Digital Business Ecosystem

Contract n° 507953

WP12: Optimisation

Del 12.5: Final software for simulated DBE model creation and prediction scenaria

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

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

Deliverable number: D12.5
Due date: 31/1/2007
Delivery Date: 31/1/2007

Short Description: The dynamic behaviour of the DBE has been studied by allowing two modifications to the prediction scenaria code described in Deliverable 12.3: 1) at each iteration step a random number of new companies is allowed to join the DBE; 2) it is assumed that each company has a copy of this prediction program and if it predicts that it is the weakest in the DBE, decides to change the portfolio of its products by dropping the one least in demand and replacing it with a product most in demand and sold by the strongest company. Such prediction scenaria can be run using as input the data of a simulated DBE one can create with the software package delivered with deliverable 12.3.
Keywords: dynamic environment; game theory; market dynamics; prediction scenaria

Partners owing: Imperial College, London
Partners contributed: Imperial College, London
Made available to: DBE Consortium

Versioning

Version	Date	Author, Organisation
1	31/1/2007	Petrou & Gautam
2		EEE, Imperial College, SW7 2AZ, UK

Quality check
1st Internal Reviewer: Philip De Wilde (Heriot-Watt)
2nd Internal Reviewer: Thomas Kurz (STU)
3rd Internal Reviewer:

Contents

1 Introduction

In Deliverable 12.3 we described a scheme for modelling the relationships of companies in a digital business ecosystem. We showed how to model seller-buyer relationships and how to take into consideration trust. We defined a fitness value with which we measured the "health" of each company, depending on the number of transactions it enters into and the demand of its products. We also described an iterative algorithm for computing the evolution of the fitness values of all the companies in a static world, ie in a world where the total number of companies was fixed (or reducing when some of them were going bankrupt), but in which no new companies were entering and the portfolios of companies did not change.

In this deliverable we allow the environment in which companies operate to change dynamically. This report is structured as follows. In section 2 we briefly summarise the model with which the mutual interactions between companies are modelled. More details on that can be found in deliverable 12.3. In section 3 we present experiments where a small number of new companies enter the system at every iteration step. We performed exhaustive experiments of how the fitness values of the companies in the system evolve when there is such an influx of new companies every so often, for all combinations of parameter values of the model. We present a small number of those results here. In section 4 we present a model inspired from game theory ideas: what if each player has a copy of this algorithm and can predict his future? What will a company do if it knows that its "fitness" value will soon make it go bankrupt? Presumably the company will take some action, like imitating the fittest company in the ecosystem and change its portfolio of products. How does this affect the evolution of the DBE? These questions are answered in part, by presenting some examples. The delivered software allows the interested user to run their own scenaria for any sets of parameters chosen. Finally, we conclude in section 5.

2 A brief review of the model used

Assume that each company C_i has with it associated a positive variable, y_i, which is its **fitness** variable. If the company is left on its own, in isolation, the value of y_i will decay exponentially with time constant τ. It was shown in deliverable 12.3 that the differential equation obeyed by y_i is

$$\frac{dy_i}{dt} = -\tau y_i + J_0 g_y(y_i) + \sum_{j\in C, j\neq i} W_{ij} J_{ij} g_y(y_j) - \frac{1}{\tilde{K}_i} \sum_{j\in C, j\neq i} K_{ij} g_y(y_j) + I_i + I_0 \tag{1}$$

where J_0 is a self-excitation constant, $g_y(y_i)$ is a sigmoid-type positive gain function,

$$g_y(y_i) = \begin{cases} 0 & \text{if } y_i < \Gamma_1, \\ \frac{y_i - \Gamma_1}{\Gamma_2 - \Gamma_1} & \text{if } \Gamma_1 \leq y_i \leq \Gamma_2 \\ 1 & \text{if } y_i > \Gamma_2 \end{cases} \tag{2}$$

with $[\Gamma_1, \Gamma_2]$ being the range of linearity of the gain function,

$$J_{ij} \equiv 1 - e^{-E_{ij}} \tag{3}$$

where E_{ij} is the number of products company C_i sells and company C_j wants to buy, and for which the matching values are less than a threshold T_1; factor W_{ij} takes values $\frac{4}{7}, \frac{2}{7}, \frac{1}{7}$ when company C_i is trusted, is indifferent or mistrusted by company C_j, respectively;

$$K_{ij} \equiv 1 - e^{-F_{ij}} \tag{4}$$

where F_{ij} is the number of products that both companies C_i and C_j sell, which are more similar than a threshold T_2; $\tilde{K}_i$ is the number of companies that send inhibitory signals to company C_i, because they sell similar products, and I_i and I_0 are some constants. The system of these differential equations is solved in an iterative way as follows:

$$y_{i;new} = y_{i;old} - \tau_y y_{i;old} + J_0 g_y(y_{i;old}) + \left\{ \sum_{j \in C, j \neq i} W_{ij} J_{ij} g_y(y_j) - \frac{1}{\tilde{K}_i} \sum_{j \in C, j \neq i} K_{ij} g_y(y_j) \right\}_{old} + I_i + I_0 \tag{5}$$

The values of y_i are initialised to be all equal to 0.5 at the first step. After each update cycle, we may remove from the system the companies the fitness value of which is below a certain threshold T_3. At the same time, we may allow the introduction of new companies with a certain rate, giving them as starting fitness value the average fitness of all other companies.

3 Allowing new companies into the system

The model we developed depends on several parameters. In table 1, we listed all these parameters.

S. No.	System Parameters	Meaning
1	y_0	Initial fitness value
2	τ	Time decay constant
3	$[\Gamma_1, \Gamma_2]$	Range of linearity of the positive gain function
4	I_i	External input to the company
5	I_0	Background input to the company
6	J_0	Self-excitation constant
7	T_1	Threshold value for a product that a company wants to buy to be considered as sufficiently similar to the same product another company sells, when the attributes of the two products are compared.
8	T_2	Threshold value for the versions of a product two companies sell to be considered as sufficiently similar to each other, so the companies to be considered competitors.
9	T_3	Threshold value for removing companies from the system; if $y_i \leq T_3$ the company is considered bankrupt.

Table 1: List of the parameters of the system

We performed a series of experiments in order to analyse the effect of each parameter on the dynamics of the system. In each series of experiments the value of one parameter was varied from one experiment to the next, while all other system parameters were kept constant.

At each iteration step, the companies with fitness below $T_3 = 0.3$ were removed from the system assumed to become too weak to survive. At the same time a random number of new companies was added to the DBE.

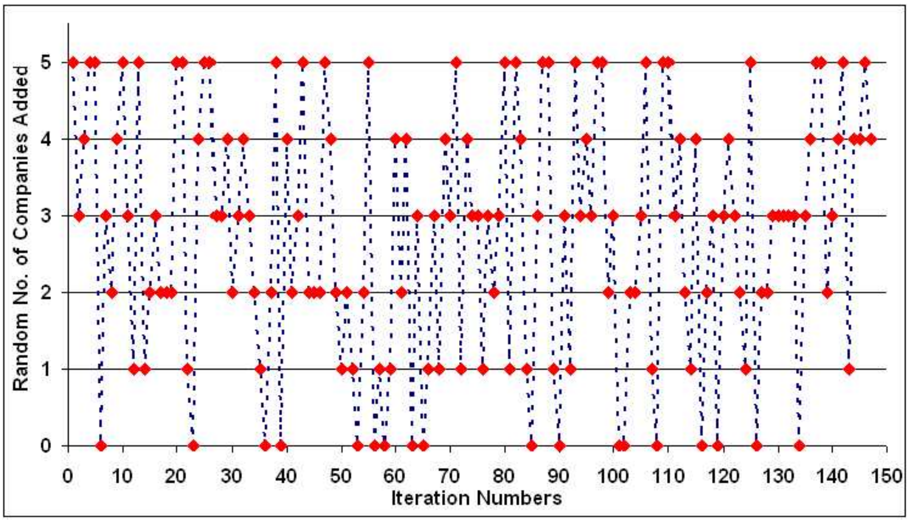


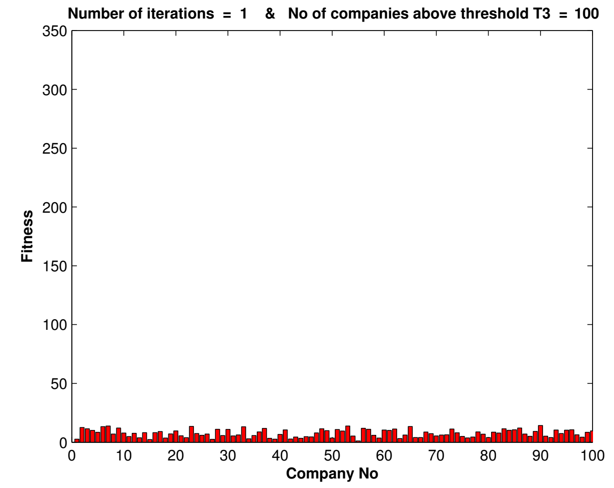
Figure 1: Random number of companies added to the DBE at each iteration step.

To emulate the entry of new companies, we run the package that allows us to create a simulated DBE[1] and created 500 companies. The DBE we started with was made up from the first 100 of these companies. At each iteration step of updating the fitness values of the companies of the DBE, a random integer number between 0 and 5 was drawn and this was the number of new companies added to the DBE from the pool of 400 extra companies that had been set up. Figure 1 shows how many companies were added to the DBE at each iteration step.

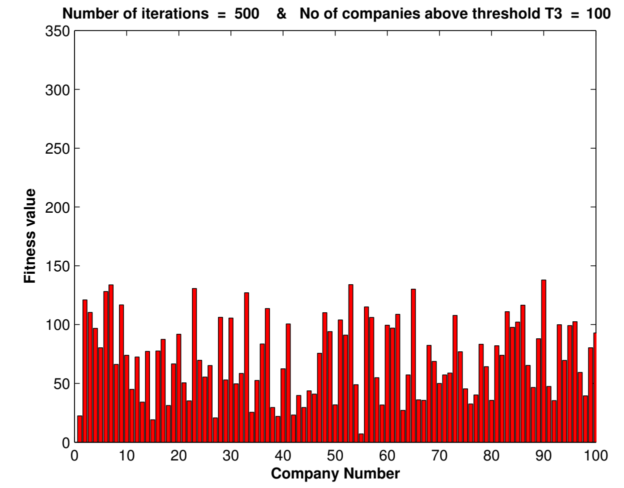
3.1 Effect of the value of τ

Figures 2–5 demonstrate the effect of parameter τ and the effect of adding new companies to the system. All these results have been obtained with values $\Gamma_1 = 0.3$, $\Gamma_2 = 0.7$, $J_0 = 0.1$, $I_i = I = 0$, $T_1 = T_2 = 0.1$ and $T_3 = 0.3$. All new companies entering the system were given an initial fitness value equal to the average fitness value of the existing companies in the DBE. We note that for small values of τ more or less all companies survive, while for large values of τ very few companies survive. This is irrespective of the injection of new companies. It seems that parameter τ is the most significant parameter of the model.

