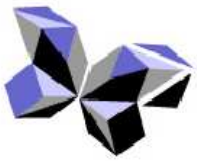


Digital Business Ecosystem

Contract nº 507953

WP12: Optimisation

Del 12.4: Issues of Trust in a DBE model

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

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

Deliverable number: D12.4
Due date: 31/10/2006
Delivery Date: 29/10/2006

Short Description: We use the software package we presented in D12.3 to study the effect of trust on the creation of chains of products. The trust between companies is modelled by using the Psychophysical law of Weber-Fechner. We present a software package that creates chains of products of any length, using either a greedy algorithm or dynamic programming. In either case the trust between the companies may or may not be taken into consideration, either as trust from the buyer to the seller of the next product in the chain, or as mutual trust between the two suppliers of two successive products in the chain. The quality of a chain may be measured by measuring the quality of matching all pairs of successive products in the chain. It is shown that chains of sub-optimal objective quality may be produced if subjective feelings of trust are taken into consideration when creating the chains. It is argued, however, that the over-all quality of a chain is not sub-optimal, if the subjective feelings of trust reflect the customer satisfaction and the smooth co-operation of the agents that form the chain. We supply the manual for using the delivered software.
Keywords: Product chains; trust; dynamic programming; greedy algorithm

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

Versioning		
Version	**Date**	**Author, Organisation**
1	31/10/2006	Petrou, Giannoutakis & 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

A Digital Business Ecosystem (DBE) is a closed or semi-closed system of Small and Medium Enterprises (SMEs), which may come together in cyberspace in the same way companies gather in a business park in the physical world. These companies will interact with each other through buyer-seller relationships.

There have not been many attempts to study DBEs quantitatively. Most published papers follow the procedure of hypothesis generation, data collection by a survey or a questionnaire and finally hypothesis testing using statistical methods (eg [6, 3, 9, 2]). There are various reasons for that: the complexity of the system, the multiplicity of the issues involved, and of course the lack of uniformity in the description of products and services, necessary to study the dynamics of a complex system [13]. The lack of such studies and lack of quantitative measures that they could yield has consequences in the formation of economic policies for the internet [8]. The problem of lack of uniformity in the data was addressed in [11], by creating a realistic simulated DBE that shares its statistical properties with a real DBE. In this report we use the data of a simulated DBE in order to study the role of trust in the creation of product chains offered in a DBE. Although trust has been acknowledged to play a significant role in internet transactions [3], it is very difficult to quantify and study it.

Manchala in [7] makes a serious attempt to quantify trust by counting the number of transactions a vendor feels they need to verify before they proceed with the actual transaction. Manchala stresses the need for quantitative measures of trust in order to make quantitative studies of such systems. In [11] trust was quantified by invoking the psychophysical law of Weber-Fechner. The approach was not incompatible with the approach of Manchala: Manchala starts from some objective measure; the authors in [11] start from qualitative categories of trust and try to infer from them some objective rankings. In a sense, if people were asked to use the quantitative measure of Manchala to create categories of trust, it is believed that they would create categories that could be modelled by the psychophysical law of Weber-Fechner. This law can bridge the gap between models like the one presented in [4], which uses qualitative terms like “low risk”, “high risk” etc, and more quantitative studies like the one in [14].

The importance of trust on web based transactions has been stressed by many researchers [5], to the point that there are even papers on how to build web-based systems that inspire trust to the customer [10, 9, 1]. Other people have studied the effect of trust by looking at the way web-sites evolve over time, their structure and of course by conducting surveys [12]. In this report we examine the effect business-to-business trust has on the creation of product chains.

This report is structured as follows. In section 2 we give a brief overview on the creation of the simulated DBE that shares its statistical properties with an observed real DBE. In section 3 we show how we can create chains of products or services and how we may quantify the quality of each chain using either a greedy algorithm or dynamic programming. We also show how the quality of a chain may be measured to reflect the trust between the co-operating agents. In section 4 we present some experimental results to demonstrate the role trust plays in product chain creation. Finally, we conclude in section 5.

2 A Simulated DBE

Here we briefly overview the methodology on how to construct a realistic simulated DBE, based on observations of a real DBE, first presented in [11]. One creates a database of virtual companies and products that have the same statistical properties as the real ones in the observed DBE. Each company created is assigned a SELL and a WANT list. The SELL list of a company is the list of the products the company wants to sell and the WANT list of a company is the list of products the company wants to buy. Either of these lists might be empty, but not both. All products which appear in the SELL lists of all companies make up the database of "real products". All products which appear in the WANT lists of all companies make up the database of "virtual products", because these products exist in the customers' minds. A product may appear in both databases, but it most likely will have different attributes in the two databases. A product has the same name, type and number of attributes no matter in which of the two databases it appears. What changes from one database to the other is the statistics of the attributes which characterise each product. For example, the average price of a product in the market may be higher than the average price a customer expects to pay. These statistics are chosen to agree with the statistics of the real products in the observed real DBE. Two types of attribute are catered for, numerical and symbolic. The statistics of the numeric attributes are characterised by their mean and standard deviation, which are assumed to be extracted by observing a real DBE. Each symbolic attribute takes values from a list of symbols, with probabilities according to the frequency with which each such value is encountered in the real DBE.

In addition, each company is assigned two other lists: the TRUST list which contains the names of the other companies in the ecosystem that it trusts, and the MISTRUST list which contains the names of the companies that it mistrusts. Any company that does not appear in either of the two lists is unknown to the company in terms of trustworthiness. The effect of the spreading of the reputation of each company is also taken care of when creating these lists. These lists are created by choosing randomly companies from the ecosystem. However, when populating the TRUST or MISTRUST list of a company, an extra weight is given to those companies which have appeared in already created corresponding lists of other companies. To model the fact that good reputation propagates faster than bad reputation (because companies try to hide their bad reputation, while they advertise greatly any positive aspect of them, like for example an award they received), the weight used to account for the good reputation of a company, when creating the TRUST list of another company, is higher than the weight used to account for the bad reputation of a company when creating the MISTRUST list of another company.

The psychophysical law of Weber-Fechner is used in order to convert the qualitative concepts of trust, indifference and mistrust to numerical weights for the case one wishes to construct numerical models to study these factors. The idea is to use this law to go from subjective classes of trust to relative numerical measurements that somehow reflect objectivity. According to this law, the degree of subjective judgement is proportional to the logarithm of an objective measure that measures the same effect. For example, if in your mind you create categories of untrustworthiness and you call them

1, 2 and 3, the people whom you classify in these categories have to lie to you twice, four times or eight times, respectively, for you to put them in the respective categories. So, categories of mistrusted, indifferent and trusted correspond to some arbitrary objective numerical values proportional to 2, 4 and 8, respectively. As these values have to be used to weigh relatively the various possible transaction partners, their exact values do not matter. To make them into relative weights, these values are normalised to sum up to 1.

