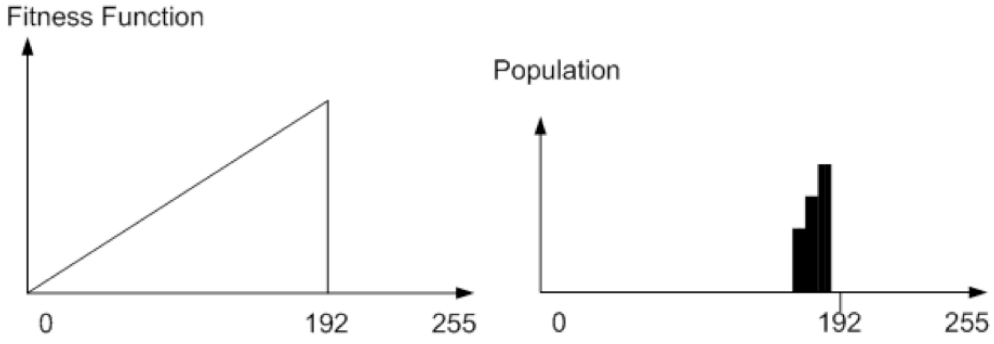


Figure 2.3: Difficulties of EC Algorithm to Find Optima

2.1.2 Fitness Function

The fitness function assigns each individual a fitness value. In the simple case it is just a mathematical function on the individual. It gets more difficult if we want to optimise according to multi objectives, or if the individual has to fulfil certain constraints, or its fitness is dependent on other individuals of the population.

For the case with constraints we have three different approaches. Either we avoid infeasible solutions, we adopt the search space or we just punish the individual not fulfilling the constraints by decreasing the fitness function.

Avoiding infeasible solutions can be achieved by initialising the population with feasible solutions only and by incorporating the constraints into the mutation and recombination operators, to ensure that no infeasible solution gets generated.

Probably the most elegant way is to find a representation where no infeasible solution occurs. Then we can solve the problem in that representation without caring about the constraints.

The third approach incorporates the constraint violation into the fitness function. Given a set of constraints g_1, g_2, $\ldots g_l$ which all should be greater or equal to zero for a feasible solution, we can construct a penalty function C.

$$C_i = \sum_{j=1}^{l} < g_j(x_i) >, \quad \text{with } < g_j(x_i) >= \begin{cases} 0 & \text{if } g_j(x_i) \geq 0 \\ g_j(x_i) & \text{else} \end{cases}$$

And the new fitness function for a maximisation problem is just the addition of the unadjusted fitness function and the penalty function. Remark: $C_i \leq 0$ therefore it decreases the fitness.

$$\hat{f}(x_j) = f(x_j) + C_j$$

The problem with this new fitness function $\hat{f}$ is that it is not guaranteed that the best solution is a feasible solution. To guarantee this, we must change the fitness function in the following way:

$$\tilde{f}(x_j) = \begin{cases} f(x_j) & \text{if } C_j = 0 \\ f_{min} + C_j & \text{else} \end{cases}$$

Where f_{min} is the minimum critical fitness value in the population. This formulation ensures us that the solution with the best fitness is a feasible solution.

Often there are multiple objectives. The return should get maximised and the weight minimised. There are many ways to solve such multi objective optimisation porblems. One is to just optimise according to a weighted average of the objectives. This is often not good enough and we search for multiple optimal solutions to this problem. In this case we probably want to find the Pareto front which is the set of all Pareto optimal solutions. A solution to a multi objective optimisation problem is Pareto optimal if there is no other solution which dominates it, and weakly Pareto optimal if there is no other solution which strictly dominates it. If there are k objectives, domination is defined in the following way:

Relation between x_1 and x_2		
x_1 dominates x_2 :	$\forall\ 1 \leq i \leq k$:	$f_i(x_1) \geq f_i(x_2)$
	$\exists\ 1 \leq j \leq k$:	$f_j(x_1) > f_j(x_2)$
x_1 strictly dominates x_2 :	$\forall\ 1 \leq i \leq k$:	$f_i(x_1) > f_i(x_2)$
x_1 and x_2 are incomparable:	$\exists\ 1 \leq i \leq k$:	$f_i(x_1) > f_i(x_2)$
	$\exists\ 1 \leq j \leq k$:	$f_j(x_1) < f_j(x_2)$

In Figure 2.4 we see in the first picture which points are better (dominating given point), worse (dominated by given point) and incomparable with the given point in the centre of the picture. In the second picture we can see the fitness value of four individuals in a two dimensional fitness function.

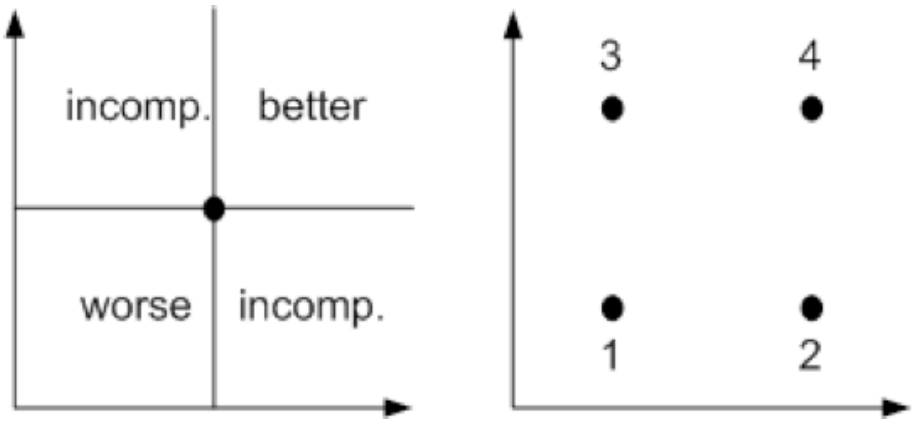


Figure 2.4: Domination with 2 Dimensional Fitness Function

The following relations between the points can be established:

4 dominates $1, 2, 3$
2&3 dominate 1
3&2 are incomparable
4 strictly dominates 1
4 is Pareto optimal
$2, 3$&4 are weakly Pareto optimal

One way to find the Pareto front is to assign the dominance rank as the fitness function [**?**]. Dominance rank counts by how many other solutions the given one is dominated. The optimal value is zero so we want to minimise. A dominance rank of zero does not mean that the solution is Pareto optimal, because we are in the process of optimisation, there are probably solutions which are not yet in the population which would dominate the given solution.

A lot of research has been conducted on finding Pareto optimal solutions. This is beyond the scope of this report. Other replacements of the fitness function are dominance depth by Goldberg [**?**] and Srinivas and Deb [**?**] or dominance count plus dominance rank by Zitzler, Laumanns and Thiele [**?**]. Please consult the indicated sources for more information.

Another approach has to be taken for game strategies and similar tasks. There the fitness is assigned after simulating games between two individuals. The fitness is dependent on other individuals of the same population. This may lead to a population where the different individuals are kind to each other. Instead of an optimal game strategies we find a local optimum. Therefore, in such scenarios it might be better to use co-evolution, where we have two evolving populations and the fitness is assigned by letting them play against individuals of the other population.

A desirable property of a population is that it doesn't have too many duplicates because this reduces the explorative behaviour of EC. Therefore a density measure is introduced and the fitness of solutions in a high density region is reduced. Different density measures can be used. K-th nearest neighbours and histograms are two of them. Please see Goldberg [**?**] for more information.

2.1.3 Mating Selection

In the mating selection individuals are selected for mating and reproduction. The mating selection is usually a randomised process to find the parent individuals for the crossover and mutation operators. For this process we assign for each individual x_i a probability Q_i according to which it is chosen for mating. There are three common fitness assignments in literature:

- fitness proportionate
- rank based
- threshold

The fitness proportionate probability assignment for a maximisation problem with non negative fitness function looks as follows, given a population size of N:

$$Q_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

The problem with fitness proportionate sampling probability assignment is that it is scaling dependent.

To overcome this dependence of the fitness function scaling, rank based sampling rate assignment can be used. We sort the population according to their fitness value and assign each individual a rank. The worst individual gets

rank $R_i = 0$ and the best $R_j = N - 1$. With an intermediate function $\hat{Q}$ we get a new rank based sampling assignment.

linear:	$\hat{Q}_i = \alpha + R_i$
quadratic:	$\hat{Q}_i = \alpha + (R_i)^2$
geometric:	$\hat{Q}_i = \alpha(1-\alpha)^{-R_i}$
exponential:	$\hat{Q}_i = 1 - e^{-R_i}$

This intermediate function $\hat{Q}_i$ needs to be normalised to get the probability Q_i:

$$Q_i = \frac{\hat{Q}_i}{\sum_{j=1}^{N} \hat{Q}_j}$$

α is a problem dependent parameter deciding how important the rank is. For a larger α we get closer to uniform sampling.

The third option is to take a threshold. Choose the threshold in such a manner that the best T individuals are above it and assign uniform probability to each individual above the threshold.

$$Q_i = \begin{cases} 1/T & \text{if } x_i \text{ is above threshold} \\ 0 & \text{else} \end{cases}$$

Now we can sample according to our sampling rate assignment, either by roulette wheel sampling or by stochastic universal sampling. An alternative sampling method is tournament selection.

For roulette wheel sampling each individual gets a range according to their probability between zero and one. If we want to sample M individuals we draw M random numbers between zero and one and return the individual x_i belonging to the range including our random number r.

$$\text{Range for } x_i : r \in \left[\sum_{j=1}^{i-1} Q_j, \sum_{j=1}^{i} Q_j \right), \quad i = 1 \ldots N$$

Roulette wheel sampling has a high variance. No individual is guaranteed to be in the mating selection and in the worst case the mating selection is just M times the worst individual.

Stochastic universal sampling was suggested to get a more predictable sampling. The ranges for the individuals are the same as in roulette wheel sampling, but this time instead of drawing M random numbers between zero and

one, we just draw one random number r between zero and $1/M$. Now we sample at the following positions: $r + k/M$ for $k = 0 \ldots M - 1$. With this it is guaranteed that individuals with a sampling probability higher than $1/M$ are included in the mating selection.

In Figure 2.5 the two methods are shown pictorially. On the left we see one selection by roulette wheel sampling and on the right sampling by stochastic universal sampling.

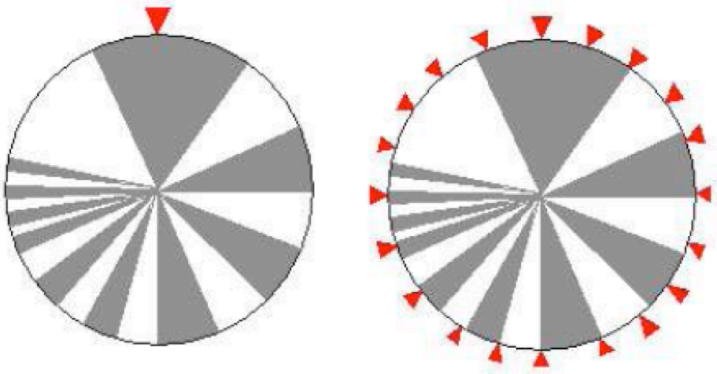


Figure 2.5: Roulette Wheel and Stochastic Universal Sampling [?]

The third sampling method is tournament selection. With tournament selection we do not need sampling probabilities because it works directly on the fitness function.

For every sample a tournament between T individuals is made. An individual wins the tournament if it has the highest fitness of the tournament participants. Tournament participants are selected uniformly out of the whole population.

The advantage of tournament selection is that no pre-processing of the population has to be done. For large populations this pre-processing might take a considerable amount of time. A common value for the tournament size T is 2.