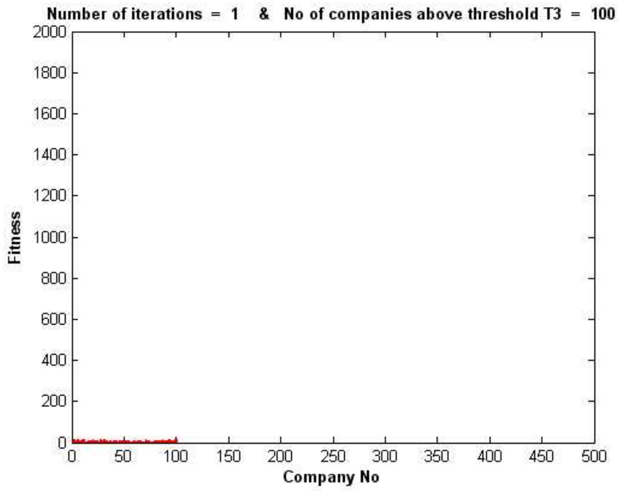
[1]This package was delivered in its final form in June 2006 and described in detail in deliverable 12.3.



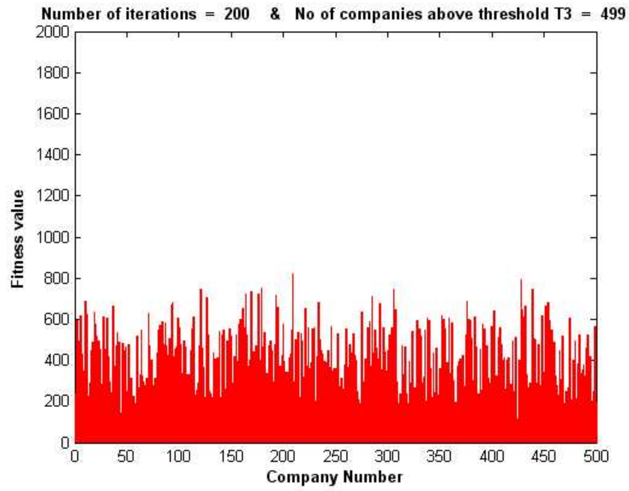
Iteration: 1 ($\tau = 0.2$)



Iteration: 500 ($\tau = 0.2$)

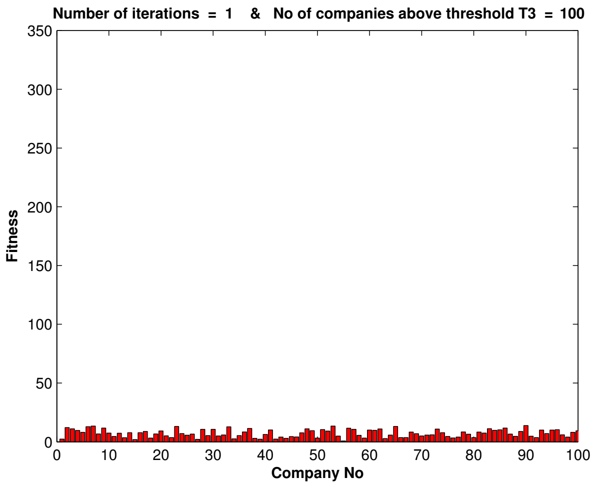


Iteration: 1 ($\tau = 0.2$)

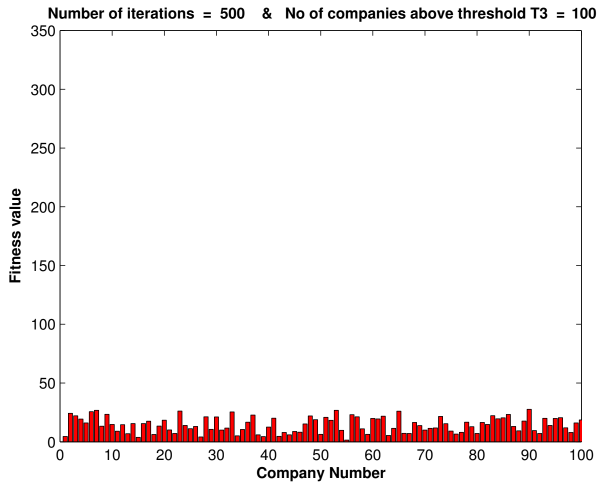


Iteration: 200 ($\tau = 0.2$)

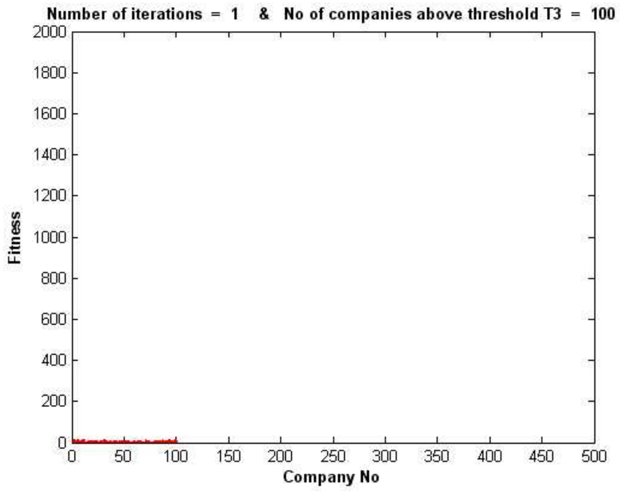
Figure 2: The effect of the value of τ in the dynamic behaviour of the DBE. $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values started as 0.5. The top set of panels corresponds to a static environment, when no new companies enter the system. The ones below correspond to the case when new companies join in.



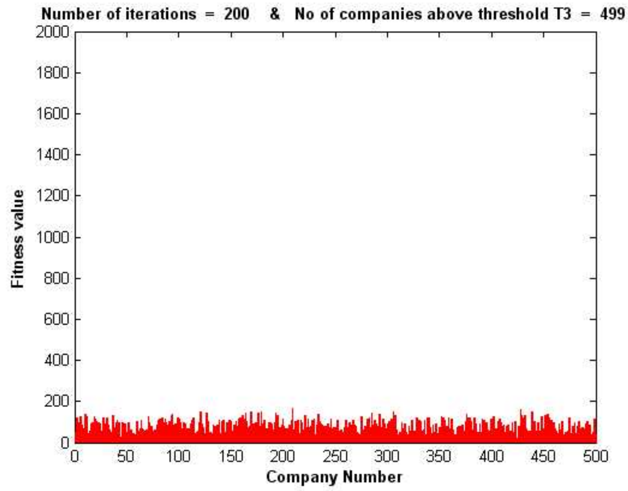
Iteration: 1 ($\tau = 1.0$)



Iteration: 500 ($\tau = 1.0$)

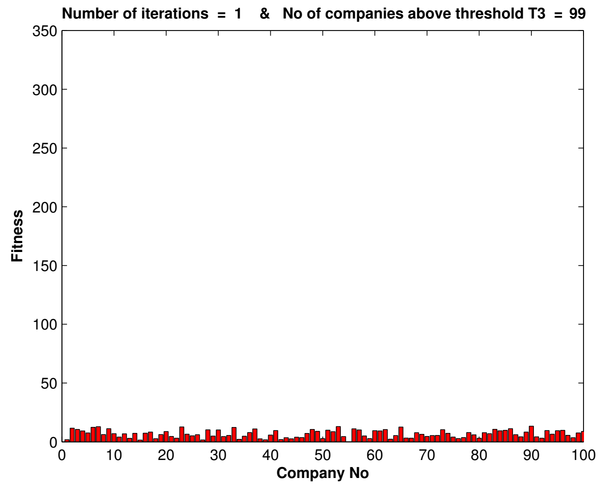


Iteration: 1 ($\tau = 1.0$)

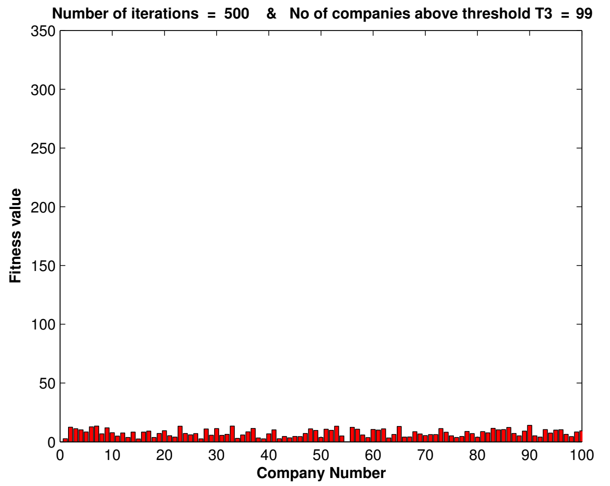


Iteration: 200 ($\tau = 1.0$)

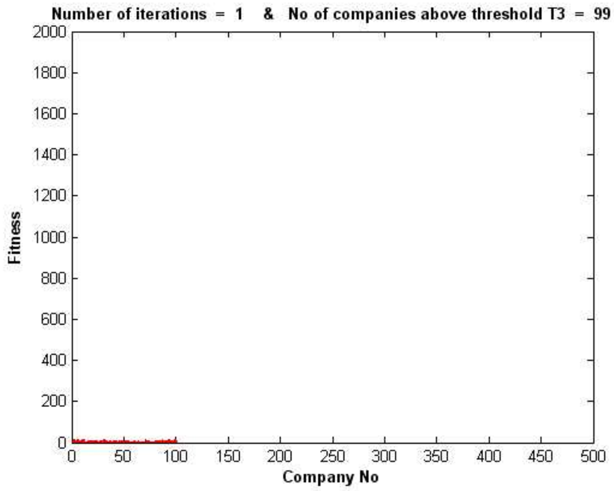
Figure 3: The effect of the value of τ in the dynamic behaviour of the DBE. $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values started as 0.5. At the top the panels correspond to a static environment, where no new companies enter the system. The ones below correspond to the case when new companies join in.