Each product in the DBE is assigned two arrays of links of products: one array of forward links and one of backward links. For example, if we are interested in a holiday package, then we may need products or services such as air-tickets, taxis, hotels etc. In order to form a successful chain of all these products, we have to match some crucial attributes that characterise them, eg the airport of destination of the flight has to match with the airport at which a taxi firm operates. If the airport of arrival matches with the airport at which the firm operates, we say that the flight has a potential to have a forward link to the taxi and the taxi has a potential to have a backward link to the flight. Only a certain number of attributes have to match in order to form a link between two products.

3 Chains of products

In order to create chains of products we have to define a measure of quality of a chain. We define this to be the sum of the matching values of the pairs of successive links in the chain. Then we need to define the matching value of two linked products. Let us assume that product P_i links forward to a product P_j and d pairs of attributes have to match. We define the quality of this match as

$$q_{i,j} \equiv e^{-V_{ij}} \tag{1}$$

where

$$V_{ij} \equiv \sum_{l=1}^{d} w_{l;ij} V_{l;ij} \tag{2}$$

is the total mismatch value for those two products, by comparing the d pairs of attributes that have to match for the two products to be linked, and

$$V_{l;ij} \equiv 2\frac{|x_{l;i} - x_{l;j}|}{x_{l;i} + x_{l;j}} \tag{3}$$

if matched attribute l is numeric, and

$$V_{l;ij} \equiv \begin{cases} 0 & \text{if symbolic attributes are identical} \\ 1 & \text{if symbolic attributes are different} \end{cases} \tag{4}$$

if matched attribute l is symbolic.

$V_{l;ij}$ in equations (3) and (4) is the mismatch value of matched attribute l. The denominator in expression (3) is used to normalise its value so that it is of order 1, like the mismatch value of matched symbolic attributes, irrespective of the different units used to measure each attribute. Weights $w_{l;ij}$ express the relative significance of the different attributes that have to be matched, according to customers. By default they may all be equal to $\frac{1}{d}$. However, in a DBE where customers are allowed

to express preference as to which attributes of a product matter to them most, the values of $w_{l;ij}$ may also be defined according to the Weber-Fechner law [11]. For example, if the duration of waiting for a connecting flight is more important to the customer than the airport at which they make the connection, the weight used when we match the arrival and departure times of the flights should be higher than the weight used when we match the airports or the terminals at which one flight arrives and the next departs. In the experiments presented in the next section we set $w_{l;ij} = \frac{1}{d}$ for all matched attributes.

The overall objective quality of a chain of products is then defined as

$$Q_i^k \equiv \sum_{i=1}^{N_k-1} q_{i,i+1} \tag{5}$$

where k is an index identifying a particular chain of N_k products and $q_{i,i+1}$ is the quality of linking (according to equation (1)) the i-th product in the chain with the next one.

In the above definitions two products are linked as long as their relative attributes match. However, for two products to be linked the supplier companies have also to trust each other and be willing to co-operate in the formation of the link. To take trust into consideration in the formation of product chains we may modify definition (5) into:

$$Q_i^k \equiv \sum_{i=1}^{N_k-1} t_{i,i+1} q_{i,i+1} \tag{6}$$

where $t_{i,i+1}$ is the trust between the company that supplies product P_i and the company that supplies product P_{i+1}. For the specification of $t_{i,i+1}$ we considered two options:

Option 1: $t_{i,i+1}$ is the trust the company that sells product P_i has to the company that sells product P_{i+1}. This is the case when *the company that sells product P_i buys the services of the company that sells product P_{i+1} in order to enhance the sale of its own product.*

Option 2: $t_{i,i+1}$ is the product of the trust the company that sells product P_i has to the company that sells product P_{i+1}, times the trust the company that sells product P_{i+1} has to the company that sells product P_i. This is the case of *mutual trust*.

The problem we are trying to solve next is: "Find the best product chain which starts from each product on sale in the DBE". This way, if a customer chooses any product, the system will immediately be ready to suggest to them a chain of follow up products that may link to the product they chose. We tested two different algorithms for finding the best possible chains. The first algorithm is a greedy algorithm which links each product with the forward link that gives the highest quality according to (1). Thus, the greedy algorithm always finds the best forward link, ignoring that accepting the second best forward link may lead to a better overall quality of chain.

The second algorithm for chain creation is based on dynamic programming. Dynamic programming does not produce the overall optimal solutions in NP complete optimisation tasks, but good

sub-optimal solutions. To obtain the optimal chains we must use stochastic optimisation approaches, like genetic algorithms or simulated annealing, which are notoriously slow. Exhaustive search is not possible as the problem is NP complete.

The dynamic programming algorithm we propose is an iterative algorithm. At step k, we assume that we have the best chain of length k starting from each product on supply. We then try to find the best chain of length $k+1$. This chain of length $k+1$ for product P_i is chosen according to the rule

$$\tilde{Q}_i^{k+1} = \max_j \left(q_{ij} + \tilde{Q}_j^k \right) \tag{7}$$

where $\tilde{Q}_j^k$ is the best chain of length k starting from product P_j with which product P_i can form a forward link. When trust is taken into consideration equation (7) is modified to

$$\tilde{Q}_i^{k+1} = \max_j \left(t_{ij} q_{ij} + \tilde{Q}_j^k \right) \tag{8}$$

The dynamic programming algorithm is as follows:

1. Set target length of chain equal to K;
2. Consider a product P_i;
3. Consider all products $P_j \neq P_i$ with which P_i can form a forward link. Say they are X_{P_i};
4. Consider the X_{P_i} chains of length $k < K$ associated with these products;
5. Form the X_{P_i} chains of length $k+1$ by placing product P_i in front of each one of the X_{P_i} chains;
6. Choose the best of them;
7. Drop the previous chain of length k that was starting from P_i and replace it with the chosen chain of length $k+1$;
8. Do that for all products P_i;
9. Repeat until you form chains of length K.

4 Experimental results

We created a simulated DBE of 100 companies and 984 products on supply. We tested both algorithms in forming chains of length 5. We compared the quality of each chain starting from each product when produced with the greedy algorithm and when produced with the dynamic programming algorithm and we concluded that, in general, the dynamic programming algorithm creates chains of higher value. The superiority of the dynamic programming algorithm over the greedy algorithm is clearly seen in the histograms of Fig. 1, Fig. 2 and Fig. 3.