2.1.4 Environmental Selection Scheme

Whereas mating selection usually is organised in a randomised fashion, the environmental selection is most often deterministic. We have two parameters μ and λ where μ stands for the population size and λ for the number of offspring. The environmental selection only allows the best μ individuals to survive, the others get killed.

There are two strategies to generate the new population, either we use environmental selection on the offspring or we use it on the previous generation plus the offspring generation. For this a short notation is introduced.

(μ,λ): Population size is μ, to form next generation λ offspring are generated and the μ best out of the offspring create the next generation. $\lambda \geq \mu$

(μ+λ): Population size is μ, to form next generation λ offspring are generated and the μ best out of the union of offspring and present generation create the next generation. $\lambda \geq 1$

2.1.5 Mutation Operator

The mutation operator is responsible for bringing the variation into EC. In biology mutation is caused by copying errors in reproduction, cell division or by exposure to radiation, chemicals or viruses. It is the driving force of evolution and it is opposing the variety decreasing process of environmental selection. It is important to keep the mutation rate high enough to guarantee genetic diversity. If the mutation rate is too low, EC might get stuck in local optima. If it is too high, EC turns into a random search rather than a guided search.

The mutation operator works on a single individual. It is recommended that a mutation operator fulfils the following two properties:

1. any solution can be generated from any other solution
2. $d(x, x') < d(x, x'') \rightarrow Prob(mut(x) = x') > Prob(mut(x) = x'')$

With the distance measure d, solutions x, x' and x'', mutation operator mut and $Prob$ standing for the probability of the given mutation taking place. The second property states that it is more likely for x to get mutated into a closer solution x' than a more distant solution x''.

For a binary vector search space $Y = \{0,1\}^n$ the strategy is to flip each bit independently with the probability p_m. Usually p_m is chosen as $1/n$. The probability that k bits are mutated is described by a binomial distribution.

$$Prob(k\text{bits mutated}) = \left(\begin{array}{c} n \\ k \end{array} \right) p_m^k (1-p_m)^{n-k}$$

This leads to an expected number of mutations of $n \cdot p_m$ and for the choice $p_m = 1/n$ the number of expected mutations is one. With that choice both of the given properties hold if we use the hamming distance as our distance measure.

This binary vector mutation can be extended easily to a discrete search space. Our search space is now $Y = Y_1 \times Y_2 \ldots \times Y_n$ with Y_i finite. With probability p_m we change each element of the vector independently. Instead of a bit flip at position i, we change the entry to another value of the set Y_i. The new value is chosen by uniform random selection.

For real vectors this solution is not applicable. We generate a Gaussian random number with mean zero and standard deviation σ_i for every position independently. Then we update our real vector $\mathbf{x} = (x_1, x_2 \ldots x_n)$ to

$$x_i' = x_i + N(0, \sigma_i)$$

Sometimes just a global σ is used.

For variable length sequences the mutation operator is extended to allow insertions with probability p_{ins} and deletions with p_{del}.

2.1.6 Recombination Operator

The recombination operator simulates the sexual reproduction. Parents give parts of their genes to children. Usually this is done with two parents as in biology, but it can get extended to more parents. In biology there is often a distinction between male and female and it needs one male and one female to produce offspring, whereas in EC the individuals are usually hermaphrodite, meaning that any two out of the population can reproduce.

There is one property that the recombination operator should fulfil. The child should have a smaller distance to each of his parents than the parents

between each other.

$$x_c = recomb(x_1, x_2) \rightarrow d(x_1, x_c) \leq d(x_1, x_2) \wedge d(x_2, x_c) \leq d(x_1, x_2)$$

This property implies that the recombination is a variance decreasing operator. The children are always in the interior of the space spanned by the parent population.

For vector or sequences the crossover is used as a recombination operator. The crossover takes place with the probability p_c. Otherwise the two parents are returned unchanged. There are three crossover schemes available:

- one-point crossover
- N-point crossover
- uniform crossover

For one-point crossover a position p between one and $l-1$ is chosen randomly. The parents are then split at that position. Two children are then generated taking the front part of one parent and the back part of the other. See Figure 2.6. For sequences with variable length, the crossover point is chosen in each sequence independently.

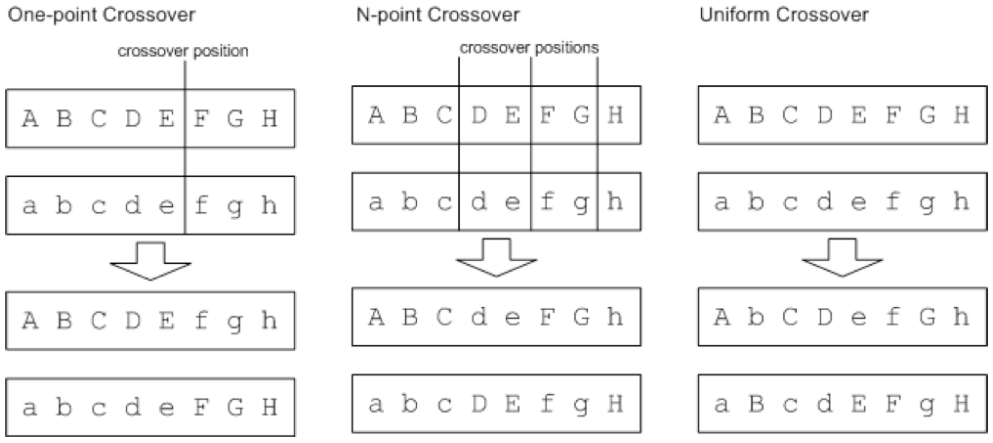


Figure 2.6: Different Crossover Operators on Fixed Length Vectors

The N-point crossover works similarly. Instead of one crossover point, N of them are generated and the parents are split into N+1 parts which are alternately chosen to form offspring.

In uniform crossover, we decide randomly for each site from which parent we take the information.

For real vectors the above mentioned crossover operators are applicable but there are specialised operators available. The intermediate recombination will be explained. We have two parent vectors $(x_1, x_2, \ldots, x_n), (x'_1, x'_2, \ldots, x'_n) \in R^n$ and we generate the child $(x''_1, x''_2, \ldots, x''_n)$ with a random variable $u \in [0, 1]$ according to the following rule:

$$x''_i = x_i + u(x'_i - x_i)$$

2.1.7 Parameter Setting

The following parameters should be set for an EC algorithm:

Symbol	Meaning
μ	population size
λ	number of offspring
α	importance of rank (only for rank based sampling)
p_m	mutation probability
σ_i	variance of mutation (only for real vectors)
p_{ins}	insertion probability (only for sequences)
p_{del}	deletion probability (only for sequences)
p_c	crossover probability

In the design of an EC algorithm there is always the trade-off between exploitation and exploration. Exploitation is the property to converge quickly towards a local solution while exploration tries to escape from local optima to find the global optima. The mutation is the only source for exploration and its variables have to be chosen such that the population maintains its diversity. The antagonist of mutation is the environmental selection scheme. It reduces the variation in the population and focuses on the best individuals. A solution to achieve a favourable balance between exploration and exploitation is to use adaptive parameters.

2.2 Complexity and Efficiency

Complexity is not yet a well defined term for biological organisms. There are different complexity measures available for different fields, but most of them lack certain properties to be used for measuring evolving populations.

> Complexity is so general a term that it seems to mean something different for everyone. Adami in [**?**]

The complexity measure in an evolutionary context tries to measure the influence of the environment on the population. An environment where every individual is treated in the same way will lead to a random population due to the lack of selection criteria. This is considered as zero complexity because by looking at the population we cannot tell anything about the environment. The other extreme case is an environment strongly favouring one special sequence above all others. This will lead towards a population of clones of similar individuals. Every site in the sequence is now defined by the environment. The complexity of this population is equal to the length of its sequence. Everything in between should result in a complexity between zero and the maximum complexity. The maximum complexity for different length sequences is not defined and may differ from one measure to another.

It is a central question of evolutionary biology whether complexity increases in evolution or not. This question cannot be answered in that generality with the current measures. However we can state that for a single population, in a non changing niche, the complexity increases with evolution. For more complex systems with changing niches and co-evolution, a prediction of the development of complexity is more difficult. A change of the niche will result in a momentarily decrease of the complexity of the population occupying this niche. Also, the process of co-evolution is not well formalised and it is not yet possible to predict whether it is increasing or decreasing the complexity.

2.3 Physical Complexity

Adami suggests to use physical complexity to measure complexity in an evolving population[**?**]. It is a conditional complexity defined by Kolmogorov and it states that the complexity of a sequence is conditional on the environment in which it is interpreted [**?**].

First we define the conditional complexity $K(s|e)$, as the length of the smallest program p that computes the sequence s given environment e. $C_T(p, e)$ is the result of running program p on Turing machine T given input e.

$$K(s|e) = \min |p| \ : s = C_T(p, e)$$

Then the physical complexity is defined as the mutual complexity $K(s:e)$.

$$K(s:e) = K(s|\emptyset) - K(s|e)$$

This will give us the number of meaningful sites in sequence s given the environment e. $K(s|\emptyset)$ is the unconditional complexity of s given an empty environment. This is always equal to the length of the sequence s because without an environment e every sequence is random.

In general we cannot determine $K(s:e)$ because we are unable to distinguish between coding and non-coding parts of the sequence. If we are given a whole population of sequences evolved in environment e we can identify coding parts by looking at the probability distribution. Coding parts will have non uniform probability distributions whereas non-coding parts are expected to have a uniform distribution. So we use Shannon's information theoretic entropy to determine the physical complexity. The average complexity of a fixed length population S can then be written as

$$C = \langle K(s:e) \rangle_S = H(S|\emptyset) - H(S|e)$$

We know that $H(S|\emptyset)$ is equal to the length $l = |s|$ with $s \in S$ for fixed length populations and the $H(S|e)$ can be estimated by the per-site entropies $H(i)$ of the sequence, where i is a site in s.

$$H(S|e) \approx \sum_{i=1}^{|s|} H(i)$$

Given the alphabet D and the probability $p_d(i)$ which is the probability of site i taking letter d in the given population, the per-site entropy $H(i)$ is defined as:

$$H(i) = -\sum_{d \in D} p_d(i) \log_{|D|} p_d(i)$$

Random sites will have almost a uniform probability and will therefore have a per-site entropy of nearly one, whereas non-random sites will contribute much less. A fixed site (the same letter at a given site for all individuals in a population) will have exactly zero per-site entropy.

So we write the complexity as:

$$C = l - \sum_{i=1}^{l} H(i)$$

The physical complexity behaves as expected. For totally defined sequences (all sequences in the population are the same) we get a maximum complexity equal to the sequence length l. And for completely random sequences we get zero complexity.

There are $|D|^l$ possible solutions and in order to get an accurate measure, the population should have a sample size of $|D|^l$ to minimise the error[**?**]. For practical applications this becomes too much even for relatively small problems. Therefore, Adami[**?**] reduced the population size to 3600 for an alphabet of size $|D| = 28$ and a sequence length of $l = 100$. In this case $|D|^l$ would have been equal to 5.19×10^{144} which is clearly not a feasible population size for a computer. So the compromise between accuracy and feasibility is to choose a population size of the around $1.29|D|l$. It is larger than $|D|l$ because the population will fluctuate in the process and we have to guarantee at least a size of $|D|l$ for statistical reliability.

2.4 Pairwise Distance

Inspired by Ulam [**?**], Needleman and Wunsch proposed the Needleman-Wunsch algorithm which uses dynamic programming to search for the best alignment for two strings a and b given an assessing function $w(a_i, b_j)$ telling us the value of a match between the letters a_i and b_j.