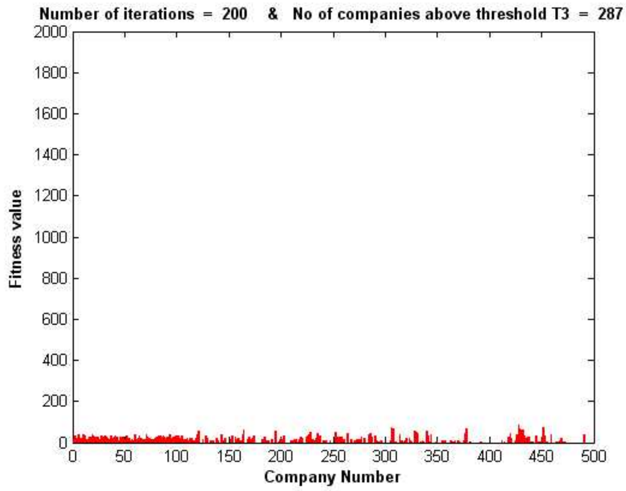
Iteration: 1 ($\tau = 2.0$)



Iteration: 500 ($\tau = 2.0$)

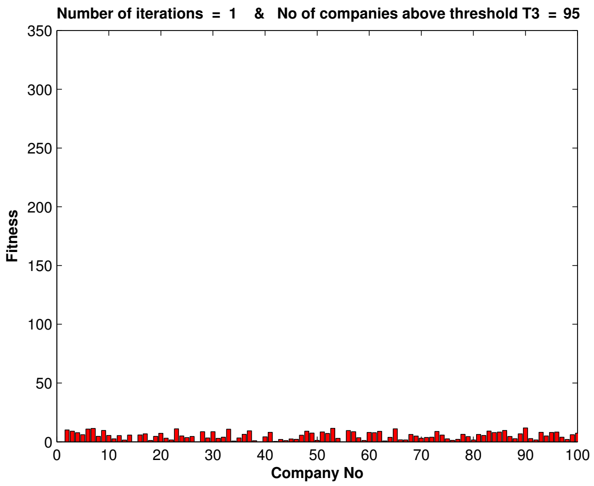


Iteration: 1 ($\tau = 2.0$)

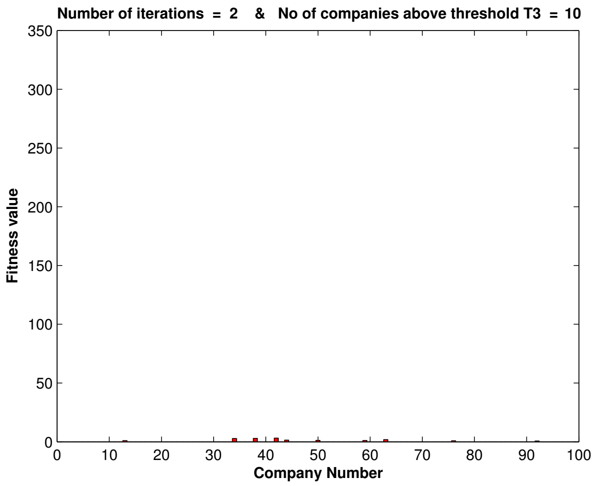


Iteration: 200 ($\tau = 2.0$)

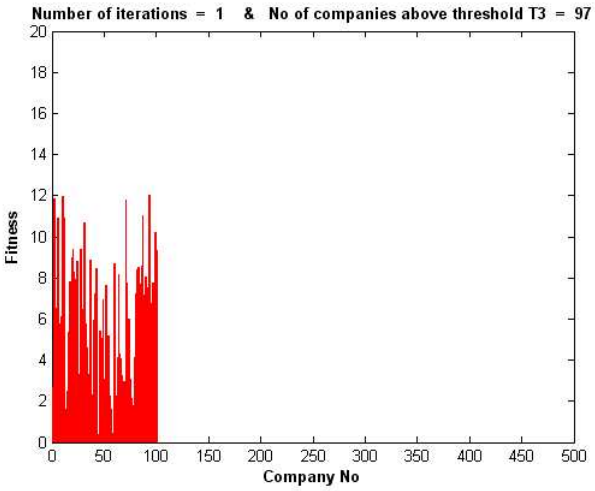
Figure 4: The effect of the value of τ in the dynamic behaviour of the DBE. $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values started as 0.5. The top set of panels correspond to a static environment, when no new companies enter the system. The ones below correspond to the case when new companies join in.



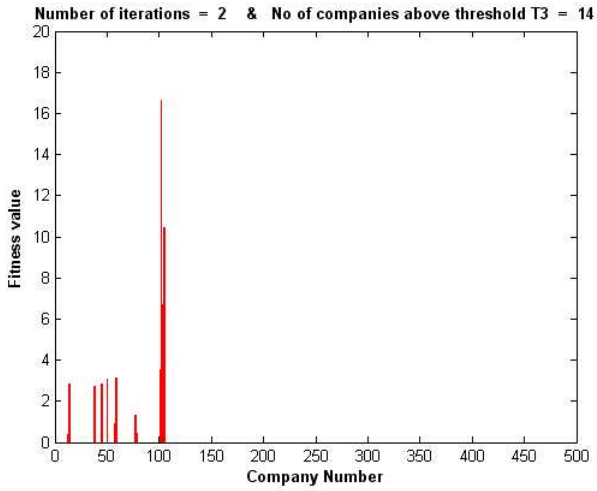
Iteration: 1 ($\tau = 5.0$)



Iteration: 2 ($\tau = 5.0$)



Iteration: 1 ($\tau = 5.0$)

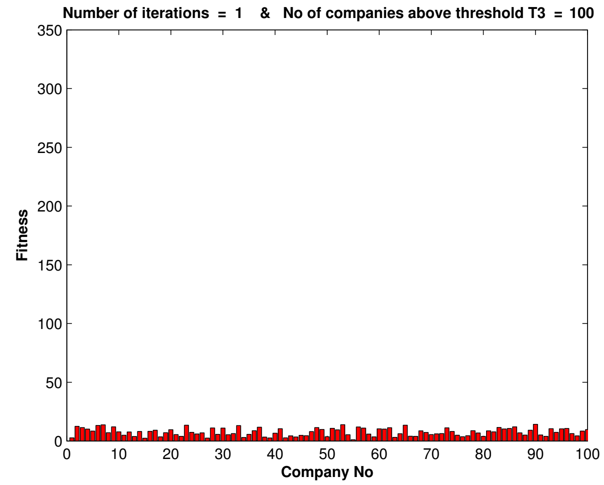


Iteration: 2 ($\tau = 5.0$)

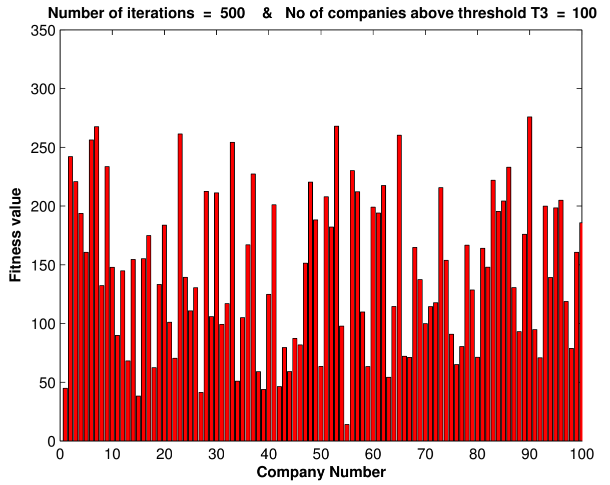
Figure 5: The effect of the value of τ in the dynamic behaviour of the DBE. $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values started as 0.5. The top set of panels correspond to a static environment, when no new companies enter the system. The ones below correspond to the case when new companies join in.

3.2 The effect of J_0

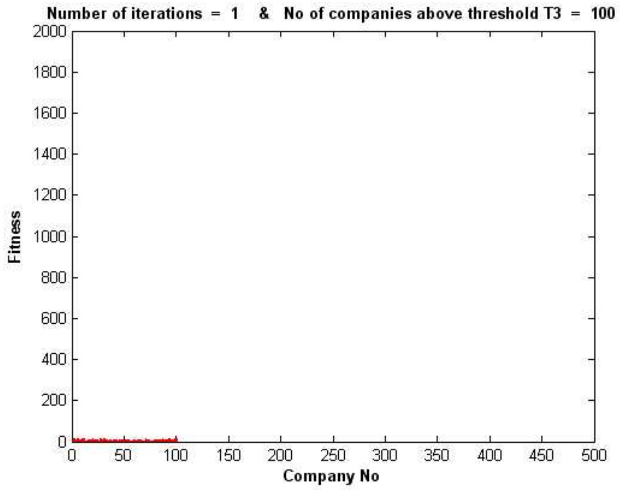
Figures 6–9 show the effect of parameter J_0 when the values of the other parameters are fixed. This parameter seems to have very little effect to the system.