In Fig. 1, we present the distribution of qualities of chains produced with the greedy algorithm (a) and with the dynamic programming algorithm (b), when the trust between companies is not considered. We can see that, when the chains of products are produced with the dynamic programming algorithm the distribution is shifted more to the right than when the chains are produced with the greedy algorithm.

We got similar results even when the trust between companies was taken into consideration. In Fig. 2 we have again a massive shift of the data towards greater values in terms of quality, when the chains are produced with the dynamic programming algorithm (b), than when created with the greedy algorithm (a). In this case, the trust is defined as in **Option 1**. However, the situation does not change much even if the trust is defined as in **Option 2** (Fig. 3).

In Fig. 4 we plot the quality of chain of length 5 starting from each product, produced with the greedy algorithm and the algorithm based on dynamic programming. The dynamic programming algorithm created chains with higher quality, compared with the greedy algorithm for 797 products, the greedy algorithm produced better chains for only 12 products and both algorithms created equal quality chains for 175 products. Since the data in Fig. 4 are too compact, in Fig. 5 we zoom in the first 100 products of Fig. 4. It is then clearly seen that the algorithm based on dynamic programming produces better results.

In the above experiments, we produced chains of products according to equations (5) and (7) for the greedy and the dynamic programming algorithm, respectively. In other words, we did not consider the trust between companies. In the following, we present how trust affects the formation of chains of products and we compare the two algorithms in terms of the quality of the chains they produce. We present the results for all 984 products, but we plot only those of the first 100.

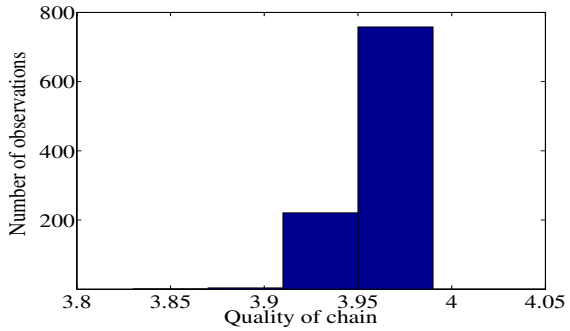
Fig. 6 shows the results when the trust issue is included according to **Option 1**. The dynamic programming algorithm did better in 633 products, worse in 140 products and both algorithms did equally well in 211 products.

Fig. 7 shows the results when trust is incorporated according to **Option 2**, ie the mutual trust. The dynamic programming performed better in 634 products, the greedy algorithm in 140 products and both algorithms did equally well in 210 products. Again, we see that trust enables the greedy algorithm to perform better than the dynamic programming algorithm for more products than in the previous experiments.

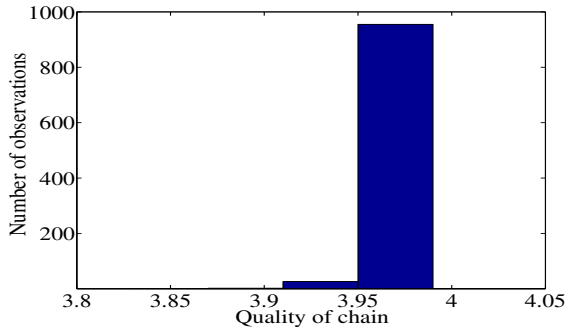
To examine the effect of trust in the formation of chains of products we decided to compare the quality values of the chains produced by dynamic programming with and without trust. To have all quality values measured in the same scale, we recomputed the quality of each chain produced by the algorithm using trust, using formula (5). The quality values of the chains starting from each product produced with and without trust being considered are shown in Fig. 8. It can be inferred that trust between companies influences the formation of chains. If one company does not trust another, that may influence their decision to co-operate. On the other hand, if a company trusts another more than it trusts a third one, it may decide to co-operate with its trusted company instead of the third company, even if the product of the third company matches with its product better than the product of the trusted one does. In both cases, we get chains of products with lower objective quality value

than the best possible.

Both algorithms are very fast, with cpu times in the order of fractions of a second for relatively small chains of products, say 5-long chains, and slightly longer times, in the order of a few seconds for chains of say 30 products.



(a)



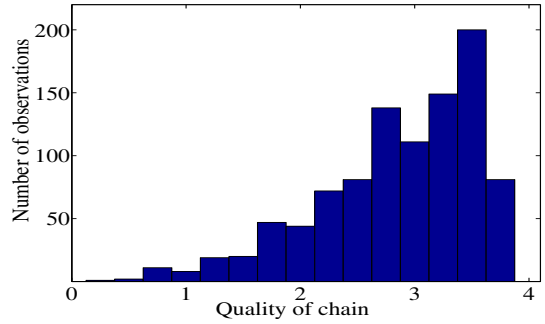
(b)

Figure 1: Histograms of the values of the qualities of the product chains produced by (a)the greedy algorithm and (b) the dynamic programming algorithm without trust.

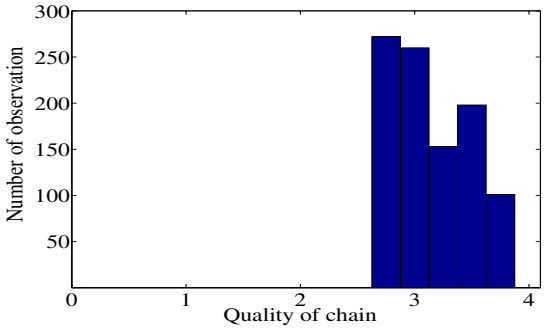
5 Conclusions

We presented a model for simulating a Digital Business Ecosystem, based on statistical data collected from a real DBE. We may relatively easily acquire statistical data by using, for example, a web robot, or another program designed for that purpose. The idea then is to use the gathered statistical data to produce a simulated version of the DBE which shares the same statistical properties as the real DBE. Each product in the DBE was allowed to form forward links with some other products, by matching specific pairs of attributes. The creation of chains of any length of products is possible. Such methodologies have been used for many years by scientists to study complex systems that cannot be modelled in a deterministic way. For example, astronomers have learnt a lot about the dynamics of galaxies by studying simulated models of them.

