Digital Business Ecosystem

Contract n^{o} 507953

Workpackage 6

Self organisation

Deliverable 6.2

Control of Self-organisation and a Performance Measure

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

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

Deliverable No: D6.2
Due date: 30/04/2005
Delivery date: 30/04/2005

Short Description:
This report investigates the self-organising behaviour of applying evolutionary computing to Multi-Agent Systems, in the model of the DBE Evolutionary Environment. Both evolutionary computing and Multi-Agent Systems are mature areas, but the application of one to the other is non-trivial because the agents are state machines, and evolutionary computing algorithms have been developed to work on numerical data. Our aim is to extend the application of evolutionary computing to Multi-Agent Systems, rather than contribute towards it directly.

Overall an insight has been achieved into where and how self-organisation occurs in the DBE, and how it can be quantified.

Partners owning: ICL
Partners contributing: ICL
Made available to: Consortium and EC

Versioning		
Version	**Date**	**Author, organisation**
1	30/04/05	Gerard Briscoe, ICL
1	30/04/05	Philippe De Wilde, ICL

Quality check
1st Internal Reviewer Paolo Dini (LSE)
2nd Internal Reviewer Jon Rowe (UBHAM)

Self-organisation of Evolving Agent Populations

Gerard Briscoe
Philippe De Wilde

Intelligent Systems and Networks Group
Department of Electrical and Electronic Engineering
Imperial College London

April 2005

Acronyms

BGS	Best Guess Solution
BPEL	Business Process Execution Language
BML	Business Modelling Language
DBE	Digital Business Ecosystem
DEC	Distributed Evolutionary Computing
DIS	Distributed Intelligence System
EvE	Evolutionary Environment
ICL	Imperial College London
ISUFI	Istituto Superiore Universitario di Formazione Interdisciplinare
LSE	London School of Economics
MAS	Multi-Agent System
MoAS	Mobile Agent System
SM	Service Manifest
SME	Small & Medium Enterprise
SOLUTA	SOLUTA.NET
SOM	Self-Organising Map
STU	Salzburg Technical University
SUN	Sun Microsystems
UBHAM	University of Birmingham
VLP	variable length population

Contents

Executive Summary

Regarding the objective 'to provide the adaptation mechanism inside the DBE' from the task S6 description of the technical annex[14], the model for the Evolutionary Environment (EvE) Digital Ecosystem is presented. It provides a framework for the Evolutionary Environment, which helps facilitate integrated work between the partners involved with the EvE. Additional technical information on the EvE is provided in Annex B.

Regarding the objective 'to provide a performance measure for the self-organisation of the services into an application'[14], the Efficiency performance measure is presented. The potential use of the Efficiency E performance measure, within the EvE, is shown for evolving populations instantiated in response to the user requests. This included the development of an EvE Demonstrator in which the abstract numerical Business Modelling Language (BML) was given meaning. It shows the Efficiency performance measure providing the probability that a currently chosen solution is optimal. This has potential value for the EvE as a supplementary control mechanism and a user information mechanism.

Regarding the objective 'to find how to control the self-organisation'[14], we attempted to control (guide) the evolutionary self-organisation by applying a form of selective breeding. This involved encouraging crossover within clusters of the evolving population, to accelerate the evolutionary process by reducing the number of generations required to evolve the optimal solution. It proved to be unsuccessful, and consideration was given to why it had failed, considering that it intuitively makes sense and is used in the human breeding of plants and animals.

After the lack of success with the clustering mechanism it was decided that a better understanding of the EvE was required. So a quantitative definition of self-organisation for any Multi-Agent System (MAS) was constructed, which is compatible with the qualitative self-organisation definition of the new DBE Science Vision[15] and the evolutionary theory presented in deliverable D8.1[33]. It was tested successfully for a MAS with evolutionary self-organisation (i.e. the EvE), and is not limited to the chain structure of aggregated agents (services). A better understanding of the EvE system was achieved, which will be valuable for the future research of the Distributed Intelligence System (DIS) that will operate within the EvE.

List of Figures

1 Introduction

This report investigates the self-organising behaviour of applying evolutionary computing to Multi-Agent Systems (MASs), in the agent based model of the DBE Evolutionary Environment (EvE). Both evolutionary computing and MASs are mature areas, but the application of one to the other is non-trivial because the agents are state machines, and evolutionary computing algorithms have been developed to work on numerical data. Our aim is to extend the application of evolutionary computing to MASs and then determine macroscopic variables to characterise the self-organisation within evolving agent populations, rather than contribute directly to the evolutionary computing theory presented in deliverable D8.1[33] 'Report on Evolution of High-Level Software Components'. The relevance of this work increases as the number of agents (services) within the DBE increases. However, we expect some practical relevance, and most certainly some research relevance, also during the bootstrap phase when the number of services will not be statistically significant.

Alternative optimisation methods have been examined in deliverable D8.1[33]. Since the exact nature of the DBE as an optimisation problem is still under development, an abstraction of the problem was put forward, based on the set cover problem. The following algorithms evaluated were: genetic algorithms, simulated annealing, steepest descent, tabu search, and random search. A random problem-generator was developed, and the candidate algorithms were tried on a large sample of problems. The results showed that genetic algorithms were the most effective algorithm, and were presented as a *prima facie* case for the use of evolutionary techniques in addressing the DBE problem[33].

Creating a Digital Ecosystem capable of the self-organising complex behaviour of a biological ecosystem is the ultimate aim of the Evolutionary Environment (EvE). Self-organisation is an emergent global 'behaviour' of a complex system, which consists of many autonomous components with local interaction rules. Over time a biological ecosystem becomes increasingly more complex, driven by the evolution of the populations within the ecosystem. In our Digital Ecosystem the components are the agents which interact in evolving populations, forming aggregated agent structures, to provide optimal solutions to user requests at the global level. The Digital Ecosystem's increasing complexity comes from these evolving populations of agents.

The final model for the EvE is presented in **Section 2**. The EvE Digital Ecosystem model is based on a Mobile Agent System (MoAS) with Distributed Evolutionary Computing (DEC) as an optimisation technique. An evolutionary optimisation process, at each Agent Station, will find the optimal combination of agents (DBE services) to meet user defined requests[8]. Deliverable D6.1 provided a way to accurately measure the organisational complexity within an evolving population of the EvE Digital Ecosystem. An effective Efficiency E performance measure was constructed, from the organisational complexity, which describes the efficiency of information storage within an evolving population. In **Section 3**, an EvE Demonstrator was created to show the potential use of the organisational complexity and the Efficiency E in measuring the performance of evolving populations.

The Efficiency E was also capable of detecting clustering and estimating the number of clusters, making it potentially useful for the form of selective breeding envisaged to

control (guide) the evolutionary self-organisation. In **Section 4**, we applied a clustering mechanism, as a form of selective breeding, to encourage crossover within clusters of the evolving population, where clusters are sub-populations of similar individuals. Possible clustering techniques were discussed, and a form of hierarchical clustering was implemented to test the effectiveness of the clustering mechanism. It was hoped that the clustering mechanism would accelerate the evolutionary process, reducing the number of generations required to evolve the optimal solution, but it ultimately proved to be ineffective.

In **Section 5**, after the lack of success with the clustering mechanism, it was decided that a better understanding of the EvE was required. A definition of self-organisation based on state transition probabilities was constructed, which is compatible with the definition of self-organisation in the new DBE Science Vision[15] and the general evolutionary theory presented in deliverable D8.1[33]. It was tested successfully for a Multi-Agent System (MAS) with evolutionary self-organisation (i.e. the EvE), and can represent any structure of aggregated agents (services).

The conclusions are presented and discussed in **Section 6**. This includes the value of the contributions both scientifically and to the DBE. Consideration is also given to the relation of this work to the other activities within the project.

Appendix A is an Interdisciplinary Dictionary of the key terms used within this document to facilitate communication within the consortium. **Appendix B** contains additional technical information about the EvE from the Evolutionary Environment Architecture Requirements document[8].

2 Digital Ecosystem Definition

At the start of the project there was no Evolutionary Environment to speak of, there was only a concept of using evolutionary computing as an optimisation technique to find the optimal combinations of services to a user request. Even at the time of writing deliverable D6.1[7] (month 5) the structure of the EvE was unclear. A more mature vision of the EvE, although still incomplete, was presented as part of deliverable D21.2 'Architecture Scope Document'[17].

The Digital Ecosystem model is now complete, so the structure of the EvE is now clear. The creation of the EvE Digital Ecosystem model, which started with the Evolutionary Environment Discussion Paper[9], has been achieved through the interactions of ICL, STU, UBHAM, SUN, SOLUTA and LSE.

The key concepts and models upon which our Digital Ecosystem is defined will be brought together to create a hybrid model, so that we can clearly define an evolving population in our Digital Ecosystem.

2.1 Ecosystem

An ecosystem is a natural unit made up of living and non-living components whose interactions make a stable, self-perpetuating system. It is made up of one or more habitats and communities of organisms, consisting of populations existing in their microhabitats. A community is a naturally occurring group of populations from different species that live together, and interacts as a self-contained unit in the same habitat. A habitat is a part of the environment, for example, a stream. A population is the total number of the individuals of a particular species, inhabiting a specific habitat or microhabitat. A microhabitat is a subdivision of a habitat that possesses its own unique properties, such as a microclimate.[29]

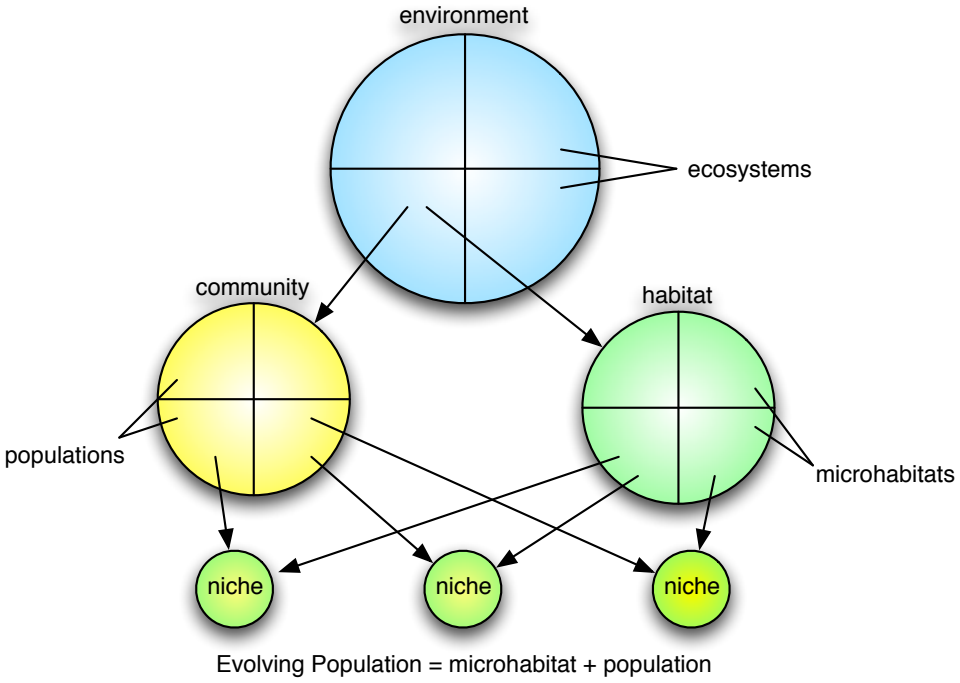


Figure 1: Abstract Ecosystem Definition

Therefore, an evolving population is a population in its microhabitat. A population comes to occupy a niche, which is the functional relationship of a population to the environment that it occupies[6]. A niche generally refers to a highly specialised adaptation of a population to its microhabitat.

2.2 Evolution

Evolution is the process that governs the adaptation of populations to their environment, involving changes to the genetic makeup of these populations. These genetic changes are passed on through successive generations[29, 21]. A population adapts over several generations through a cyclic process of mutation, replication, and death.

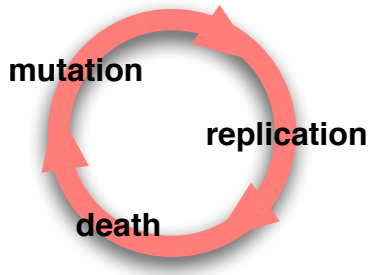


Figure 2: Cyclic Process of Evolution

Genetic change is introduced through the mutation and replication of individuals within a population. A 'selection pressure' is defined by the environment and is the sum total of the forces acting upon a population, which results in genetic change and natural selection. Those individuals best fitted to survive the selection pressure operating upon them, will pass on their biological fitness to their progeny through the replication process.[29, 21]

2.3 Evolutionary Computing

The subject of evolutionary computation is discussed extensively in deliverable D8.1 'Report on Evolution of High-Level Software Components'[33], but we will briefly introduce the key concepts necessary to construct our Digital Ecosystem. Evolutionary Computing is an optimisation technique capable of finding solutions in a large search space of possible solutions. It works on the principle of using evolution to generate a solution to a specific problem. The cyclic process, of mutation, replication and death, is applied to a population of possible solutions. The 'selection pressure' comes from a **fitness function** which is instantiated with the specific problem for which a solution is to be found. The fitness function determines the fitness of the different solutions relative to the environment, and therefore their replication rate. So the better solutions get to replicate more, potentially providing improved solutions in the next generation.

2.4 Distributed Evolutionary Computing

The subject of distributed evolutionary computation is discussed extensively as well as simulated using the NetEvo program in deliverable D11.1 'The flow of software

components in a static network'[32], but we will briefly present the key concepts necessary to construct our Digital Ecosystem. Distributed Evolutionary Computing aims to achieve parallel processing to find solutions more efficiently. We will focus on the 'Island Model' which is probably the most analogous to the notion of migration in nature[20]. There is a distance between the sub-populations on each island, and a probability of migration between one island and another.

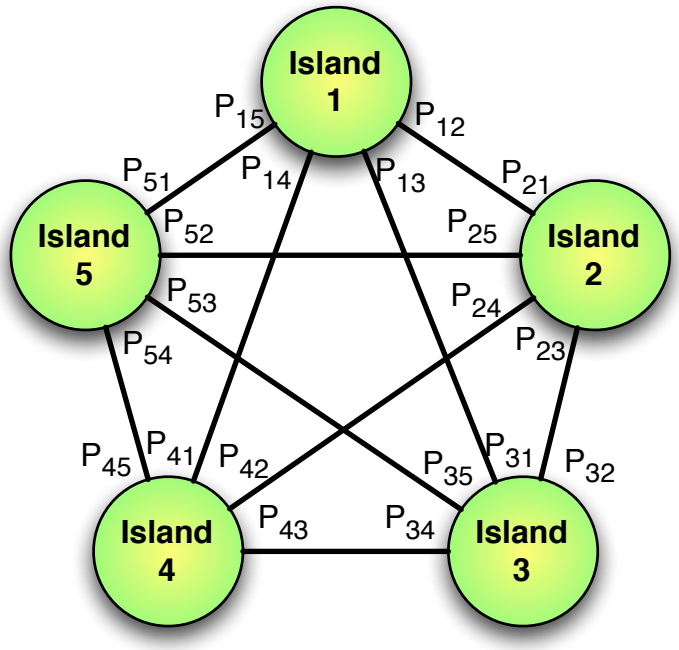


Figure 3: Island Model of Distributed Evolutionary Computing

In Figure 3, there are different probabilities of going from island '1' to island '2' as there is of going from island '2' to island '1'. This allows maximum flexibility for the migration process. It also mirrors the naturally inspired quality that, although two populations have the same separation no matter how you look at them, it may be easier for one population to migrate to another than vice-versa. An example of this from nature could be the situation whereby it is easier for one species of fish to migrate down-stream than for a different species to migrate upstream.

This model has been used successfully in the determination of investment strategies in the commercial sector, in a product known as the Galapagos toolkit.[41]

2.5 Mobile Agent System

A mobile agent is a program that can migrate from one host to another in a network of heterogeneous computer systems and perform a task specified by its owner. The agents self-initiate migration, work autonomously and communicate with other agents and host systems. On each host they visit, mobile agents need a special software called an Agent Station, which is responsible for executing the agents, providing a safe execution environment, and offers several services for the agents that reside on the host. A Mobile Agent System is the union of all the Agent Stations, with agents running on these Agent Stations being part of agent-based applications.

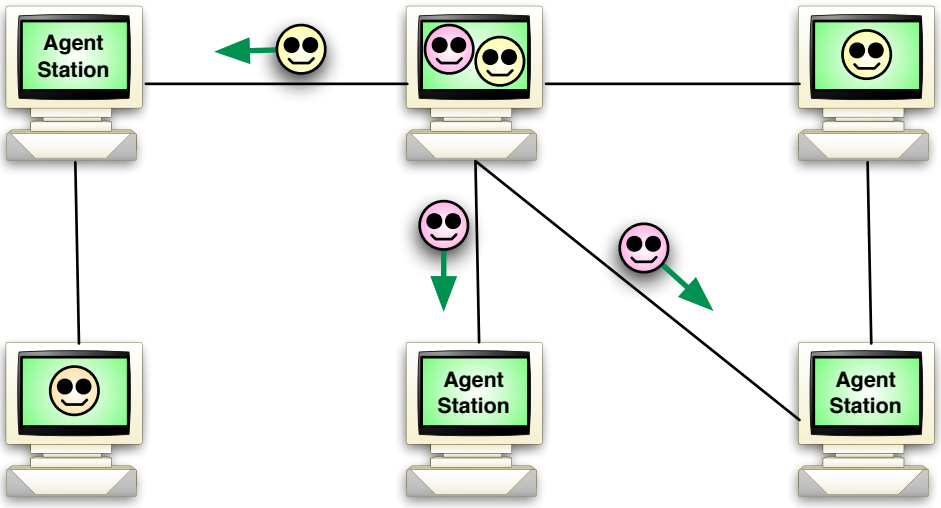


Figure 4: Mobile Agent System

Migration of agents is based on an infrastructure that has to provide the necessary services in the network. The infrastructure is a set of Agent Stations that run on platforms (nodes) within a possibly heterogeneous network. Each Agent Station hides the vendor specific aspects of its host platform and offers standardised services to agents visiting such an Agent Station. Services include access to local resources and applications, for example, Web servers or Web services, the local exchange of information between agents via message passing, basic security services, creation of new agents, etc.

2.6 Agents and Aggregated Agents

I have purposely left defining the agents until the last as it is difficult to find any definition for an agent that includes all the things that most researchers and developers consider agents to be, and excludes all the things they aren't. However, a generally accepted abstract definition is a computer system that is situated in some environment, and that is capable of autonomous action in this environment to meet pre-defined objectives[43].

Generally, the definition of an agent depends on the context in which it is used. So it can be more accurately defined in the context of our Digital Ecosystem, in which the agent is the base unit for the evolutionary mechanism. An evolutionary process will optimise the combination of available agents to respond to a user request, but will not change (mutate) the agents themselves. A question from the very beginning, even before the project started, was the structure of combined or aggregated agents (services). Initially, 'chain' and 'tree' structures where contemplated, since then 'sets' and 'workflows' have been proposed.

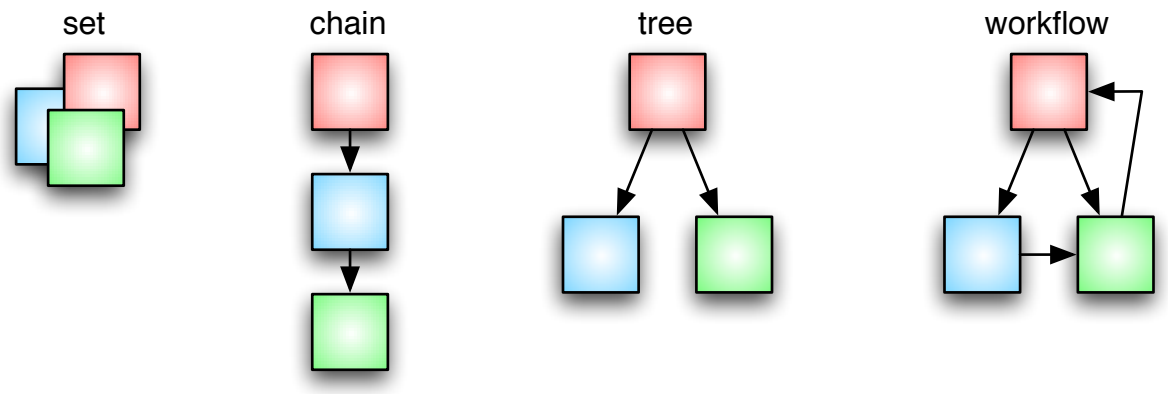


Figure 5: Possible Structures of Aggregated Agents

The 'workflow', which is the most sophisticated and can represent the other structures, has not only been proposed but implemented by the computing stream using the Business Process Execution Language (BPEL). The decision of what aggregated structure to use in the EvE has not been finalised[8]. Discussions are continuing regarding the most suitable structure, including the UBHAM's theoretical work on 'trees' in deliverable D8.1[33] 'Report on Evolution of High-Level Software Components'.

The current thinking does envisage using 'chains' at some stage[8], which is potentially a first step towards genetic programming. We will use the term agent-chain to represent agents aggregated into a 'chain' structure, and we will continue to use agent-chains as the aggregated structure of agents, as in deliverable D6.1[7]

2.7 Digital Ecosystem Model

The Digital Ecosystem model contains elements from MoASs, DEC and ecosystem theory. The Digital Ecosystem will consist of interconnected habitats just as in a biological ecosystem. Each Habitat will provide facilities similar to an Agent Station, including an 'Agent Pool' similar to a 'gene pool' (please see Appendix A). The set of agents registered at the Habitat is used as the 'gene pool' for Evolving Populations. An Evolving Population, similar to an 'Island' in the DEC Island Model, will be created to use Evolutionary Computing and evolve the fittest (optimal) solution(s) to a request defined by the user. The agents can migrate through the interconnected Habitats combining with one another in the Evolving Populations to meet user requests.

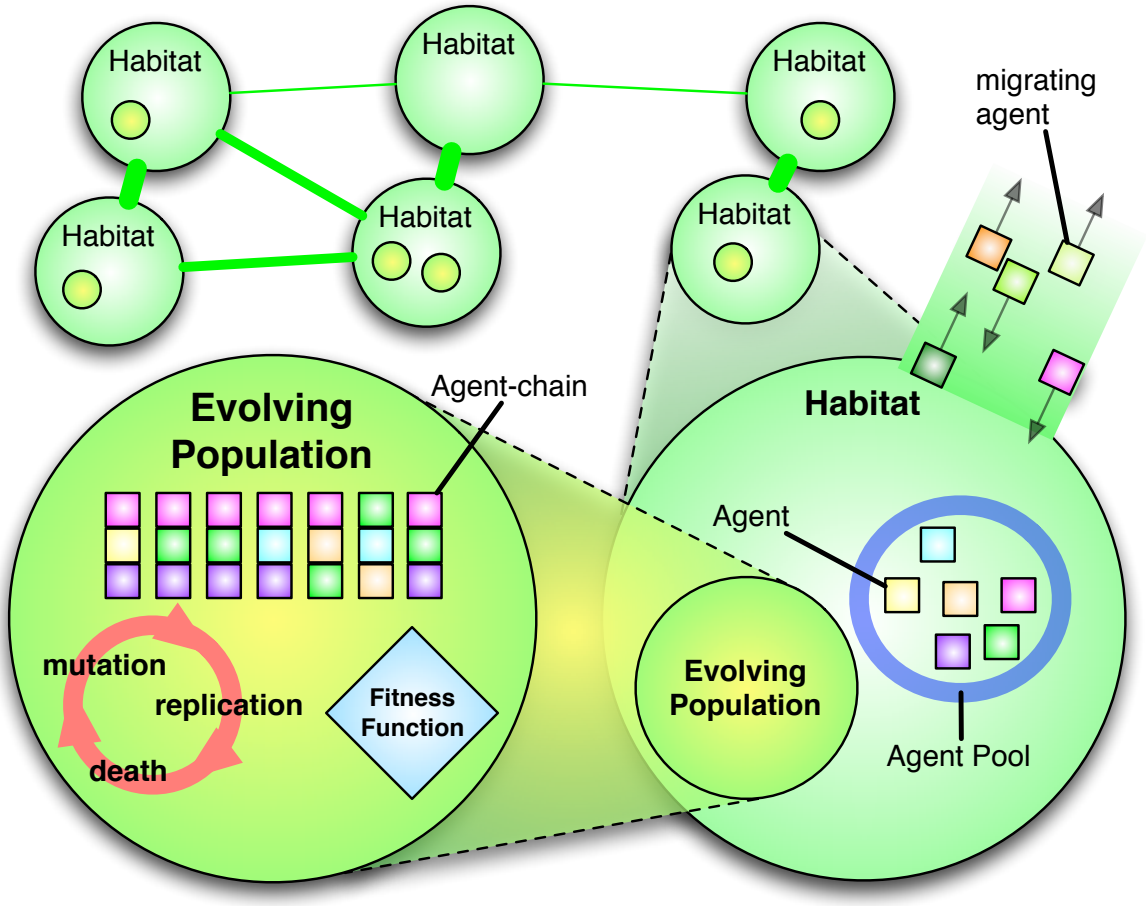


Figure 6: Digital Ecosystem Model

The agents in the Digital Ecosystem are light-weight entities consisting of a semantic description of their functionality and a reference (local or remote) to their executable component. The description contained within each agent acts as a guarantee of its functionality, and is the inheritable component from one generation to the next. There is no constraint on the language used for the executable components, but there should be compatible interfacing so that agents can aggregate to perform more complex tasks. The fitness will be based on comparing the descriptive components of the agents with the complex description of the user request. So the aggregated agents have to be comparable to the user request. Therefore, the semantic descriptions in both are written in the same language.