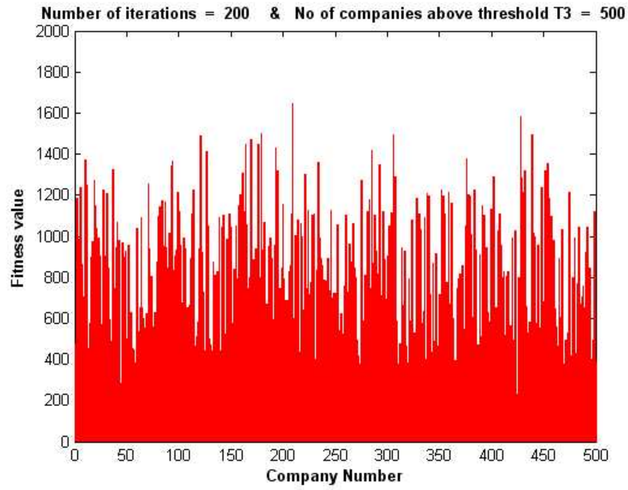
Iteration: 1 ($J_0 = 0.1$)



Iteration: 500 ($J_0 = 0.1$)

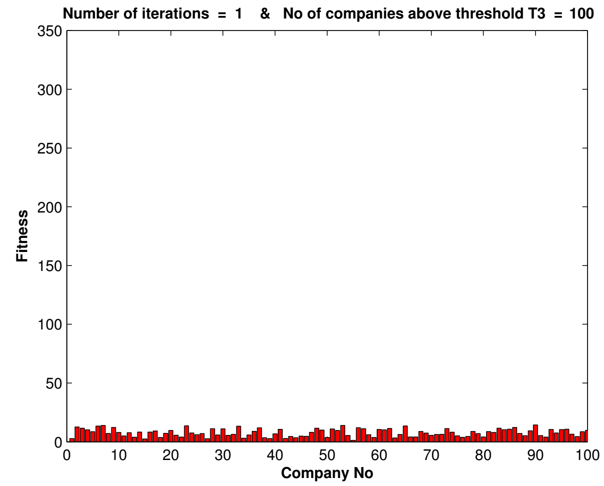


Iteration: 1 ($J_0 = 0.1$)

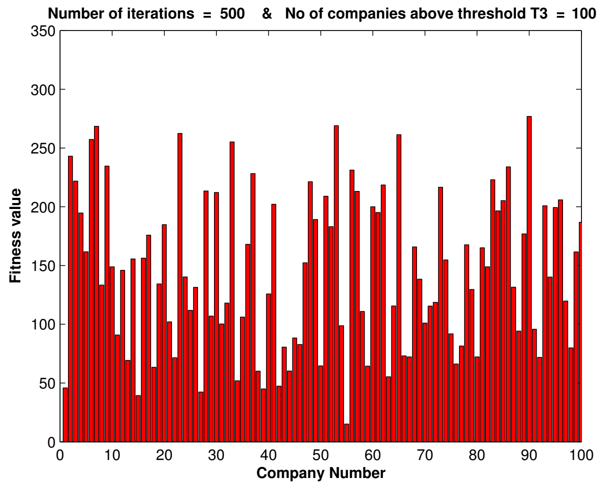


Iteration: 200 ($J_0 = 0.1$)

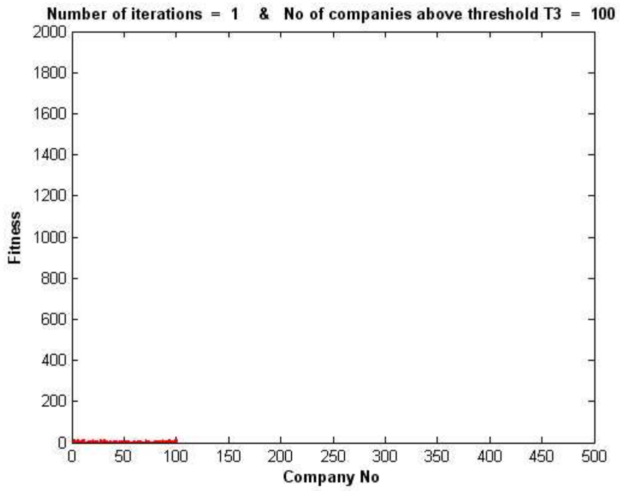
Figure 6: Effect of value of J_0 on the dynamic behaviour of the DBE. The values of the parameters are: $\tau = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values were initialised to 0.5. At the top the results in a static environment. At the bottom the results when new companies were allowed to join the system.



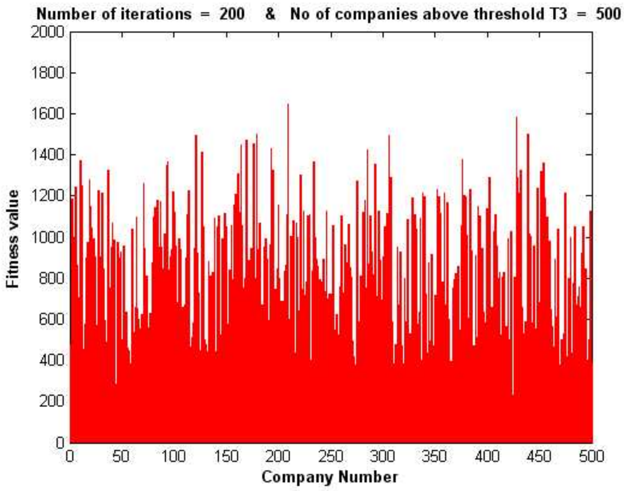
Iteration: 1 ($J_0 = 0.2$)



Iteration: 500 ($J_0 = 0.2$)

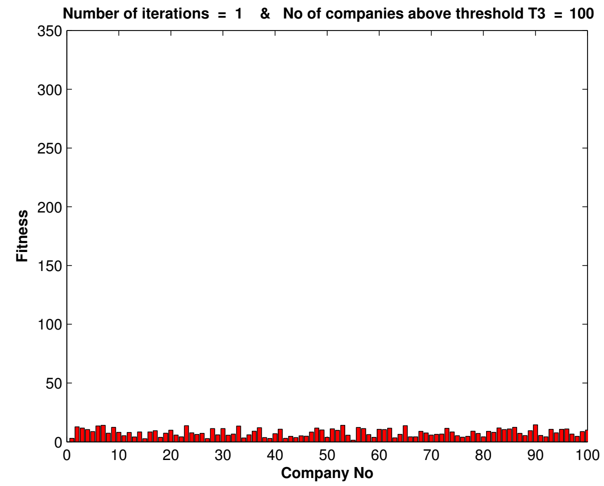


Iteration: 1 ($J_0 = 0.2$)

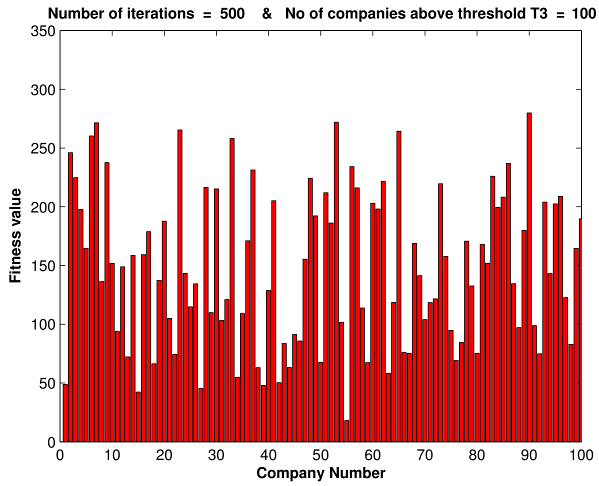


Iteration: 200 ($J_0 = 0.2$)

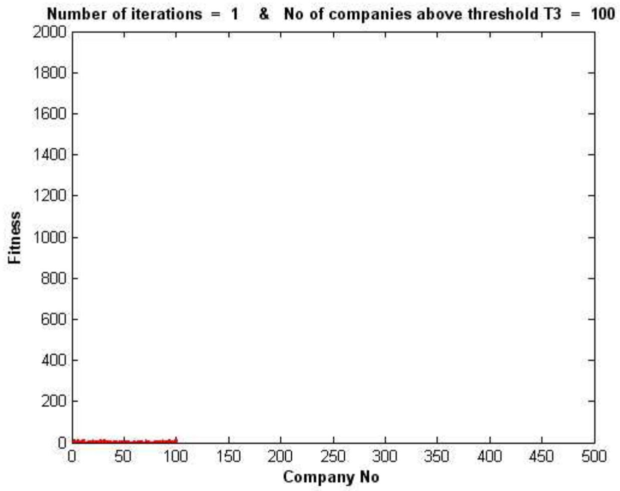
Figure 7: Effect of value of J_0 on the dynamic behaviour of the DBE. The values of the parameters are: $\tau = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values were initialised to 0.5. At the top the results in a static environment. At the bottom the results when new companies were allowed to join the system.



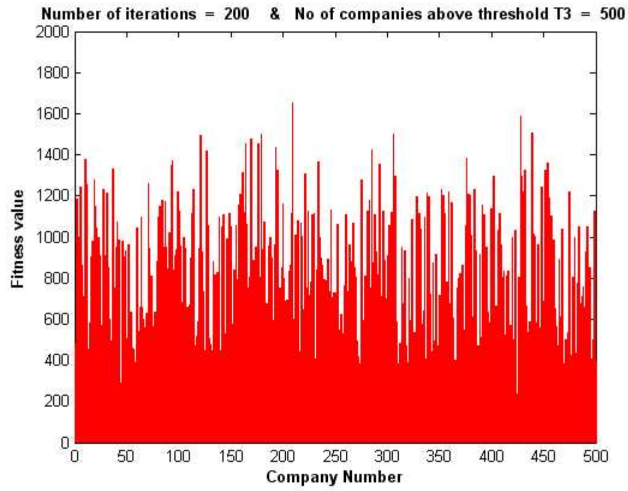
Iteration: 1 ($J_0 = 0.5$)



Iteration: 500 ($J_0 = 0.5$)

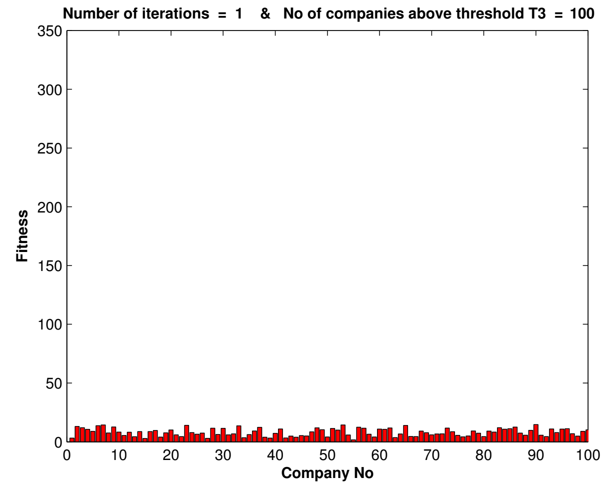


Iteration: 1 ($J_0 = 0.5$)