The simplest w function will just assign one to a match and zero else:

$$w(x, y) = \begin{cases} 1 & x = y \\ 0 & else \end{cases}$$

The Needleman-Wunsch algorithm needs a table D of the size $n + 1$ times $m + 1$ where n is the length of string a and m the length of string b. The table gets filled according to the following recursive equation.

$$D(i, j) = max \begin{cases} D(i-1, j-1) & + & w(a_i, b_j) \\ D(i-1, j) & + & w(a_i, -) \\ D(i, j-1) & + & w(-, b_j) \end{cases} \qquad \forall i = 1 \ldots n, j = 1 \ldots m$$

In the initialisation $D(0, 0)$ is set to 0 and the first row and the first column are set according to the gap value $w(a_i, -)$ respectively $w(-, b_j)$. In our case it is set to 0. The algorithm starts at the upper left corner $D(1, 1)$ and fills the whole table progressively. The distance d_{ab} of the alignment is in the cell $D(n, m)$ and the path can be constructed by backtracking through the table

D.

This algorithm is widely used in genetics on genes and amino acid-chains as well as in language studies. To illustrate the algorithm I will demonstrate a matching between an English word and its German equivalent. The example is taken out of the lecture notes of computational biology [**?**] but adapted to the given w function.

$$a = enough$$

$$b = genug$$

The table is filled using the w function given before.

D		E	N	O	U	G	H
	0	0	0	0	0	0	0
G	0	0	0	0	0	1	1
E	0	1	1	1	1	1	1
N	0	1	2	2	2	2	2
U	0	1	2	2	3	3	3
G	0	1	2	2	3	4	4

Therefore the maximum score is 4 leading to the following matching:

$$\begin{array}{cccccccc} - & E & N & O & U & G & H \\ G & E & N & - & U & G & - \end{array}$$

2.5 Phylogenetic Tree Construction

The task of phylogenetic tree construction is to build a tree which describes the relationship between objects. It's origin is biology. A phylogenetic tree

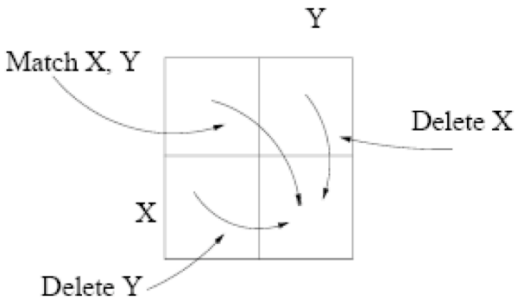


*Figure 2.7: Illustration of the Recursion [**?**]*

is showing the evolutionary relationships among different species. It can be seen as a family tree of life (see Figure 2.8).

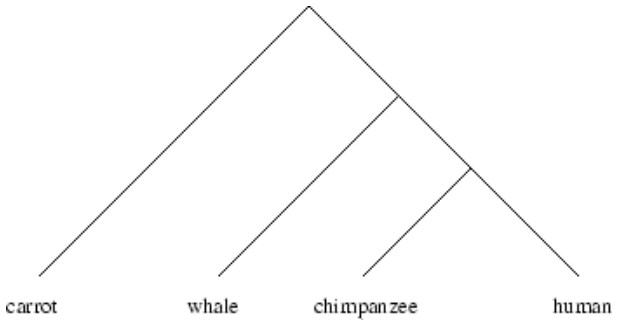


Figure 2.8: Phylogenetic Tree of 4 Species [?]

The basic assumptions behind phylogenetic tree construction are:

1. All species are related by descent from a common ancestor.
2. Species emerge through bifurcation (binary pattern).
3. Changes in characteristics occur in lineages over time.

Phylogenetic trees can be constructed with different classification types. There are three main types. These are:

1. Distance based methods
2. Character based methods
3. Probabilistic methods

The distance based methods require the definition of a distance function between two species. A tree with distances assigned to edges is then constructed in such a way that the distances between a pair of species map to the distances between the according leaves in the tree as accurately as possible.

The character based method defines a set of characters. For each character it can be decided if a species fulfils it or not. Such characters can be mammal, unicellular, eye-colour, etc. Species then are grouped according to the

maximum number of matching characters.

Probabilistic methods try to maximise the likelihood of the tree explaining the given set of objects.

The focus for this report will be on the distance based methods. For further information concerning the other two methods please consult the script and the slides of the course scientific computation[**?**].

The optimal tree construction is an NP-complete problem. There are $\prod_{i=2}^{n}(2i-3)$ different topologies for rooted trees with n leaves. And there are uncountable infinite many trees with real edge lengths. Therefore heuristics have to be applied in order to construct this tree. One heuristic is neighbour-joining.

2.5.1 Neighbour-Joining

The idea behind neighbour-joining is to join close species into clusters. The problem is that it is not sufficient to just pick the two closest species and unite them in a cluster and proceed like that. Since this may lead to considerable errors.

Given the distance table:

Distance	A	B	C
B	5		
C	6	3	
D	10	7	6

This table represents the unrooted binary tree shown in Figure 2.9. But if we always join the closest neighbours and calculate the distance of the new node to other nodes as the mean of the distance of the joined nodes to the other node minus half the distance between the joined nodes this would lead to the result shown in Figure 2.10.
This is not satisfactory because it leads to a different topology and the distance table gets changed to:

Distance	A	B	C
B	5.5		
C	5.5	3	
D	8.25	8.25	8.25

Therefore, instead of joining just the two nearest species, we allow to join the two nearest neighbour which are far away from all other and we allow the edges of the tree to be asymmetric. For that the distance u_i is introduced:

$$u_i = \sum_{k \neq i} \frac{d_{ik}}{n-2}$$

With d_{ij} as the distance between species i and species j and n as the number of species. Instead of merging the minimum out of the distance table d_{ij} we create a new distance table D_{ij} which is calculated as follows:

$$D_{ij} = d_{ij} - (u_i + u_j)$$

The neighbour joining theorem by Studier and Keppler states that for a tree with additive lengths, a minimal D_{ij} implies that i and j are neighbours. So we join i and j to form the inner node k. The distance of this new node k to other nodes m, $m \neq i$ and $m \neq j$ is defined as before:

$$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$$

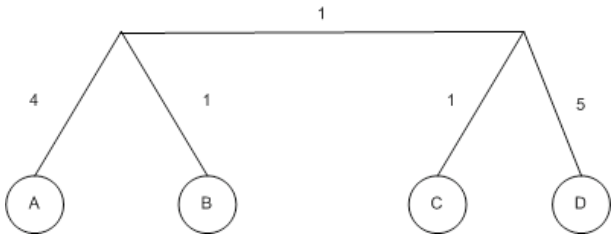


Figure 2.9: Tree Matching Distance Table Exactly

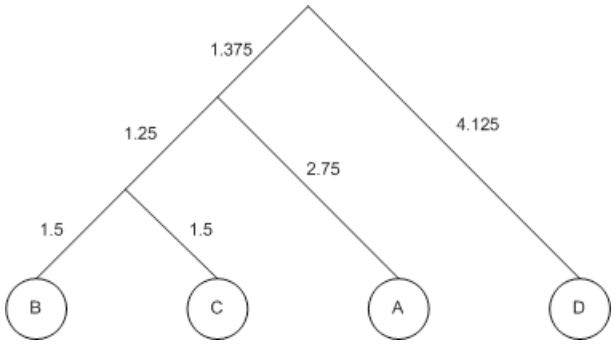


Figure 2.10: Tree Constructed by Joining Nearest Neighbours

The lengths d_{ik} and d_{jk} are now not taken to be equal to $d_{ij}/2$. They are adapted according to the following rule:

$$d_{ik} = \frac{1}{2}d_{ij} + \frac{1}{2}(u_i - u_j), \quad d_{jk} = \frac{1}{2}d_{ij} + \frac{1}{2}(u_j - u_i)$$

Now we replace nodes i and j by node k and repeat this until there is only one node left.

To demonstrate this neighbour-joining algorithm we first calculate the u_i for our example:

i	u_i
A	10.5
B	7.5
C	7.5
D	11.5

Now we construct the changed distance table D:

Distance D	A	B	C
B	-13		
C	-12	-12	
D	-12	-12	-13

There are two minima in this table. Either A and B will get joined or C and D. In the original tree in Figure 2.9 these are also neighbours. Lets call the new inner node E. It is constructed by joining A and B. The new distance table d looks as follows:

Distance D	E	C
C	2	
D	6	6

The distances of A and B to E are:

$$d_{AE} = \frac{1}{2}d_{AB} + \frac{1}{2}(u_A - u_B) = 2.5 + \frac{(10.5 - 7.5)}{2} = 4$$

$$d_{BE} = \frac{1}{2}d_{AB} + \frac{1}{2}(u_B - u_A) = 2.5 + \frac{(7.5 - 10.5)}{2} = 1$$

For the next step the table of the adapted distances D has to be calculated with $u_E = 8,\ u_C = 8,\ u_D = 12$:

Distance D	E	C
C	-14	
D	-14	-14

Since an unrooted tree gets constructed, it does not matter which two out of three we will join together. So we merge E and C to F and calculate all new distances:

$$d_{DF} = \frac{1}{2}(d_{ED} + d_{CD} - d_{EC}) = \frac{6+6-2}{2} = 5$$

$$d_{EF} = \frac{1}{2}d_{CE} + \frac{1}{2}(u_E - u_C) = 1 + \frac{(8-8)}{2} = 1$$

$$d_{CF} = \frac{1}{2}d_{CE} + \frac{1}{2}(u_C - u_E) = 1 + \frac{(8-8)}{2} = 1$$

Therefore, at the end we get the same distances and the same topology for the unrooted tree as in Figure 2.9. So for a species set with additive distances the neighbour joining algorithm will construct exactly the right tree. Distances are additive if there exists a tree which can satisfy all distances without any error. Usually the distance measure will not give us distances which are additive. Therefore an error will be made. The goal is to keep this error as low as possible. This error can be measured as a sum of squares. If we use $tdist(i, j)$ as the sum of the edges on the path from i to j in the tree and d_{ij} as the distance between them given by the distance table then the error is given as:

$$E^2(\text{edge lengths in tree}) = \sum_{i \neq j}(tdist(i, j) - d_{ij})^2$$

By adapting the edge lengths in the tree, the error can be minimised. This procedure just finds the optimal edge length for the given topology. If the given topology is not the best one, a local search can be done to find a better one. The algorithm used for the local search is called 4-optim.

4-optim Algorithm:

1. Save current minimal error E_{minold}
2. Foreach inner edge (not connected to leaf) do
3. Generate the three different topologies for the given edge
4. Perform a local least square fit to obtain the optimal length of the 5 internal edges.
5. Select best topology and update E_{min}
6. End foreach

7. If $E_{minold} > E_{min}$ goto 1. Else return current tree configuration and E_{min}.

It is not guaranteed that this will lead to the best possible tree, but in most cases returns a reasonably good solution.

Chapter 3

From Biology to Single Population DBE

The main difference from biology to the DBE is the alphabet size and the sequence length. Usually in biology a rather small alphabet of size four for DNA/RNA, and a size of 20 for amino acids is used. In the DBE every agent offered by SME's is part of the alphabet. This will lead to an estimated alphabet size of one thousand up to one million. For the sequence length the situation is inverted. In biology the sequences are very long (3 billion for human genome) and are usually split up and processed in blocks of 500. The DBE will have rather short distances because the building blocks are already quite complex. So the length of DBE service chains (sequences) will be between 5 and 20. The search space for biology is around $4^{500} \approx 10^{300}$ and for the DBE it is $1'000'000^{20} = 10^{120}$. So our search space in the DBE is much smaller than in biology. Bear in mind however, that we are trying to solve the optimisation in the DBE, whereas in biology we just analyse the solution found by life.