Each Evolving Population of agent-chains will search the agent chain combination space through evolution to find the optimal solution(s) to a user request. The fitness of individual agent-chains within a population will be determined by a selection pressure applied as a fitness function instantiated from the user request, and works primarily on comparing the semantic descriptions of the agent-chains with the semantic description in the user request. Mutations can occur by switching agents in and out of the agent-chain structure. Recombination (Crossover) can occur by combining elements of two agent-chains into a new agent-chain.

2.8 Summary

The EvE Digital Ecosystem model, the platform upon which this research is to be conducted, has now been finalised. It provides a digital ecosystem for the project, where the word ecosystem is more than just a metaphor. Additional technical information is available in Appendix B.

The EvE Digital Ecosystem is based on much of the scientific research, will require most of the computing infrastructure, and the Habitat clustering represents the business opportunity spaces. Therefore, it has potential to facilitate integrated work within the project.

Now that the Digital Ecosystem model has been defined, the question of a performance measure for the self-organisation of an evolving population of agent-chains can be addressed.

3 Efficiency Performance Measure

The model of the EvE Digital Ecosystem has now been finalised. So the application of the Efficiency performance measure, presented in deliverable D6.1 'Self-Organistaion in Multi-Agent Systems'[7], can be completed. The Efficiency performance measure was constructed from the definition of organisational complexity developed for an evolving population, which itself was constructed in deliverable D6.1[7] by extending Physical complexity to Multi-Agent Systems and variable length populations. The **Efficiency** E **performance measure** describes the efficiency of information storage within an evolving population of the EvE. A version of the simulation called the EvE Demonstrator was created to show its applicability. This included 'translating' an abstract numerical BML into more meaningful textual descriptions based around the tourism industry, which is one of the opportunity spaces (business sectors) that the project is targeting[28, 31].

3.1 Efficiency Definition

The definition of Efficiency from deliverable D6.1[7] is summarised here for clarity, and the definition of organisational complexity which it is based on, is summarised from deliverable D6.1[7] in Appendix C. A **performance measure** was constructed which shows the use of the information space by using the organisational complexity measure of an evolving population, i.e., the organisational complexity relative to the maximum organisational complexity. So the efficiency E for a variable length population is,

$$E = \frac{C_V}{C_{V_P}}, \tag{1}$$

the actual complexity C_V over the complexity potential C_{V_P}, which ranges between zero and one:

$$0 \leq E \leq 1 \tag{2}$$

The Efficiency is maximum, $E = 1$, when there is no randomness in the population, so the actual complexity C_V equals the complexity potential C_{V_P}. Two agent-chain populations are presented in Figure 7 with their organisational complexity values and their respective efficiencies.

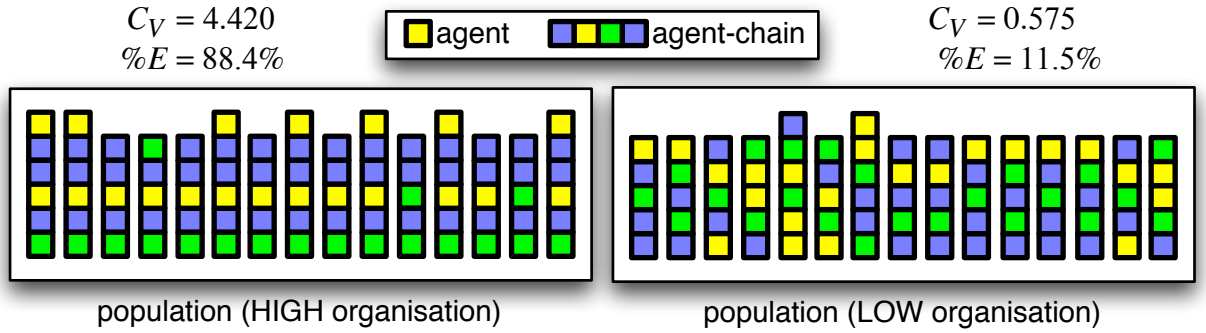


Figure 7: Efficiency in Populations of agent-chains

The efficiency of the populations in Figure 7 is as expected. It shows the population with high organisational complexity is efficient, but that there is still some randomness

in the population. For the population with low organisational complexity, it shows that the population is almost entirely random. The performance measure is concise, succinct, and matches ones intuition.

3.2 EvE Demonstrator - BML Semantic Descriptions

Each agent of the simulation contains a 'business process' object to represent the agent's semantic description. It provides an abstract representation as a list of integer tuples. Each tuple represents an attribute of the semantic description, one integer for the attribute identifier and one for the attribute value. The attribute identifiers range in value between one and ten, and in quantity, for each process, between three and five. The values of the attributes range between one and a hundred. A sample is shown below in Figure 8.

$$business\ process = [(1, 40), (5, 67), (3, 88)]$$

Figure 8: Agent's Abstract Semantic Description

The user requests are handled by the Habitat instantiating an evolving population, which uses evolutionary computing to find the optimal solution(s). The request consists of a list of business processes. A sample is shown below in Figure 9.

$$user\ request = [[(3, 91), (4, 15), (2, 9)], [(8, 57), (6, 45), (3, 88)], [(1, 40), (5, 67), (7, 15)]]$$

Figure 9: User Request - List of Business Processes

This numerical representation is similar to the simplified service definition for the fitness landscape simulator developed by STU, which is presented in deliverable D9.1 'Report on Fitness Landscape'[38]. However, it is more sophisticated as each service has multiple attributes. Our abstract numerical definition makes it widely applicable, but feedback suggested that it could be clearer. So a version of the simulation was constructed showing the agents and the user request in the context of the travel industry, which is one of the opportunity spaces (business sectors) that the project is targeting[28, 31]. Basically, the numerical semantic descriptions were given meaning that would be compliant with the BML metamodel[12] as defined by ISUFI, compliant in the sense that the information (data) could be represented in a BML model. The simulation still works on the numerical representation for operational efficiency, but a filter was created that assigns meaning to the numbers. In one of the project meetings a business partner explained the basic properties of any business process are cost, quality and time; so this was followed in the simulation. A sample is shown below in Figure 10.

agent's *abstract* semantic description = (1,25) , (2,35) , (3,55) , (4,6) , (5,37)

agent's semantic description =
Business = Airline
Company = Luthansa
quality = economy
cost = 60 per person
London to Rome

Figure 10: Agent's Semantic Description

The output from the semantic filter in Figure 10 not only shows that the numerical semantic descriptions are a reasonable modelling assumption, but can now also show the numerical semantic descriptions in a human readable form. The filter acts as a type of translation matrix, so theoretically it can be altered without too much difficulty to show output for applications from other opportunity spaces (business sectors).

3.3 EvE Demonstrator - Efficiency Scaling

It is desirable that optimally self-organised service chains should also exhibit a high fitness[14], so the raw Efficiency E has been scaled by the average fitness for the Efficiency performance measure,

$$\text{performance measure} = E \times F_{avg}, \tag{3}$$

where F_{avg} is the average fitness. So the performance measure will only be high when both the fitness and Efficiency are high.

3.4 EvE Demonstrator - Sample Output

The scenario envisaged from the travel industry opportunity space was for a travel agent SME looking for a package holiday, compatible with the BML example in the ISUFI document 'Very simple BML'[13]. So the user places a request, which would look like Figure 11.

User Request:

Business	**Company**	**Quality**	**Price**	**Time**
Airline	Air France	economy	60 per person	Paris to Rome
Hotel	Intercontinental	2*	110 per night	Rome for 3 nights
Airline	Air France	economy	60 per person	Rome to Monte Carlo
Hotel	Intercontinental	2*	250 per night	Monte Carlo for 2 nights
Airline	*	economy	50 per person	Monte Carlo to London
Hotel	Intercontinental	2*	250 per night	London for 2 nights

Figure 11: EvE Demonstrator - Sample User Request

The star '*' symbol in the Company column, of Figure 11, represents a wildcard. Other minor changes were made to the simulation to provide the best solution or solutions at each generation, which can be seen in Figure 12.

The aim with the textual semantic descriptions is to show the relevance of the Efficiency performance measure in the evolutionary processing of BML data to the other project partners, particularly ISUFI and STU. The semantic meaning assigned to the numerical descriptions is a feature not required in the fitness landscape simulator developed by STU in deliverable D9.1 'Report on Fitness[38], but then we wish to show the applicability of the Efficiency E performance measure to the EvE, whereas they wish to demonstrate various aspects of fitness in the EvE.

Generation = 153
Population size = 58
Highest % Match = 95.885%
Average % Match = 93.57928571428583%
organisational Complexity = 5.795137251578087
Complexity Potential = 6
Efficiency = 0.965856208596
Efficiency Performance Measure (Probability Best Solution) = 90.384134103155%

Best Solution(s) = 1

Service: fitness=85.18500000000002%

Business	Company	Quality	Price	Time
Airline	Luthansa	economy	70 per person	Paris to Milan
Hotel	Marriot	2*	120 per night	Milan for 2 nights
Airline	KLM	economy	70 per person	Milan to Monte Carlo
Hotel	Hilton	3*	350 per night	Monte Carlo for 3 nights
Airline	Air France	economy	60 per person	Monte Carlo to London
Hotel	Marriot	2*	350 per night	London for 1 nights

Figure 12: EvE Demonstrator - Sample - Generation 127

The Efficiency E has been scaled by the average fitness as defined in equation (3), which is not well shown in Figure 12, but is easily seen in Figure 13. Therefore, the Efficiency E performance measure estimates the probability that a solution chosen at the current generation is the optimal solution.

Generation = 37
Population size = 58
Highest % Match = 56.52
Average % Match = 53.78612068965517
organisational Complexity = 3.2159970180147273
Complexity Potential = 4
Efficiency = 0.803999254504
Efficiency Performance Measure (Probability Best Solution) = 43.244000937128%

Best Solution(s) = 1

Service: fitness=56.52%

Business	Company	Quality	Price	Time
Airline	Luthansa	economy	70 per person	Paris to Florence
Hotel	Marriot	2*	120 per night	Florence for 1 nights
Airline	KLM	economy	70 per person	Madrid to Milan
Hotel	Marriot	2*	350 per night	London for 1 nights

Figure 13: EvE Demonstrator - Sample - Generation 37

The Efficiency E is always between zero and one, independent of the population size and the lengths of the individuals within the population. So it can compare different populations, as can the Efficiency E performance measure.

3.5 Summary

An EvE Demonstrator was successfully created to show the application of the Efficiency performance measure to the Evolutionary Environment. To do this a more sophisticated model was developed for the service's BML semantic description, compatible with the BML metamodel[12]. This replaced the abstract numerical BML with meaningful textual descriptions based around the tourism industry, which is one of the opportunity spaces (business sectors) that the project is targeting[28, 31].

The value of the Efficiency performance measure to the DBE, is that it can measure the self-organisation of evolving populations in the EvE. These evolving populations are instantiated in response to user requests. So it can provide status information, to the control mechanism of the EvE and the user, on the progress of finding the desired services. The applicability to the DBE will depend on the aggregated EvEservice structure and the user interaction paradigm, which is discussed further in the Conclusion, subsection 6.4.

The organisational complexity and the Efficiency provide a way to measure the organisation within an evolving population. This understanding is a first step in being able to control the EvE system.

4 Clustering Control of Self-organisation

Now that the self-organisation of an evolving population can be measured, in the theoretical context of the Digital Ecosystem model using the raw Efficiency measure, and applied to the EvE Digital Ecosystem with the Efficiency performance measure, we can consider how to control the self-organisation. A form of selective breeding was chosen to achieve this, as it is known to be viable and effective in accelerating the evolutionary process, but had not yet been applied to evolving agent populations.

We will attempt to control (guide) the evolutionary self-organisation using a form of selective breeding, by encouraging recombination (crossover) within clusters of the evolving population. A cluster being a grouping of similar and identical individuals. This is similar to when humans breed plants or animals for specific features and qualities, recombination between individuals is enforced that will encourage the desired features and qualities. So there is potential to accelerate the evolutionary process towards the desired goal.

The Efficiency measure's use in detecting clustering, presented in deliverable D6.1[7], is summarised in Appendix D. The Efficiency measure can estimate the number of clusters, but it does not provide a way to determine which individuals of the population are in which clusters. An exhaustive search cannot be used as it would be computationally impractical. So, a technique is sought to determine efficiently the clusters within the population.

4.1 Distance Matrix

The most significant information available, for efficiently determining the clusters, is a matrix of the distances between the individuals. An example is shown in Figure 14.

	Agent Chain 1	Agent Chain 2	Agent Chain 3	Agent Chain 4	Agent Chain 5
Agent Chain 1	0	8.96	12.04	6.5	13.19
Agent Chain 2	8.96	0	3.23	14	5.46
Agent Chain 3	12.04	3.23	0	17.23	8.69
Agent Chain 4	6.5	14	17.23	0	8.54
Agent Chain 5	13.19	5.46	8.69	8.54	0

Figure 14: Example Distance Matrix

A distance matrix can be calculated from the fitness function, which itself works by calculating the distance between a user request and an individual solution within the evolving population. A distance matrix for a population of agent-chains, such as the one shown in Figure 14, can be used to determine the clusters of agent-chains within the population.

4.2 Clustering Techniques [27, 25, 22, 24]

Generally, the problem faced by any clustering technique is to determine the number of clusters to be able to perform the clustering. Calculating the number of clusters is often more challenging than the clustering itself. Fortunately, the number of clusters can be estimated from the organisational complexity. Clustering techniques come in one of two forms.

4.2.1 Hierarchical Clustering

This type of clustering involves successive fusions (agglomerative clustering) or separations (divisive clustering) of objects.

Agglomerative Clustering
Agglomerative clustering starts with all objects as individual clusters. The most similar objects are first grouped. These are then merged according to their similarities, until all are fused into a single cluster. A distance matrix can be clustered using linkage methods. There are three different types of linkage analysis:

- Single linkage (nearest neighbour or minimum distance): is obtained by fusing clusters according to the distance between their nearest members.
- Complete linkage (farthest neighbour or maximum distance): is obtained by fusing clusters according to the distance between their farthest members.
- Average linkage (average distance): is obtained by fusing clusters according to the average distance between pairs of members in the respective sets.

Linkage methods are relatively simple and can be used successfully for the analysis of a distance matrix[27].

Divisive Clustering
Divisive clustering methods start with one cluster containing all objects. These are successively separated into smaller subgroups till the number of clusters equals the number of objects.

4.2.2 Non-Hierarchical Clustering

Non-hierarchical clustering: In this method the number of clusters is assumed at the start. Objects are allocated among the clusters so that a chosen criterion is minimised, e.g. within cluster sum of squares. K-means[24] and Self-Organising Maps[23] are examples of this method.

K-means clustering
In this algorithm each object is assigned to the cluster having the nearest centroid (mean). The objects are partitioned into K (pre-specified) clusters, where K is less than or equal to the number of objects. Proceed by assigning an object to the cluster with the nearest centroid (mean). Recalculate the centroid for the cluster gaining the object and the cluster losing the object. Repeat the two steps till no more reassignments happen. The purpose is to minimize some chosen criterion, eg, squared distance to the group centroids. K-means is also sensitive to outliers. The result obtained must be carefully analyzed to confirm the clusters make sense based on previous knowledge. The algorithm must be run for different values of K to determine the correct value.[24, 27]

Self-Organising Maps
Self-Organising Map (SOM) is a neural-network based on the divisive clustering approach. The aim is to assign genes to a series of partitions because of the similarity of their expression vectors to reference vectors that are defined for each partition.[24, 23]

The first paper about using SOM in distance matrix analysis was published by Pablo Tamayo et al, where they have described the use of this method to interpret patterns of gene expression during the hematopoietic differentiation.[40]

The disadvantages of SOM are:[27]

- It requires some pre-processing of data such as elimination of genes that are not changing, normalisation of the data.
- There needs to be some idea about the number of clusters to expect.

4.2.3 Choice of Technique

The range of clustering techniques available has been shown. The choice of the optimal technique very much depends on the structure of the data. The data should be analysed with the different methods. The stability of the resulting solution can be tested in the presence and absence of small errors.[27]

At this stage, an effective clustering mechanism is sought to test whether encouraging intra-cluster recombination accelerates the evolutionary process. If the intra-cluster recombination is effective, then a more in-depth investigation into the optimal clustering technique can be done. Hierarchal agglomerative average linkage clustering was chosen, as it is known to be effective and does not require any specialist customisation to be implemented.

4.2.4 Hierarchal Agglomerative Average Linkage

The result a hierarchical clustering can be represented as a dendrogram. Figure 15 shows the result of applying the chosen clustering technique to the distance matrix in Figure 14, as a dendrogram.

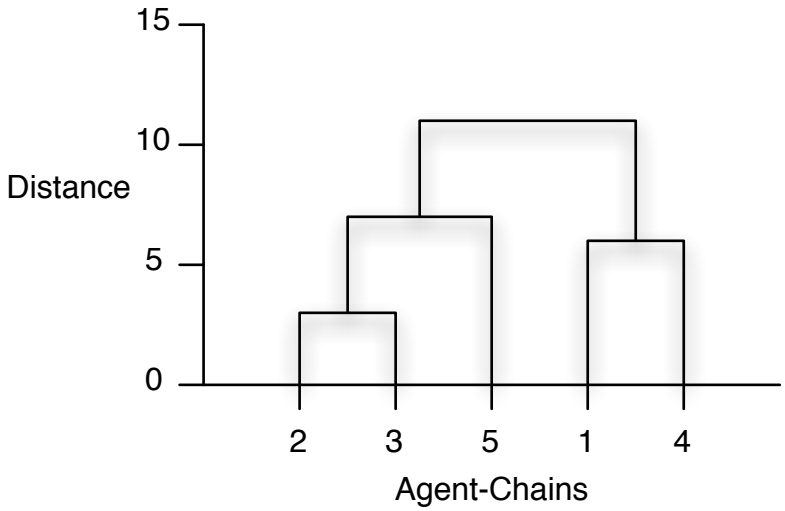


Figure 15: Dendrogram for the Clustering of Figure 14

4.3 Simulation Specifications

The core specifications for the simulation are provided in deliverable D6.1[7]. A summary of these same specifications is provided below.

An Agent Station provides an alphabet (agent pool) consisting of a set of 15 randomly generated agents. Each agent represents a DBE service, containing a process object to represent its BML description. A user request consists of several process objects to represent the BML request. So, both the user request and agents have BML components which can be compared in the evolutionary process by the fitness function. The semantic descriptions are described in more detail in section 3.2.

A population is created to find a solution to the user request, from the set of agents available at the Agent Station. The solution is found by evolving the population of solutions, seeded with the set of agents available at the Agent Station.

This simulation implements a simple evolutionary process, which works by assigning fitness values to the current population using the Fitness Function. The Fitness Function determines the fitness of the solutions in the population. This is done by comparing the process objects of the agent-chains with the list of process objects of the user request, and calculating the distance between them as a percentage value. So, the Fitness Function assigns fitness values between 0.0 and 100.0 to each agent-chain in the population.

The type of selection used is non-elitist, as the best individual from one generation is not guaranteed to survive into the next generation. It has a high probability of surviving into the next generation, but it is not guaranteed as it may be mutated. Then the weakest (poor fitness) 10% of the population is deleted (death rate), and then the strongest (high fitness) 10% of the population is allowed to replicate multiple times (replication rate). So the middle 80% of the population is neither replicated or deleted, which is equivalent to being allowed to reproduce once before dying. Subsequently, 10% of the population is mutated randomly using point mutations (mutation rate), with agents drawn from the set of agents in the Agent Station. The point mutation consists of insertions (an agent is

inserted into an agent-chain), replacements (an agent is replaced in an agent-chain), and deletions (an agent is deleted from the agent-chain).

4.4 Modifications to the Simulation

The simulation was modified to include a crossover operator. A crossover rate of 25% is applied to the evolving population. That is 25% of the individuals within the population are randomly chosen to perform a crossover with an individual chosen randomly from the remaining 75% of the population.

The clustering was achieved by first generating a distance matrix for the current generation of the population, with the distances calculated by the Fitness Function. Hierarchal agglomerative average linkage clustering is applied to this distance matrix, and used to cluster the current generation of the population into sub-populations of similar or identical individuals. The crossover operator is then favoured within the clusters, so that only individuals within the same clusters can perform a crossover. Again, the 25% crossover rate is applied to each cluster, where 25% of the individuals within each cluster are randomly chosen to perform a crossover with an individual chosen randomly from the remaining 75% of the same cluster.

The simulation was also changed to allow for the averaging of results over thousands of runs. The simulation is written in Java so it was easy be run simultaneously on different platforms to acquire sufficient results for statistical significance.

4.5 Experiments

4.5.1 Effect of Clustering

The addition of the crossover operator could potentially reduce the average number of generations to find a solution. Taking this into account we expect to see a further reduction from the clustering mechanism itself. These predictions are shown graphically in Figure 16.

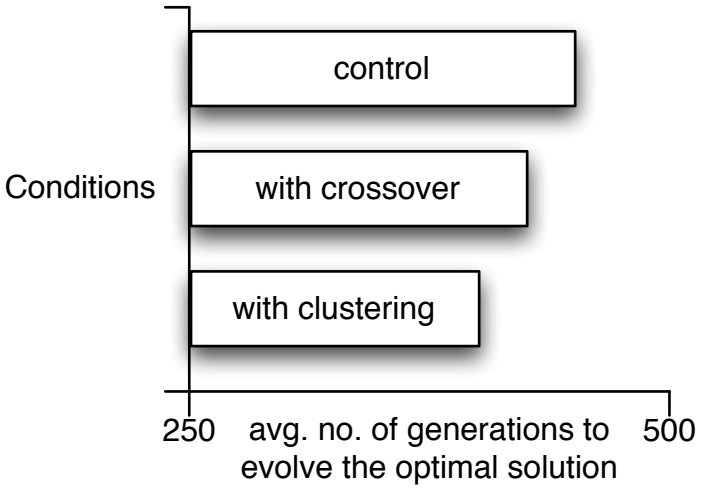


Figure 16: Graphical Predictions for Effect of Crossover and Clustering

The addition of a crossover operator can potentially reduce the number of generations required to evolve the optimal solution, depending on the system specifics. It causes

more recombination in the evolving population. So there is increased variation in the population to explore potential solutions, thereby providing potential to reduce the number of generations required to evolve the optimal solution. The clustering encourages crossover within the clusters. So it will encourage those individuals with similar features to perform a crossover, potentially reinforcing those features. If the reinforced features are desirable there will be a significant increase in fitness, thereby providing further potential to reduce the number of generations required to evolve the optimal solution.