Further, we presented a model for the creation of chains of products and we introduced a measure for the quality of a chain of products, by matching pairs of attributes between two products. Two algorithms were tested: a greedy algorithm and an iterative algorithm based on dynamic programming. The algorithm based on dynamic programming created, in general, chains of higher quality

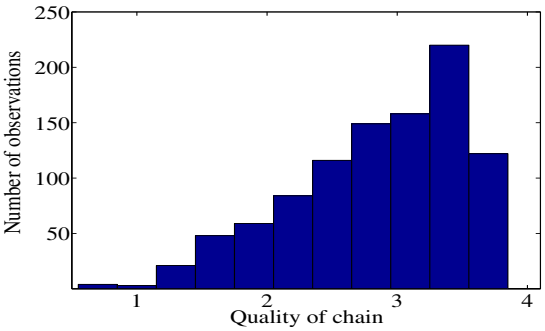


(a)

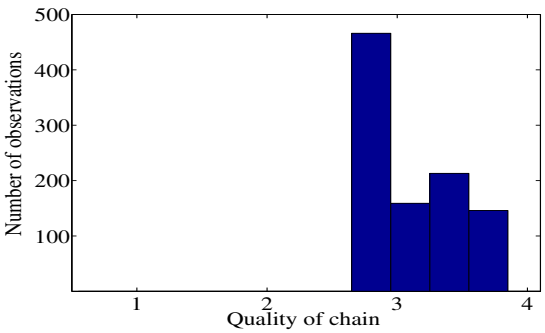


(b)

Figure 2: Histograms of the values of the qualities of the product chains produced by (a) the greedy algorithm and (b) the dynamic programming algorithm with trust according to **Option 1**.



(a)



(b)

Figure 3: Histograms of the values of the qualities of the product chains produced by (a) the greedy algorithm and (b) the dynamic programming algorithm with trust according to **Option 2**.

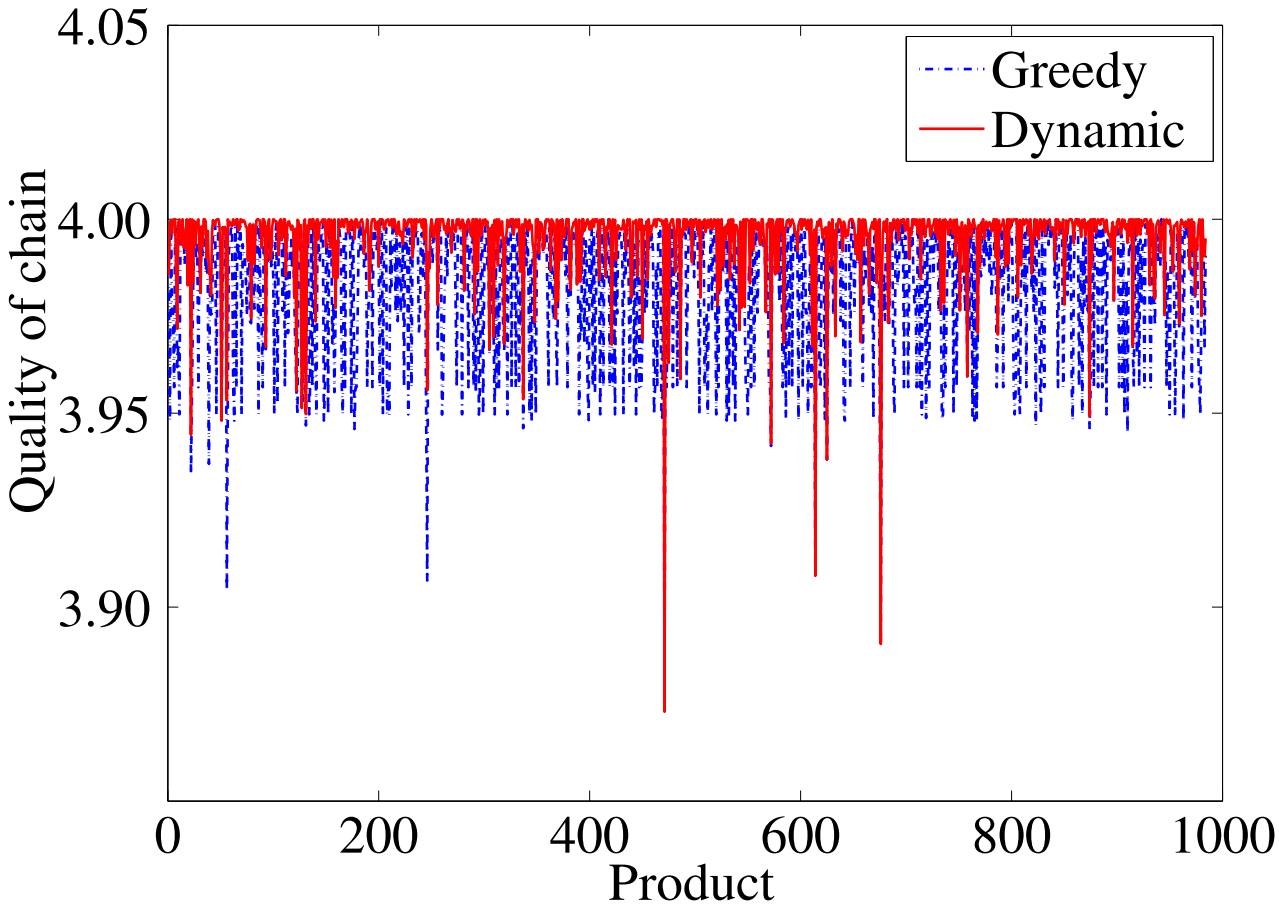


Figure 4: Difference in quality of chains of products produced by the dynamic programming and the greedy algorithm, without the trust issue.