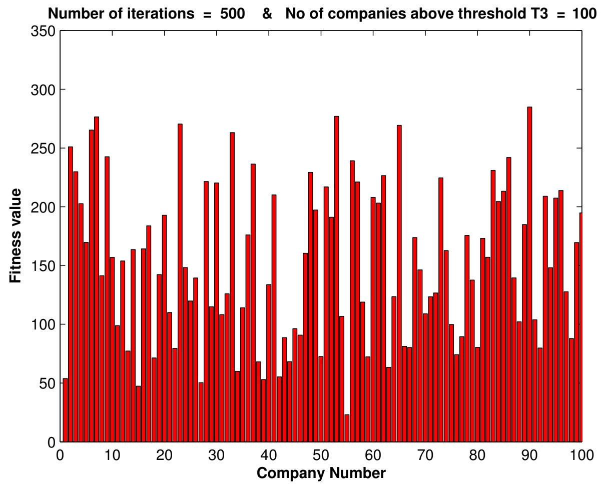


Iteration: 200 ($J_0 = 0.5$)

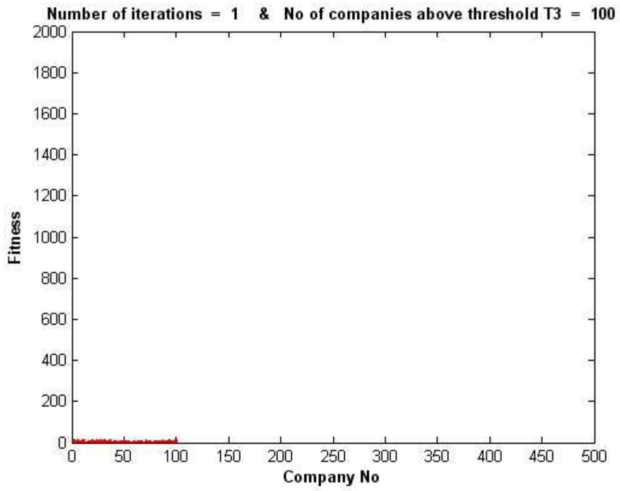
Figure 8: Effect of value of J_0 on the dynamic behaviour of the DBE. The values of the parameters are: $\tau = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values were initialised to 0.5. At the top the results in a static environment. At the bottom the results when new companies were allowed to join the system.



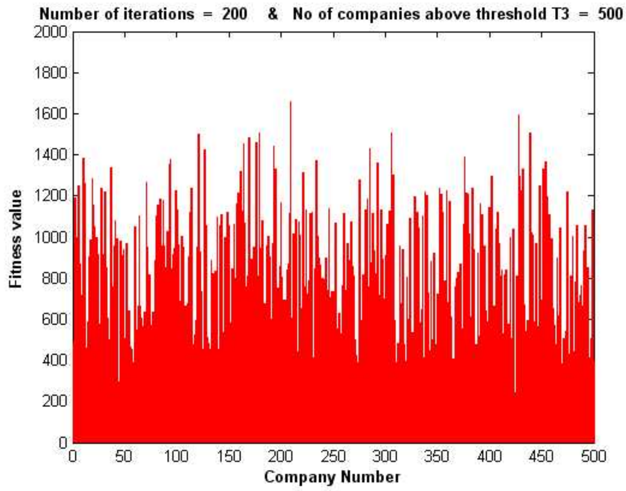
Iteration: 1 ($J_0 = 1.0$)



Iteration: 500 ($J_0 = 1.0$)



Iteration: 1 ($J_0 = 1.0$)

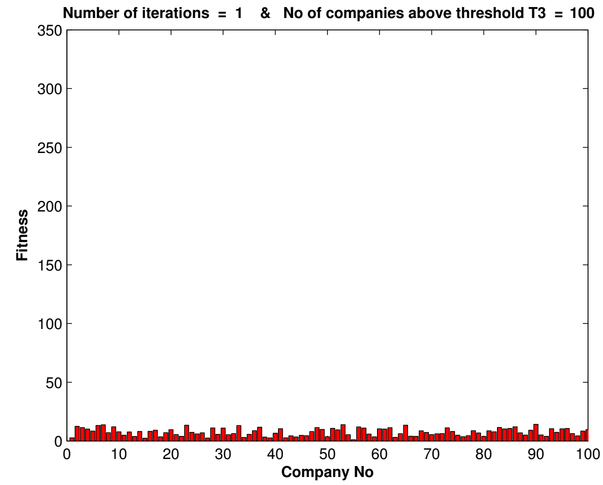


Iteration: 200 ($J_0 = 1.0$)

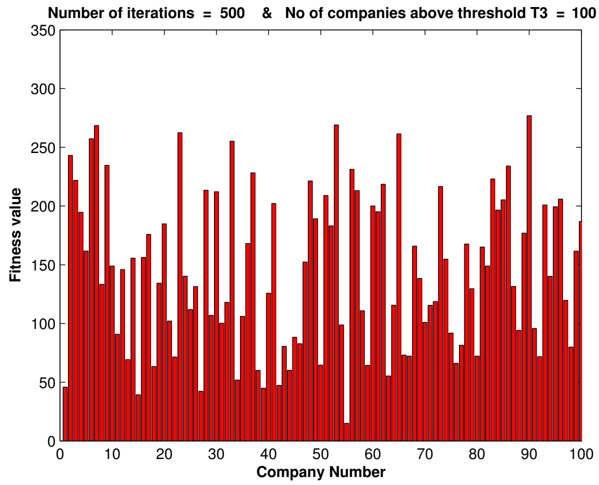
Figure 9: Effect of value of J_0 on the dynamic behaviour of the DBE. The values of the parameters are: $\tau = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$, $I_0 = 0.0$, $I_i = 0.0$. All fitness values were initialised to 0.5. At the top the results in a static environment. At the bottom the results when new companies were allowed to join the system.

3.3 The effect of parameters $I_0 + I_i$

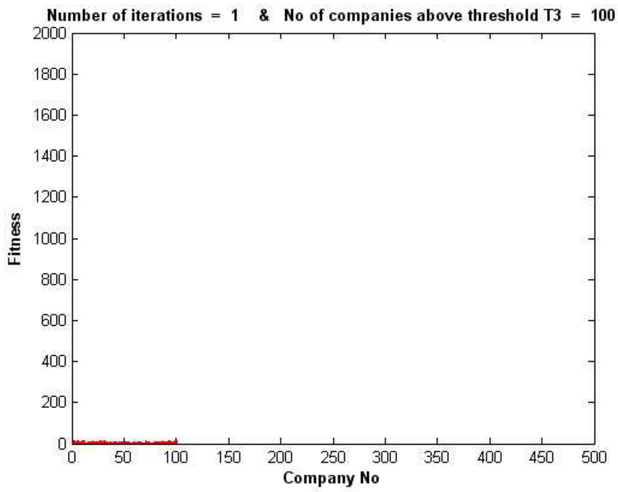
Figures 10–13 show the effect of the values of combined parameters $I_0 + I_i$ when the values of the other parameters are fixed. These two parameters do not appear to play significant role.



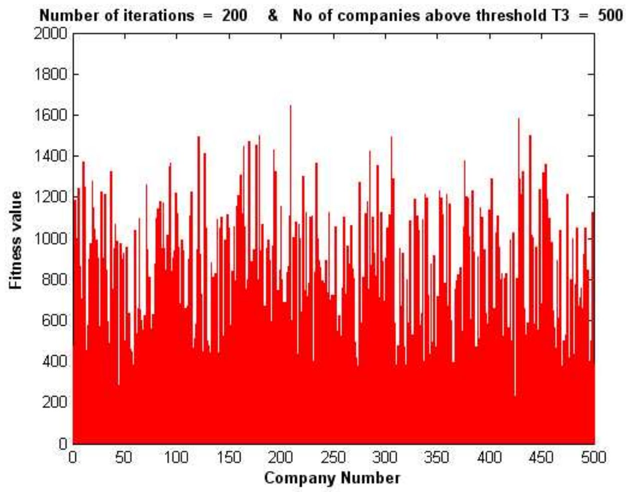
Iteration: 1 ($I_0 + I_i = 0.1$)



Iteration: 500 ($I_0 + I_i = 0.1$)

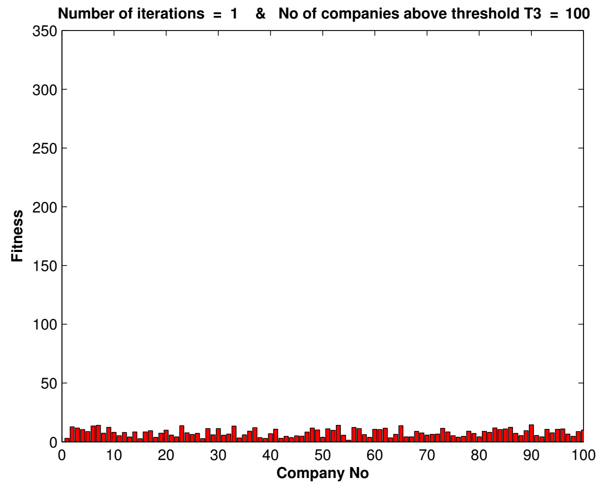


Iteration: 1 ($I_0 + I_i = 0.1$)

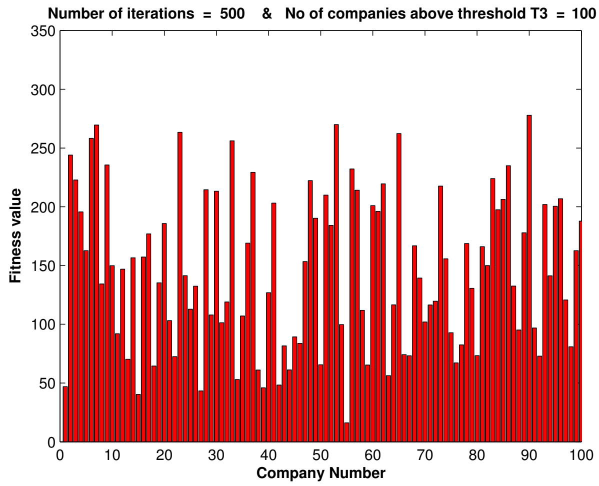


Iteration: 200 ($I_0 + I_i = 0.1$)

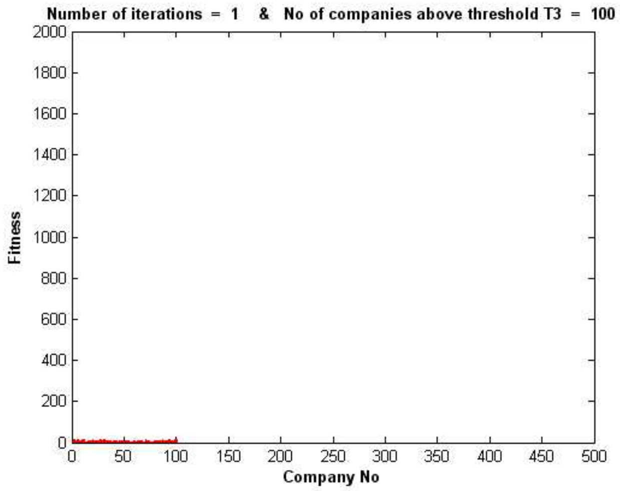
Figure 10: The effect of parameters $I_0 + I_i$ on the dynamic behaviour of the DBE. The values of all other parameters are: $\tau = 0.1$, $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. All fitness values are initialised to 0.5. At the top the result when no new companies are added to the system. At the bottom the result when a small number of new companies is added at each iteration step.