3.1 Physical Complexity Adapted to DBE

The problem with Adamis[**?**] physical complexity is that it needs fixed length sequences. In the DBE our mutation operator allows insertion and deletions. Therefore we cannot guarantee a fixed length.

For this reason Briscoe[**?**] introduced the physical complexity for variable length populations. Since there is no fixed length l anymore, a new length l_v is introduced. l_v is chosen so that the number of samples per site $i \leq l_v$ is always bigger than $|D|l_v$. The reason for this is that there are not enough

samples for the sites $i > l_v$ to include them in the calculation. If we have a function $sampleSize(i)$ which returns the number of samples at given site i we can calculate l_v as:

$$l_v = \max\{i | sampleSize(i) \geq |D| i\}$$

So l_v will be in the range of zero to l_{max} with l_{max} being the length of the longest sequence in the population.

With l_v instead of l the per-site entropy $H(i)$ is changed to

$$H_V(i) = -\sum_{d \in D} p_d(i) \log_{|D|} p_d(i)$$

Where i ranges from one to l_v and the probabilities are calculated on the sequences which have samples at site i. The sites above l_v are ignored. The variable length complexity can now be written as:

$$C_V = l_V - \sum_{i=1}^{l_v} H_V(i)$$

Based on the complexity, Briscoe[**?**] further introduced the performance measure which shows the complexity relative to the maximum possible complexity. This measure is called efficiency E and is calculated as:

$$E = \frac{C_V}{l_v}$$

The efficiency is between zero and one and shows how many percent of the sequence length is used. The efficiency is not dependent on sequence length. This makes it a good measure to compare different evolving populations with different lengths.

3.2 Distance Based Complexity and Efficiency

The variable length complexity measure suggested by Briscoe has some flaws which I tried to bypass by taking a completely different approach and creating the distance based complexity measure. Instead of the physical complexity, I use pairwise distances with an adopted version of the Needleman-Wunsch algorithm described in the background chapter.

The distance based complexity measure uses a table of pairwise distances to derive a measure for the organisational complexity. It is based on the

Needleman-Wunsch algorithm to find a sequence alignment for two agent chains (sequences).

There are two types of sequence alignments, global and local. In a global sequence alignment, both sequences have to be aligned over the whole length. For local sequence alignment deleting letters at the beginning and at the end of the two sequences is cost free. Since we are interested in the alignment of the whole sequence, global alignment is chosen.

3.2.1 Find Minimum Distance

In our case, instead of maximising this matching, we prefer to minimise the mutational distance. In a mutation step three things can happen. A given letter is exchanged, a new one is inserted or one gets deleted. Each of these mutations should lead to an increase of the pairwise distance of one. To achieve this the function w is changed to the following.

$$w(x,y) = \begin{cases} 1 & x \neq y \\ 1 & x = - \\ 1 & y = - \\ 0 & x = y \end{cases}$$

And we want to minimise the distance so the recursion changes to

$$D(i,j) = minimum \begin{cases} D(i-1,j-1) & + & w(a_i,b_i) \\ D(i-1,j) & + & w(a_i,-) \\ D(i,j-1) & + & w(-,b_i) \end{cases} \quad \forall i = 1 \ldots n, j = 1 \ldots m$$

A small example from biology:

$$a = ACTG$$

$$b = ACGTG$$

We use the w for the mutational distance and try to minimise the distance.

D		A	C	T	G
	0	1	2	3	4
A	1	0	1	2	3
C	2	1	0	1	2
G	3	2	1	1	2
T	4	3	2	1	2
G	5	4	3	2	1

This gives a mutational distance of one with the following matching.

$$\begin{array}{cccccc} A & C & - & T & G \\ A & C & G & T & G \end{array}$$

The matching can be interpreted that in sequence a there was a deletion of G at the third position or in sequence b was an insertion of G at the third position. Usually this is called an indel for insertion or deletion.

3.2.2 Disadvantages of Pairwise Distance

The main problem with this algorithm is its computational complexity of $O(nm)$ which gets too large for very long sequences. In biology sequence alignment is just applied to sections of the genome. In the DBE environment we have very short agent chains compared to DNA sequences in biology. Therefore the algorithm will not run into difficulties and is well suitable for our purpose for comparing two sequences.

Another problem is that the distance is dependent on the length. A distance of 1 in a sequence of length 3 results in a huge change of the sequence, one third is changed. In comparison to that, a distance of 100 in a sequence of 10'000 is a rather small change of just 1 percent. As we are interested in the relative change rather than the absolute distance, we can divide the calculated distance by the maximum distance possible with two strings of length n and m, which is equal to the maximum of n and m.

$$\text{Relative Change} = r_{ij} = \frac{d_{ij}}{\max(m, n)}$$

Where m is the length of sequence number i and n of sequence number j. The values of r_{ij} are between zero and one.

3.2.3 Multiple Sequence Distance

The problem to get the distance with multiple sequence alignment is, that it gets high dimensional very quickly. While pairwise distance calculation has a computational complexity of $O(n \times m)$ which is feasible for relatively short n and m. The computational complexity of a multiple sequence alignment with k sequences s_i of length l_i is $O(\prod_{i=1}^{k} l_i)$. For sequences of the same length l this becomes $O(l^k)$ which is growing exponentially in k. Even for short lengths this becomes infeasible quite quickly so we have to search for an alternative.

3.2.4 Pairwise Distance Table

Instead of solving the problem in one step with a multiple sequence distance, we calculate a table with the pairwise distances. If we construct this table for the samples in Figure 3.1 this leads to the distances in Table 3.1.

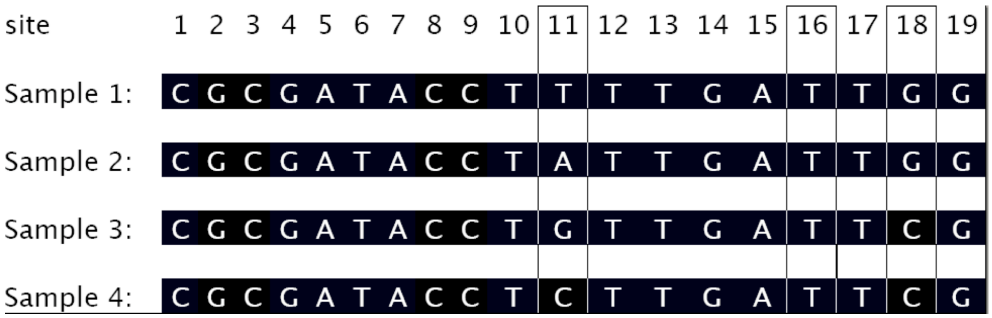


*Figure 3.1: Samples of the Same Genome [**?**]*

Sample	1	2	3	4
1	0	1	2	2
2	1	0	2	2
3	2	2	0	1
4	2	2	1	0

Table 3.1: Pairwise Distance of Samples

Obviously the table is symmetric with zeros on its diagonal, therefore we can reduce the Table 3.1 to the triangular Table 3.2 without loosing any information.

Sample	1	2	3
2	1		
3	2	2	
4	2	2	1

Table 3.2: Pairwise Distance of Samples (just the relevant part)

3.2.5 Complexity from Pairwise Distance Table

The complexity wants to measure how much of a sequence is representing its environment. We know that the maximum pairwise distance is equal to the length of the longer sequence. We are interested in the 'inverse' of the distance which is the length of the longer sequence minus the calculated distance. If we have m and n as the length of sequences i and j, the pairwise complexity c_{ij} can be written as:

$$c_{ij} = \max(m, n) - d_{ij}$$

Since in our example all the sequences have length 19, we just calculate Table 3.3, showing the pairwise complexity of the sequences.

Sample	1	2	3
2	18		
3	17	17	
4	17	17	18

Table 3.3: Pairwise Complexity of Samples

The total complexity of the sequences is just the mean of the pairwise complexities. So, if we denote the pairwise complexity between sequence i and j as c_{ij} and assuming we have k sequences, we can write the total complexity as

$$C_d = \frac{\sum_{i \neq j} c_{ij}}{k(k-1)}$$

Which in our example leads to a complexity C_d of $17\frac{1}{3}$.

3.2.6 Efficiency from Relative Change Table

The efficiency is the normalised complexity. It ranges between zero and one. To get the efficiency we need the relative change. Since in the example form Figure 3.1 all sequences have the same length we can just divide the distances of Table 3.2 to get the relative changes in Table 3.4. If we have varying lengths, we divide the pairwise distance by the length of the longer sequence to get the pairwise relative change.

The total efficiency can be calculated in a similar way to the total complexity, bearing in mind that the relative change is the inverse of efficiency. Therefore, if we denote the pairwise relative change between sequence i and j as r_{ij} and assuming we have k sequences, we can write the efficiency as

$$E_d = \frac{\sum_{i \neq j} 1 - r_{ij}}{k(k-1)} = 1 - \frac{\sum_{i \neq j} r_{ij}}{k(k-1)}$$

In our example the efficiency E is equal to 91.23%. Since the sequence length was constant this is equal to $C_d/l = 17.33/19 = 0.9123$.

3.3 Comparison of Complexity Measures

There are some weak points in the variable length complexity measure suggested by Briscoe. The first one is that the sites above l_v are ignored, since they are considered as noise. Secondly, it is not reflecting the way the mutation operator works and thirdly, it has only discrete maximum complexities. Since the physical complexity works on the per-site probabilities it is invariant to changes in population size (duplication of every individual) under the assumption that the maximum length l_v stays the same. This is not the case for the distance based measure. This can be seen as superior or inferior.

I will illustrate situations in which these flaws will cause problems and it will be shown that the distance based measure can handle these situations more successfully. For all the examples I will use an alphabet of size two (black or white site) and a sequence length between 3 and 5 because otherwise the requirement of having a population with more than $|D|l$ individuals would be too large to represent it on paper.

Sample	1	2	3
2	0.05		
3	0.11	0.11	
4	0.11	0.11	0.05

Table 3.4: Relative Change of Samples

3.3.1 Long Sequences

In the case of long sequences it is unnatural to ignore the last sites of the longest sequences. It is not justified that the last sites are noise. The noise could as well be in the middle or at the beginning of the sequence because it is not known where the new site was inserted.

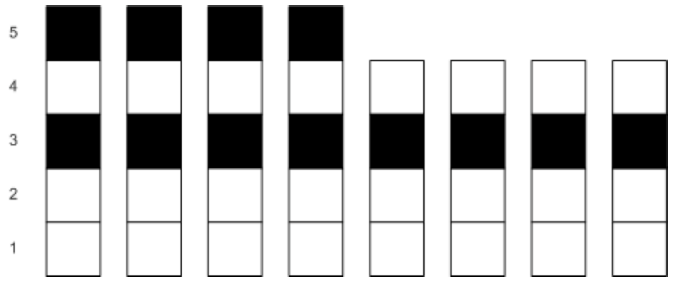


Figure 3.2: Population with Variable Length

Firstly, we calculate the physical based complexity and efficiency for Figure 3.2. The alphabet size $|D|$ is two and the population size $|S|$ is eight. For all individuals the first four sites are defined. l_v is set to four because the following inequality holds:

$$sampleSize(l_v) \geq |D|l_v = 2 \cdot 4 = 8 \leq sampleSize(4) = 8$$

Since all the sites from one to four are equal, the complexity is maximal $C_V = l_v = 4$. The efficiency is equal to $E = C_V/l_v = 1 = 100\%$. This seems quite odd because with a 50% chance there is another site of a known colour.