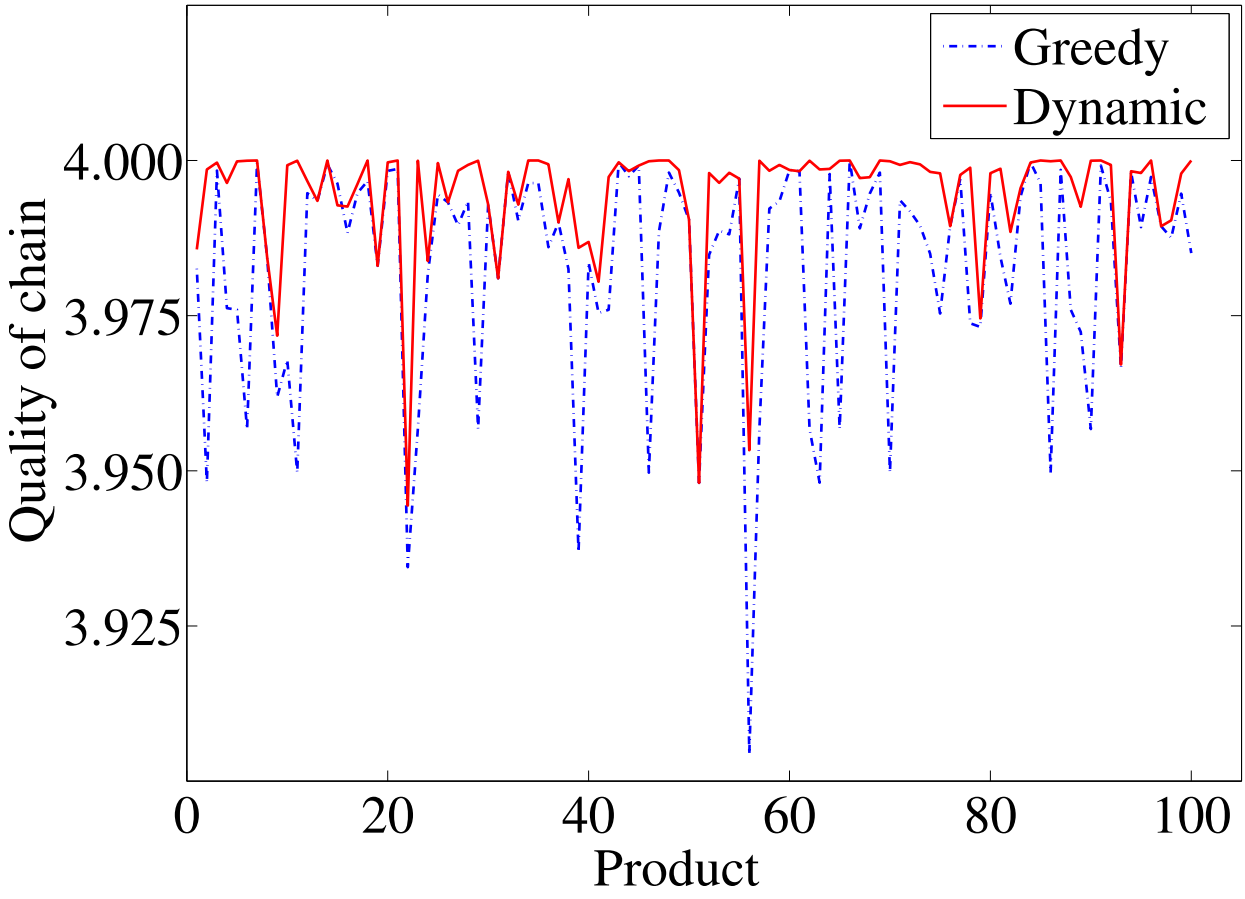


Figure 5: Difference in quality of chains for the first 100 products produced by the greedy algorithm and the dynamic programming algorithm, without the trust issue.

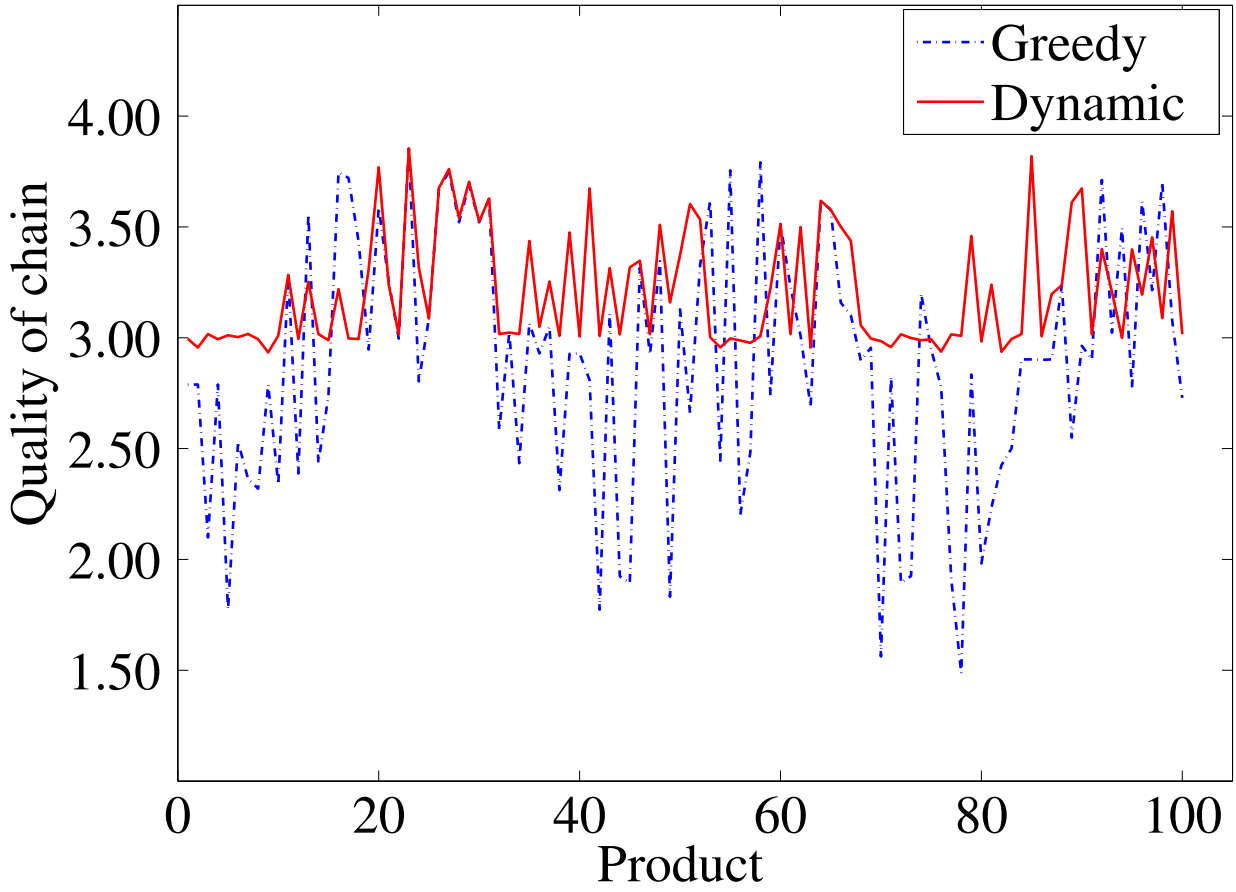


Figure 6: Difference in quality of chains for the first 100 products produced by the greedy algorithm and the dynamic programming algorithm, with trust according to **Option 1**.