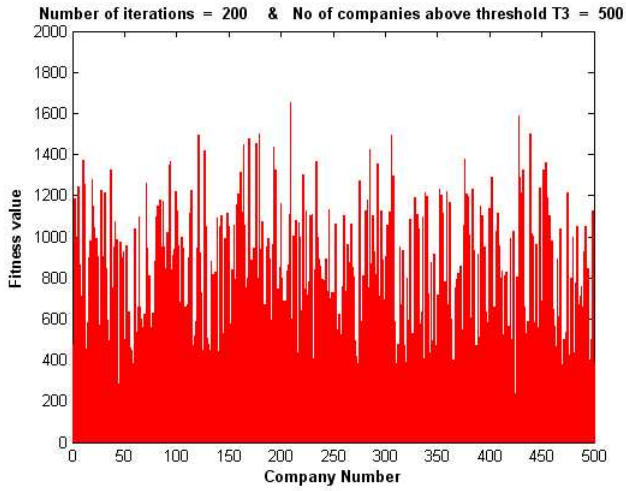
Iteration: 1 ($I_0 + I_i = 0.2$)



Iteration: 500 ($I_0 + I_i = 0.2$)

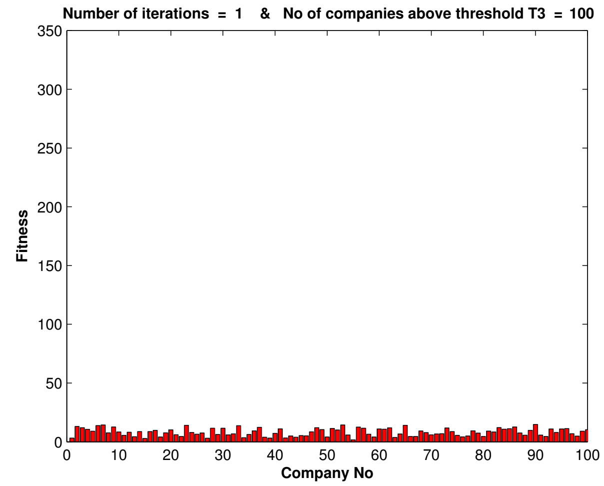


Iteration: 1 ($I_0 + I_i = 0.2$)

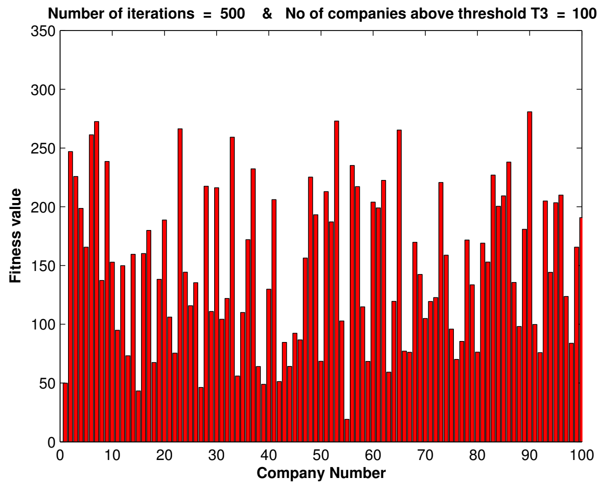


Iteration: 200 ($I_0 + I_i = 0.2$)

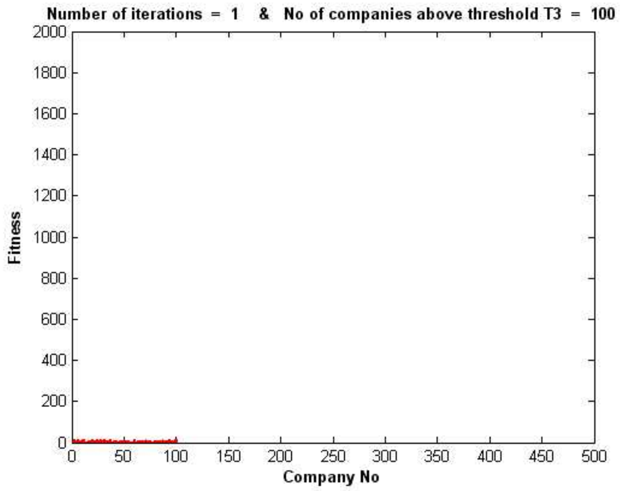
Figure 11: The effect of parameters $I_0 + I_i$ on the dynamic behaviour of the DBE. The values of all other parameters are: $\tau = 0.1$, $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. All fitness values are initialised to 0.5. At the top the result when no new companies are added to the system. At the bottom the result when a small number of new companies is added at each iteration step.



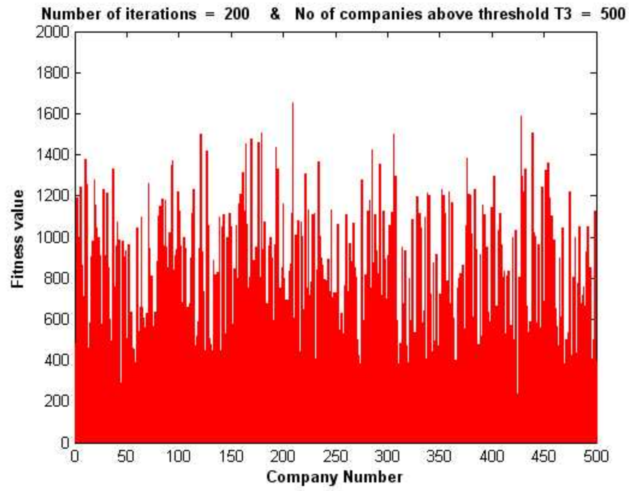
Iteration: 1 ($I_0 + I_i = 0.5$)



Iteration: 500 ($I_0 + I_i = 0.5$)

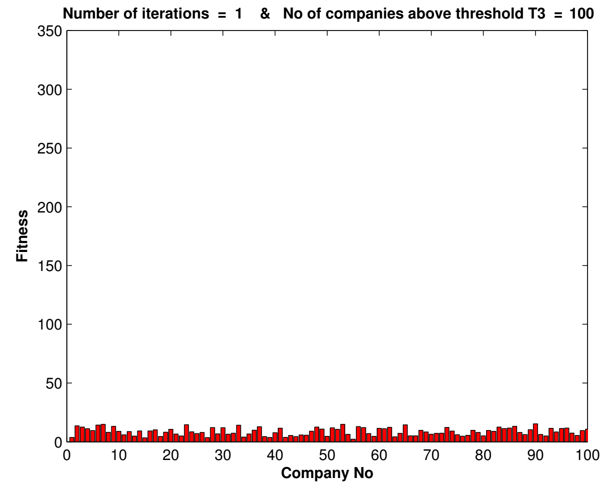


Iteration: 1 ($I_0 + I_i = 0.5$)

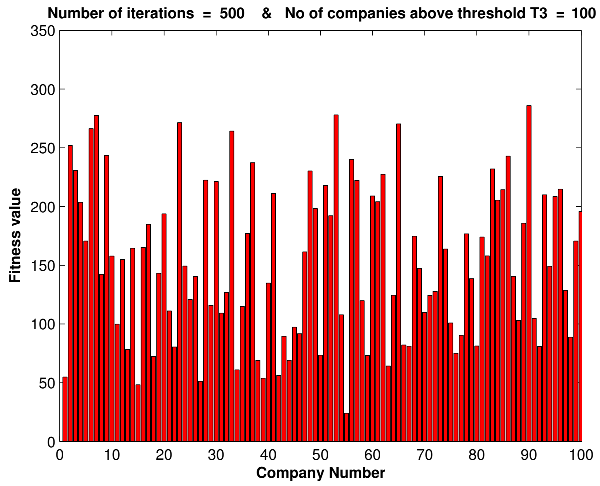


Iteration: 200 ($I_0 + I_i = 0.5$)

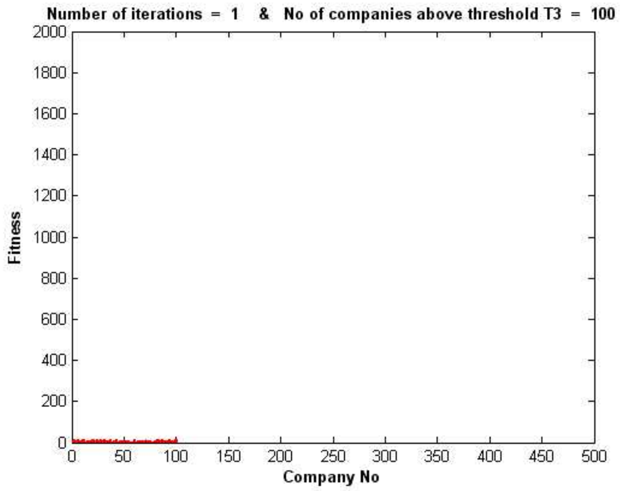
Figure 12: The effect of parameters $I_0 + I_i$ on the dynamic behaviour of the DBE. The values of all other parameters are: $\tau = 0.1$, $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. All fitness values are initialised to 0.5. At the top the result when no new companies are added to the system. At the bottom the result when a small number of new companies is added at each iteration step.