The distance based complexity will return the following pairwise complexity table:

Sample	1	2	3	4	5	6	7
2	5						
3	5	5					
4	5	5	5				
5	4	4	4	4			
6	4	4	4	4	4		
7	4	4	4	4	4	4	
8	4	4	4	4	4	4	4

and the relative change of samples table will look as follows:

Sample	1	2	3	4	5	6	7
2	0						
3	0	0					
4	0	0	0				
5	0.2	0.2	0.2	0.2			
6	0.2	0.2	0.2	0.2	0		
7	0.2	0.2	0.2	0.2	0	0	
8	0.2	0.2	0.2	0.2	0	0	0

This leads to a distance based complexity of $C_d = 118/28 = 4.21$ and an efficiency of $E_d = 1 - 3.2/28 = 88.6\%$. This result seems quite reasonable. It incorporates the additional sites in a sensible way, without giving the additional site too much weight pushing the complexity to a value of almost five.

3.3.2 Insertion and Deletion

Now we start from a population with the highest complexity (see Figure 3.3) for which both measures return a complexity of four and an efficiency of 100%.

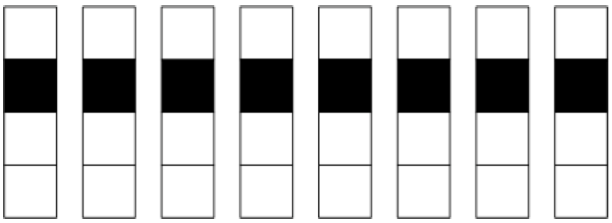


Figure 3.3: Highest Complexity

If we insert one site in individual number three (see Figure 3.4), we get new values for complexity and efficiency. For the physical complexity measure l_v is again set to four, ignoring the additional site in individual number three. The per-site entropies are:

Site	black	white	p(black)	p(white)	entropy
1	1	7	0.125	0.875	0.54
2	0	8	0	1	0
3	7	1	0.875	0.125	0.54
4	1	7	0.125	0.875	0.54
5	–	–	–	–	–

So the complexity is equal to $C_V = 4 - 1.63 = 2.37$ and the efficiency is reduced to $E = 2.36/4 = 59.2\%$. Therefore by just one mutation the complexity and efficiency are almost halved.
For the distance based complexity first the pairwise complexity table is calculated.

Sample	1	2	3	4	5	6	7
2	4						
3	4	4					
4	4	4	4				
5	4	4	4	4			
6	4	4	4	4	4		
7	4	4	4	4	4	4	
8	4	4	4	4	4	4	4

and the table for the relative change of samples looks as follows:

Sample	1	2	3	4	5	6	7
2	0						
3	0.2	0.2					
4	0	0	0.2				
5	0	0	0.2	0			
6	0	0	0.2	0	0		
7	0	0	0.2	0	0	0	
8	0	0	0.2	0	0	0	0

We can see that the complexity stays at $C_d = 4$. The efficiency is reduced to $E_d = 1 - 1.4/28 = 95\%$. Again this result seems to be superior to the physical complexity measure.

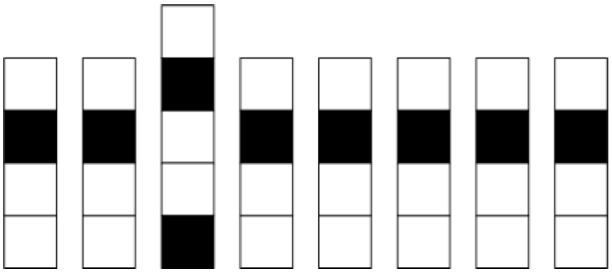


Figure 3.4: Site Inserted

3.3.3 Discrete Maximum Complexities

In the evolutionary process it is possible that the whole population is growing or shrinking. The physical complexity measure has only discrete maximum complexities equal to l_v this will lead to jumps in efficiency and complexity.

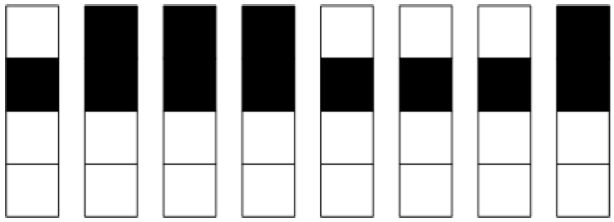


Figure 3.5: Same Length

We start with Figure 3.5. Here the complexity is $C_V = 3$ and the length is $l_v = 4$. This leads to an efficiency of $E = 75\%$. The distance based complexity is $C_d = 96/28 = 3.42$ and the efficiency is equal to $E_d = 3.42/4 = 85.7\%$.

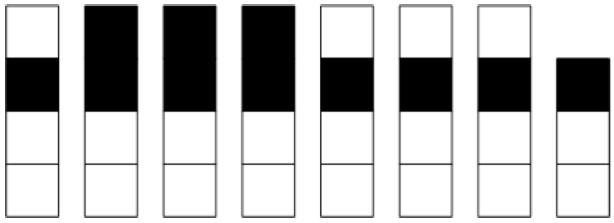


Figure 3.6: After Deletion

After a deletion in site eight (see Figure 3.6) the length l_v is reduced to three because for site four we just have seven samples which would violate the inequality for $l_v = 4$.

$$sampleSize(l_v) \geq |D| l_v = 2 \cdot 3 = 6 \leq sampleSize(3) = 8$$

Therefore the per-site entropies are:

Site	black	white	p(black)	p(white)	entropy
1	8	0	1	0	0
2	0	8	0	1	0
3	0	8	0	1	0
4	–	–	–	–	–

This leads to a complexity of $C_V = 3 - 0 = 3$ and an efficiency of $E = 100\%$.

For the distance based complexity first the pairwise complexity table is calculated.

Sample	1	2	3	4	5	6	7
2	3						
3	3	4					
4	3	4	4				
5	4	3	3	3			
6	4	3	3	3	4		
7	4	3	3	3	4	4	
8	3	3	3	3	3	3	3

and the table for the relative change of samples looks as follows:

Sample	1	2	3	4	5	6	7
2	0.25						
3	0.25	0					
4	0.25	0	0				
5	0	0.25	0.25	0.25			
6	0	0.25	0.25	0.25	0		
7	0	0.25	0.25	0.25	0	0	
8	0.25	0.25	0.25	0.25	0.25	0.25	0.25

This results in a complexity $C_d = 89/28 = 3.32$ and a efficiency of $E_d = 83\%$. This is again smoother than the physical complexity measure.

	C Fig. 3.5	C Fig. 3.6	ΔC	E Fig. 3.5	E Fig. 3.6	ΔE
Physical	3	3	0	75%	100%	25%
Distance	3.42	3.32	0.10	85.7%	83.0%	2.7%

It can be observed that in this case the complexity is almost unaltered for both measures but the physical efficiency makes a 25% change. Here again the distance based measure looks better.

There are also cases in which the efficiency stays the same and the physical complexity changes drastically. If we take Figure 3.3 as starting population we get a complexity of four and an efficiency of 100% for both measures. If we now delete site number four in one individual, the physical complexity jumps to a value of three because l_v is reduced from four to three. The physical efficiency stays at 100%. The distance based complexity will fall to 3.75 and the distance based efficiency to 93.8%.

3.3.4 Population Size

The distance based complexity measure is dependent on the population size. If we have completely random sequences we expect the complexity to be zero. If we now clone every individual the physical complexity stays zero, because it is independent of the population size and only changes its value if there is a change in the per-site probability under the assumption of a fixed l_v. If we look at Figure 3.7 we see the random population A and its derived population B, where every individual of A becomes a cloned twin.

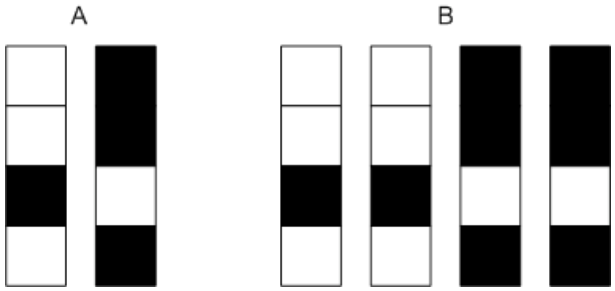


Figure 3.7: Population A and Cloned Twins Population B

Ignoring the fact that the population size is too small for the physical complexity, we calculate the physical complexity with a length of $l_v = 4$. Because the per-site entropy is one for every site in both populations, the physical complexity becomes $C_V^A = 0$ and $C_V^B = 0$.

For the distance based complexity for population A we get the pairwise distance of three by two deletions, one mutation and two matches. Therefore the complexity of the two individuals is $C_d^A = 4 - 3 = 1$. Since for the calculation of the complexity of B we also compare twin individuals with each other, the complexity will not stay one. The pairwise complexity table for B looks as follows:

Sample	1	2	3
2	4		
3	1	1	
4	1	1	4

This leads to a complexity of $C_d^B = 12/6 = 2$. One could argue that this is not a reasonable result because we don't know more than in population A. But one could also claim that it is true that the complexity is higher,

because there are two clusters of maximal complexity in our population.

However the distance based complexity goes wrong when measuring B' (see Figure 3.8) which is obviously more random than B.

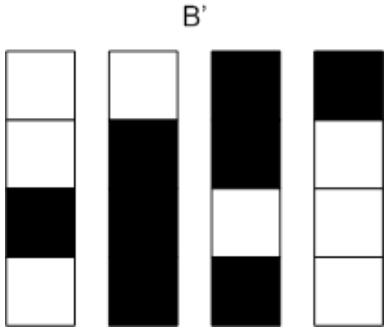


Figure 3.8: Population B' with Same Per-Site Probabilities

Here the pairwise complexity between two individuals is always two as can be seen in the following table:

Sample	1	2	3
2	2		
3	2	2	
4	2	2	2

This leads again to a complexity of $C_d^{B'} = 2$. This is an undesirable result. We would expect the complexity to be lower since there is obviously more randomness in population B' than in population B. The physical complexity assigns to all the three populations A, B and B' a complexity of $C_V = 0$.

It can be shown that populations larger than the size of the alphabet never have zero distance based complexity C_d because there is always at least one match per site. This is not a desired property especially because in the simulation, the population size is always set to the request length multiplied by the alphabet size.

3.3.5 Computing Time

The computing time needed to calculate the complexity clearly favours the physical complexity. For a population size of n, a mean individual length of

l and an alphabet size of m we get the following approximate running time for the two measures:

In a first step, for the physical complexity we need to calculate probabilities for each site. This needs $O(nl)$ time. After that the per-site entropies are calculated, needing $O(ml)$ time. Then the complexity is calculated as the length minus the sum of the per-site entropies needing $O(l)$ time. Since the physical complexity needs n to be greater than lm this leads to an overall runtime in the order of $O(nl)$.

The distance based complexity needs to calculate the pairwise complexities of the population. The pairwise complexity of a pair can be calculated in $O(l^2)$ time. This needs to be done for all possible pairs. There are $(n-1)n/2$ pairs. Therefore this can be done in $O(n^2l^2)$ time. After that, the mean of the pairwise complexities has to be calculated. This needs additional time in the order $O(n^2)$. This leads to an overall runtime in the order of $O(n^2l^2)$.

If we want to compare the two measures in a simulation, the population size is given by the requirement of the physical complexity measure. It has to be larger than lm. Therefore the runtime can be written as $O(ml^2)$ for the physical complexity and as $O(m^2l^4)$ for the distance based complexity. So the runtime for distance based complexity is growing very fast with the requirement of the population size given by the physical complexity measure. This prevents us from doing large scale simulations to compare the two measures on the same population.