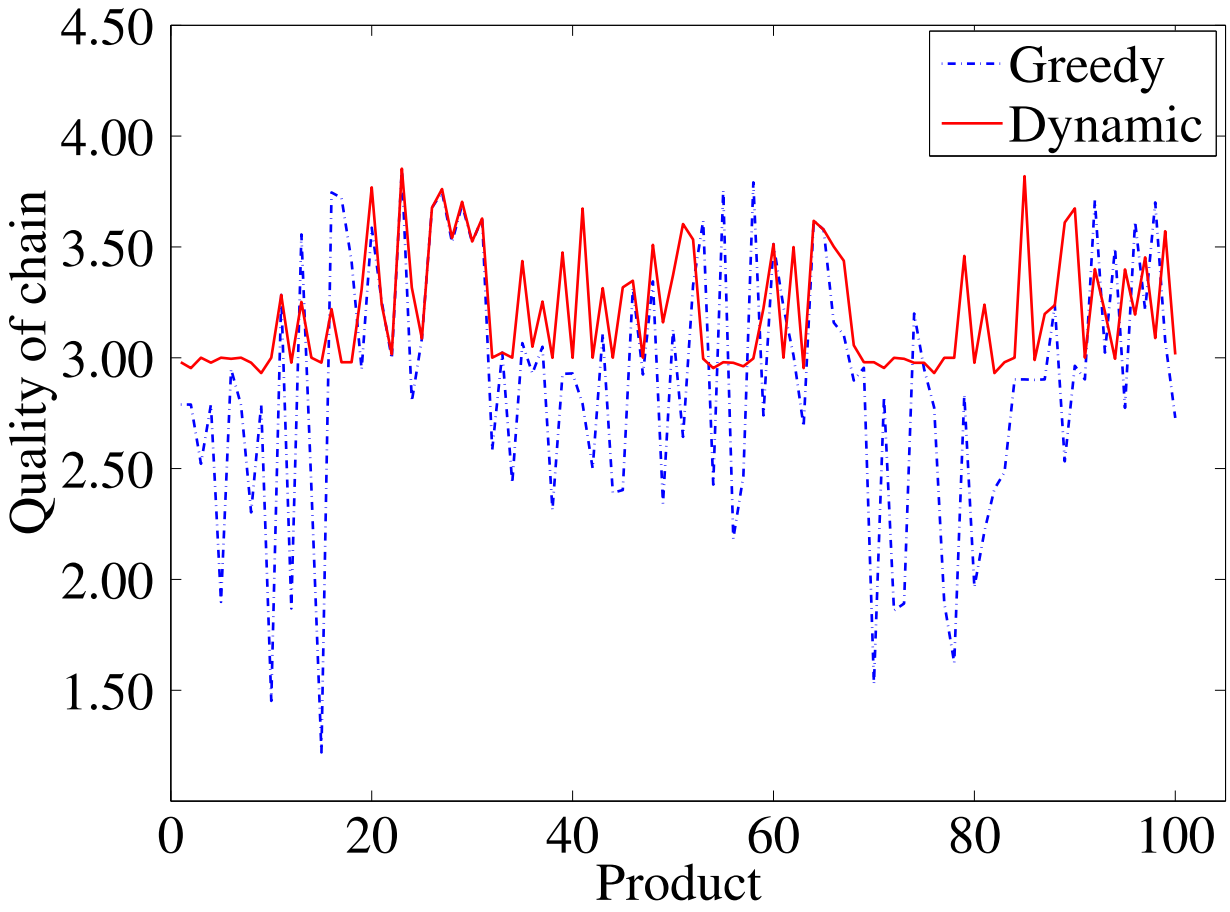


Figure 7: Difference in quality of chains for the first 100 products produced by the greedy algorithm and the dynamic programming algorithm, with trust according to **Option 2**.

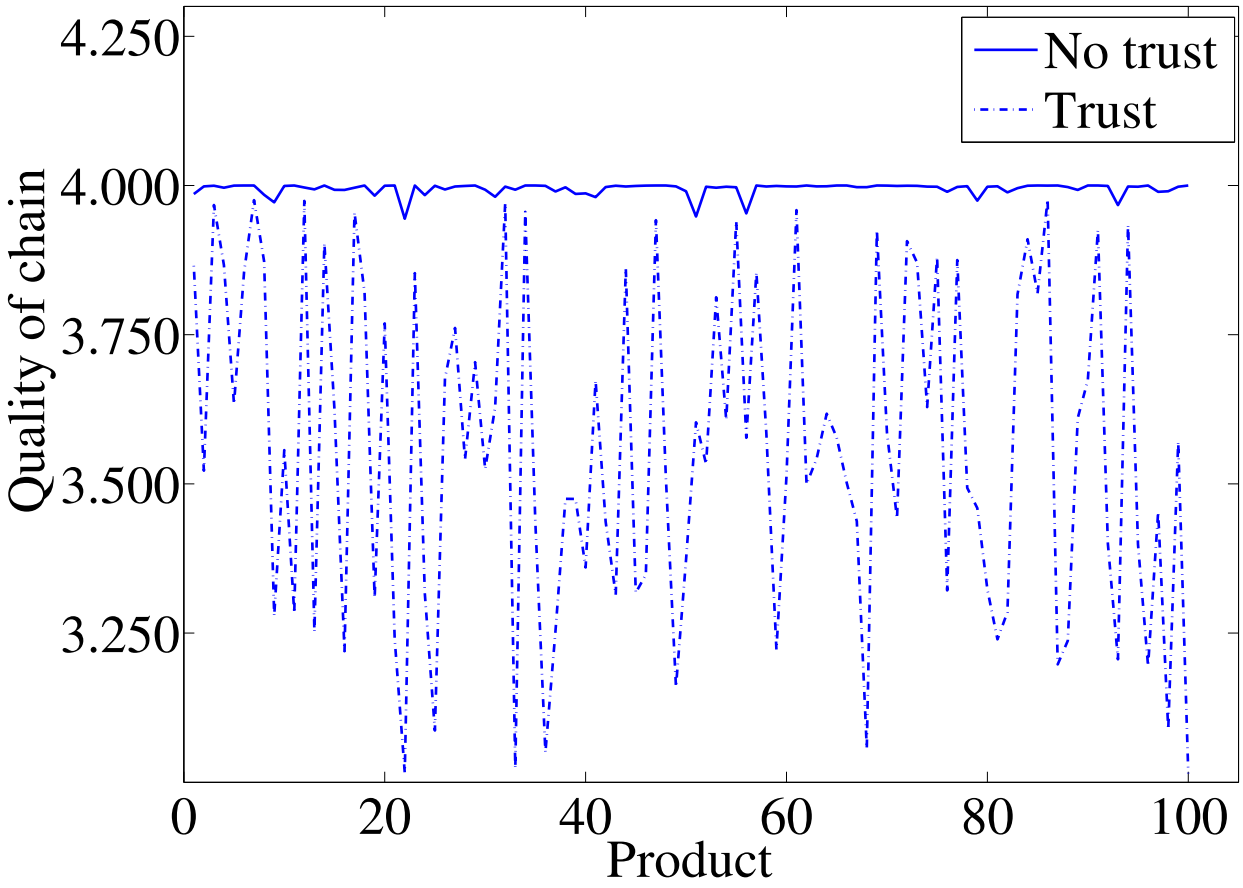


Figure 8: The effect of trust on the quality of chains of products produced.