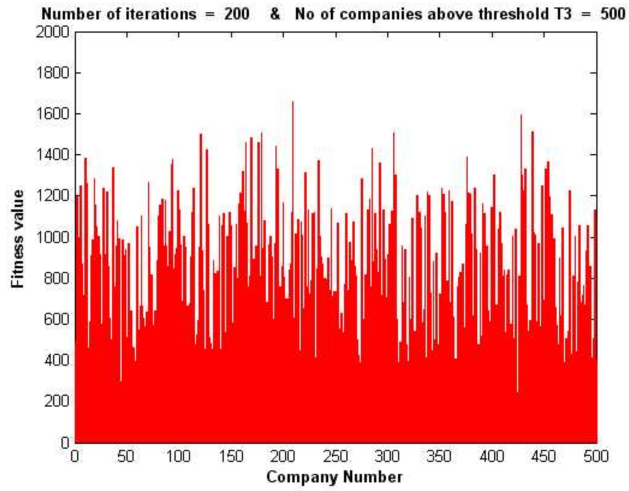
Iteration: 1 ($I_0 + I_i = 1.0$)



Iteration: 500 ($I_0 + I_i = 1.0$)



Iteration: 1 ($I_0 + I_i = 1.0$)



Iteration: 200 ($I_0 + I_i = 1.0$)

Figure 13: The effect of parameters $I_0 + I_i$ on the dynamic behaviour of the DBE. The values of all other parameters are: $\tau = 0.1$, $J_0 = 0.1$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. All fitness values are initialised to 0.5. At the top the result when no new companies are added to the system. At the bottom the result when a small number of new companies is added at each iteration step.

4 When each company can predict its future

We assume now that each company has a copy of the algorithm we described in deliverable 12.3 (and in brief here in section 2) and it can foresee its future in the ecosystem. If a company thinks that it is too weak to survive, it is expected to take some action to prevent its bad fate from becoming reality. A plausible course of action would be to look at the fittest company in the ecosystem and identify which of that company's products is most in demand and enhance its own portfolio by starting to sell that product, while dropping from its portfolio one of its own products that was least in demand. The following algorithm was designed to simulate this behaviour.

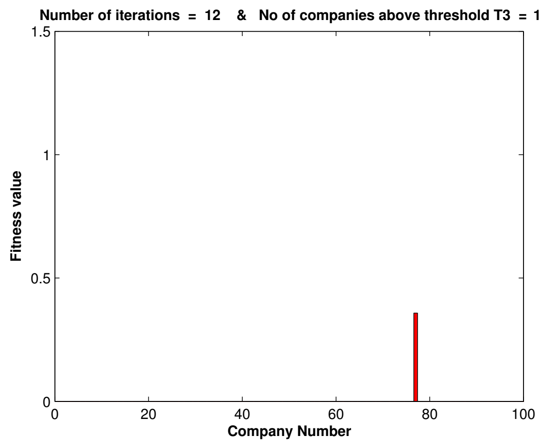
- 1: At every iteration step, consider the **X** weakest companies, **X** = 1, 2, 3, 4, 5.
- 2: For every product at each iteration step, count

2.1: how many companies want to buy it;

2.2: rank products in decreasing order of demand.

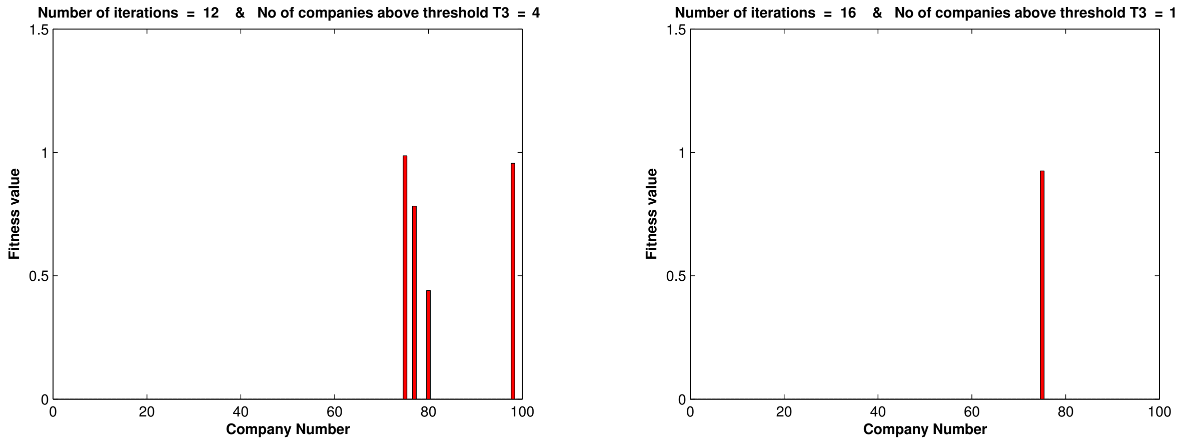
- 3: Start from the strongest company. See which one of the products it sells is most in demand.
- 4: Next look for the weakest company. See which one of the products it sells is least in demand.
- 5: Replace it with the product which is most in demand and sold by the strongest company.
- 6: Check now if any other company wants to buy or sell the product you just replaced in the portfolio of the weakest company. If even one company still wants to buy or sell it, do nothing. If no other company sells of wants to buy it, then remove this product from the list of products in the DBE.

We present here some example results of running this algorithm. Figure 14 shows the situation observed in a static environment emulated by a certain set of parameter values: after 12 iterations a monopoly was created. Figure 15 shows the same system after 12 iteration steps when at each step the weakest company was changing its portfolio. This set of parameter values eventually led to a monopoly, but it was created at iteration step 16, ie the DBE survived for longer, even though only a single company was changing only one of its products at each iteration step. Some more examples are shown in figures 16–18.



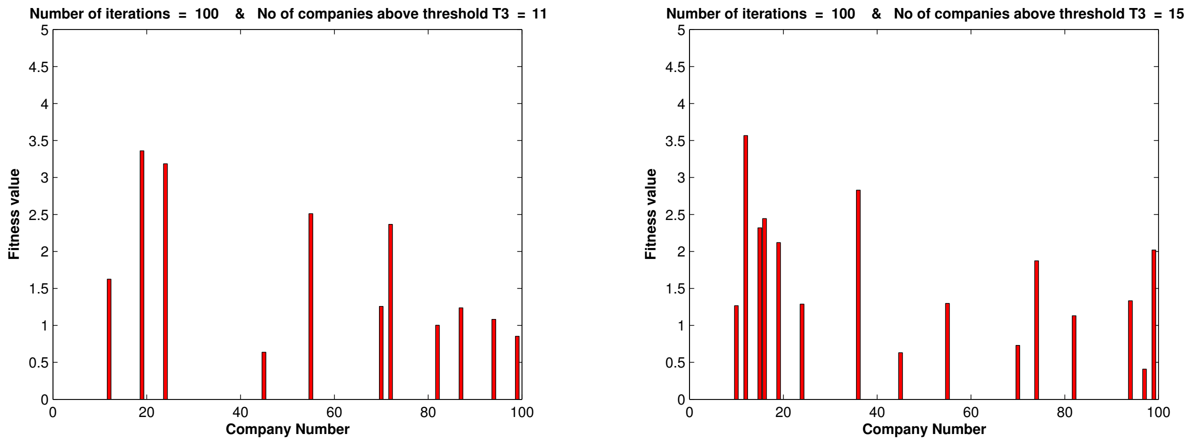
Iteration: 12

Figure 14: The values of the parameters are: $I_0 = 0.05$, $I_i = 0.05$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $\tau = 2.4$, $J_0 = 2.2$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. In a static environment, these values led to the creation of a monopoly after 12 iteration steps.



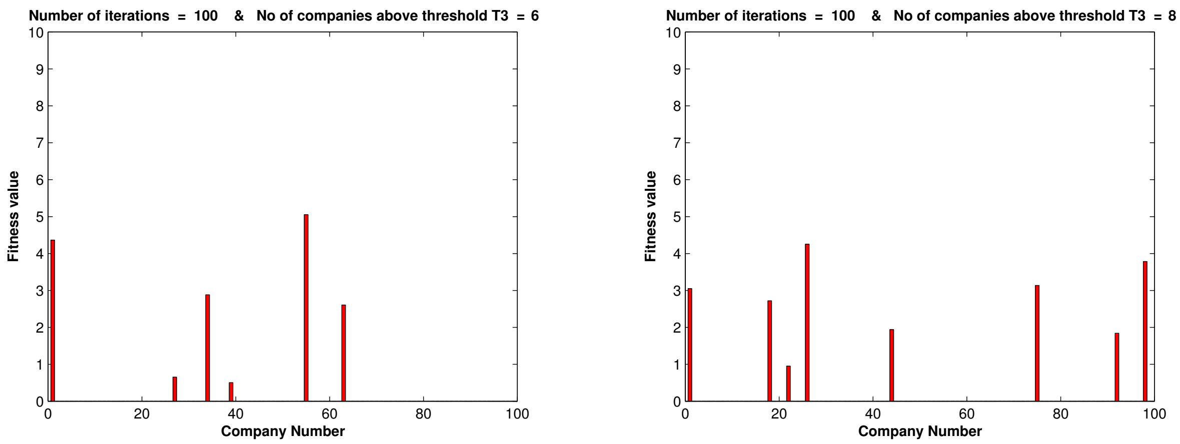
Iteration: 12 Iteration: 16

Figure 15: The values of the parameters are the same as in figure 14. However, now the weakest company at each iteration step is allowed to replace its least selling product with the most selling one supplied by the strongest company. The monopoly appears at a later stage, ie at iteration 16.