4.6 Results

The simulation was run ten thousand times for each scenario for statistical significance, of the averages, standard deviations and t-tests to be calculated.

4.6.1 Effect of Clustering

Below shows a graph of the average number of generations to evolve the optimal solutions under the scenarios of crossover and clustering. A control scenario without clustering or crossover was also included for comparison.

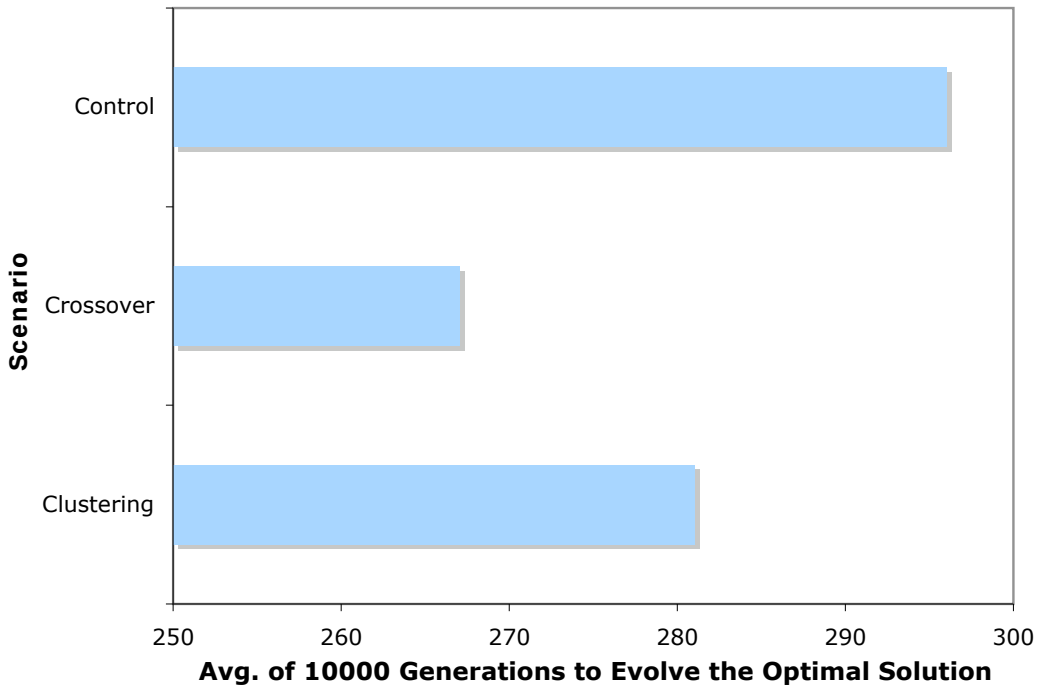


Figure 17: Graph 1 : Effect of Crossover and Clustering

The control scenario averaged to 296.035 generations with a standard deviation of 35.76, the clustering showed about a 10% reduction to 266.967 generations with a standard deviation 30.91. The clustering failed to further reduce the average number of generations to evolve the optimal solutions. It was actually about 5% more than the crossover scenario, at 281.098 generations with a standard deviation of 24.25. A paired t-test of both the clustering and the crossover scenarios with the control scenario, indicated no significant difference at the 95% level.

4.6.2 Investigation of Clustering Result

As the clustering was unsuccessful a typical run of each scenario was plotted to observe the behaviour, and better understand why the clustering did not succeed.

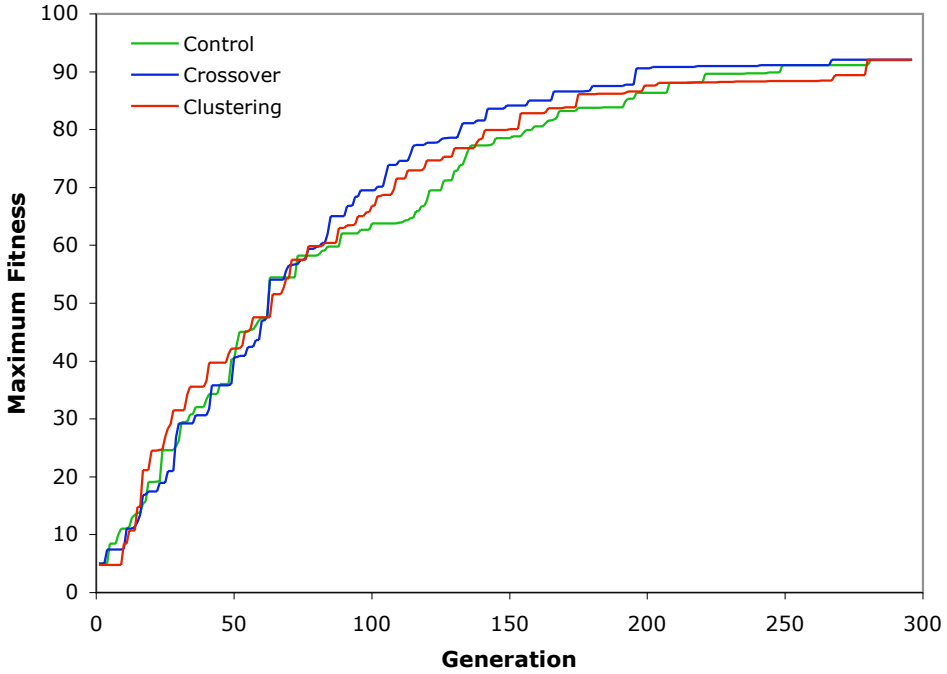


Figure 18: Graph 2 : Typical Runs from Crossover and Clustering

Graph 2 (Figure 18) did not show any unexpected behaviour. The evidence would therefore suggest that the assignment of crossover by the clustering mechanism is less efficient than the random assignment of crossover.

4.6.3 Organisational Complexity Clustering

As the main contribution to determining the clusters comes from the distance metric used to generate the distance matrix, it was decided to use the organisational complexity instead of the fitness function. This scenario with complexity clustering was run ten thousands times to calculate the average number generations to evolve the optimal solution.

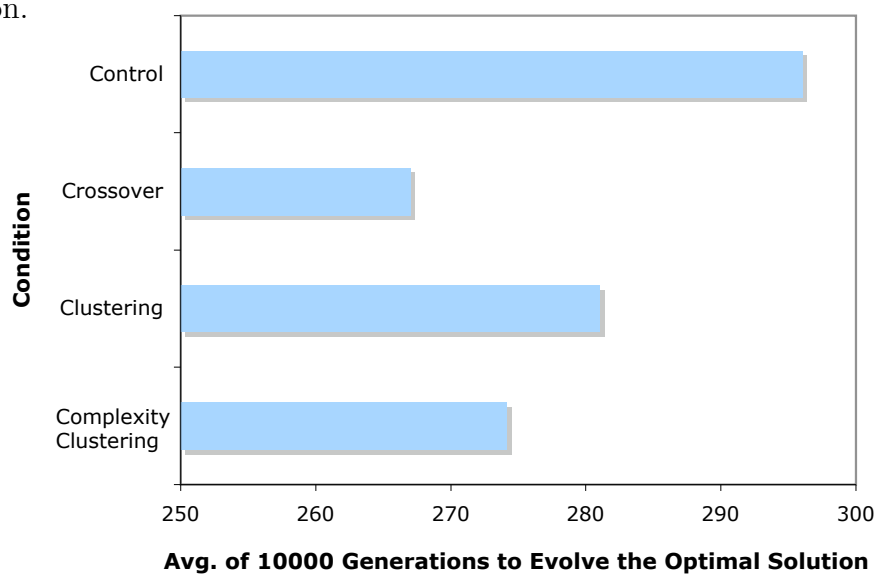


Figure 19: Graph 3 : Effect of Complexity Clustering

The complexity clustering averaged 274.016 generations with a standard deviation of 28.53, and a paired t-test of with the control scenario again indicated no significant difference at the 95% level. So the complexity clustering, although more effective then the original clustering, still failed to reduce the average number of generations required to evolve the optimal solution.

4.7 Summary

All the results presented showed the clustering controlled crossover to be less efficient than random crossover. Additional experiments were conducted to vary several parameters to determine if there were any conditions under which the clustering was effective. The varied parameters included the population size, the mutation rate and the crossover rate. Extensive testing through multiple scenarios failed to show a significant reduction in the number of generations required to evolve the optimal solution. More extreme scenarios could have been tried, but they would have taken the EvE Digital Ecosystem model and simulation outside the scope of the DBE. The application of selective breeding intuitively had potential, but it is surmised that the individuals within the evolving population lacked sufficient complexity (relative to biological evolving populations) for the mechanism to be effective. Also, in any form of selective breeding intelligence is used in determining which individuals should be crossed, which was not done with the clustering mechanism.

Another reason, why the clustering mechanism might not have worked is that crossing very similar individuals will produce children that are again very similar to their parents. So it does not actually achieve valuable change. When selective breeding is applied to animals, the forced mating is done to preserve rare traits that have arisen through mutation which is different to clustering mechanism that does not explicitly identify traits.

The clustering mechanism does however allow for the application of the Efficiency measure for populations with multiple clusters E_m in equation (D.9) of Appendix D. It was reformulated from the Efficiency E performance measure in deliverable D6.1[7], and can manage populations with multiple clusters E_m. It is equivalent to E if the population consists of only one cluster, and if there are multiple clusters E_m is the average of the Efficiencies of the clusters. Whether the E_m or E is required for evolving populations within the Evolutionary Environment will depend on whether a mechanism is implemented to maintain variation within evolving populations, such as 'fitness sharing'[8].

5 Transitional Self-organisation

A better understanding of the EvE Digital Ecosystem was sought, motivated by the lack of success with the clustering mechanism. We decided to extend an existing definition of agent stability[11] based on the following reasons:

- The literature review earlier conducted on self-organisation of evolving agent populations[7].
- The capability of representing any aggregated agent structure.
- Its compatibility with the new Science Vision[15] definition of self-organisation.
- Complementary to the theoretical Markov processes based definition of genetic algorithms in deliverable D8.1[33] 'Report on Evolution of High-Level Software Components'
- There has been past work on modelling genetic algorithms as Markov chains, but as far as we can check none of the results [4, 3, 19, 16, 36, 37] are 'directly' applicable to the EvE Digital Ecosystem model to be implemented in the DBE, as none have been applied to Multi-Agent Systems.
- It allows the rare application of the statistical physics measure of entropy to Multi-Agent Systems.

Both evolutionary computing and Multi-Agent Systems are mature areas, but the application of one to the other is non-trivial because the agents are state machines, and evolutionary computing algorithms have generally been developed to work on numerical data. We will attempt to bridge these areas by extending a definition of agent stability to include evolutionary dynamics. The definition of agent stability is based on modelling the agents as Markov processes, which will be extended to apply to Multi-Agent Systems with evolutionary dynamics. A definition for self-organisation, and the degree of self-organisation which will be entropy based, will be constructed from the extended agent stability. We will therefore achieve a definition compatible with Multi-Agent Systems, from the agent stability, and evolutionary computing as defined in deliverable D8.1[33], from the extension to include the evolutionary dynamics.

5.1 Definition

Deliverable D8.1 goes much further in the theory then we do for our definition. We choose not to go any further to prevent it becoming too theoretical for our purposes, as we wish to focus on its application to Multi-Agent Systems with evolutionary dynamics, and determine a macroscopic variable to characterise the evolutionary self-organisation of an evolving agent population.

5.1.1 Definition of Agent Stability

A system is composed of n agents, and each agent i is in a state $\xi_i(t)$ at time t, where $i = 1, 2, ..., n$. The states of the agents are random variables. The state vector for the

Multi-Agent System is a vector of random variables $\boldsymbol{\xi}(t)$. The time is discrete, $t = 1, 2, \ldots$. The interactions among the agents are noisy, and are given by the probability distributions

$$P_i(x_i, \mathbf{y}) = \Pr(\xi_i(t+1) = x_i | \boldsymbol{\xi}(t) = \mathbf{y}), \quad 1, \ldots, n, \tag{4}$$

where x_i is a value for the state of agent i, and $\mathbf{y}$ is a value for the state vector of the Multi-Agent System. The distributions implement a Markov process[39]. The noise can be caused by mutations. If there is a single mutation rate r for all agents, this rate can be related[30] to a temperature T. Deliverable D8.1[33] has a less specific definition of states, as their state vector is the genotype frequencies of the individuals within the population, and the ordering of the individuals within the population does not affect the state vector. This is the only significant difference to the definition presented in deliverable D8.1[33]. However, as the genotype frequencies can be determined from our state description, we can determine the *population vector* p in deliverable D8.1[33]. So this does not make the definitions incompatible.

The agents are individually subject to a selection pressure from the environment of the system, and the selection pressure and evolutionary dynamics are applied equally to all agents. This means that the probability distributions P_i are statistically independent, and

$$P(\mathbf{x}, \mathbf{y}) = \Pi_{i=1}^{n} \Pr(\xi_i(t+1) = x_i | \boldsymbol{\xi}(t) = \mathbf{y}). \tag{5}$$

If the occupation probability of state $\mathbf{x}$ at time t is denoted by $p_{\mathbf{x}}(t)$, then

$$p_{\mathbf{x}}(t) = \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}) p_{\mathbf{y}}(t-1). \tag{6}$$

This equation is used to calculate the evolution of the state occupation probabilities from $t = 0$.

Equation (5) is our version of the *transition matrix* Q_{ij} as defined in deliverable D8.1[33], which is the probability of moving from one state to another. This *transition matrix* is a discrete time equation, referred to as the Hardy-Weinberg model in population genetics (and the Nix-Vose model in genetic algorithm theory). In the "Mathematical background" section of deliverable D8.1[33] it is explicitly derived from the underlying replicator equation for a system at the infinite population limit. Our Multi-Agent System model of the EvE is not of the scale that many theoretical results would be applicable. There is not an infinite number of states, or infinite time, so theoretical results might have limited applicability. We would require further tests to establish how close our system is to the thermodynamic limit.

The Multi-Agent System is self-organising if the limit distribution of the transition probabilities exists and is non-uniform, i.e.

$$p_{\mathbf{x}}^{\infty} = lim_{t \to \infty} p_{\mathbf{x}}(t) \tag{7}$$

exists for all states $\mathbf{x}$, and there exist states $\mathbf{x}$ and $\mathbf{y}$ such that

$$p_{\mathbf{x}}^{\infty} \neq p_{\mathbf{y}}^{\infty}. \tag{8}$$

In words this means that some configurations of the system are more likely to occur than others after a long time, if the likelihood of occurrence does not change any more.

A system where the likelihood of occurrence of the states does not change anymore is stable. This is the definition of stability developed in [11]. Equation (7) is equivalent to equation $Qv = v$ as defined in deliverable D8.1[33] and such states are termed fixed-points of the mapping G, but we do not presume that they exist. In a Multi-Agent System with changing tasks (user requests) over time to be fullfilled, it is non-trivial to determine a fixed fitness function, so it is not guaranteed for there to be a limit distribution. Also, we agree that the limit probability is a property of Markov chains.

Stable systems with uniform limit probability distributions over a subspace of the state space are called 'fair' in [42]. The convergence of fixed-length binary strings is a basic theorem of population genetics known as Geiringers Theorem, and the 'Geiringer' type theorems in deliverable D8.1[33] give examples of 'fair' systems.

5.1.2 Extention to Self-Organisation

A stable system is self-organising if there is a limit probability and it is not uniform.

Equation (7) is equivalent to the existence of an attractor in a Multi-Agent System with deterministic interactions. This is the definition of self-organisation that is given in the new DBE Science Vision[15]. We had to extend this to a stochastic process, because mutation is essential in evolutionary dynamics.

The number of agents in this formalism is fixed. To model a system with a varying number of agents, introduce a new state d for each agent. If the agent is in this state, $\xi_i(t) = d$, it is "dead". When a new agent has to be created, one of the "dead" agents gets assigned a state that is different from d. Dead agents do not affect the state of the other agents. If agent i has a low fitness, then that agent is likely to die,

$$P_i(d, \mathbf{y}) = \Pr(\xi_i(t+1) = d | \boldsymbol{\xi}(t) = \mathbf{y}) \tag{9}$$

is high. If the agent has high fitness, then a dead agent is likely to assume the same state of the successful agent (the successful agent duplicates), or a crossover might occur, changing the state of the successful agent and one other agent. The state of an agent can be a **bitstring** of variable length. The string is padded with zeros to give a fixed length. The state of the agent is simply the binary number represented by the string. We chose to use a bitstring representation as it is a general purpose representation of data used in computing.

A degree d_s of self-organisation can be defined based on equation (8), as the entropy of the probability distribution at infinite time,

$$d_s = H(p_{\mathbf{x}}(\infty)) = -\sum_{\mathbf{x}} log_N(p_{\mathbf{x}}(\infty)), \tag{10}$$

where N is the number of possible states. Taking log to the base N proportions the number of limit states against the total number of states that the system can occupy. The requirement of equation (8) in the definition of equations (7-8) is weak. Fairness is rarer in Multi-Agent Systems than self-organisation, so the degree of self-organisation is vital in differentiating how much self-organisation there is from one Multi-Agents System to the next.

5.2 Application to the EvE Digital Ecosystem

To apply the definition to the agent based EvE Digital Ecosystem model, the state description of the agents and aggregated agents will be bitstrings. We chose to represent the aggregated agents as bitstrings because the data, the semantic descriptions of agents, are not programs or methods, but the descriptions of business processes, such as holidays, or manufacturing chains. This type of information has an inherent linear nature, so a string representation is suitable. Therefore, the state of the evolving agent population is the collection (concatenation) of the aggregated agent bitstrings.

5.2.1 Agents as Bitstrings

The semantic descriptions of agents can of course be represented as bitstrings, as all data within digital systems exist in binary format. An example encoding strategy is shown in Figure 20, which encodes the semantic description of the agent shown in Figure 10.

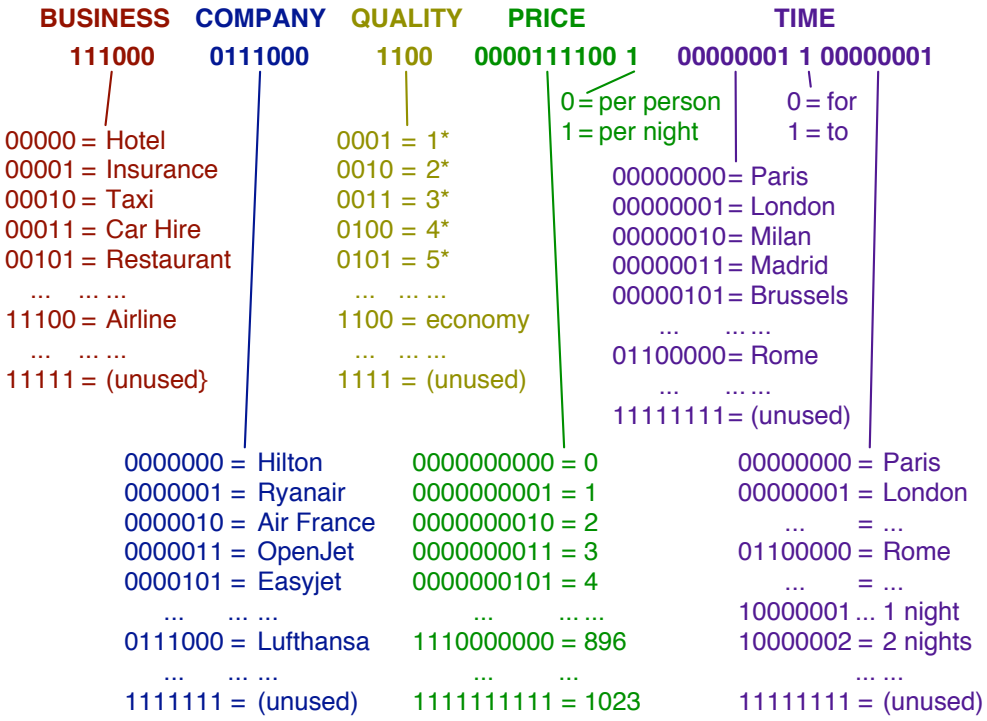


Figure 20: Example Coding Scheme for the Semantic Descriptions of Agents

The structure of aggregated agents (EvEservices) is still not finalised for the EvE[8]. So we will provide an encoding scheme for simple workflows that can represent the functionality of the BPEL workflows used by the computing stream, because the workflow aggregated structure can also represent the other structures. A simple workflow representation was created using a matrix, which is functionally representative of a simple BPEL workflow. An example workflow encoding is shown in Figure 21.

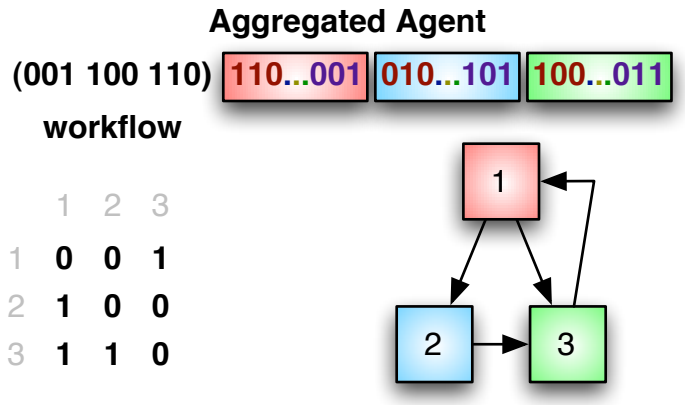


Figure 21: Example Coding Scheme for Workflows

So an individual within an evolving population is a single agent as shown in Figure 20, or a group of aggregated agents as shown in Figure 21, with padding to standardise the length. So the state of an evolving population is the bitstring representation of every agent within the population. For an evolving population of the EvE Digital Ecosystem model, an example of a state description that meets the requirements of the definition in section 5.1, is shown in Figure 22.

000110110101...101001...111111...1110000...00000
000110100101...101001...111111...1110000...00000
010010110101...101001...111111...1110000...00000
010111001101...101001...111111...1110000...00000
000011001001101...101001...111111...111001...111
000110100101...101001...111111...1110000...00000
000011001001101...101001...111111...111001...111
000110100101...101001...111111...1110000...00000
000011001001101...101001...111111...111001...111
000110100101...101001...111111...1110000...00000

Figure 22: An Example State Description of an Evolving Population

In Figure 22 the example state description has padding at the end to compensate for the different number of agents within each aggregated agent structure, because in this example each service has the same number of attributes. The padding would also have to compensate additionally if the agents had different numbers of attributes.

Including the workflow string allows us to model any of the aggregated agent structures that may be chosen for the EvE. If 'sets' are chosen then the workflow strings would be all zero and not affected by the mutation operator. If 'chains' or 'trees' are chosen then the workflow strings would be affected by the mutation operator, ensuring unidirectional information flow. Finally, if 'workflows' are chosen then the workflow strings would be subjected to an unrestricted mutation operator.

5.2.2 Evolving Agent Population State Space

The state space is easier to understand by considering the possible macro-states, which is visualised in state representation shown in Figure 23. A macro-state is a cluster of

states all with the same maximum fitness individual, and each state differs in the number of maximum fitness individuals within the population and the population size. So two populations are considered equivalent if they have the same maximal fitness. Within each cluster is a centroid state which is composed entirely of copies of the maximum fitness individual.