than the greedy algorithm.

Finally, we examined the effect of trust between companies in the formation of chains of products. It was demonstrated, not surprisingly, that trust will affect the quality of a chain in objective terms because it incorporates subjectivity in the choices made. However, the overall quality of the chains produced this way may be better in terms of reliability and customer satisfaction, as the constituent parts will work better together and will interact with each other more effectively than in chains where the subjective feelings of the companies that have to co-operate are not taken into consideration.

References

[1] D. P. Cook and W. Luo, The role of third-party seals in building trust on line, e-Service Journal, Indiana University Press, Vol. 2, Issue 3, Summer 2003, pp. 71–84 .

[2] D. Gefen, E-commerce: The role of familiarity and trust, Omega, Vol. 28, Issue 6, December 2000, pp. 725–737.

[3] D. Gefen and D. Straub, Managing user-trust in B2C e-services, e-Service Journal, Indiana University Press, Winter 2003, Vol. 2, Issue 2, pp.7–24.

[4] J. Jahng, H. Jain and K. Ramamurthy, Effective design of electronic commerce environments: a proposed theory of congruence and an illustration, IEEE Transactions on System, Man and Cybernetics, Part A: Systems and Humans, Vol. 30, Issue 4, July 2000, pp. 456–471.

[5] S.Y.X. Komiak, W. Wang and I. Benbasat, Trust building in virtual sales-persons versus in human sales-persons; similarities and differences, e-Service Journal, Indiana University Press, Summer 2004/2005, Vol. 3, Issue 3, pp. 49–63.

[6] M. Limayem, M. Khalifa and A. Frini, What makes consumers buy from internet? A longitudinal study of on-line shopping, IEEE Transactions on System, Man and Cybernetics, Part A: Systems and Humans, Vol. 30, Issue 4, July 2000, pp. 421–432.

[7] D. W. Manchala, E-commerce trust metrics and models, IEEE Internet Computing, Vol. 4, Issue 2, March - April 2000, pp. 36–44.

[8] L. W. McKnight and J. P. Bailey, Internet economics: when constitutions collide in cyberspace, IEEE Internet Computing, Vol. 1, Issue 6, November - December 1997, pp. 30–37.

[9] A. Noteberg, E. Christiaanse and P Wallage, Consumer Trust in electronic channels, e-Service Journal, Indiana University Press, Winter 2003, Vol. 2, Issue 2, pp. 46–67.

[10] A. S. Patrick, Building trustworthy software agents, IEEE Internet Computing, Vol. 6, Issue 6, November - December 2002, pp. 46–52.

[11] M. Petrou, S. Gautam and K. N. Giannoutakis, Simulating a Digital Business Ecosystem, Proceedings of the 2nd international conference in computational finance and its applications, June 2006, pp. 277–287.

[12] C. Ruppel, L. Underwood-Queen and S. J. Harrington, e-Commerce: The roles of Trust, security and type of e-commerce involvement, e-Service Journal, Indiana University Press, Winter 2003, Vol. 2, Issue 2, pp. 25–45.

[13] M. P. Singh, The e-commerce Inversion, IEEE Internet Computing, Vol. 3, Issue 5, September - October 1999, pp. 4–5.

[14] W. K. Sung, D. Yang, S. M. Yiu, D. W. Cheung, W. S. Ho and T. W. Lam, Automatic Construction of on-line catalogue topologies, IEEE Transactions on System, Man and Cybernetics, Part C: Applications and Reviews, Vol. 32, Issue 4, November 2002, pp. 382–391.

A User's manual for the package used to create chains of products

The developed software, written in C, functions as follows:

1. It receives as input a file with the data of a simulated DBE and a file of allowed forward and backward links for each product.
2. It creates chains of products according to the target length of chains and specifications defined by the user.
3. It stores the produced chains of products in a separate file.
4. It reads the produced file by calling a routine designed for that purpose.

B Getting started

There are three packages: **greedy.tar** for creating chains using the greedy algorithm, **dynamic.tar** for creating chains using the iterative algorithm based on dynamic programming and **readchains.tar** for reading the file with the produced chains of products. To run the programs in Linux, untar the packages in their own directories. The executable for chains using the greedy algorithm is called **chain-greedy** and the executable for chains using the dynamic programming algorithm is called **chain-dynamic**. The first two packages contains 17 files each, with most of the files being common between the packages. The common files are:

1. **ComputeQuality.c** with the function **ComputeQuality()**, which receives as input two successive links in a chain and calculates the quality between the two products.
2. **CreateChains.c** with the function **CreateChains()**, which receives as input the target length of the chains and options about the trust and repetition of products in a chain and forms chains of products. When it is called with the dynamic programming algorithm it creates chains of length 3 and above.
3. **Gauss_random.c** that includes **ran1()**, the function for drawing a random number from a uniform distribution and the function **gasdev()**, which returns a random number from a normal distribution with mean 0 and standard deviation 1.
4. **Get_routines**, three functions for user and file input, ***getString()** for inserting strings, **getInt()** for inserting integers and **getFloat()** for inserting floats.
5. **InitFunctions.c** that has all the initialisation functions for all structures.
6. **links.txt** a text file with the forward and backward links of each product.
7. **Main.c**, which contains the function **main()** (it has slightly different structure from one package to another).