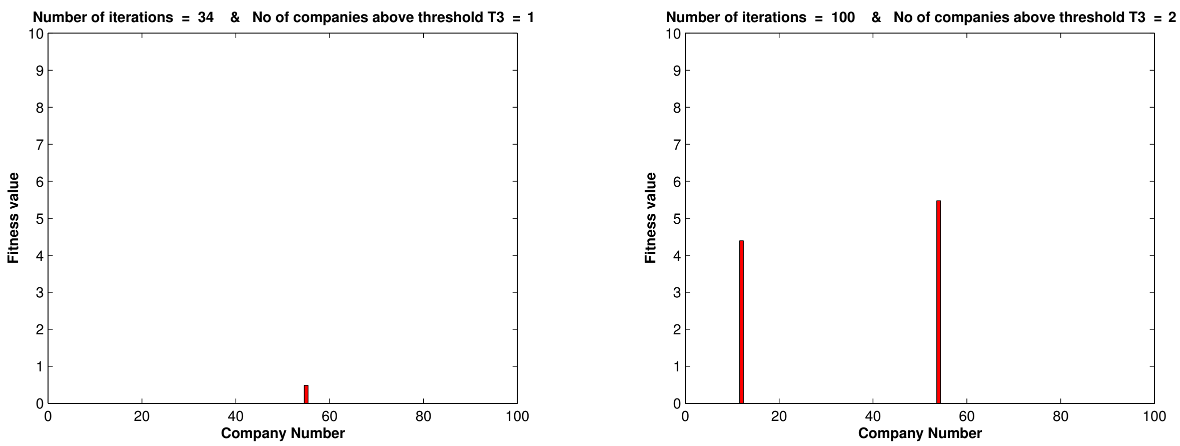
Static world: Iteration= 100 Dynamic world: Iteration= 100

Figure 16: The values of the parameters are: $y_i = 0.5$ (initial fitness value), $I_0 = 0.05$, $I_i = 0.05$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $\tau = 2.1$, $J_0 = 3.5$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. When companies change their portfolio even to a small degree, more companies survive and different companies dominate the DBE. Note that the fittest company is different in the two panels.



Static world: Iteration= 100 Dynamic world: Iteration= 100

Figure 17: The values of the parameters are: $y_i = 0.5$ (initial fitness value), $I_0 = 0.5$, $I_i = 0.5$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $\tau = 2.1$, $J_0 = 4.5$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$.



Static world: Iteration= 34 Dynamic world: Iteration= 100

Figure 18: The values of the parameters are: $y_i = 0.5$ (initial fitness value), $I_0 = 0.5$, $I_i = 0.5$, $T_1 = T_2 = 0.1$, $T_3 = 0.3$, $\tau = 2.1$, $J_0 = 6.5$, $[\Gamma_1, \Gamma_2] = [0.3, 0.7]$. In the static world a monopoly was created at iteration 34. When companies were allowed to change their portfolios, even at iteration 100 two had survived.

5 Discussion and Conclusions

We presented in this deliverable algorithms that can be used to study the dynamics of a digital business ecosystem in a dynamic environment, ie when companies are allowed to join in and when companies are allowed to change their portfolios. The manuals of the software packages that can be used to perform such experiments are given in the two appendices. These programs rely on the use of simulated data that have been created by using the DBE simulator package delivered in June 2006 and described in deliverable D12.3.

6 APPENDIX A: User's manual for the package studying the dynamics of a DBE when new companies can join the system

This software is in package code_add.tar.gz

The purpose of this software package is to allow one to study a simulated DBE under various controlled conditions.

The developed software is written in C. It uses the stored data of the simulated DBE, which has the same statistical characteristics as the real DBE, in order to study the dynamics of the model.

The program has to run in a directory where a simulated DBE is stored. This simulated DBE has to have been created by using the software package **dbe-mc**, which was delivered in June 2006. The file with the data of the created DBE has to be called **aditya500.txt**. In the same directory you need to have the following files to read these data: **get_routines.c**, **initfunctions.c**, **structs.h**, **market.c**, **linkdealers.c**, **readfunction.c**, **input.txt** and **main.c**. To run the program in linux, type: **./main**. This is the name of the executable file of the program. Only the executable is supplied. It is made up from the following functions:

- **main.c**, includes the main function (starts from reading the DBE model), and contains the following functions:
  excitatory_function: which computes the excitatory signal a company C_i receives from company C_j.
  sigmoid : which computes the value of the sigmoid function in the range $[\Gamma 1, \Gamma 2]$.
  excitatory_sigmoid: computes the term that models all excitatory signals the company C_i receives from all other companies.
  inhibitory_function: which computes the inhibitory signal C_i receives from C_j. and the value of $\tilde{K}_i$, the number of companies that send inhibitory signals to company C_i.
  inhibitory_sigmoid: computes the term that models all inhibitory signals the company C_i receives from all other companies.
  inhibitory_excitatory: computes the difference between the terms computed from **excitatory_sigmoid** and **inhibitory_sigmoid**.

- If the program is successfully executed, the program returns a message on the screen, with the number of companies remaining in the system and the corresponding iteration number. The fitness values for each company at each iteration are stored in a file named **fitness.txt**. **0** value indicates that the corresponding company was removed from the ecosystem.

The parameters that determine the behaviour of the system are read from file **input.tex**. An example entry of these parameters is shown in table 2. In this file the parameters are written in a single line in the order they appear in this table. The meaning of these parameters is as follows.

- y_i **(initial fitness value)**: The values of y_i are initialised to be equal to 0.5 at the first step.

- T_1 : Threshold of the dissimilarity value used to decide when two products are similar when a company is to buy a product another company sells. Recommended values $0.1 - 0.2$.
- T_2 : Threshold of the dissimilarity value used to decide when two products are similar, when two companies sell the same product. Recommended values $0.1 - 0.2$.
- T_3: Threshold value used to remove companies from the ecosystem when their fitness value is below T_3 during each updating cycle.
- I_i **(external input to the company)**: Its value should be different for different companies. For a closed system it should be 0.
- I_0 **(background input to the company)**: Its value should be the same for all companies. For a closed system it should be 0.
- τ **(time decay constant)**: Its value must be positive and of order one.
- J_0 **(Self-excitation constant)**: Its value must be positive and of order one.
- $[\Gamma 1, \Gamma 2]$ **(range of linearity of the positive gain function)**: values of order one. It is recommended that the starting value of the y_i is between $\Gamma 1$ and $\Gamma 2$
- **Iteration**: Maximum numbers of update cycles allowed.

The random number of companies added at each iteration step are precomputed and stored in file **A2.txt**. The extra number of companies have to be included in the simulated DBE stored in file **aditya500.txt**. The last entry in input file **input.tex** indicates how many of these companies will be used to start the DBE. The random numbers in file **A2.txt** have to add up to be equal to the number of the remaining companies in the **aditya500.txt** file. These remaining companies will be added to the DBE gradually. It is recommended that at least a DBE with 500 companies is created and that the initial number of companies is 100, so the number of gradually added companies is 400.

y_i	T_1	T_2	T_3	I_i	I_0	τ	J_0	Γ_1	Γ_2	Iteration	Initial No. of Companies
0.5	0.1	0.1	0.3	0.0	0.0	2.1	0.3	0.3	0.7	200	100

Table 2: Input parameters as they should appear in file **input.tex**

7 APPENDIX B: User's manual for the package studying the dynamics of a DBE when companies can predict their future

This software is in package code_change.tar.gz

The program has to run in a directory where a simulated DBE is stored. This simulated DBE has to have been created by using the software package **dbe-mc** which was delivered in June 2006 with deliverable 12.3.

The file with the data of the created DBE has to be called **aditya.txt**. In the same directory you need to have the following files to read these data: **get_routines.c**, **initfunctions.c**, **structs.h**, **market.c**, **linkdealers.c**, **readfunction.c**, **input.txt** and **main.c**. To run the program in linux, type: **./main**. This is the name of the executable file of the program. Only the executable is supplied. The package is made up from the following functions:

- **main.c**, includes the main function (starts from reading the DBE model), and contains the following functions:
  excitatory_function: which computes the excitatory signal a company C_i receives from company C_j.
  sigmoid : which computes the value of the sigmoid function in the range $[\Gamma 1, \Gamma 2]$.
  excitatory_sigmoid: computes the term that models all excitatory signals the company C_i receives from all other companies.
  inhibitory_function: which computes the inhibitory signal C_i receives from C_j. and the value of $\tilde{K}_i$, the number of companies that send inhibitory signals to company C_i.
  inhibitory_sigmoid: computes the term that models all inhibitory signals the company C_i receives from all other companies.
  inhibitory_excitatory: computes the difference between the terms computed from **excitatory_sigmoid** and **inhibitory_sigmoid**.
  list_for_want_product: Stores the products in demand in the DBE in decreasing order of demand.
  GameTheory_SelectProduct: It selects from the products the strongest company sells the one that is in most demand. It also selects from the products the weakest company sells, the one that is in least demand.
  ChangingProduct: It replaces the product least in demand of the weakest company with the product most in demand of the strongest company.

- If the program is successfully executed, the program returns a message on the screen, with the number of companies remaining in the system and the corresponding iteration number. It also returns the name of the strongest company, with its best product and the name/names of the weakest company, with the product (which was replaced). The fitness values for each company at each iteration are stored in a file named **fitness.txt**. **0** value indicates that the

corresponding company was removed from the ecosystem.

y_i	T_1	T_2	T_3	I_0	I_i	τ	J_0	Γ_1	Γ_2	Iteration	InputComp
0.5	0.1	0.1	0.3	0.0	0.0	2.1	0.1	0.3	0.7	110	2

Table 3: Input parameters as they should appear in file **input.tex**

The parameters that determine the behaviour of the system are read from file **input.tex**. An example entry of these parameters is shown in table 3. In this file the parameters are written in a single line in the order they appear in this table. The meaning of these parameters is as follows.

- y_i **(initial fitness value)**: The values of y_i are initialised to be equal to 0.5 at the first step.
- T_1 : Threshold of the dissimilarity value used to decide when two products are similar when a company is to buy a product another company sells. Recommended values $0.1 - 0.2$.
- T_2 : Threshold of the dissimilarity value used to decide when two products are similar, when two companies sell the same product. Recommended values $0.1 - 0.2$.
- T_3: Threshold value used to remove companies from the ecosystem when their fitness value is below T_3 during each updating cycle.
- I_0 **(background input to the company)**: Its value should be the same for all companies. For a closed system it should be 0.
- I_i **(external input to the company)**: Its value should be different for different companies. For a closed system it should be 0.
- τ **(time decay constant)**: Its value must be positive and of order one.
- J_0 **(Self-excitation constant)**: Its value must be positive and of order one.
- $[\Gamma 1, \Gamma 2]$ **(range of linearity of the positive gain function)**: values of order one. It is recommended that the starting value of the y_i is between $\Gamma 1$ and $\Gamma 2$
- **Iteration**: Maximum number of update cycles allowed.
- **InputComp**: Maximum number of weak companies, one product of which will be replaced.