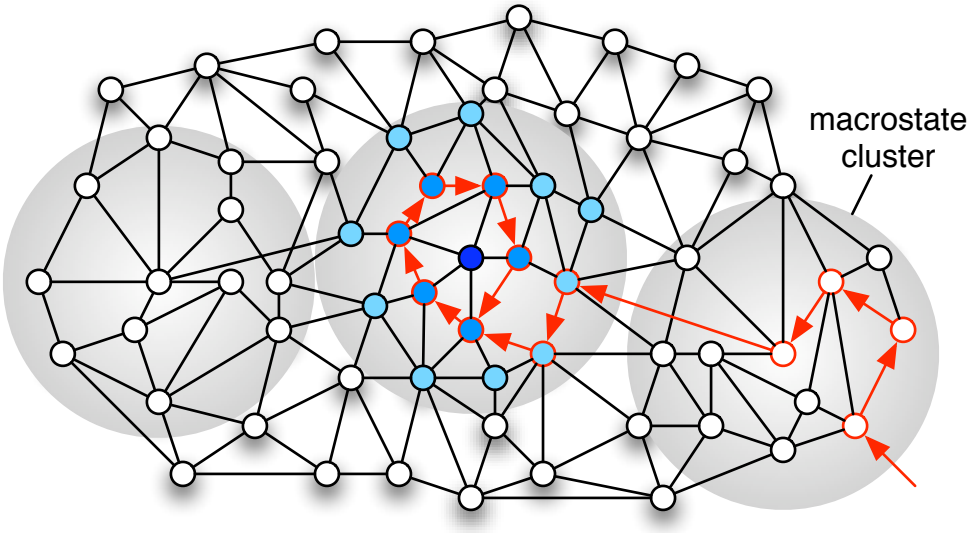


Figure 23: Evolving Population State Space

A possible path of evolution is shown through the state space in Figure 23. The states of the central cluster all contain at least one copy of the optimal solution, which has maximum fitness F_{max}. The centroid state of the central cluster is when all the individuals are copies of the optimal solution. This is the equilibrium state that the system is forever falling towards, propelled by the selection pressure, without ever reaching it because of mutation. This optimal macro-state is certain to be reached (if there is only one optimal solution), i.e., the probability of the being in the state at infinite time is $p_{\mathbf{x}}(\infty) = 1$ as defined in equation (6). Rudolph[35] has shown that the optimal population (central cluster) is an absorbing state of the Markov process, and the evolutionary algorithm is almost guaranteed to converge to it, but may take a significant length of time.

5.3 Experiments

The same simulation specifications as defined in section 4.3 will be used for the experiments.

5.3.1 Test for Self-organisation

The Multi-Agent System is self-organising if the distribution of the limit probabilities exists and is non-uniform, as defined in equation (8). This will be verified for an evolving agent population, with one global optimal solution, by finding at least two macro-states with different transition probabilities.

The maximum fitness of any individual at any generation is F_{max}, and F_{med} is half of F_{max}. The probability of the system being in macro-state F_{max}, which includes at least one individual with maximum fitness F_{max}, at the thousandth generation will be one,

$p_{\mathbf{F_{max}}}(1000) = 1$. Furthermore, the probability of the system being in the sub-optimal macro-state F_{med}, which includes at least one individual with median fitness F_{med}, at the thousandth generation will be one, $p_{\mathbf{F_{max}}}(1000) = 1$. This will fulfil the requirement of equation (8). This is presented graphically in Figure 24.

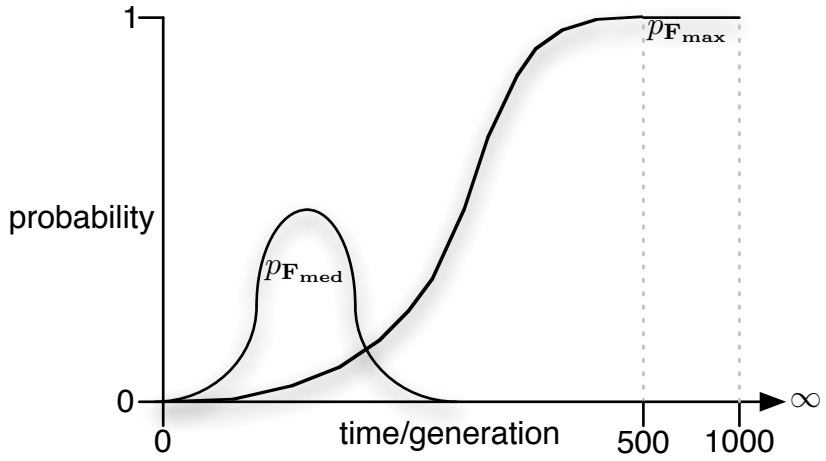


Figure 24: Graphical Predictions for $p_{\mathbf{F_{max}}}$ and $p_{\mathbf{F_{med}}}$

The sub-optimal macro-state F_{med} is predicted to be seen earlier in the evolutionary process as it has a lower fitness F_{med}, and then to disappear as higher fitness states are reached. The system will take longer to reach the optimal macro-state F_{max}, but once it does it is unlikely to leave it (depending on the strength of the mutation rate), and even if it does it will return quickly.

The convergence at $t = \infty$ of an evolving population of binary strings is formalised mathematically in deliverable D8.1[33]. The convergence of fixed-length binary strings is a basic theorem of population genetics known as Geiringers Theorem, and has recently been extended by Rowe and Mitavskiy to variable-length structures and trees[33]. A specific value of $t = 1000$ was chosen to represent experimentally $t = \infty$, based on past experience with this simulation. Effectively, $t = 1000$ is equivalent to $t = \infty$ as this simulation has generally been observed to reach the optimal macro-state F_{max} within 500 generations.

5.3.2 Estimate Degree of Self-organisation

The degree of self-organisation d_s in equation (10) is heavily affected by the number of optimal F_{max} macro-states. As the simulation is known to have only one global optimum solution it is predicted

$$\begin{aligned} d_s = H(p_{\mathbf{x}}(1000)) &= -\sum_{\mathbf{x}} log_N(p_{\mathbf{x}}(\infty)) \\ &= -1 log_N(1) \\ &= 0, \end{aligned} \tag{11}$$

as the probability of being in the macro-state F_{max} at the thousandth generation will be one, $p_{\mathbf{F_{max}}}(1000) = 1$. Therefore, showing the maximum degree of self-organisation as there is no entropy at $t = 1000$ (infinite time).

5.3.3 Occupancy of Optimal Macro-state F_{max}

Although the system will reach the optimal macro-state F_{max}, it will not reach the centroid state where all individuals have the same maximum fitness F_{max}, as shown in Figure 23. The end states will be tested, and a visualisation presented of a population at the thousandth generation, which will show there is not total uniformity in the population.

5.3.4 Relation of Mutation Rate to Temperature

It is a well known issue to find the optimal mutation rate for the evolutionary problem being solved. A variable mutation rate that starts high and that decreases over the generations[26] is often recommended.

It has been proved[37, 34] theoretically how to guarantee convergence to the optimum, by lowering the mutation (and crossover) rate while simultaneously increasing the selection strength. It is also necessary to have a sufficiently large population for this to work. The proof is rather similar to the proof of convergence for simulated annealing and relates the mutation/crossover rate to 'temperature'[34]. We wish to investigate if the mutation rate can be related to temperature[30, 37] in the context of the EvE Digital Ecosystem model. So we will disable the crossover and predict the consequences of changes in the mutation rate in accordance with changes of temperature in physical systems.

The mutation rate of the simulation is at 10%. Meaning the number of point mutations that are applied is 10% of the population size. Increasing the mutation rate (temperature) moderately will generally reduce the time to evolve the optimal solution (i.e. reach the optimal macro-state F_{max}). If the mutation rate (temperature) is 0%, then the system will never reach the optimal macro-state F_{max}. Also, if the mutation rate (temperature) is too high (100%) then the system will also fail to reach the optimal macro-state F_{max}. This is presented graphically in Figure 25.

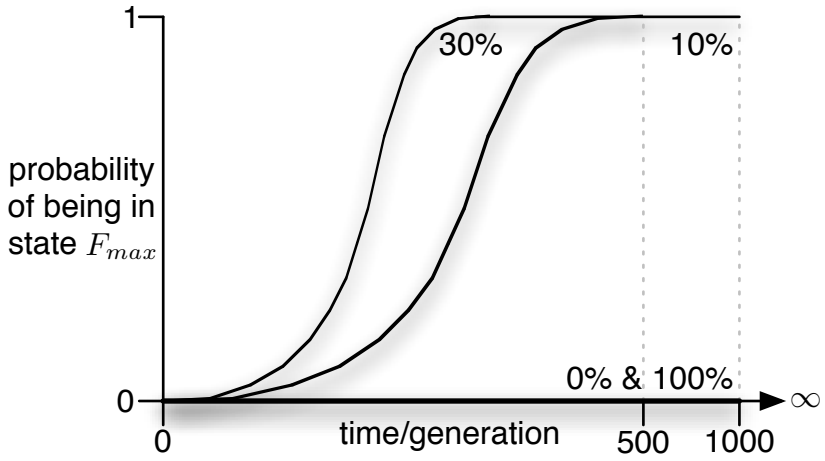


Figure 25: Graphical Predictions for Variation of Mutation Rate (Temperature)

The system will fail to reach the optimal macro-state F_{max} if the temperature (mutation rate) is zero, as there is no opportunity for the agent solutions to grow. The system will also fail to reach the optimal macro-state F_{max} if the temperature (mutation rate) is too high, as there will be too much opportunity for high fitness individuals to be sub-optimally mutated before they can replicate. It is predicted that a moderate increase to 30% will

reduce the number generations to evolve the optimal solution because the initial rate is quite low at only 10%.

5.4 Results

The simulation was run ten thousand times for statistical significance of the probabilities to be calculated.

5.4.1 Test for Self-organisation

Below shows a graph of the probability, as defined in equation (6), of the macro-state F_{max} and the macro-state F_{med} at each generation.

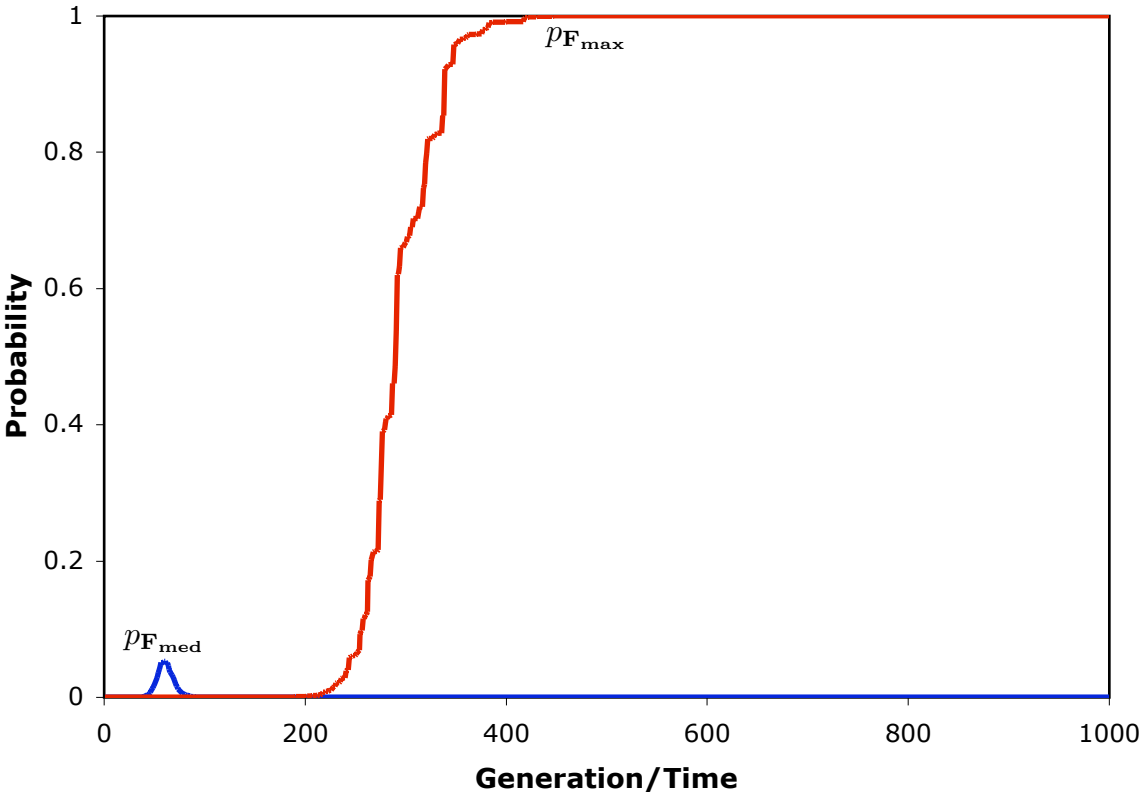


Figure 26: Graph 4 : State Occupation Probabilities - $p_{\mathbf{F}_{\max}}$ and $p_{\mathbf{F}_{\text{med}}}$

The prediction of the behaviour of the system (shape of the curves) was correct. The system can only be in the optimal macro-state F_{max} after generation 178 and is always in the optimal macro-state F_{max} after generation 482.

The highest probability of the system being in the sub-optimal macro-state F_{med} is 0.053, because the macro-state F_{med} does not have to be visited during the simulation, it can be skipped altogether depending on the state transitions caused by mutation. The system only has the possibility of being in the sub-optimal macro-state F_{med} between generations 37 and 113.

5.4.2 Degree of Self-organisation

The degree of self-organisation d_s, as defined in equation (10), for the system calculated from the limit probabilities is,

$$\begin{aligned} d_s = H(p_{\mathbf{x}}(1000)) &= -\sum_{\mathbf{x}} log_N(p_{\mathbf{x}}(1000)) \\ &= -(p_{\mathbf{F_{med}}}(1000) log_N(_{\mathbf{F_{med}}}(1000)) + .. + p_{\mathbf{F_{max}}}(1000) log_N(p_{\mathbf{F_{max}}}(1000))) \\ &= -(0 log_N(0) + .. + 0 log_N(0) + .. + 1 log_N(1)) \\ &= 0, \end{aligned} \tag{12}$$

where $t = 1000$ is a reasonable estimate for $t = \infty$ based on the results graphed in Figure 26. The result is as predicted, because also as predicted the probaility of being in the F_{max} macro-state at the thousandth generation was one, $p_{\mathbf{F_{max}}}(1000) = 1$, and so the probability of being in the other macro-states at the thousandth generation was zero. The system therefore shows the maximum degree of self-organisation, as there is no entropy in the occupied macro-states at infinite time.

5.4.3 Occupancy of Optimal Macro-state F_{max}

As predicted the system reached the optimal macro-state F_{max}, and never reached the centroid state where all individuals were the same and had maximum fitness F_{max}. All tests for the centroid state were negative. A visualisation for the state of an evolving population at the thousandth generation is shown in Figure 27.

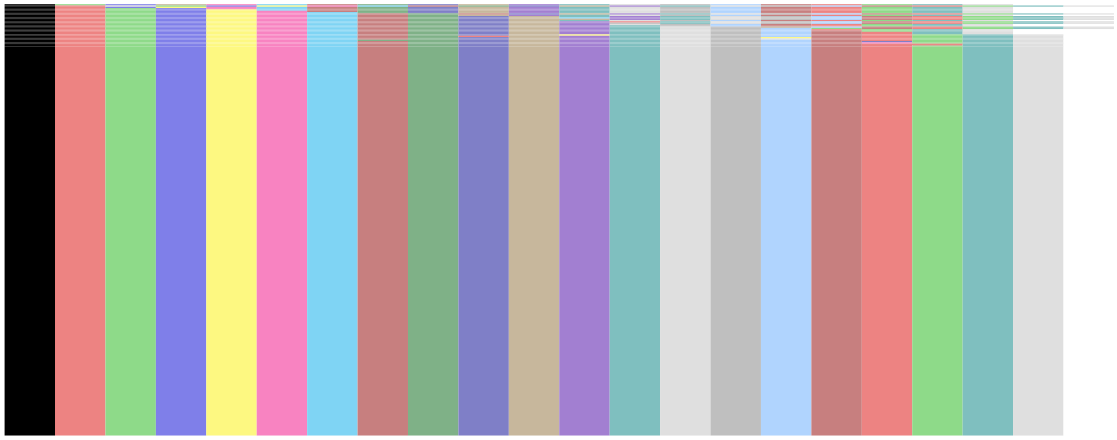


Figure 27: Visualisation of Evolving Population at Generation 1000

The lack of total uniformity in Figure 27 shows that the system is not at the centroid state of the optimal macro-state F_{max}, because of the mutation within the evolutionary process.

5.4.4 Investigate Relation of Mutation Rate to Temperature

Figure 28 shows the effect of varying the temperature (mutation rate) on the probability of the system reaching the optimal macro-state F_{max}.

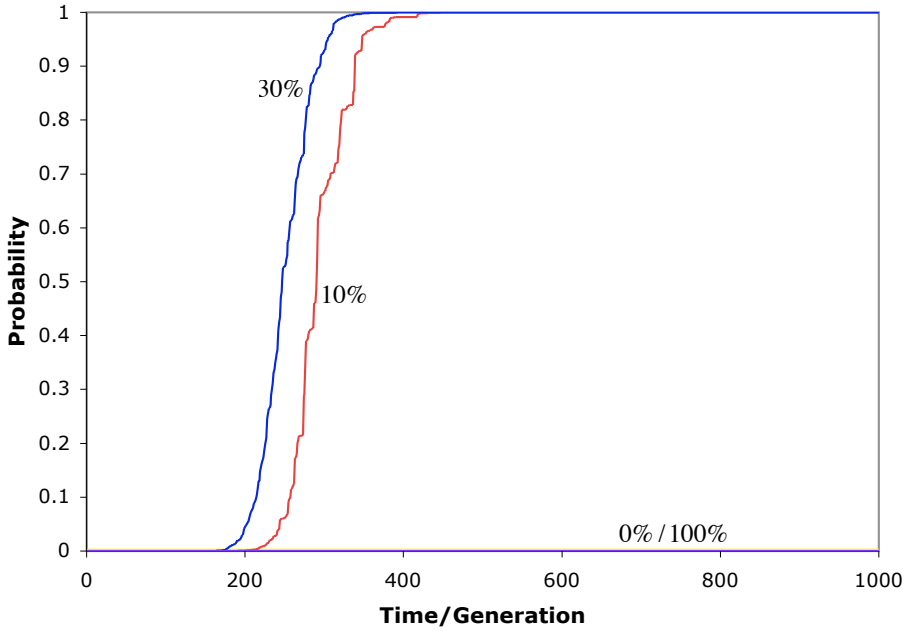


Figure 28: Graph 5 : Variation of Temperature (Mutation Rate)

Both extremes of temperature (mutation rate), 0% and 100%, prevented the system from reaching the optimal macro-state F_{max}. The increase of the temperature (mutation rate) to 30% did reduce the time to reach the optimal macro-state F_{max}. So the mutation rate can be considered a temperature like control parameter for evolutionary self-organisation of an evolving agent population in the EvE Digital Ecosystem model.

5.4.5 Summary

The transitional self-organisation was developed to provide a better understanding of a Multi-Agent System with evolutionary dynamics, specifically an evolving agent population of the EvE Digital Ecosystem model. The definition of self-organisation is based on the limit probability of the system when modelled as Markov processes, and the degree of self-organisation (entropy of the limit probabilities) provides a macroscopic variable to characterise the evolutionary self-organisation, both of which are complementary to the more generally applicable theory presented in deliverable D8.1[33]. The results show the transitional self-organisation has been successfully applied to the EvE Digital Ecosystem model. The results also show that the mutation rate can be considered a temperature like control parameter for evolutionary self-organisation of evolving agent populations. None of the results are contradictory to the results presented in deliverable D8.1[33], as they use complementary definitions.

The scientific value comes from the non-trivial application of evolutionary computing theory[33] to a Multi-Agent System by extending an existing definition of agent stability[11]. Also, it allows the rare application of the statistical physics measure of entropy to Multi-Agent Systems, and does in fact define self-organisation for any Multi-Agent System provided that its state can be represented as a bitstring. The value to the DBE that it provides is a better understanding of the EvE Digital Ecosystem, and a measure of self-organisation applicable whatever the aggregated agent structure (i.e. it is not limited to agent-chains). Furthermore, it is a quantitative definition compatible with the qualitative self-organisation definition from the new DBE Science Vision[15].

6 Conclusions

6.1 Experimental Results

6.1.1 Efficiency Performance Measure

The EvE Demonstrator successfully showed the potential use of the Efficiency performance measure in handling user requests within the EvE, Figures 12 and 13.

6.1.2 Selective Breeding - Clustering

The results showed that the selective breeding clustering mechanism, whether the fitness based (Graph 1 - Figure 17) or complexity based (Graph 3 - Figure 19), failed to optimise the evolutionary process. Typical runs from each scenario showed no adverse behaviour, which can be seen in Graph 2 (Figure 18), so the failure to optimise the evolutionary process did not result from any unexpected side effects. All the evidence suggests that the crossover assignment of the clustering mechanism is less efficient than the random crossover assignment.

6.1.3 Transitional Self-organisation

The results show that the transitional self-organisation definition does accurately describe the behaviour of the system, shown by the similarity of Graph 4 (Figure 26) to its prediction in Figure 24.

The results show that the degree of self-organisation estimated for the system was accurate, shown by the prediction in equation (11) and result in equation (12) being identical.

The results support the proposition that the 'mutation rate' is a temperature like parameter for an evolving population in the EvE system, shown by the similarity of Graph 5 (Figure 28) to its prediction Figure 25.

6.2 Achievements

6.2.1 EvE Digital Ecosystem Model

The EvE Digital Ecosystem model, the platform upon which this research has been conducted, has now been finalised. The model has been presented, and further technical information made available in the Appendix B.

Its scientific value is a novel approach to distributed evolutionary computing. Instead of having multiple populations sharing solutions to find the optimal solution for one problem, there are multiple populations to find optimal solutions for multiple similar problems.

The value it provides to the DBE is a digital ecosystem for the project, where the word ecosystem is more than just a metaphor. It is a platform upon which ideas from the science stream can be more easily envisaged within the DBE. The EvE Digital Ecosystem is based on much of the scientific research, will require most of the computing infrastructure,

and the Habitat clustering represents the business opportunity spaces. Therefore, it has potential to facilitate integrated work within the project.

6.2.2 Efficiency Performance Measure

An EvE Demonstrator was successfully created to show the application of the Efficiency performance measure to the Evolutionary Environment. To do this a more sophisticated model was developed for the service's BML semantic description. This replaced the abstract numerical BML with meaningful textual descriptions based around the tourism industry, which is one of the opportunity spaces (business sectors) that the project is targeting[28, 31]. The numerical semantic descriptions were given meaning that would be compliant with the BML metamodel[12], which helped to envisage more realistic user requests. A translation matrix was used, which theoretically can be altered without too much difficulty to show output for applications from other opportunity spaces (business sectors), if so desired.

The Efficiency performance measure itself is a simple construct of the organisational complexity developed. The significance of the scientific contribution from developing the organisational complexity has been discussed in deliverable D6.1[7] where it was presented.

The value of the Efficiency performance measure to the DBE, is that it can measure the organisation of an evolving population in the EvE. This understanding was a first step in being able to control the EvE system. Also, these evolving populations are instantiated in response to user requests. So it can provide status information, to the control mechanism of the EvE and the user, on the progress of finding the desired services.

6.2.3 Transitional Self-organisation

The transitional self-organisation was developed to provide a better understanding of a Multi-Agent System with evolutionary dynamics, specifically an evolving agent population of the EvE Digital Ecosystem model. The definition of self-organisation is based on the limit probability of the system when modelled as Markov processes, and the degree of self-organisation (entropy of the limit probabilities) provides a macroscopic variable to characterise the evolutionary self-organisation, both of which are complementary to the more generally applicable theory presented in deliverable D8.1[33]. The results show the transitional self-organisation has been successfully applied to the EvE Digital Ecosystem model. The results also show that the mutation rate can be considered a temperature like control parameter for evolutionary self-organisation of evolving agent populations. None of the results are contradictory to the results presented in deliverable D8.1[33], as they use complementary definitions.