3.3.6 Complexities in Simulation

To further compare the two measures some simulations were set up to see whether the flaws of the two measures found in the examples given before, can also be observed in a simulation. The simulation has some restrictions which are described in detail in the simulation chapter. One of them is that the request has a given size and high fitness individuals must have exactly the same size. This leads to a population with almost constant size. To be able to observe some behaviour of long sequences and changing maximum complexity, the population should have different length individuals. Therefore a simulation was set up with a request length of 10. The population was initialized with all possible services of length one. So it is necessary for the population to grow during the evolutionary process. To comply with the requirement of the physical complexity measure regarding the population size and bearing in mind the high runtime requirements of the distance based

complexity measure, the alphabet size was set to 30. This simulation ran for over two hours to produce 50 generations.

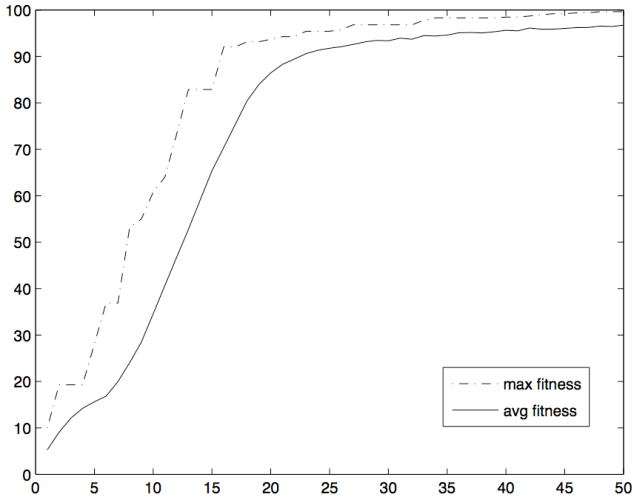


Figure 3.9: Maximum and Average Fitness

Since the fitness function is very sensitive to the length of the individuals, we can see the stair like behaviour of the maximum fitness (see Figure 3.9). The maximum fitness takes a step when a matching letter gets inserted. The average fitness follows more slowly because it takes some time for a successful mutation to produce enough offspring to increase the average fitness observably. By then, the next dominant (highest fitness) individual might have emerged already, leading the mutation into a new direction.

In the plot of the physical complexity measure there is also this stair like behaviour (see Figure 3.10). It has to be noted though, that it is not caused by the same reason as the stair like behaviour of the maximum fitness. The steps are usually two to three generations after we observe a step in the maximum fitness. It shows how long it takes for the dominant individual to spread enough to allow the increase of l_v which is also the maximum complexity for the physical complexity measure. It can be seen that the maximum complexity just takes integer values. This stair like behaviour is not only restricted to the maximum complexity, it can also be observed in the physical complexity measure for the whole population. Increasing the length l_v by one immediately leads to an erratic increase of the complexity because

the complexity of the old sites (below l_v) will not change drastically, and on the additional site, there has been enough time (two to three generations) to build up some complexity. This is exactly the erratic behaviour described in the parts about long sequences, insertion and deletion and discrete maximum complexities.
Compared to the physical complexity the curves of the distance based complexity and its maximum complexity are very smooth (see Figure 3.10). Once the maximum complexity of approximately 10 is achieved (around the twentieth generation) both complexities qualitatively behave in a similar way. The small changes in the physical complexity after the twentieth generation are also present in the distance based complexity but seem to be amplified considerably.

The comparison of the two efficiencies shows again the erratic behaviour of the physical based efficiency E during the first twenty generations and the smooth increase in the distance based efficiency E_d. The peak in the physical based efficiency shows us that the efficiency of the first site was close to 100% before the length l_v got extended to two. After the twentieth generation the efficiencies are very similar, with an amplification of the changes for the distance based efficiency E_d compared to E. This is due to the almost fixed maximum complexity and the similar behaviour of the complexities.

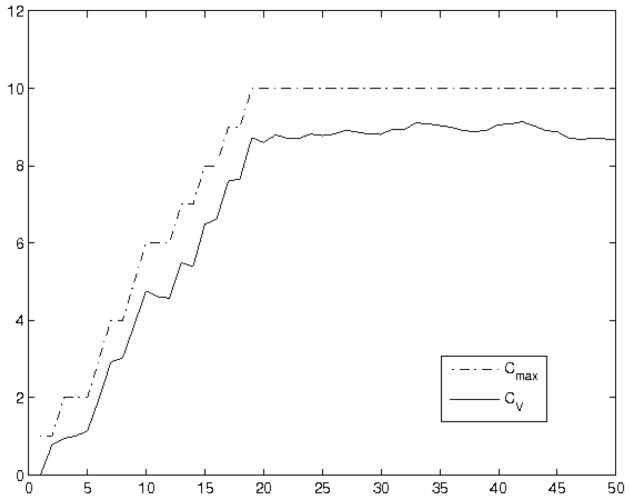


Figure 3.10: Complexity and Maximum Complexity by Physical Complexity Measure

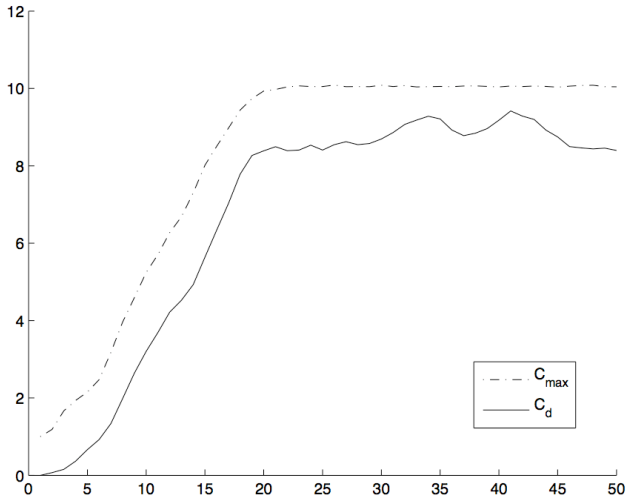


Figure 3.11: Complexity and Maximum Complexity by Distance Based Measure

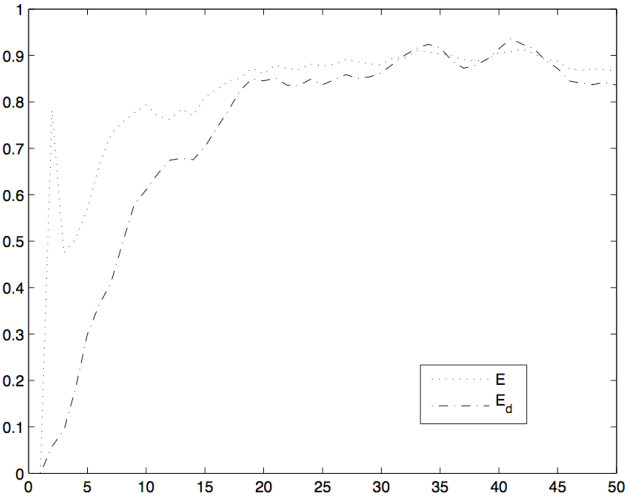


Figure 3.12: Comparison of Efficiency

3.3.7 Summary

The distance based complexity seems to handle different length populations better than the physical based complexity measure. It is better adapted to the way mutation changes the individuals. The negative effects of the change in population size for the distance based measure could not be found in the simulation because the population size was given by the requirements of the physical complexity measure. Although this population size was above the alphabet size not allowing the distance based complexity to become zero, the results of the physical complexity measure seem to be good. For almost fixed sequence lengths both complexity measures seem to behave similar.

Chapter 4

Measure for Multiple Populations

The initial goal was to find a similar measure to the complexity applicable for multiple populations. There are various aspects which have to be clarified before the construction of such a a measure is possible.

The first question is naturally what exactly should be measured. I discussed several ideas during the meetings with Dr. De Wilde and Briscoe. Some suggestions were quality of the services, similarity between different SME's, value of single services.

Another question is whether the measure is based on the whole populations or rather restricted to the best solutions returned by the evolutionary process. In discussions with Gerard, we decided to restrict it to the solutions because we are interested in the properties of these.

4.1 Quality Measure

In my opinion we cannot measure the quality of a service by just having the user request, the solution, the complexity and the efficiency of the evolved population and the fitness function. What could be measured from that, is how close the solution came to the initial request. From my point of view this has nothing to do with the quality of the generated service because we will ignore the quality of the request. Dr. De Wilde disagreed on that. He argued that the quality of the request can be incorporated into the fitness function in such a way that low quality requests will never achieve a high

fitness. I agree that simple dumb requests like *a weekend trip to London for 100 pound with a stay in the Ritz* for a travel agency habitat can get identified by a clever fitness function and get punished.

In my opinion though, there are subtler forms of dumb request which can get answered very well by the evolutionary algorithm but cannot get incorporated into the fitness function. A request like *staying in a hostel and renting a Rolls Royce* will probably not be the best selling holiday packet although it can be satisfied easily by a travel agency habitat, assumed both, hostel and Rolls Royce rental are services in the habitat. However, if the customers of the given travel agency are impostors it might nonetheless be a good selling offer. We can see that quality is not an objective measure. It is difficult to formalise and it may vary for different client groups.

Therefore for a quality measure we would need some kind of user feedback assessing this service. This could be done by the agency or the customers of the agency. It still would be difficult to compare these qualities because they are dependent on the customer base of the agencies.

4.2 Similarity of SME's

The goal of the DBE is that SMEs exchange their service chains and single services to stay competitive against the big companies. Without any control, this will lead to uniform SME's, all having access to every service. This might not be in the interest of the more successful SMEs, because they would lose their advantage over the other SME's. So some sort of control has to be introduced, allowing SME's to deny their services to other SMEs or increase the price. To judge how much a service is worth for another company, it is useful to know how similar the other SME is. If it is very similar it might be a competitor. The SME can now decide on the price of this service. It can even decide not to sell this service at all or to sell it at a later stage.

4.2.1 Solutions Distance Table

To find similarities between different SMEs, the distance measure which is already used to calculate complexities, is used again. This time we will use the pairwise distance rather than the pairwise complexity. The distance measure is now used on a set of solutions instead of an evolving population.

The similarity between different solutions is much less than the similarity of individuals in the same population. If the measure is used unadapted it will therefore return the length of the longer sequence in most of the cases because of the large alphabet and short sequence length of DBE services it is difficult to find a match in the sequences. This is unsatisfactory in most of the cases. Therefore a hierarchical clustering of the simple services is introduced. The services have to be described in BML anyway, so the creator can also define which cluster and sub clusters a single service belongs to. The top level identifiers could be the following categories: payment, transport, good or service. Then for each category there can be subcategories. For payment this could be credit card, debit card, pay-in slip, cheque or cash. If we introduce a function $cdepth(x, y)$ returning the depth of the lowest cluster in the clustering tree which contains single service (letter of alphabet) x and y and zero, if the two single services have no common cluster, we change the function w to:

$$w(x,y) = \begin{cases} \left(\frac{1}{2}\right)^{cdepth(x,y)} & x \neq y \\ 1 & x = - \\ 1 & y = - \\ 0 & x = y \end{cases}$$

This will lead to a distance table of the solutions. Which allows us to compare solutions of different SMEs.

One first measure that just operates on this table could be called market coverage. It shows how much of the market is covered by a given SME. It can be calculated as the mean of all distances of a given SME divided by the mean of the whole distance table. SMEs with low market coverage are rather specialised in one or two fields while those with high values are all-rounders. So the market coverage mc for SME A can be calculated as:

$$mc_A = \frac{m_A}{m_{all}}$$

With m_A as the mean distance in set A of solutions:

$$m_A = \frac{1}{(|A| - 1)\,|A|} \sum_{i \in A, j \in A} d_{ij}$$

And d_{ij} is the distance between solution i and solution j calculated with our adapted function w. For m_{all} we just use the union of all sets of solutions.