The scientific value comes from the non-trivial application of evolutionary computing theory[33] to a Multi-Agent System by extending an existing definition of agent stability[11]. Also, it allows the rare application of the statistical physics measure of entropy to Multi-Agent Systems, and does in fact define self-organisation for any Multi-Agent System provided that its state can be represented as a bitstring.

The value to the DBE that it provides is a better understanding of the EvE Digital Ecosystem, and a measure of self-organisation applicable whatever the aggregated agent structure (i.e. it is not limited to agent-chains). Furthermore, it is a quantitative definition compatible with the qualitative self-organisation definition from the new DBE Science Vision[15]. Qualitatively self-organising systems are forever falling toward equilibrium without ever reaching it[15]. The equilibrium for a self-organising evolving agent population is the centroid state of the central cluster, in which all its individuals are copies of the optimal solution. The 'fall' towards the equilibrium state is propelled by the selection pressure, without ever reaching it because of mutation. Quantitatively this is measured by the probability distribution of the states that the system occupies at infinite time. The system is self-organising if the limit probability is non-uniform, and the entropy of the limit probability provides a degree of the self-organisation.

6.3 Limitations

6.3.1 Selective Breeding - Clustering

The application of selective breeding intuitively had potential, but it is surmised that the individuals within the evolving population lacked sufficient complexity (relative to biological evolving populations) for the mechanism to be effective. Also, in any form of selective breeding intelligence is used in determining which individuals should be crossed, which was not done with the clustering mechanism. Another reason, why the clustering mechanism might not have worked is that crossing very similar individuals will produce children that are again very similar to their parents. So it does not actually achieve valuable change. When selective breeding is applied to animals, the forced mating is done to preserve rare traits that have arisen through mutation which is different to clustering mechanism that does not explicitly identify traits.

The clustering mechanism does however allow for the application of the Efficiency measure for populations with multiple clusters E_m in equation (D.9) of Appendix D. It was reformulated from the Efficiency E performance measure in deliverable D6.1[7], and can manage populations with multiple clusters E_m.

The value both scientifically and to the DBE is a better understanding regarding the limitations of the EvE Digital Ecosystem model, specifically the relative simplicity of the individuals (aggregated agents) within the EvE in comparison to a real ecosystem.

6.4 Relation to Project Activities, WPs and Tasks

The EvE Digital Ecosystem model has provided a practical platform for the application of evolutionary computing within the DBE context. It is a framework for all activities within the project related to the EvE Digital Ecosystem and includes some of the more ambitious knowledge sharing mechanisms envisaged since the beginning of the project, which are described in Appendix B.5. The EvE Digital Ecosystem is based on much of the scientific research, will require most of the computing infrastructure, and the Habitat clustering represents the business opportunity spaces. Utilising ideas from all three streams of

the project, it helps to facilitate integrated work. Nevertheless, its main benefit is a framework for collaboration between the science and computing partners involved with the EvE, including the UBHAM, STU, SUN, SOLUTA, ISUFI and ICL.

The Efficiency performance measure has potential use within the EvE, so it directly impacts upon WP6 task C42 and indirectly on other work-packages and tasks associated with the EvE. The lead author of this document is involved in the specification of the EvE architecture. From that perspective, the Efficiency performance measure has potential to be used within the EvE as a supplementary control mechanism for determining when to stop non-trivial evolving populations, and as a user progress mechanism. Ultimately, this will depend on the final decisions taken regarding the aggregated EvEservice structure and the user interaction paradigm.

The transitional self-organisation provides a definition that can be used in the investigation of intelligence within the EvE, with the aim of developing a Distributed Intelligence System (DIS) that can optimise the EvE. This directly impacts on future work scheduled to begin at month 18, specifically task S5 from WP6. It will also indirectly affect other tasks from WP6. The aim of WP6 is the research (S5), design (C42) and implementation (C53) of a DIS. Commenting as someone who works on the EvE architecture specification and will work on the DIS architecture specification, the more in-depth understanding of these systems that the research provides is considered more valuable than any specific algorithms.

To communicate with the business partners in the project, deliverable D6.3 will provide a non-technical summary of the research presented in deliverables D6.1 and D6.2.

6.5 Future Work

The EvE Digital Ecosystem model will be used as a platform for our future research. We have developed an understanding, as it was being created, of how the EvE Digital Ecosystem works from the organisational complexity research (deliverable D6.1[7]), and the application of the transitional self-organisation definition. Our future effort, as described in the task S5 description from WP6 of the new Technical Annex[18], will be to answer the following questions:

- Can intelligence optimise the evolutionary process?
- How does this intelligence interact with the ecosystem dynamics?
- Can the software components that are part of genetic selection be intelligent in themselves, as in an adaptive technology?

6.6 Summary

Regarding the objective to 'provide the adaptation mechanism inside the DBE'[14], the Evolutionary Environment Digital Ecosystem model is presented. The model provides a

distributed platform, which uses a novel variant of distributed evolutionary computing, to find the optimal combinations of services to meet user requests.

Regarding the objective to 'provide a performance measure for the self-organisation of the services into an application'[14], the Efficiency performance measure is presented. The Efficiency performance measure does 'give more meaning to the optimisation that is implicit in self-organisation'[14] as it provides a single summative value of the current organisational complexity, the probability of currently having the optimal solution, and is finally scaled by the average fitness. The scaling by the average fitness is performed as 'optimally self-organised service chains should also exhibit a high fitness'[14].

Regarding the objective 'control of self-organisation'[14], the relative simplicity of individuals within the EvE made the proposed selective breeding (clustering mechanism) ineffective. It was decided, in response to this, to construct a definition of transitional self-organisation to better understand the EvE. It was successfully applied to understand the EvE Digital Ecosystem model and its behaviour in more depth. This included the proposition that the mutation rate can be considered a 'temperature like control parameter for the self-organisation'[14]. We have not been able to achieve the control of self-organisation, contrary to the expectations raised by the deliverable title. However, in the transitional self-organisation we have found an effective understanding and quantification of the evolutionary optimisation. It was concluded that some kind of intelligence is required to achieve control. We plan to address this issue in the second half of the project in task S5 of WP6[18], which will investigate distributed intelligence in our Digital Ecosystem model.

Overall an insight has been achieved into where and how self-organisation occurs in the DBE, and how it can be quantified.

References

[1] C Adami and N Cerf. Physical complexity of symbolic sequences. *Physica D*, 137:62–69, 2000.

[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] A Agapie. Genetic algorithms: Minimal conditions for convergence. *j-LECT-NOTES-COMP-SCI*, 1363:183, 1998.

[4] A Agapie. Modelling genetic algorithms: From markov chains to dependence with complete connections. *PARALLEL PROBLEM SOLVING FROM NATURE - PPSN V*, 1498:3–12, 1998.

[5] G Basharin. On a statistical estimate for the entropy of a sequence of independent random variables. *Theory Probability and its Applications*, 4:333–336, 1959.

[6] W Beck, K Liem, and G Simpson. *LIFE: An Introduction to Biology*. HarperCollins Publishers Inc., 3 edition, 1991.

[7] G Briscoe. Self-organisation in multi-agent systems. *Digital Business Ecosystem*, Contract no 507953, 2004.

[8] G Briscoe. Evolutionary Environment Requirements (draft). *Internal*, April 2005. Available from: `http://www.iis.ee.ic.ac.uk/~g.briscoe/Eve/EveArchRequirements.pdf`.

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

[10] M Chli. Market efficiency of a digital business ecosystem. Available from: `http://www.iis.ee.ic.ac.uk/~maria/publications/DBESummary.pdf` [cited 04/05/05].

[11] M Chli, P De Wilde, J Goossenaerts, V Abramov, N Szirbik, L Correia, P Mariano, and R Ribeiro. Stability of multi-agent systems. In P Willett E Santos Jr, editor, *2003 IEEE International Conference on Systems, Man, and Cybernetics*, pages 551–556. Piscataway, NJ, 2003.

[12] A Corallo. Bml metamodel (draft) v2.0.1. *Internal*, 2004. Available from: `https://languages.digital-ecosystem.net/files/documents/18/310/BML_metamodel_2.0.1.zip`.

[13] A Corallo. Very simple bml v2. *Internal*, 2005. Available from: `https://languages.digital-ecosystem.net/files/documents/18/224/VerySimpleBml_v2.doc`.

[14] P Dini. Annex I - "description of work". *Internal*, (146-147), 2002. Available from: `https://dbe.digital-ecosystem.net/files/documents/8/6/file_6.dat?filename=DBE%20Annex%20I_ffinal.doc.gz`.

[15] P Dini. DBE science vision. *Internal*, March 2005.

[16] Agoston E. Eiben, E. H. L. Aarts, and K. M. Van Hee. Global convergence of genetic algorithms: A markov chain analysis. In Hans-Paul Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature*, pages 4–12, 1991.

[17] P Ferronato. D21.2 architecture scope document. *Internal*, 2005. Available from: `https://dbe.digital-ecosystem.net/files/documents/8/483/file_483.dat?filename=Del%5f%32%31%2e%32%5fDBE%20Core%20Architecture%20Scoping%20Document%5fRelease%5fA%2epdf`.

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

[19] David E. Goldberg and Philip Segrest. Finite markov chain analysis of genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 1–8, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

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

[21] W G Hale, J P Margham, and V A Saunders. *Dictionary of BIOLOGY*. HarperCollins Publishers Inc., 2 edition, 1995.

[22] W Härdle and L Simar. 11.3 cluster algorithms. Available from: `http://www.quantlet.com/mdstat/scripts/mva/htmlbook/mvahtmlnode80.html` [cited 14/10/04].

[23] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag New York, Inc., 2001.

[24] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[25] B Mobasher. Details of clustering algorithms. Available from: `http://maya.cs.depaul.edu/~classes/ds575/clustering/CL-alg-details.html` [cited 12/10/04].

[26] G Ochoa, I Harvey, and H Buxton. Genetic and evolutionary computation conference. In Morgan Kaufmann, editor, *On Recombination and Optimal Mutation Rates*, pages 488–495, San Francisco, CA, 1999.

[27] University of South California. Clustering algorithms. Available from: `http://www.cmb.usc.edu/cbmp/2001/Microarray/algorithms.htm` [cited 12/10/04].

[28] Petri Räsänen. Dbe target domains. *Internal*, 2004. Available from: `https://regions.digital-ecosystem.net/files/documents/26/134/file_134.dat?filename=Target%20Domains%20by%20TTC%20short%2eppt`.

[29] A Redmore and M Griffen. *Longman Reference Guides: Biology*. Longman Group Limited, 7th edition, 1994.

[30] A Reimann and W Ebeling. Ensemble-based control of evolutionary optimization algorithms. *Physical Review E*, 65(046106), 2002.

[31] T Romberg. Target domain meeting (summarised and extended minutes). *Internal*, 2004. Available from: `https://regions.digital-ecosystem.net/files/documents/26/132/TargetDomainsMinutes.doc`.

[32] Jonathan E. Rowe. D11.1 report on the flow of software components in a static network. *Internal*, 2005.

[33] Jonathan E. Rowe and Boris Mitavskiy. D8.1 report on evolution of high-level software components. *Internal*, 2005.

[34] Jonathan E. Rowe and Colin R. Reeves. *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[35] G Rudolph. Convergence of non-elitist strategies. In *First IEEE Conf on Evolutionary Computation*, volume 1, pages 63–66. Piscataway, NJ. IEEE Press, 1994.

[36] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[37] Lothar M. Schmitt, Chrystopher L. Nehaniv, and Robert H. Fujii. Linear analysis of genetic algorithms. *Theor. Comput. Sci.*, 200(1-2):101–134, 1998.

[38] STU. Report on fitness landscape. *Internal*, 2005.

[39] J Suzuki. A markov chain analysis on simple genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(655–659), 1995.

[40] P Tamayo, D Slonim, J Mesirov, Q Zhu, S Kitareewan, E Dmitrovsky, E Lander, and T Golub. Interpreting gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. In *Proceedings of the National Academy of Sciences*, volume 96, page 29072912, 1999.

[41] M Ward. Life offers lessons for business. Available from: `http://news.bbc.co.uk/2/hi/technology/3752725.stm` [cited 13/10/2004].

[42] P De Wilde, H Nwana, and L Lee. Stability, fairness and scalability of multi-agent systems. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 3:84–91, 1999.

[43] M Wooldridge. *Introduction to MultiAgent Systems*. John Wiley & Sons Inc, 2002.

A Interdisciplinary Dictionary

The key scientific terms used within this document will be explained here, in general terms and in terms of the Multi-Agent Systems model of the Evolutionary Environment. These terms when first mentioned in the main text, will be defined there and then, relative to their context. These definitions in context, and this dictionary should hopefully make the meaning of these terms clear.

The scientific definitions are given in green italics[29, 6], and the MAS definitions are given in red. It is hoped that this dictionary will complement, and can be added to the other efforts to create a cross-disciplinary glossary of terms within the DBE.

DNA
DeoxyriboNucleic Acid whose sequence encodes the genetic information of living organisms, provides two primary functions. The holder of virtually all information in inheritance, and the controller of protein synthesis.
The equivalent is the process descriptions associated with each agent.

ecosystem
An ecosystem is a natural unit made up of living and non-living components whose interactions give rise to a stable, self-perpetuating system. It is made up of one of more communities of organisms, consisting of populations existing in their microhabitats.

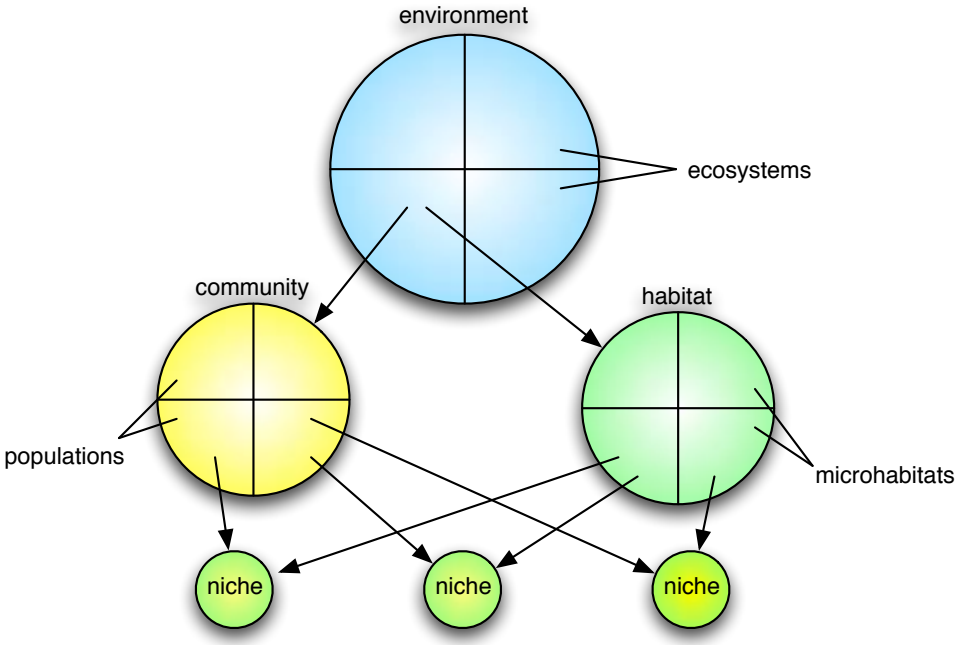


Figure A.1: Ecosystem

The agent-stations are equivalent to the habitats in Figure A.1, and the population objects are equivalent to the populations in Figure A.1. The evolving population, under the influence of a fitness function (selection pressure), is equivalent to the niche in Figure A.1.

epistasis
An interaction between genes in which one gene masks or modifies the effects of another gene.
This is equivalent to the probability of finding a specific agent at a specific site in an agent-chain, being affected by the presence of other agents at other sites within the agent-chain.

evolutionary stable strategies (ESS)
It is based on the concept of a population of organisms, playing a certain strategy, and the mutant form of a gene that causes organisms to adopt a different strategy cannot invade the population, but will instead be selected out by natural selection.
This is equivalent to a population of agent-chains that have found all the global optima in the search space (fitness landscape). So any mutant agent-chains invading the population will be sub-optimal, and therefore removed by the selection pressure (fitness function).

fitness
Measure of the ability to produce mature offspring in the next generation which will also be able to reproduce.
In evolutionary computing the fitness is how well it performs relative to the desired goal.

gene
A small length of the total DNA sequence of an individual, to which a specific function can be assigned.
The equivalent unit is the process description of an agent.

gene pool
All the genes in a population or species, and therefore its genetic constitution.
This is the set of agents registered at an agent station.

genotype
The genetic makeup of an organism, independent of it physical or functional traits.
This is equivalent to the process descriptions of agents and agent-chains.

migration
This is the movement of individuals between different habitats (areas), often because of seasonal variations.
This is equivalent to the movement of mobile agents from one agent-station to another.

mutation
A permanent transmissible change in genetic material (DNA) of an individual.
In its simplest form, this is equivalent to replacing one of the agents in an agent-chain.

organism
An organism(phenotype) is an individual that is capable of reproducing itself and existing as a separate entity.
This is equivalent to the instantiation of an agent or agent-chain, which is generated by evolutionary computing, in response to a user request.

population
This is all the members of a species that occupies a particular area at a given time. Statistically, the group of entities under study.
This is the population of agent-chains created, using Evolutionary Computing at an agent station, to generate a solution to a user request.

selection pressure/natural selection
Describes the sum total of the forces acting upon a population, resulting in genetic change and natural selection. Those organisms best fitted to survive the selection pressures operating upon them will pass on their biological fitness to their progeny through the inheritance process.
This is equivalent to the fitness function in the MAS model, which assigns fitness to the agent-chains of the evolving population, and thereby determines which will survive.

species
A series of populations within which significant gene flow occurs, so groups of organisms showing a very similar genetic makeup.
Pure clusters are an example of different species within the MAS model, as no agents are exchanged (no gene flow occurs) between them.

wild-type
This is the most frequent genotype within a population.
This is equivalent to the most frequent agent-chain within the population, which will potentially be instantiated at an agent-station in response to a user request.

B EvE Digital Ecosystem Information[8]

We will define the behaviour of the components within the Evolutionary Environment (EvE), from the scientific point of view but tempered and constrained by computational realities. It will provide the behaviour of the components within the system, relative to one another and the expected SME users. The EvE will exist within the Execution Environment. The Habitat will be a core service located upon the DBE servent. All other EvE components will exist within this Habitat service.

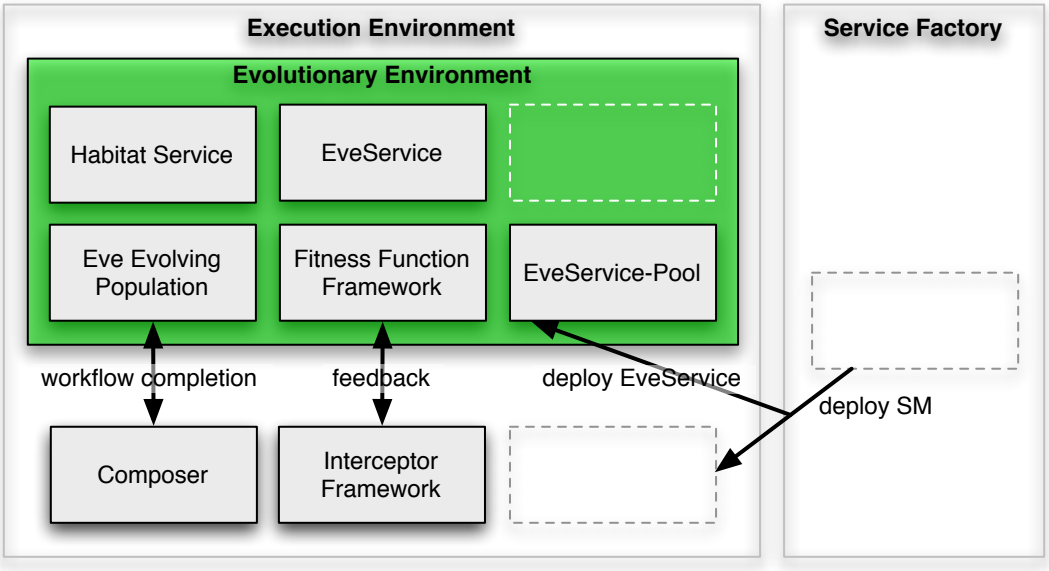


Figure B.1: Evolutionary Environment Components

B.1 Evolutionary Environment Service (Agent)

B.1.1 Relation to DBE Service

An EvEservice is an individual within the EvE, and its DNA is the Service Manifest (SM) it references. An EvEservice is primarily a reference to a Service Manifest, and many EvEservices can point to the same Service Manifest.

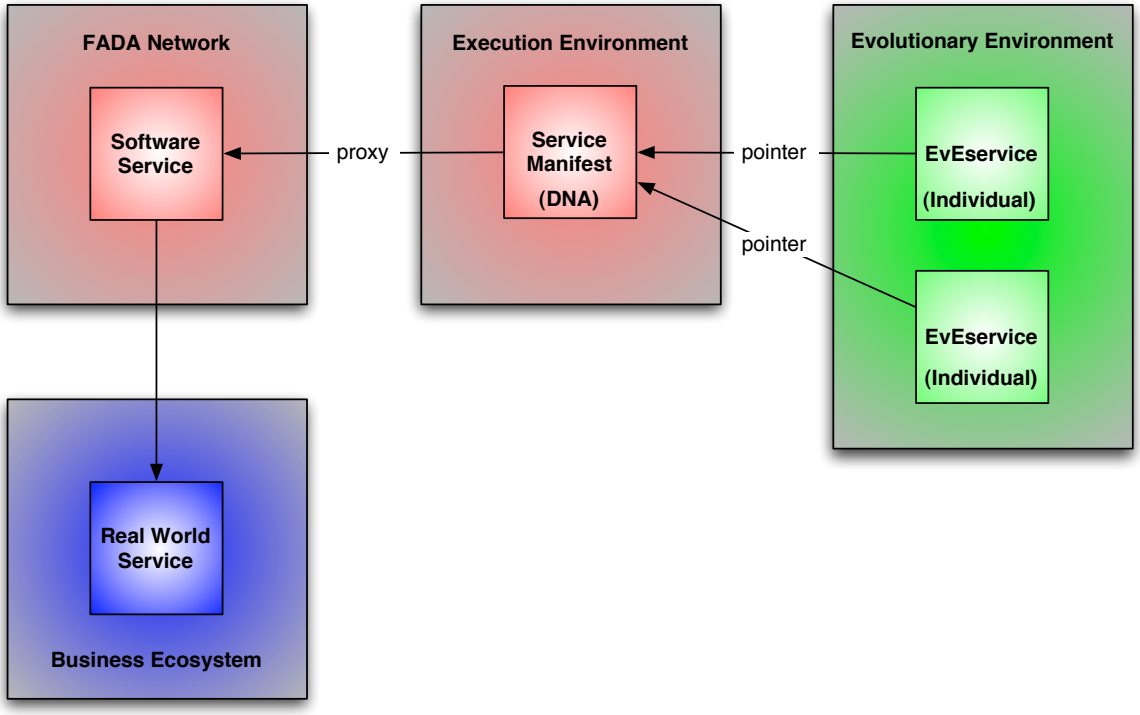


Figure B.2: EvEservice relation to DBE service

A service in the real world, such as selling books, is represented by a software service on a computer of the SME. The service is listed on the FADA network via a proxy reference. The Service Manifest describes the service in the Business Modelling Language (BML) and is stored in the distributed Semantic Registry of the Execution Environment. It also includes a reference to the proxy which activates the software service.

B.1.2 Life Cycle

The life cycle of an EvEservice exists within the life cycle of the DBEService (Service Manifest) it represents. The EvEservice life cycle, with the major interactions within the DBE life cycle, is shown in Figure B.3.

The first stage of the service life cycle is when a user creates a Service Manifest using the Service Factory. Once created the Service Manifest is deployed in the distributed Semantic Registry, and then an EvEservice with the Service Manifest ID is created in the user's home Habitat. The EvEservice is then copied (migrated) to any Habitat connected to the home Habitat, conditionally on the probabilities associated with the connections. The migrated EvEservice is now available in Habitats where it is potentially useful.

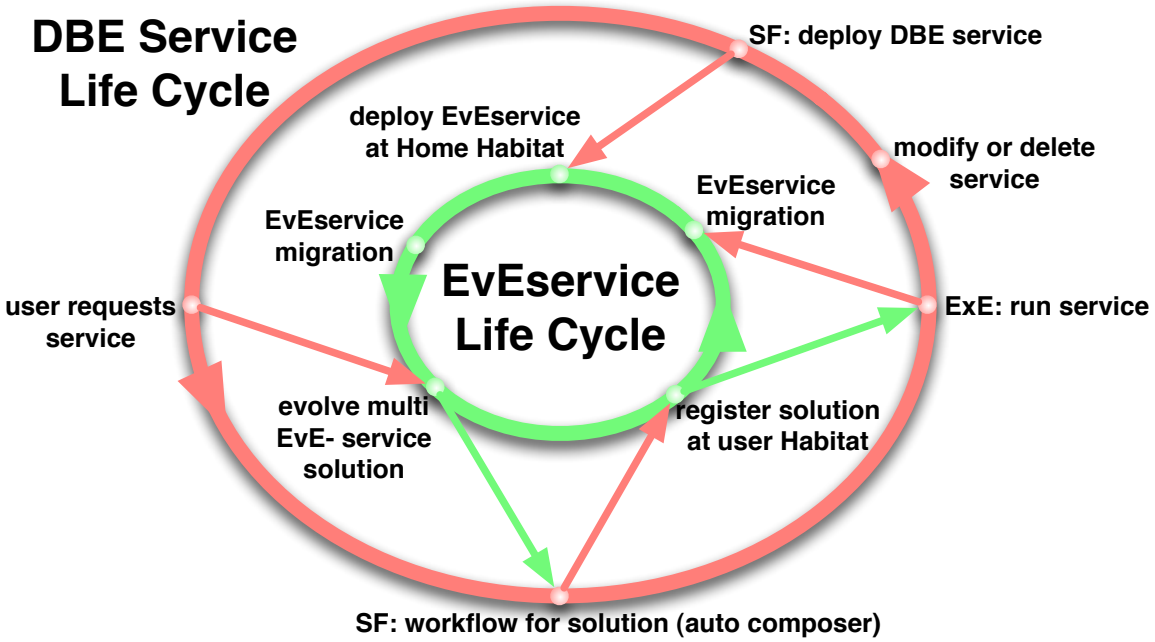


Figure B.3: EvEservice Life Cycle

The second stage of the life cycle is when the EvEservice is chosen by the evolutionary optimisation in response to a user request. When the user uses the EvEservice, it is then attempted to copy (migrate) it to every Habitat connected to the Habitat where the EvEservice is currently located. It is 'attempted' as there is a probability associated with the connection when attempting a migration, similarly to the Island Model shown in Figure 3.

If an EvEservice is not used, at the Habitat where it is currently located, after several user requests it risks deletion. It will have a number of escape migrations, and when threatened with deletion it will relocate to a connected Habitat. After the escape migrations finish, it is then deleted. If a Service Manifest is removed from the distributed Semantic Registry, then all EvEservices pointing to it must also be deleted. It is suggested that this is done passively rather than proactively. So when the EvEservice is being used and when resolution of the pointer is attempted for retrieving the Service Manifest fails, then the EvEservice should be deleted.

B.1.3 Migration and Usage History

Each EvEservice requires a record of its migration and usage through its life span. This migration and usage history are essential to the self-management of the EvE. It is simply a record of the Habitats that the EvEservice has migrated through, and its use at these Habitats. Also which other EvEservices it was used with and migrated with.

B.1.4 EvEservice-Pool

From this point on we will use the name EvEservice-Pool, rather than the name Agent Pool which can be seen in Figure 6. This is the set of EvEservices (agents) registered at the Habitat (Agent Station). The EvEservice-Pool is updated with new EvEservices

that arrive, from other Habitats, and is the location from which EvEservices are deleted when they have not been used over many user requests. The most important aspect of the EvEservice-Pool is that the multi-EvEservice solutions found using the evolutionary mechanism are stored in the EvEservice-Pool. They are then available later for reuse in bootstrapping that same evolutionary mechanism in the future.

B.2 Evolving Populations

An Evolving Population object is instantiated in response to a user request. It uses an evolutionary optimisation technique to generate the optimal combination of available EvEservices that fulfil the user request.

B.2.1 Aggregated EvEservice Structure

The EvEservice is the base unit for the evolutionary mechanism, which means that the evolutionary process optimises the combination of EvEservices to a user request. It does not change the EvEservices themselves. A question from the very beginning, even before the project started, was the structure of aggregated services. Initially 'chain' and 'tree' structures where considered, since then 'sets' and 'workflows' have been proposed. The 'workflow', which is the most sophisticated, has not only been proposed but implemented by the computing stream. The 'workflow', which is the most sophisticated, has not only been proposed but implemented by the computing stream.

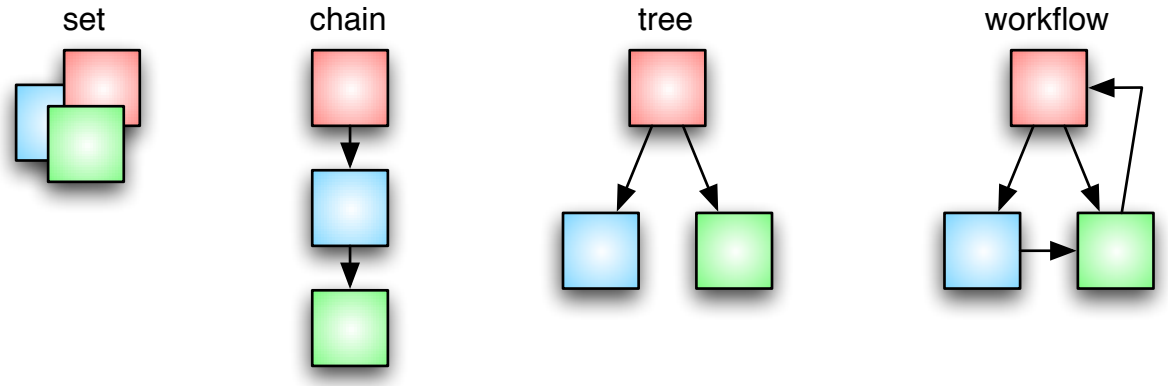


Figure B.4: Possible Structures of Aggregated EvEservices

Ultimately, we would like the aggregated structure of EvEservices to be 'workflows', however for the present they will be 'sets' as the production of 'workflows' cannot be automated. This is caused by the EvE's dependency on the Automatic Composer, which is specifically for generating the 'glue-logic' of 'workflows' between services. The Automatic Composer faces a very significant challenge and the current approach it to be semi-automated, requiring some minimal user input.

The structure of aggregated services mostly affects the Fitness Function as it has to have an understanding of the structure for the comparisons to work. As the 'workflow' can represent the other structures, one way to progress would be to start with 'sets', then progress to 'chains', followed by 'trees', and then finally 'workflows'. Therefore, supporting functionality in increasing complexity.

B.2.2 Population Properties

An evolving population of EvEservice(-Set)s is the key component for supporting evolution within the DBE. Each evolving population is an artificial environment, having everything for evolution - a population, replication, mutation and a selection pressure. The selection pressure guides the evolution of the population towards the desired solution (EvEservice-Set) by means of a Fitness Function. Over many generations, the optimal solution (EvEservice-Set) will be generated. Once the solution is found it is stored in the EvEservice-Pool of the user's Habitat, and then a copy is migrated to every connected Habitat conditionally on the connection probabilities.

It is suggested that an Evolutionary Computing library is used to provide the core evolutionary process, and then the appropriate interfacing completed for connection to the EvEservices and the Fitness Function. Therefore, the details of the replication rate, death rate, and population size can be managed by the library.

As many generations must pass in the evolutionary process to generate an optimal solution, the possibility for bootstrapping the process using the 'Best Guess Solution' from the 'Recommender' should be considered.

B.2.3 Fitness Function

As stated previously the Fitness Function will provide the Selection Pressure required to evolve a population of EvEservice-Sets to the user request. The upcoming deliverable D9.1 'Report on Fitness Landscape' will deal with the task of creating a Fitness Function. So I will deal with the issue of population diversity controlled by the Fitness Function.

A situation that often arises in evolutionary computation is that there may be more than one way to meet the fitness criteria (that is, there are multiple global optima). There is a theorem in theoretical biology that says that, in this case, exactly one solution will survive - the alternative solutions will die out. If we want to present all possible solutions to the user, then we have to prevent this happening. We do that by creating a separate "niche" around each optimal solution. This can be done in a number of ways, which are usually based on the idea of "fitness sharing". That is, if more than one solution in the population is at a particular optimum, they have to share the fitness associated with that point. That makes it less likely that they will all stay there - many of the individuals will move off to other optimum. Using fitness sharing like this, the population will spread out and occupy all the global optima. The more distinct solutions are; the more such a mechanism is required. Generally, more generations will be required to find an optimal solution, but ultimately multiple optimal solutions can be found.[9]

B.2.4 Runtime Switching

The potential to switch services at runtime to manage service failure is one of the long standing challenges of the computing stream. The EvE can contribute to this goal by providing the alternative services for the switching. Basically, this requires that one service can functionally replace another in a DbeService workflow.

Assuming we have an EvEservice-Pool of 26 agents, labelled from A to Z, and assuming X can functionally replace Y. If either is used in the composition of an individual within an evolving population, so will the other. Assuming no other services can functionally replace either, then potentially half the evolving population will consist of individuals with X, and the other half of individuals with Y. Therefore, available alternatives to services within a solution can be provided to the Execution Environment for the dynamic runtime switching.

B.2.5 Ownership

The actual issue is not 'ownership', but 'responsibility' when running a multi-service solution which has been composed by a third party. Although the participants in the EvE have discussed this issue, it very much depends on the general information sharing within the DBE, and the DBE legal framework. Nevertheless, we will endeavour to follow basic Open Source Software principles. If users are willing to share their solutions (EvEservice-Sets and DbeService-Workflows), it will be automatically without liability. The users choosing to make use of existing solutions are responsible for their running.

B.3 Habitats

The Habitat is the key component for supporting the ecosystem concept within the DBE. Each SME user has a Habitat, which contains an EvEservice-Pool with services of potential use to the SME. New solutions evolved from the EvEservice-Pool in response to user requests are also stored within the EvEservice-Pool, as the combination of services has value as well as the atomic services. The Habitat also provides a place for service migration to occur, as the Habitats are interconnected with one another. The union of the Habitats creates the EvE Digital Ecosystem.

The connections between the Habitats will be self-managed, and the mechanism for this will be discussed shortly. The initial connections are based on the SME profiling. An SME's Habitat connections will be copied from an SME with a similar profile. If no similar SME is found then the user will be asked to find a similar SME. A Habitat may become disconnected if placed in the wrong cluster, to prevent this a fail-safe mechanism will be required.

B.3.1 EvEservice Migration

The connectivity in the Habitats will be parallel to the 'opportunity spaces' (business sector and cross business sector interaction) that exist within the SME user base. The network will find the optimal topology based on the rules of connectivity provided. The target topology is a 'caveman' network in which there are 'caves' of high connectivity.

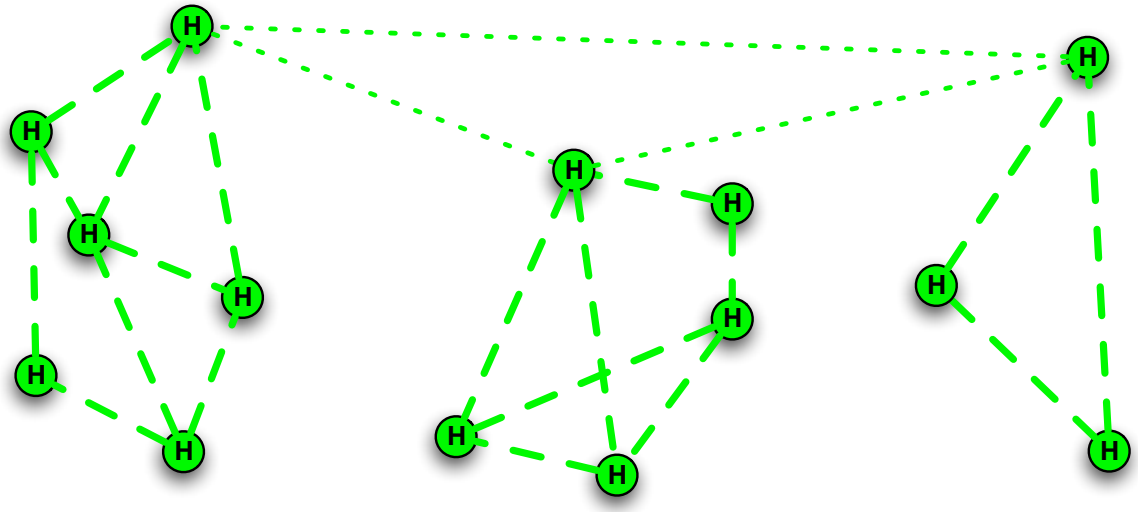


Figure B.5: Habitat Network - Caveman Topology

These 'caves' are the Habitat clusters which represent the opportunity spaces within the SME user base. The interaction and connectivity are the migration of EvEservices from one Habitat to another. Each connection between the Habitats is bi-directional, and there is a probability associated with moving in either direction across the connection (like the connections shown in Figure 3). The connection probabilities affect the migration of EvEservices and are updated by the success or failure of the migration.

The migration of an EvEservice is triggered by it being deployed into its Home Habitat or when it is used by a user at any Habitat. When migration occurs, depending on the probabilities associated with the connections; an exact copy of the EvEservice is made at the connected Habitat. The copy of the EvEservice is identical, then its migration history is updated which will differentiate it from the original.

The success of the migration, the **migration feedback**, is determined by the use of EvEservices at the Habitats to which they migrate. When a solution (EvEservice-Set) is found to a user request, then the individual EvEservice migration histories can be used to determine where the used EvEservice has come from and update the connection probabilities. The challenge is in managing the 'feedback' to the connection probabilities for migrating EvEservice-Sets, because for an EvEservice-Set the value is within the combination. This benefit of where the combination or subsets of the combination were created needs to be passed on to the connection probabilities.

The **escape range** is the number of escape migrations possible upon the risk of death. If an EvEservice migrates to a Habitat and is not used after several user requests, then it will have the opportunity to migrate (moved not copied) randomly to another Habitat. After this happens several times the EvEservice will be deleted (die). The escape range is a parameter that will be better determined by future work from the partners and practical experience. Ideally the escape range will be 'tuned' to the size of the Habitat cluster it exists within. This is not as simple as it may sound, as the view of a cluster very much depends on the point of view of the Habitat. Temporary values will be suggested later, on the premise of updating them later or creating a definition that is dynamically responsive to the state of the system.

B.3.2 Habitat Clustering

There will be hundreds of Habitats at least, and potentially three or more times the number of populations at any one time. There will then be thousands of EvEservice(-Set)s. Each Habitat is localised to the SME it was created for, and therefore the EvEservice(-Set)s within it will have properties which match the properties of the SME.

This infrastructure allows for the spontaneous formation of communities via Habitat clustering. Many successful service migrations will reinforce Habitat connections increasing the probability of service migration. If a successful migration occurs through multi-hop migration, then a new link can be formed between those Habitats. Unsuccessful migrations will lead to connections (migration probabilities) decreasing, until finally the connection is closed.

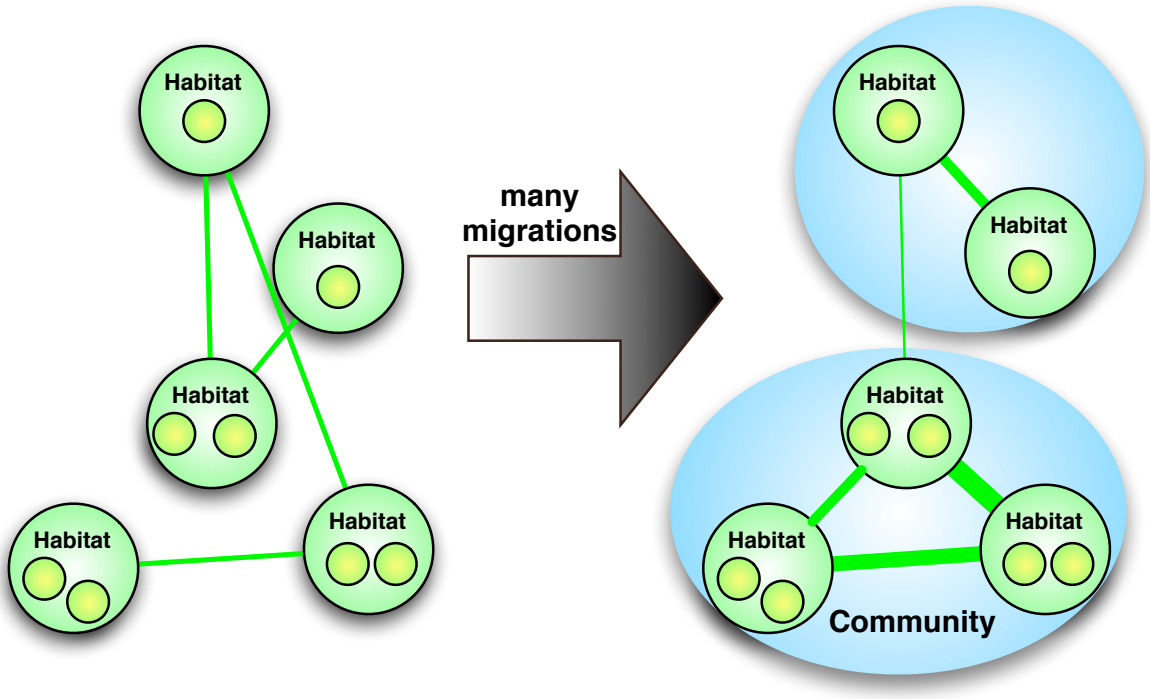


Figure B.6: Habitat Clustering - Community Formation

The clustering/community formation is like joining a club; you can be a member of more than one. In the same way Habitats can be clustered into more than one community. The clustering will lead to communities clustered over language, opportunity spaces, nationality, geography, etc.

B.3.3 Distributed Evolutionary Computing

The 'distributed' evolutionary computing component comes from evolving populations being created in response to 'similar' requests. So whereas in Figure 3 there are multiple evolving populations in response to one request, here there will be multiple evolving populations in response to similar requests. This is shown in Figure B.7 where the colour of the evolving populations indicates similarity in the requests being managed.

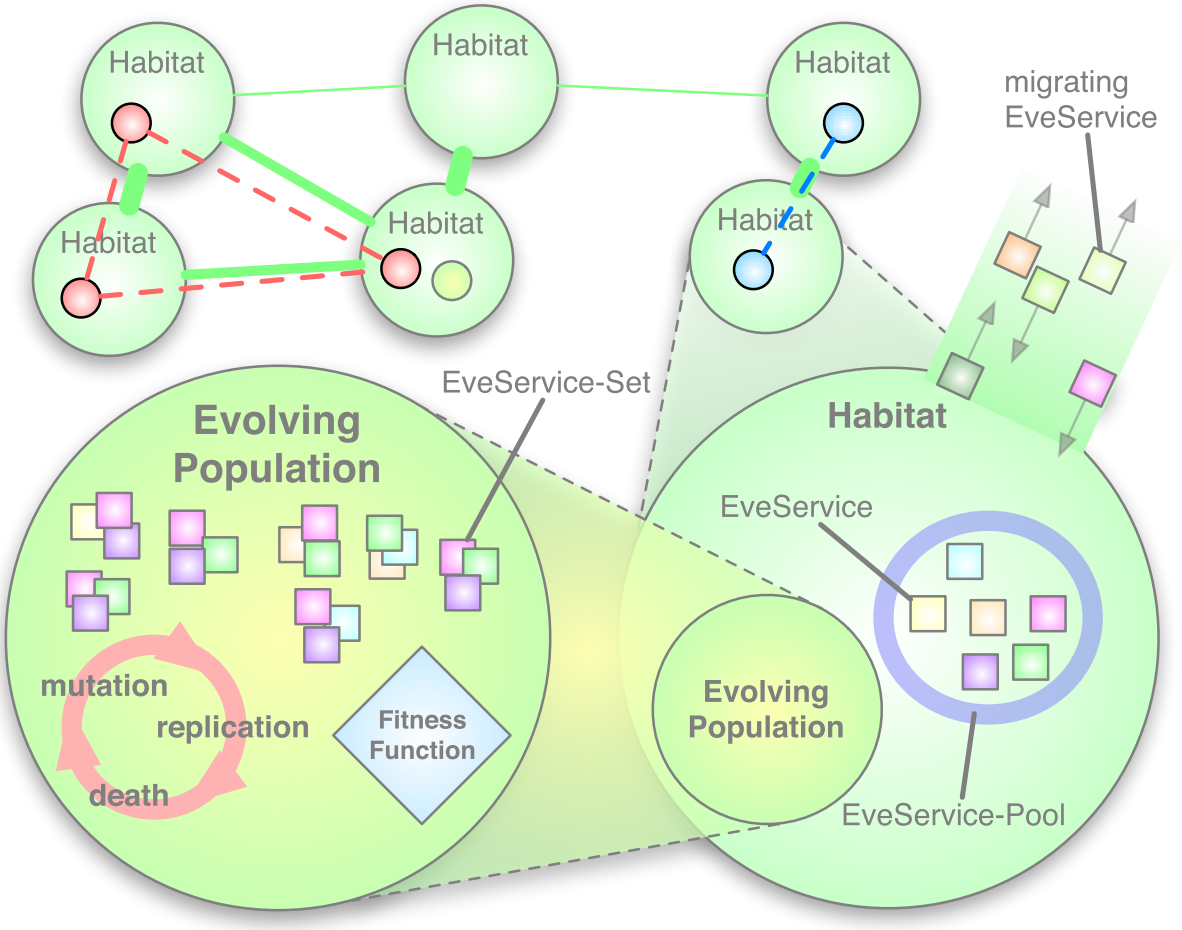


Figure B.7: Distributed Evolutionary Computing in the EvE Digital Ecosystem

This is based on the premise that once the DBE has sufficient 'critical mass', in terms of the SME user base, there will then be similarity in the requests of SMEs within the same opportunity space (business sector).

B.3.4 Fail-Safe Mechanisms

A Habitat may become disconnected or located in the wrong cluster. This will most likely occur when a Habitat is initially connected to the wrong cluster to begin with, or when the facilities provided are not used by the user. In both scenarios, the user should be allowed to reconnect. It may also occur if a user changes its business portfolio. If the system fails to connect the Habitat to the correct cluster in the first place, then it is doubtful it would succeed a second time. This is why it is better to allow the user to perform the reconnection. It is expected that there will be a DBE local business directory for the region in which the SME is located.

If the EvEservice-Pool is insufficient to find even a single solution, then the EvE is probably incapable of finding an optimal solution even in the long-term over several requests. In this scenario, a Habitat would be located in the wrong cluster. This problem will be managed by boot-strapping the evolving populations with the 'best guess solution' from the Recommender. If this 'Best Guess Solution' from the Recommender cannot be

provided then, an alternative method will need to be found to introduce EvEservices to Habitats when it is not capable of providing a solution.

B.3.5 Relation to DBE Network Architecture

Consideration has been given to integrating the EvE Digital Ecosystem with the DBE core architecture by making each Habitat a core service. For each SME, whether producer or consumer, their Habitat service will be located on their DBE servent application.

Figure B.8: EvE and ExE : Habitat Service and the DBE Servent

The evolving populations, instantiated in response to a user request, are contained within the Habitats. A Habitat service is created when an SME joins the DBE. The connections between the EvE Habitats in Figure B.8 and Figure B.9 represent the bi-directional sharing of EvEservices between the Habitats.