Now we construct a distance table for an example of three SMEs given in Figure 4.1:

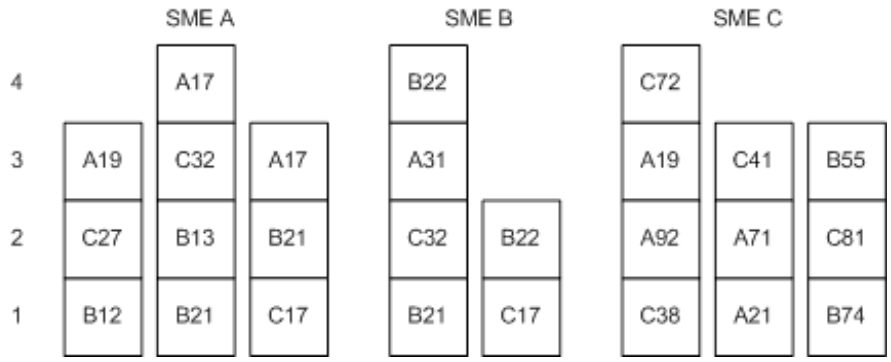


Figure 4.1: Solution Service-Chains of Three SMEs

Each single service is given by a letter and two digits. The first letter stands for the top level cluster this single service is assigned to. The first number stands for the second level clusters and the second number is the number of the service. With our w we will get the following values for the matchings:

x	y	$w(x,y)$
A17	A17	0
A17	A19	0.25
A17	A31	0.5
A17	B22	1

The distance between the different solutions gets calculated for each pair. The calculation is shown in detail for the first pair. For the remaining distances only the distance table is given.

D	B12	C27	A19
	0	1	2	3
B21	*1*	0.5	1.5	2.5
B13	2	*1.25*	1.5	2.5
C32	3	2.25	*1.75*	2.5
A17	4	3.25	2.75	*2*

Instead of a matching of distance four which would result using the w without clustering, this matching with hierarchical clustering returns a distance of two. The path leading to the shortest distance is emphasised. The matching looks as follows:

–	B12	C27	A19
B21	B13	C32	A17

If the distance calculation is done for every pair the solutions distance table will look as follows:

Distance	$A1$	$A2$	$A3$	$B1$	$B2$	$C1$	$C2$
$A2$	2						
$A3$	2.25	2.5					
$B1$	2.5	2.5	3.5				
$B2$	2.5	3.5	1.25	2.5			
$C1$	3	4	2.75	3.5	3.5		
$C2$	3	3.5	3	3.5	3	2.5	
$C3$	2	3	3	2.5	2	4	3

This leads to the following mean distances:

$$m_{all} = 2.85$$
$$m_A = 2.25$$
$$m_B = 2.5$$
$$m_C = 3.17$$

Now the market coverage measure can be calculated for each SME. This will return the following values:

$$mc_A = 0.79$$
$$mc_B = 0.88$$
$$mc_C = 1.11$$

Therefore, SME C has the largest market coverage and is the least specialised compared to A and B. This measure though, tells us nothing about the closeness of the three SMEs.

4.2.2 Distance of SMEs

To get a distance measure for SMEs, we further process the solutions distance table. It will most certainly not be possible to put the solutions on a plane reflecting all the measured distances, but to visualise the ideas behind the measure it is well acceptable to use a two-dimensional representation.

A first approach is to take the mean distance of services of SME A to services of SME B. For Figure 4.2 and the following approximate distance table this returns:

Figure 4.2: Distribution of Solutions, Well Separated

Solution	$A1$	$A2$	$A3$	$B1$	$B2$
$A2$	1				
$A3$	1	1			
$B1$	2	3	2		
$B2$	2	2	1	1	
$B3$	3	3	2	1	1

So the average distance between SME A and SME B is 2.22. This is an acceptable result. However, if we look at the special case where both SMEs have exactly the same solutions (see Figure 4.3) and the following distance table:

Solution	$A1$	$A2$	$A3$	$B1$	$B2$
$A2$	4				
$A3$	4	4			
$B1$	0	4	4		
$B2$	4	0	4	4	
$B3$	4	4	0	4	4

This will lead to a higher distance of 2.67 although both SMEs have exactly the same solutions. Therefore, the mean distance is not a suitable measure for SME distances.

We need to consider the mutual distances inside a set of solutions to be able to measure the distance of intermingled solutions. What we actually want to measure is the distance between the centre of mass of one solution set to another. The problem is to create the centre of mass. This can not be done because the mean is not defined on service chains. However, the mean distance to the centre of mass can be estimated by the mean distance in a set multiplied by a constant $c_m \in (0, 1)$. The centre of mass in is the point with the shortest mean distance to all the other solutions in a given set.

To specify the constant c_m we look at a minimal set of points in a given dimension spanning the whole space in the given dimension and having a distance between each point of exactly one.

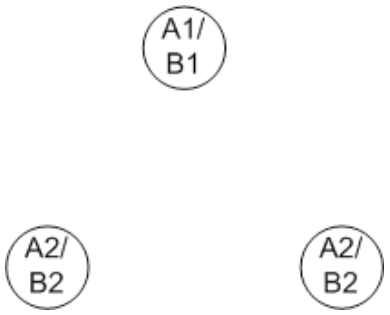


Figure 4.3: Distribution of Solutions, same solutions

Points	x	y	z	a	b	c	d
1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	0.5	0.866	0	0	0	0	0
4	0.5	0.289	0.816	0	0	0	0
5	0.5	0.289	0.204	0.791	0	0	0
6	0.5	0.289	0.204	0.158	0.775	0	0
7	0.5	0.289	0.204	0.158	0.129	0.764	0
8	0.5	0.289	0.204	0.158	0.129	0.109	0.756

Dimension	Mean Distance	Dist. To Centre (c_m)	Used Points
1	1	0.500	$\{1, 2\}$
2	1	0.577	$\{1, 2, 3\}$
3	1	0.612	$\{1, 2, 3, 4\}$
4	1	0.632	$\{1, 2, \ldots, 5\}$
5	1	0.646	$\{1, 2, \ldots, 6\}$
6	1	0.655	$\{1, 2, \ldots, 7\}$
7	1	0.662	$\{1, 2, \ldots, 8\}$

The dimensionality of the problem is approximately its sequence length. For variable length solutions an estimate will be made. Now we calculate the distance between the two sets of solutions A and B as follows:

$$D_{AB} = \frac{1}{|A||B|} \sum_{i \in A, j \in B} d_{ij} - c_m \left(\frac{|A| m_A}{|A| + |B|} + \frac{|B| m_B}{|A| + |B|} \right)$$

With m_A and m_B as defined before and d_{ij} as the distance between solution i and solution j.

Applied to the example given in Figure 4.2 we now get a distance of:

$$D_{AB} = \frac{20}{9} - 0.577 \left(\frac{3}{6}\ 1 + \frac{3}{6}\ 1 \right) = 1.645$$

and for the example in Figure 4.3 we get:

$$D_{AB} = \frac{24}{9} - 0.577 \left(\frac{3}{6}\ 4 + \frac{3}{6}\ 4 \right) = 0.259$$

We can now construct a distance table giving us the distance of two SMEs. SMEs can use this table to look up which other companies are relatively close and which operate in a far away market. It might be useful to divide these distances by the mean distance between all the companies. Now every company can define a level below which another company is treated a

competitor and define rules how prices for service exchanges should be raised.

Now we construct this SME distance table for the example for which we already calculated the solutions distance table before. The dimension of this example is estimated to be approximately three. Therefore, we use a value for c_m of 0.612. This will lead to the following distances (attention: from now on A, B and C refer back to Figure 4.1):

$$D_{AB} = \frac{15.75}{6} - 0.612\left(\frac{3}{5}\ 2.25 + \frac{2}{5}\ 2.5\right) = 1.187$$
$$D_{AC} = \frac{27.25}{9} - 0.612\left(\frac{3}{6}\ 2.25 + \frac{3}{6}\ 3.17\right) = 1.370$$
$$D_{BC} = \frac{18}{6} - 0.612\left(\frac{2}{5}\ 2.5 + \frac{3}{5}\ 3.17\right) = 1.224$$

These results allow us to compare the different SMEs. A and B are more similar to each other than to C. Therefore, B is more likely a competitor of A than C.

4.3 Clustering with Tree

The SME distance table is interesting for SMEs searching for their competitors. They can sort all other SMEs according to their mutual distance and find like this the closest competitors. However, with a growing number of SMEs the distance table becomes confusing and it will be difficult to extract a general overview. Therefore, it would be useful to have a clustering mechanism allowing to divide the SMEs in a given number of clusters.

In a first step a tree is built out of the distance table using the neighbour-joining algorithm described in the background chapter. Then it is defined in how many clusters the SMEs should get divided. For n clusters, the longest edge in the tree gets cut and the tree is split into two parts. The adjoint inner nodes to the deleted edge are deleted. Then we cut again the longest edge of the remaining two trees and build three trees. We repeat this procedure $n-1$ times building n trees. The connected trees now stand for a cluster.

Tree clustering algorithm:

1. for $i = 1; i < n; i++$
2. find longest edge e in T

3. delete edge e and inner nodes connected to e

4. end for

5. connected parts are returned as clusters.

If we apply this to the example of Figure 4.1 we get an unrooted tree with three leaves as shown in Figure 4.4. Although this tree does not look binary it still is binary. Just pick one edge and define the root along this edge. Now every inner node will have two child nodes and one parent node.

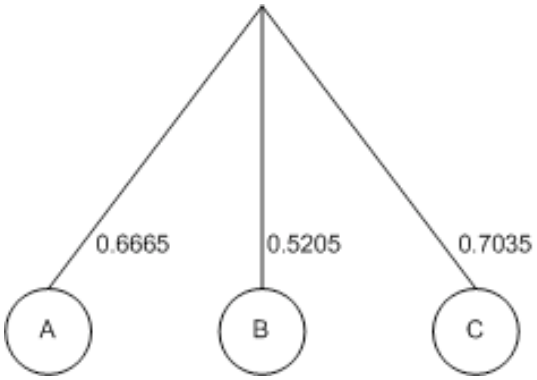


Figure 4.4: Distance Tree of SMEs A,B and C

This tree is too small to illustrate clustering algorithms on it. The only clusters apart from single SME clusters that this tree can generate is A and B together in one cluster and C in its own cluster. Therefore, a larger tree will be introduced to show the clustering behaviour (see Figure 4.5).

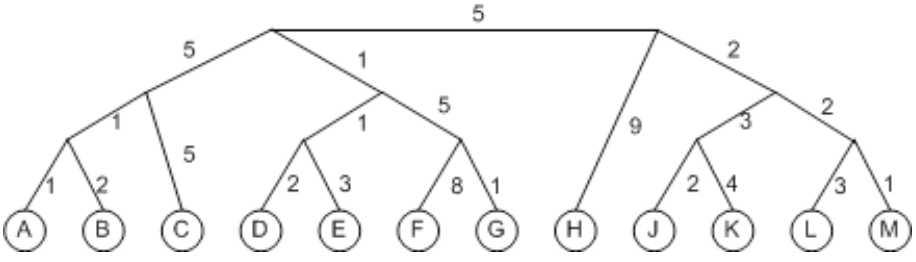


Figure 4.5: Distance Tree of 12 SMEs

To build four clusters we apply the algorithm three times to this tree. The longest edge has length nine and connects SME H to an inner node. This edge gets deleted and the adjoint edges get joined together leading to Figure 4.6.