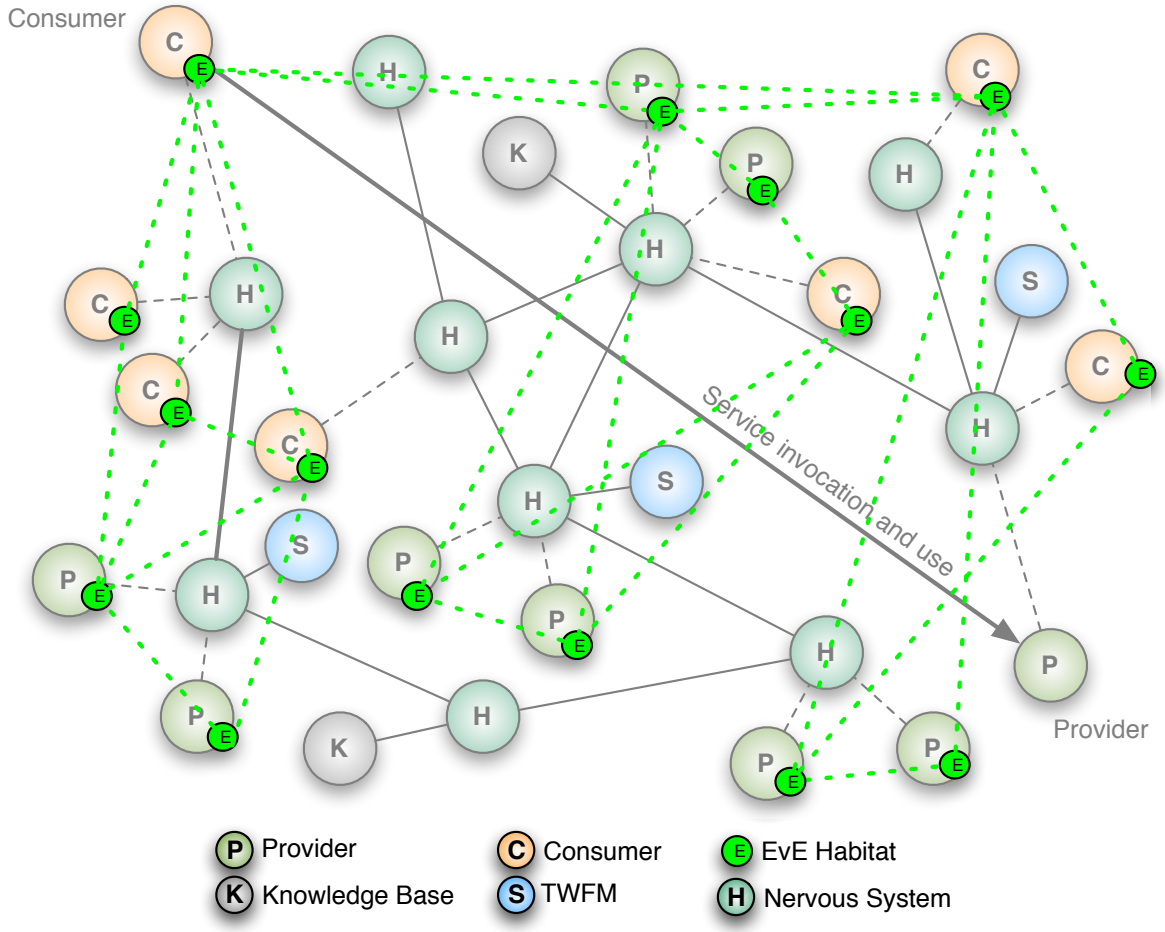


Figure B.9: EvE Digital Ecosystem relation to DBE Network Architecture

The Habitats can be run anywhere, just as in grid computing where the user is blind to where the processing occurs. A distributed processing arrangement, where supernode DBE servents handle several Habitats for several SMEs should be possible. The EvE will be distributed within the DBE network as shown in Figure B.9.

B.3.6 User Interaction

Although we will enforce receiving of shared EvEservices, we cannot and should not force users to share the solutions that are evolved for them. We hope they will share, and this will be the default setting, as it will encourage EvE system efficiency and market efficiency. So user interaction is required for the user to set and update their sharing preferences.

B.4 EvE Architecture Behavioural Requirements

This subsection will specify the interactions between the components within the EvE. The aim here is to provide the behaviour of the EvE described earlier in more detail. This information is intended to supplement any master use cases that already exist.

B.4.1 New User

When a new user joins the DBE, their Habitat needs to be created and most importantly connected to the correct cluster(s). The best way to achieve this is to find an SME with a similar profile or ask the user to identify an SME within the same business sector. The automated approach is preferred, but may not always be possible.

The scenario presented in Figure B.10 requires that the SME found with a similar profile has sharing enabled on their Habitat. Sharing will be enabled by default and SME users will be given the chance to turn it off.

B.4.2 Deploy Service

When a user deploys a service to the DBE, it must also be deployed as an EvEservice to their Habitat. The SM-ID of the new DbeService is required to create the new EvEservice. It is then copied to the EvEservice-Pool of the user's Habitat. From there the migration of the EvEservice occurs, which involves migrating (copying) the EvEservice probabilistically to all the connected Habitats. The copying of an EvEservice to a connected Habitat depends on the associated migration probability. If the probability were one, then it would definitely be sent. This is all shown in Figure B.11.

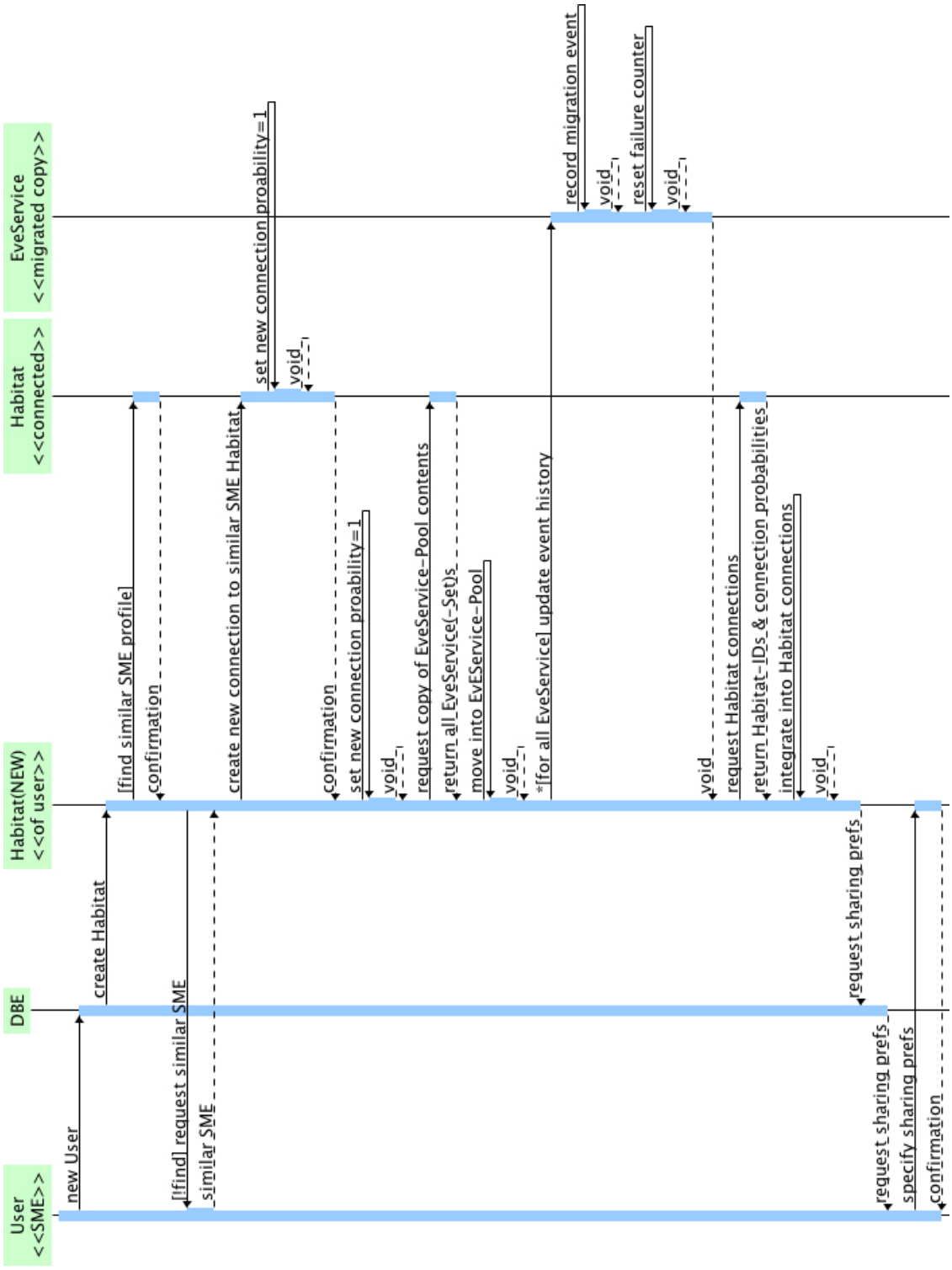


Figure B.10: New User

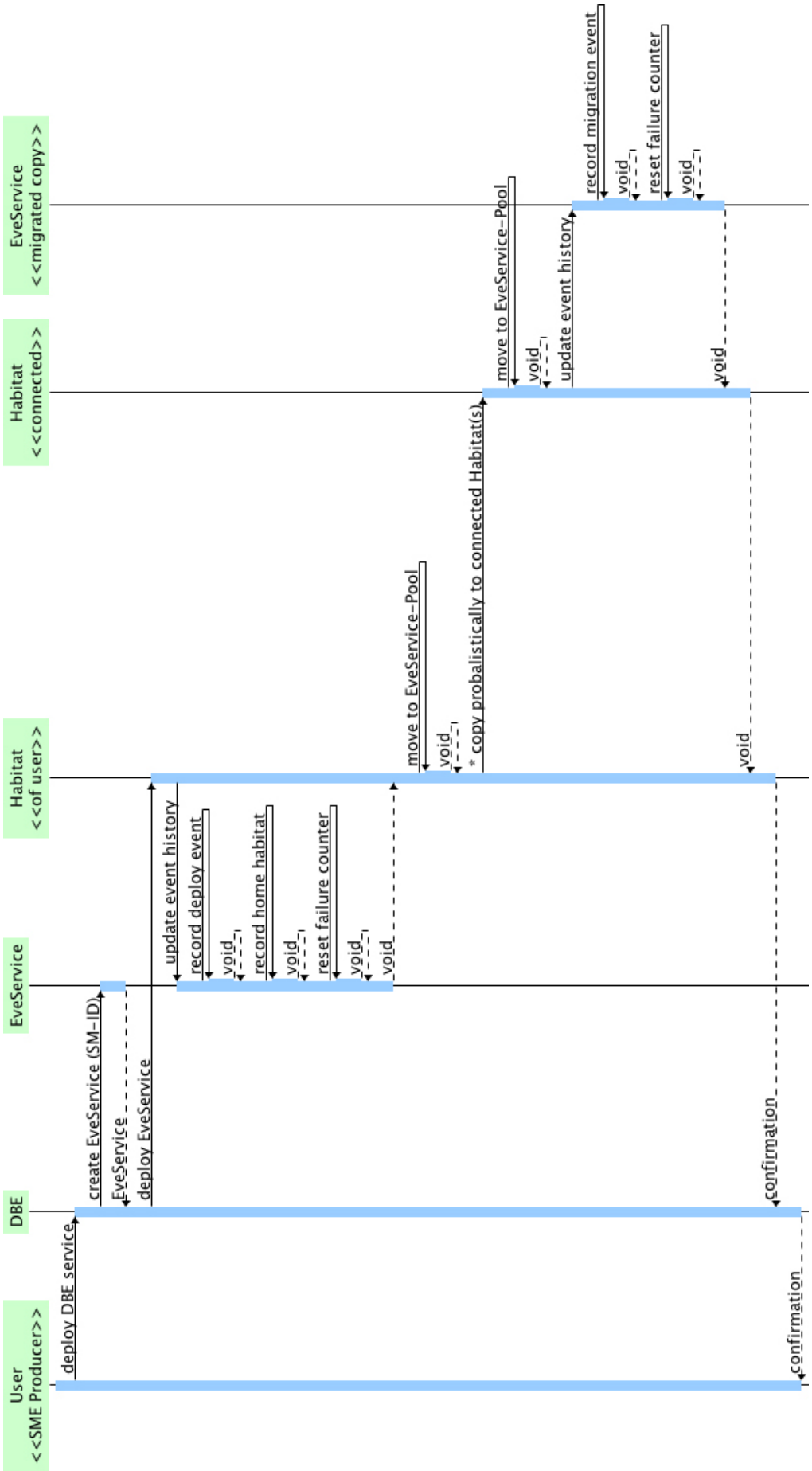


Figure B.11: Deploy Service

B.4.3 User Request for Services

When a user requests a service(s) from the DBE, the request is passed to their Habitat where it is used to create an evolving population to find the optimal solution. This involves generating a fitness function from the request and then allowing an evolutionary process of mutation, replication and death to occur until the optimal solution is found.

If a user chooses to execute the EvEservice(-Set) optimal solution they will have to complete the workflow using the Composer. The execution of a service is the trigger for the EvEservice(-Set) migration and Habitat clustering mechanism.

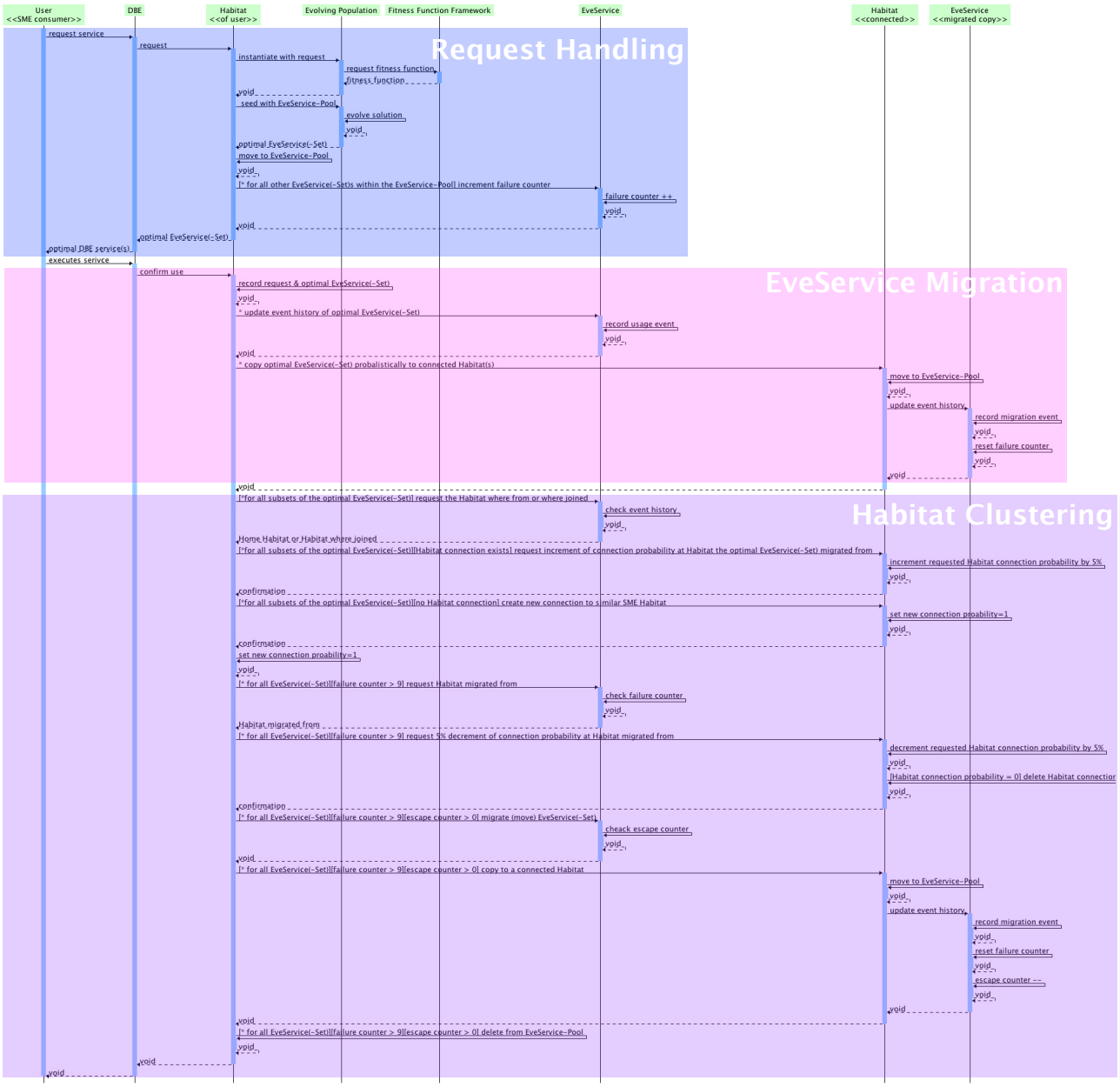


Figure B.12: User Request

Figure B.12 has been significantly reduced in size. If viewing the PDF then please zoom in, the image is of sufficient quality for 400% magnification.

The numerical parameters which specify the conditionals in Figure B.12 are tentatively proposed. They will be better informed by future work from the partners and practical experience. Ideally in a future version the values can be determined dynamically depending on the user behaviour.

A Habitat can become totally disconnected under certain conditions. One condition being that the EvEservice(-Set)s within the EvEservice-Pool consistently fail to satisfy user requests. Another condition being when the user chooses not to share the EvEservice-Sets they evolve, or their shared services are undesirable. These scenarios can be because the Habitat is located within the wrong cluster. If located within the wrong cluster the user can be asked to join another Habitat cluster. This assumes there is sufficient critical mass in the SME user base such that there is a viable Habitat cluster to join.

The availability of what is known as the Best Guess Solution (BGS) from the Recommender is currently not clear. Using the BGS is a relatively simple way to implement a fail-safe against a Habitat being located in the wrong cluster and/or lacking sufficient diversity. If the BGS or any part of it where found in the optimal EvEservice(-Set) solution in response to a user request, then the Habitat connections can be updated to include the Habitat(s) of the BGS. If ultimately the BGS is not available, then an alternative method to 'top-up' the EvEservice-Pool and correct Habitat cluster membership will have to be found.

B.5 Knowledge Ideas

Once the EvE has been created, it will provide a platform to support some of the knowledge sharing mechanisms envisaged from the beginning of the project.

B.5.1 New Service Notifications

Informing users of new services that may be of use to them would be extremely beneficial. This is very simple to state, but once the scale of the DBE is taken into account it can prove difficult, because the number of services in the DBE is potentially thousands. However, a positive side effect of how the EvE works is that this functionality is provided by the Habitats. The EvEservice-Pool is a subset of the useful services available, and is updated when new services of interest become available. All that is required is way to inform the users. Ideally we should provide a way for users to control how they are informed. For example, by pop-up, by email, daily, weekly, etc.

B.5.2 Strategic Business Partnerships

In terms of the DBE, specifically the EvE, a strategic business partnership means sharing the solutions evolved in your Habitat with others. The EvE will be set by default to share solutions as it sees fit, with potentially anyone. The users are of course free to turn sharing off. However, users could be given the option to share more proactively. Users may not share generally, but will share with their partners. This will benefit the SME users who enter partnerships and the overall market efficiency.[10]

B.5.3 Evolving Futures Market

Each DbeService consists of a remote executable component and a descriptive component. The descriptive component acting as a guarantee of behaviour. The evolutionary process only requires the existence of the guarantees to work. The actual underlying code or service only comes into play once the whole package has been assembled. There is therefore no strict requirement for the underlying code to exist until this stage. That is, evolution could happen entirely at the level of the specifications. Software component providers would only need to supply the code once there is a demand for it. That is, when one of their specifications (or guarantees) has been used in the construction of a software solution which meets a user's requirements. The whole system then becomes more like an evolving futures market.[9]

B.5.4 Potential Markets

In the early stages of the project, it was suggested that unfulfilled requests for services should be sent to possible suppliers. This is somewhat the reverse of the future markets idea. This has always struck me as problematic since its inception, because how is a provider to decide to devote resources into creating a new service(s) without any guarantee of payment.

The EvE/DBE could help mediate/automate business propositions from the SME consumers to the SME providers. Imagine an SME provider getting several dozen unfulfilled requests for the same service they are able to provide. I should imagine they could easily see the huge potential of offering the desired service(s). It would even be possible to have a virtual advertising board where all the requests for desired services (unfulfilled requests) are listed so that SME providers can see the potential markets (niches) to enter. Market niches will be more easily identified and much easier to target, as the DBE will show the demand for specific services and a pathway to providing it. This is either the creation or the starting point for networked organisations that can adapt quickly to new markets, supported by the DBE.

A more sophisticated version using a two stage evolutionary process could be created. It would be used to present SME providers with suggestions of modifications to their existing services (cheaper and easier to produce than entirely new services), with an estimation of the demand for such modified services with a list of potential customers. In the first stage the evolutionary process will attempt to construct a solution out of existing services, as before. If this fails to find a solution above a user specified threshold then the second stage evolutionary process will start. The second stage will take the population being evolved to find a solution and evolve the services themselves (mutate them), rather than just the combination of services. Once the optimal service chain has been created then the non-existing service specifications (which are mutated from existing service specifications) can be sent to the SME providers who provided the original services. The value of this approach is that it reduces the risk for the SME providers, because instead of creating new services they can modify their existing services.

C Definition of Organisational Complexity

A summary is provided for the organisational complexity presented in deliverable D6.1[7], which was constructed by extending Physical complexity to variable length populations of agent-chains.

C.1 Origins of Physical Complexity

Physical complexity was born from the need to determine the proportion of information in sequences of DNA. It has long been established that the length of DNA sequences is an unsatisfactory measure of the information content, as they contain significant redundancy. So the information contained is not directly proportional to the length[2]. Understanding the DNA requires knowing the environment (context) in which it exists.

The purpose of DNA in the context of life may initially sound clear-cut, as DNA is considered to be the language of life. However, consider viruses which can consist of only a strand of DNA covered by a protein coating. Scientists are still split on whether viruses are alive or not. The only function viruses perform is to replicate, not showing any of the other commonly accepted signs of life in more complex organisms, such as respiration, growth, etc. We shall consider viruses to be alive and accept that their only function is to replicate. The process of replication requires resources, energy and matter, to be harvested. Viruses are the simplest form of life known. More complex forms of life have evolved far more complex, specialised, specific and effective ways to acquire the necessary energy and matter for replication.

So consider that for any individual the environment represents the problem of extracting energy for replication. Then the DNA sequence of an individual represents a solution to this problem. This hopefully alleviates the misconception that the DNA of individuals encodes their environment; it does not; it encodes a solution to extract energy and matter from their environment for its replication. Furthermore, the individual DNA solution is not a simple inverse of the 'problem' that the environment represents.

Even with this understanding, the problem remains of the need to define the environment, to be able to distinguish the information from the redundancy in the solution. This situation is resolved in Physical complexity measure by analysing a group of solutions to the same problem. The consistency between the different solutions shows the information, and the differences are the redundancy. Entropy is a measure of disorder, as it measures the number of states a system can access. A large number of accessible states is usually associated with disorder. So entropy is used to determine the redundancy from the information, in a population of solutions. The measure therefore provides a context-relative measure of organisational complexity, by measuring the population, which relieves us of the need to define the context (environment).

C.2 Definition of Physical Complexity

Physical complexity is derived[1] from the notion of conditional complexity defined by Kolmogorov, which is different from traditional Kolmogorov complexity. It states that the determination of complexity of a sequence is conditional on the environment in which the sequence is interpreted. The complexity C of a population of sequences is

$$C = \ell - \sum_{i=1}^{\ell} H(i), \tag{C.1}$$

where ℓ is the maximal entropy (length) of the population minus the sum over the length ℓ of the per-site entropies. The per-site entropy is

$$H(i) = -\sum_{d \in D} p_d(i) \log_{|D|} p_d(i), \tag{C.2}$$

where ℓ is the length, D is the alphabet of characters found in the sequences, the site i varies between $1 \leq i \leq \ell$ and the $p_d(i)$ is the probability that at a site i (in the sequences) takes on character d from the alphabet D, rather than any other character. So the sum of the $p_d(i)$ probabilities equals one, $\sum_{d \in D} p_d(i) = 1$. Taking the log to the base $|D|$ results in $H(i)$ ranging between 0 and 1.

The equivalence of the maximum complexity to the length matches the intuitive understanding that, if a population of sequences of length ℓ has no redundancy, then their complexity is their length ℓ.

If G represents the set of all possible genotypes constructed from an alphabet D and of length ℓ, then

$$|G| = |D|^{\ell}, \tag{C.3}$$

where the size(cardinality) of G is equal to the size of the alphabet $|D|$ raised to the length ℓ. For the complexity measure to be accurate, a population sample size of $|D|^{\ell}$ is suggested to minimise the error[1, 5]. This quantity can be computationally unfeasible. For practical applications, Adami[2] chooses a population size of 3600 for an alphabet of size twenty eight, $|D| = 28$, and a length of one hundred, $\ell = 100$. This is a population size of about $1.29\,|D|\,\ell$. The reason it is greater than $|D|\,\ell$ is because the population size will fluctuate in the simulation, and it is necessary to maintain a minimum of $|D|\,\ell$ for statistical reliability of any trends present. So, for a population S, we choose with Adami a computationally feasible population size of $|D|$ times ℓ, which is sufficient to show any trends present:

$$|S| \geq |D|\,\ell \tag{C.4}$$

C.3 Physical Complexity Extension

Managing populations of variable length agent-chains is the most significant part of the reformulation, because it requires changing and re-justifying the fundamental assumptions.

It is necessary to understand the measures conditions and limits, in terms suitable for extending the measure to variable length populations. In equation (C.1) the Physical

Complexity C is defined for a population in which the individuals all have length ℓ. The most important question is what does the length ℓ equal if the population is of variable length? The problem is what ℓ represents, which is the **complexity potential** C_p, the maximum complexity possible for the population. The maximum complexity occurs when the per-site entropies sum to zero, as there is no randomness in the sites (all contain information),

$$in\ equation\ (C.1) \quad if\ \sum_{i=1}^{\ell} H(i) \to 0 \quad then\ C \to \ell. \tag{C.5}$$

So the complexity potential equals the length,

$$C_P = \ell, \tag{C.6}$$

provided the population S is of sufficient size $|S|$ for accurate calculations, i.e. $|S|$ is equal or greater than $|D|\,\ell$, as found in equation (C.4), justified at the end of section C.2. For a variable length population (VLP) S, the complexity potential for a VLP, C_{V_P}, cannot be equivalent to the length ℓ, because it does not exist. Based on the minimum sample size from equation (C.4) when $\ell = 1$, if the population size is less than the alphabet size, $|S| \leq |D|$, then C_{V_P} equals zero. If $|S| \geq |D|$, then there are at least $|D|$ individual samples of length one or more, so the C_{V_P} will be greater than zero and will be equivalent to the length of a VLP ℓ_V (which will be defined shortly),

$$C_{V_P} \quad = \quad \begin{cases} 0 & if\ \ |S| < |D| \\ \ell_V & if\ \ |S| \geq |D|, \end{cases} \tag{C.7}$$

where S is the population, D is the alphabet, $\ell_V \geq 1$ and $|D| > 0$. If ℓ_V where to be equal to the length of the longest individual ℓ_{max} in the VLP S, then the operational problem is that for some of the later sites i in the range of one to ℓ_{max}, the number of samples per-site will be less than the population size. So the complexity potential $C_{V_P} = \ell_V = \ell_{max}$ would be incorrect, as there are insufficient samples at the later sites. Consider the samples of DNA sequences shown below in Figure C.1.

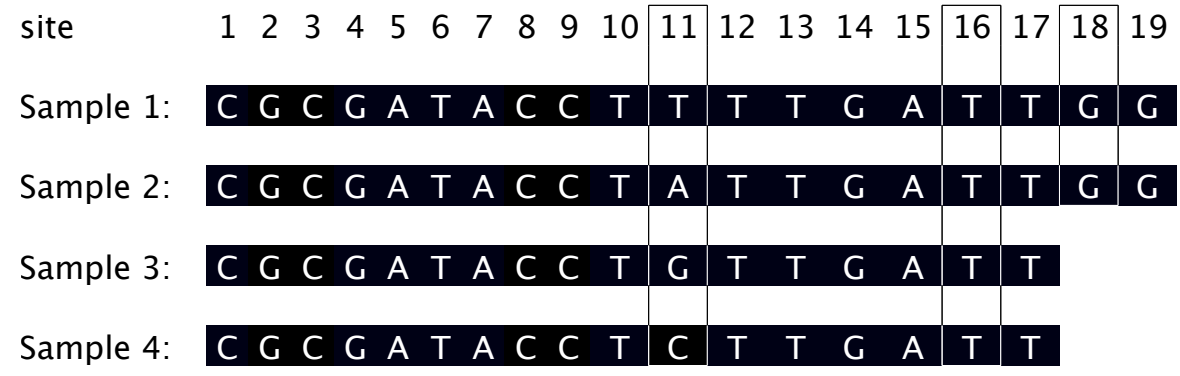


Figure C.1: Genome Samples

If the entropy is calculated for site 18 in Figure C.1 it is $H(18) = 0$, but there is an insufficient sample size to be accurate, for the estimated probabilities that are used in the calculation. Therefore, the length ℓ_V for a VLP is defined as the highest value within the range between one and the maximum length, $1 \leq \ell_V \leq \ell_{\max}$, for which there are sufficient

samples to calculate the complexity. To specify the value of ℓ_V precisely, a function is required which provides the number of samples at a given site,

$$sampleSize(i : site)output : int, \tag{C.8}$$

where the *output* varies between $1 \leq output \leq |S|$ and $|S|$ is the population size. Therefore, the length ℓ_V for a VLP is defined as the highest value within the range between one and the maximum length $1 \leq \ell_V \leq \ell_{\max}$, for which the number of samples at a site specified by ℓ_V is greater than or equal to the alphabet size $|D|$ multiplied by the length ℓ_V,

$$sampleSize(\ell_V + x) \leq sampleSize(\ell_V) \geq |D|\,\ell_V, \tag{C.9}$$

where x varies between $0 < x < \ell_{max} - \ell_V$, ℓ_V varies between $1 \leq \ell_V \leq \ell_{max}$, D is the alphabet and $|D| > 0$, ℓ_V is the length for a VLP, and ℓ_{max} is the maximum length in a VLP. This definition intrinsically includes the minimum population size for VLPs, $|D|\,\ell_V$. This replaces the minimum population size for same length populations as specified in equation (C.4).

C_{V_P} in equation (C.7) will always equal ℓ_V provided the condition $|S| \geq |D|$ is true. If it is false, the C_{V_P} will be zero and therefore the complexity of the VLP S will be zero, so no calculation involving ℓ_V will be performed.

The length ℓ used in the limits of equation (C.2) no longer exists, and therefore equation (C.2) must be updated. So the per-site entropy calculation for VLPs will be denoted by $H_V(i)$,

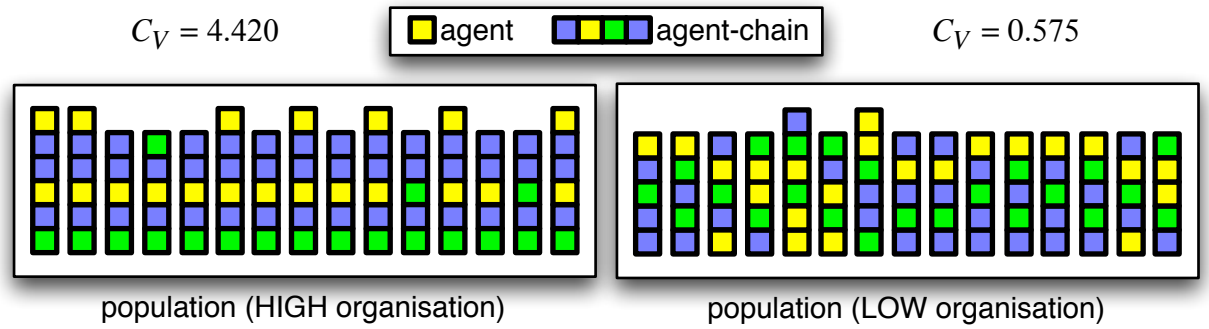
$$H_V(i) = -\sum_{d \in D} p_d(i) \log_{|D|} p_d(i), \tag{C.10}$$

where the site i ranges beteeen $1 \leq i \leq \ell_V$, the probability $p_d(i)$ ranges between $0 \leq p_d(i) \leq 1$, the sum over D of the $p_d(i)$ is 1, ℓ_V is the length for a VLP, and D is the alphabet. It remains algebraically almost identical to equation (C.2), but the conditions and constraints of its use will change, specifically ℓ is replaced by ℓ_V. $H_V(i)$ in equation (C.10) ranges between 0 and 1. When the entropy is minimum, zero, then the site i holds information, as every sample shows the same character of the alphabet at that site. When the entropy is maximum, the character found in the site i is uniformly random and therefore holds no information.

The complexity C_V of a variable length population (VLP) is the complexity potential of the acsVLP C_{V_P} from equation (C.7), minus the sum over the length of the acsVLP ℓ_V of the per-site entropies from equation (C.10),

$$C_V = \begin{cases} 0 & if\ C_{V_P} = 0 \\ \ell_V - \sum\limits_{i=1}^{\ell_V} H_V(i) & if\ C_{V_P} > 0, \end{cases} \tag{C.11}$$

where ℓ_V is the length for a VLP, and $H_V(i)$ is the VLP entropy for a site i. Now that the Physical complexity can be applied to VLPs, it can be used to measure the organisational complexity of agent-chain populations such as in Figure C.2, where l_V is calculated from equation (C.9):



$$D = alphabet = \{\blacksquare, \blacksquare, \blacksquare\}$$
$$|D| = alphabet\ size = 3 \quad \ell_{max} = maximum\ length = 6$$
$$\ell_V = length\ for\ a\ VLP = 5 \quad complexity\ potential\ C_{V_P} = \ell_V = 5$$

Figure C.2: Organisational Complexity in Populations of agent-chains

If we recall the definition of **organisational complexity** (organisation), which measures the complexity of the coherent patterns and structures, formed by the clustering of the agents within the population to form agent-chains. The Physical complexity values match the intuitive understanding that one gets visually about the organisational complexity in the populations.

D Efficiency and Clustering

Clustering in a population of evolving agent-chains is the grouping of same or similar sequences, around an optimum genome on the fitness landscape. If the evolutionary process does not become trapped at local optima, then it will reach the global optima. It is important to realise that the fitness landscape is the combination space of the agents weighted with their fitness values. As such it is dependent on the set of agents (alphabet) from which solutions are constructed as much as it is dependent on the selection pressure (fitness function).

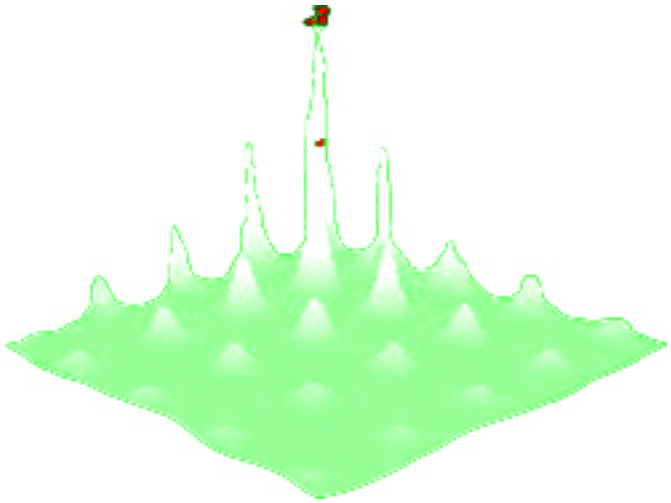


Figure D.1: Fitness Landscape - Single Global Optimum

The population will move across the fitness landscape, temporarily clustering over local optima, on the way to clustering around the optimal genome at the peak of the global optimum. In Figure D.1, the population will cluster into a single cluster at the peak of the global optimum. This process of clustering at the peak of a global optimum is indicated by the average population fitness F_{avg} tending towards the maximum fitness F_{max}, the **clustering indicator**. The optimal solution(agent-chain) which has maximum fitness F_{max} is increasingly becoming the dominant agent-chain in the population.

$$F_{avg} \rightarrow F_{\max} \tag{D.1}$$

The clustering indicator is especially useful as it easily and clearly indicates the occurrence of clustering, independent of the number clusters in the population. Also, its output is much more reliable to measure and programme, than the clustering coefficient which is about to be introduced.

At the same time, the population complexity C_V from equation (C.11) tends towards the complexity potential C_{V_P} from equation (C.7), because the uniformity of sites across the population is increasing as the optimal solution (sequence) is increasingly becoming the dominant sequence in the population,

$$E = \frac{C_V}{C_{V_P}} \rightarrow 1 \wedge |T| \rightarrow 1, \tag{D.2}$$

when $C_V \rightarrow C_{V_P}$, given equation (1), and where $|T|$ is the number of clusters. The efficiency E will not quite reach one due to the occurrence of mutations. The efficiency E acts as a **clustering coefficient**, tending towards its maximum, and when the population consists of only one cluster then $|T| = 1$. The other extreme is when the number of clusters equals the size of the population, $|T| = |S|$. This would only occur with a non-discriminating selection pressure, in which the fitness landscape would be perfectly flat, as shown in Figure D.2.

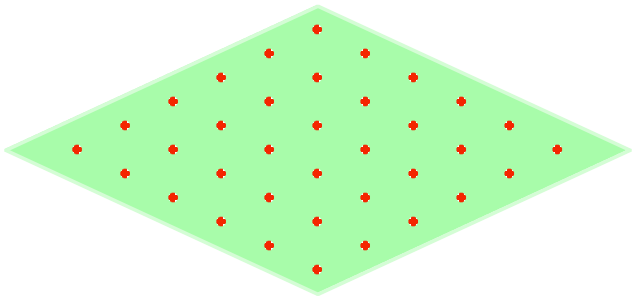


Figure D.2: 3D Fitness Landscape - Perfectly Flat

The population occupancy of the fitness landscape would be uniformly random, as any position(agent-chain combination) has the same fitness for any agent-chain in the population. So the entropy(randomness) would be tending towards maximum, resulting in the complexity C_V from equation (C.11) tending towards zero, so the efficiency E would tend to zero,

$$E = \frac{C_V}{C_{V_P}} \rightarrow 0 \wedge |T| \rightarrow |S|\,, \tag{D.3}$$

when $C_V \rightarrow 0$, given equation (1), where $|T|$ is the number of clusters and $|S|$ is the population size. It will generally never quite reach zero, due to the occurrence of mutation. So the number of clusters $|T|$ will tend to the population size and each cluster would in fact consist of only one agent-chain. The clustering indicator from equation (D.1) would be showing clustering, as the average fitness is always the maximum fitness, $F_{avg} = F_{max}$.

If there are multiple global optima, the clustering indicator $F_{avg} = F_{max}$ behaves as before, because it is independent of the number of clusters.

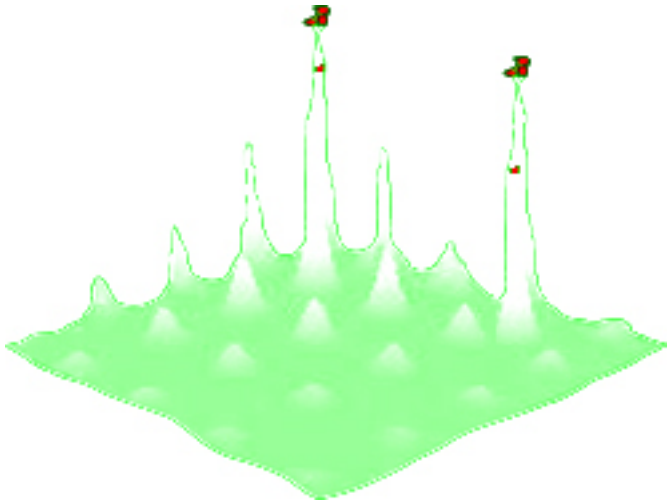


Figure D.3: 3D Fitness Landscape - Multiple Global Optima

However, the efficiency E will no longer tend towards its maximum from equation (2), precisely because the population consists of more than one cluster. For a population S, a cluster is a sub-population with an efficiency that tends towards the maximum (efficienyRange).

The simplest scenario of multiple clusters, is when there are **pure clusters**. Pure meaning that there are no agents shared between the clusters. So, there are as many totally distinct optimal solutions with maximum fitness F_{max}, as there are clusters. In this scenario, the value the efficiency E no longer tends towards one, but a value based on the number of clusters $|T|$, because a *number* of the probabilities $p_d(i)$ in equation (C.10) at each site become one over the number of clusters $|T|$. Given $p_d(i) = \frac{1}{|T|}$, equation (C.7), equation (C.10), equation (1) and the fact that the *number* of probabilities is equal to the number of clusters. Then the per-site entropy calculation for VLPs is

$$H_V(i) = \log_{|D|} |T| \,, \tag{D.4}$$

where i is the site, $|D|$ is the alphabet size, and $|T|$ is the number of clusters. Hence, the efficiency is

$$E \rightarrow 1 - (\log_{|D|} |T|) \tag{D.5}$$

given equation (C.7), equation (C.11) and equation (1). So for pure clusters, the value to which the clustering coefficient E tends towards, can be used to determine the number of clusters $|T|$.

For clusters that are impure, the relationship cannot be specified so succinctly. Environments with multiple optima will potentially lead to the population clustering around each global optimum, in which case the efficiency E will no longer tend towards the maximum. For a population S with clusters, each cluster is a sub-population with an efficiency that tends towards the maximum. To specify this more accurately the following function is required,

$$efficiency(input : population)output : int, \tag{D.6}$$

which provides the efficiency E between the range $0 \leq output \leq 1$, as defined in equation (1), from the input of a population.

Assuming that the clustering indicator is actively indicating clustering ($F_{avg} \rightarrow F_{\max}$) for a multiple global optima fitness landscape, then a cluster t,

$$t \in T \rightarrow \left(t \subseteq S \wedge efficiency(t) \rightarrow 1 \wedge |t| \approx \frac{|S|}{|T|} \wedge \sum_{t \in T} |t| = |S| \right), \qquad \text{(D.7)}$$

is a member of the set of clusters T in the population S, and therefore a sub-population of the population S. A cluster by definition has an efficiency E tending towards the maximum, one, from equation (1). Furthermore, the cluster size $|t|$ is roughly equal to the population size $|S|$ divided by the number of clusters $|T|$. It is only roughly equal, as the division may not result in a whole number. These conditions are true for all members of the set of clusters T, and the summation of the cluster sizes in T equals the size of the population $|S|$. This last condition ensures that clusters are non-overlapping, i.e. do not share agent-chains.

If we visualise the population on the 3D fitness landscape of Figure D.3, in Figure D.4, even though the clustering indicator is active, the clustering cannot be seen in the visualisation of the population.

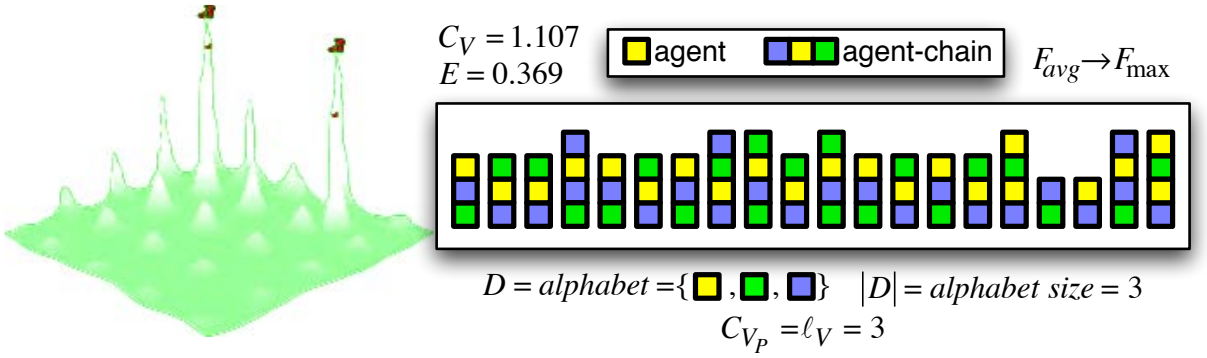


Figure D.4: Population with Hidden Clusters

If we arrange the population to show the clustering, then we can clearly see the two clusters present in the population, in Figure D.5.

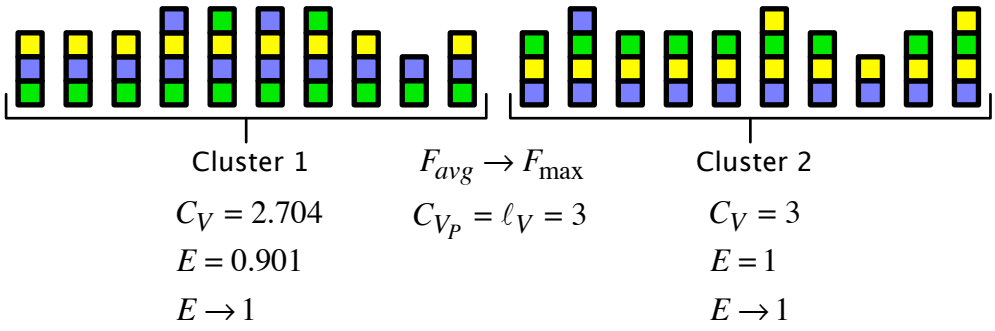


Figure D.5: Population with Clusters Showing

Figure D.5 clearly shows that the clusters in the population both have efficiencies E tending towards their maximum, compared to the efficiency E of the population as a whole which is tending towards a value significantly below the maximum. This is the behaviour of clusters as specified in equation (D.7).

The population size $|S|$, in Figures D.4 and D.5, is purposely double the minimum requirement specified in equation (C.9). So that, the complexity C_V in equation (C.11) and efficiency E in equation (1) could be applied to the clusters, without having to introduce new definitions. However, when determining the variable length ℓ_V of a cluster t, the sample size requirements are different, specifically a cluster t is a sub-population of S, and therefore by definition cannot have a population size equivalent to S (unless the population consists of only one cluster). Therefore, equation (C.9) must be updated to manage clusters,

$$\ell_V = \begin{cases} sampleSize(\ell_V + x) < sampleSize(\ell_V) \geq |D|\,\ell_V & if\ for\ population\ S \\ sampleSize(\ell_V + x) < sampleSize(\ell_V) \approx \frac{|D|\ell_V}{|T|} & if\ for\ cluster\ t \end{cases} \tag{D.8}$$

where S is a population, D is the alphabet, T is the set of clusters of population S, $\ell_{\max}$ is the maximum length in a VLP; t is a member of the set of clusters T, ℓ_V ranges between $1 \leq \ell_V \leq \ell_{max}$, x ranges between $0 < x < \ell_{max} - \ell_V$, and $|S| \geq |D| > 0$. A population with multiple clusters will always have an efficiency E that never tends towards the maximum. A reformulation is required of the efficiency measure E, to an efficiency measure capable of managing populations with multiple clusters

$$E_m(S) = \begin{cases} \frac{C_V}{C_{V_P}} & if\,|T| = 1 \\ \frac{\sum_{t \in T} E_m(t)}{|T|} & if\,|T| > 1 \end{cases}, \tag{D.9}$$

where T is the set of clusters of population S as defined in equation (D.5). It is equivalent to E if the population consists of only one cluster, and if there are multiple clusters E_m is the average of the efficiencies of the clusters.