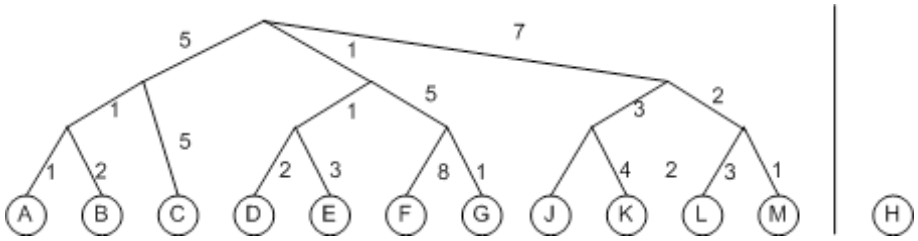


Figure 4.6: Division into Two Clusters

Now the connection between F and an inner node is the longest edge with a value of 8. So again this edge will be deleted leading to the clustering in Figure 4.7.

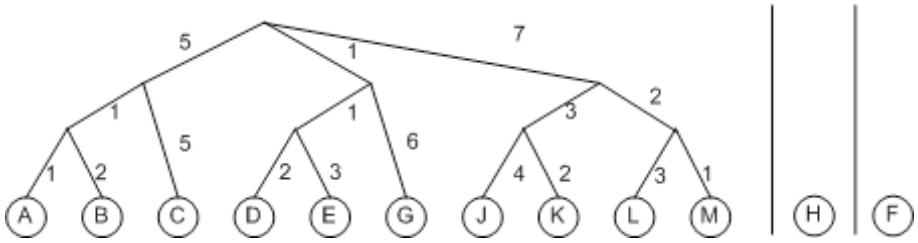


Figure 4.7: Division into Three Clusters

In the last step the inner edge of length seven is the longest one. Therefore, this edge is deleted which leads to the desired four clusters in Figure 4.8.

If it is not desirable to create single SME clusters out of outlier SMEs, restrictions can be imposed on the line deleting process. If a minimum cluster size of two is required we just allow the algorithm to delete inner edges. For our example this leads to the four clusters given in Figure 4.9.

If a more balanced clustering is preferred, the distance could be weighted according to the size of the two clusters generated by the cut. Given the

cluster T as an unrooted three with given edge lengths e_{ij}, we calculate the weighted edge length ew_{ij} as follows:

$$ew_{ij} = |T_1||T_2|e_{ij}$$

With T_1 and T_2 as the clusters generated by deleting edge e_{ij} and $|T_i|$ as the number of leaves in cluster T_i. This favours the further division of larger clusters and inside clusters a balanced division is preferred. For the given example this leads to the clustering given in Figure 4.10.

Which clustering is used by the user is heavily dependent on what he requires. If he is interested in companies which are rather outliers, it is well acceptable to build clusters of single SMEs identifying these companies as outliers. If

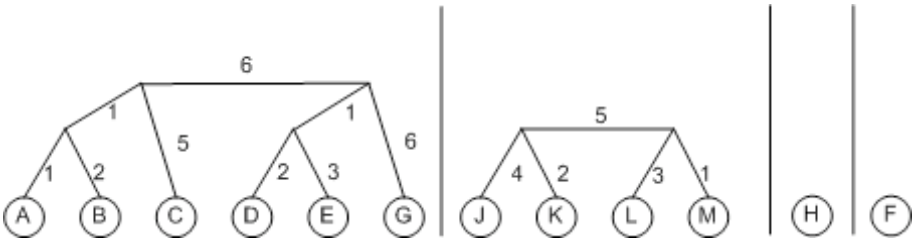


Figure 4.8: Division into Four Clusters

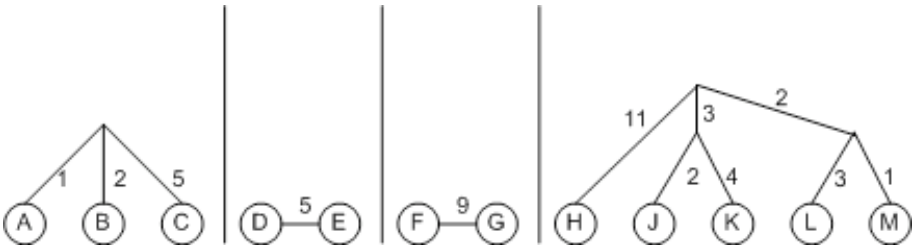


Figure 4.9: Division into Four Clusters without Single SMEs Clusters

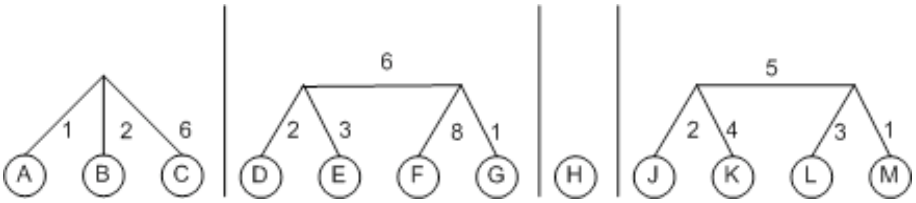


Figure 4.10: Division into Four Clusters with Weighted Clustering

he rather wants a general overview, he might not be pleased with one big cluster containing almost all SMEs and many small clusters with only one or two SMEs inside.

4.4 Computing Time

The computing time to build the solutions distance table is enormous. It has the same time complexity as the distance based complexity measure of $O(n^2l^2)$. With n as the number of solutions and l as the average length. This time we apply it not only to one population. We use it on all solutions off all SMEs. This will be soon much bigger than the normal population size. However, this should not be such a problem because the solutions distance table can be built by successively adding the new solutions. It is also easily parallelisable, letting each habitat calculate the distances of its solutions to the newly generated solution. Nonetheless there should be a deletion mechanism deleting solutions after a certain time period to prevent the table to grow forever. SMEs should also be able to delete their unused solutions, because these are biasing the measure.

The time needed to build the SMEs distance table and for the clustering is much smaller than the time needed to build the solutions distance table. So the overall computing time for the suggested measures for multiple populations is in the order of $O(n^2l^2)$.

Chapter 5

Simulation

The original simulation by Briscoe was restricted to a single population in a single habitat. It was used for simulating the evolution of one population and keeping track of its complexity. The desired population size μ is set to the 1.1 times request length times alphabet size. The number of offspring λ generated is adaptive, depending on how close the current population size is to the desired μ. The environmental selection scheme used is the selection on both, parent and child generation. The mutation probability p_m, the insertion probability p_{ins}, the deletion probability p_{del} and the crossover probability p_c are all adaptive. The mutation operator fulfils the two properties given in the background chapter. The crossover operator also adheres to the rule from the background chapter. The fitness function receives a request given as a service vector and returns values between 0% and 100%. The service chains are represented as sequences of single services.

In a first phase, multi habitat support was added and each habitat was allowed to have multiple populations. This was achieved by creating a new main class `DBE`, coordinating all habitats and issuing requests to habitats. The classes `Habitat` and `Population` were changed to allow the evolution of a population to take place in its own thread, thus running quasi parallel.

The service pool of each habitat is generated randomly and the representation was changed back to vectors of integers from the textual description of travel services which restricted the original simulation to a few user chosen single services. Minor changes in almost all the classes have been made to change the simulation back to vector representation, which is more flexible and easily extendable. Further, a new function generating random requests with a given length out of a given service pool was added.

Additionally a class to measure mutational distance `DistanceMeasure` and another one for measuring the distance based complexity and efficiency `CEMeasure` were added. These two classes implement the mutational distance, the distance based complexity and efficiency described in chapter 3.

In its current implementation the simulation takes the following arguments: number of habitats, populations per habitat, number of single services in a habitat, request length and the number of generations until the algorithm terminates. The population is initialised with all possible services of length one. The output for each generation is the generation number, the population size, the maximum fitness, the average fitness, the maximum complexity of the physical measure (l_v), the physical complexity for variable length, the physical efficiency, the distance based complexity, the distance based efficiency and the distance based maximum complexity. Other output schemes are available in the source code but have to be uncommented before the use of the simulation.

The source code for the simulation can be found on `http://www.doc.ic.ac.uk/~fh104/`.

Chapter 6

Conclusion

6.1 Complexity

The distance based complexity has to prove its value in a real DBE environment. It has been shown that it is superior in a varying length population. So the additional overhead for computing the distance based complexity can only be justified if the evolutionary environment of the DBE produces different length populations. In the current state it is not definite if there will be a varying length population or if it will be almost fixed, like in the current simulation. The trend towards an almost fixed or a varying length population is highly dependent on the fitness function and the request.

As long as the request is given as a sequence of desired single services of a fixed length, I expect the population length to converge towards a fixed length as it happens in the current simulation. In such an environment the variable length physical complexity measure should be used to save computing power.

If the fitness function is multi modal and tries to identify the Pareto front rather than the optimum, it is more likely that the population will contain individuals with considerably different lengths. Here the smooth behaviour of the distance based complexity measure justifies the higher computing cost.

6.2 Multiple Populations

The measures suggested satisfy just part of what the desired measure should have been representing. To further improve these measures we need more information like user feedback, SMEs customer base, location and size.

The suggested measures for multiple populations have not yet been tested on a simulation due to the lack of a good request model. The results of the evolutionary process come very close to the request. As long as the requests are just random requests, we will not measure anything useful. It is expected that with random requests all simulated SMEs will have a comparable distance to each other. To run tests a model for a request generation is needed. This model should incorporate the business field each SME is working in.

Later these measures also have to prove to be useful in the evolutionary environment of a running DBE. Because this measure is mainly concerned to help SMEs, the feedback of SMEs concerning these measures is important.

6.3 Future Work

The simulation should be extended in various ways. The first priority should be given to the implementation of a clever request generator producing SME dependent requests. This will allow the testing of the measures for multiple populations.

Further an adaptive exchange mechanism between the different habitats has to be introduced and its influence on the evolutionary environment has to be measured.

The fitness function should get changed completely, freeing it from the fixed length request and allowing populations with different lengths to evolve.

For the improvement of the multiple populations measure feedback mechanisms have to be introduced to judge the quality of service chains.

Bibliography

[1] C Adami. What is complexity? *BioEssays*, 24:1085–1094, 2002.

[2] C Adami, C Ofria, and T Collier. Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97(9):4463–4468, 2000.

[3] G Brisoce. Deliverable 6.1, self-organisation in multi-agent systems, entropy-based complexity measure for the evolution-based self-organisation of agent populations, 2004.

[4] G Brisoce, J Rowe, and P Dini. Evolutionary environment discussion paper, 2004.

[5] P. Dini and A. Nicolai. D.b.e. - the digital business ecosystem, 2003.

[6] N. G. Cooper (Editor). *From Cardinals to Chaos, Reflections on the Life and Legacy of Stanislaw Ulam*. Cambridge University Press, 1990.

[7] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann, 1993.

[8] P von Rohr G. Gonnet and G. Cannarozzi. Slides of computational biologie. Available from: `http://www.inf.ethz.ch/personal/pvrohr/Courses/CompBio/2002/week2/week2.slides.pdf` [cited 12/02/05].

[9] D. Gaertner. Natural algorithms for optimisation problems. Available from: `http://www.doc.ic.ac.uk/teaching/projects/Distinguished04/DorianGaertner.pdf` [cited 15/05/05].

[10] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.

[11] G. Gonnet and J. M. Baumann. Lecture notes and slides of computational science lecture. Available from: `http://www.inf.ethz.ch/personal/vroth/wrk/index.html` [cited 11/03/05].

[12] E. Zitzler P. Koumoutsakos and N. Hansen. Lecture notes and slides of bioinspired computation. Available from: `http://www.icos.ethz.ch/cse/education/bioinspired_computation/` [cited 15/06/04].

[13] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[14] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.