8. **mydbe.txt**, a text file with a simulated DBE.
9. **ReadText.c** with the **ReadText()** routine that reads the data of a simulated DBE stored in a plain text file.
10. **ReadTextLinks.c** with the **ReadTextLinks()** routine that reads the forward and backward links of each product when the DBE data are inserted from a plain text file.
11. **ReadXML.c** with the **ReadXML()** routine that reads the data of a simulated DBE stored in an XML file.
12. **ReadXMLLinks.c** with the **ReadXMLLinks()** routine that reads the forward and backward links of each product when the DBE data are inserted from an XML file.
13. **Save.c** with the function **Save()** that saves the chains of products in a separate file.
14. **Structs.h**, a header file that has all the data structures used in these programs.
15. **xmldbe.xml** an XML file with a simulated DBE.

In addition, the greedy algorithm package **greedy.tar**, contains the following file:

- **Quick.c**, which contains the Quicksort algorithm. It is called inside the function **Create-Chains()** to sort in ascending order of quality all possible forward links of a product before choosing the one with the highest quality.

In addition, the dynamic programming package, **dynamic.tar**, contains the following file:

- **Two.c** with the function **Two()** which creates chains of products of length 2. This function is called at the beginning of the program and links all products in the DBE with the match with the highest quality. This is the starting point of the algorithm which then continues by calling the function **CreateChains()** for creating chains of length 3 and above.

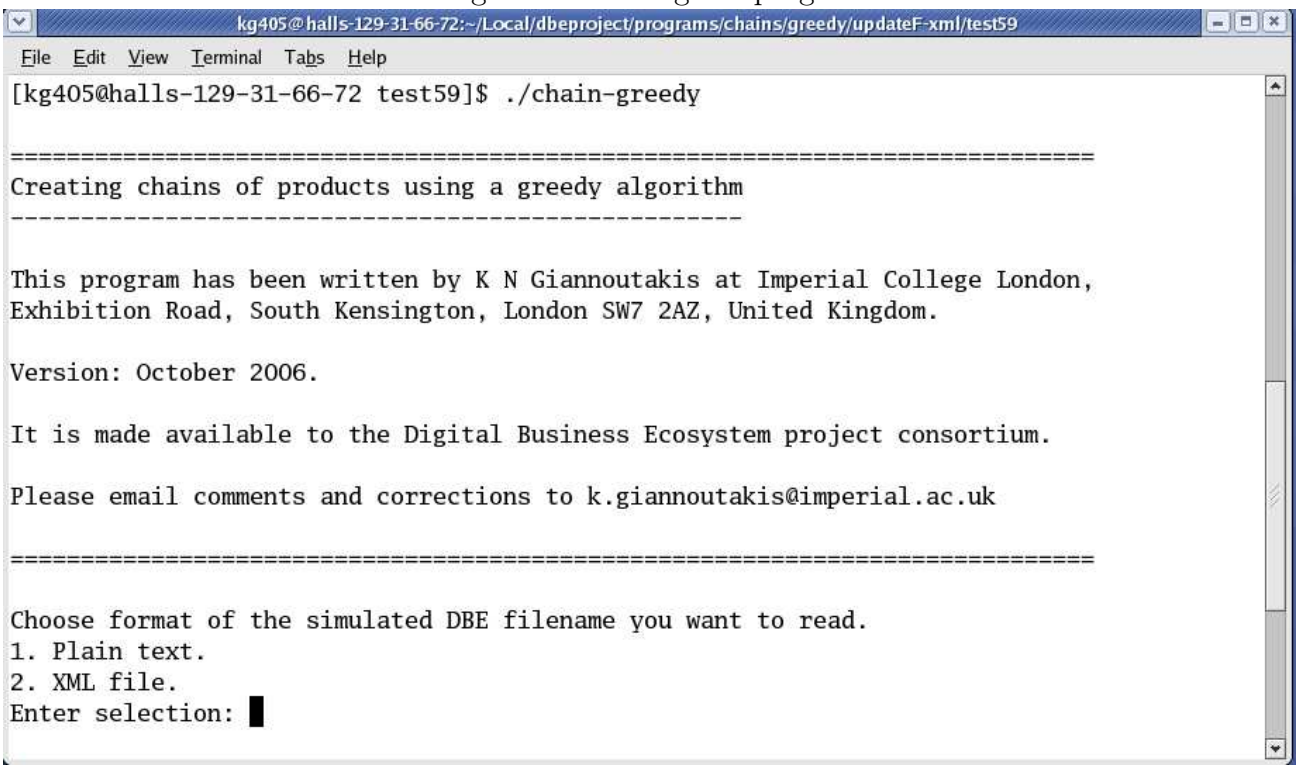
The package **readchains.tar** is explained in section "Reading the chains of products".

C Creating chains of products

The process is the same whether you are using the greedy algorithm or the dynamic programming algorithm, or whether you are inserting your DBE data from an XML file or from a plain text file. To create chains of products using the greedy algorithm type **./chain-greedy**, as shown in figure 9. To create chains using the dynamic programming algorithm type **./chain-dynamic**[1]. You can choose between plain text file and XML file to input your simulated DBE data. If you choose plain text, you only have to type the name of the simulated DBE file and it will find and open the file with the product links automatically. If you choose XML file, then you will also have to type in the name of the file with the links of products, since the XML file does not include that filename in it.

[1]If you need to compile them first type "gcc -lm -o chain-greedy Main.c", for the greedy algorithm program and "gcc -lm -o chain-dynamic Main.c" for the dynamic programming program.

Figure 9: Starting the program



C.1 Reading a simulated DBE from plain text

You have to give the name of the simulated DBE you want to read. The program opens the DBE file and finds automatically the accompanying file of the links of products, so that you do not have to type its name. Once it is done, the program displays the number of companies that exist in the DBE and then prompts you to enter the number of companies you wish to include in your simulation (figure 10). By choosing a number within the range of companies, the program displays the total number of products sold by them. Any invalid input will prompt you to try again.

One of the most necessary parameters of the function that creates chains of products is the target length of chains, ie how many products you want a chain to be formed by. As figure 11 shows, this is the parameter you enter next and it should be within the range of the total number of products on sale. By entering any wrong input the program will ask you to try again. The next option is to decide whether you want a product to appear more than once in a chain. You can answer by typing "yes" or "no" or "y" or "n" either lowercase or uppercase.

The last parameter for creating chains of products is the issue of trust. You can opt to include trust between the sellers of the products in your chains by answering yes or no (same as before) in the question about trust (figure 12). If you choose "yes", you will be asked how you prefer the trust to be considered. There are two options:

1. The trust of the seller to the company linked forward, ie the company whose products or services the seller uses to enhance their own product.

2. The level of trust between the two companies, ie the mutual trust, given as the product of the level of trust one has to another.

Figure 10: Reading from plain text

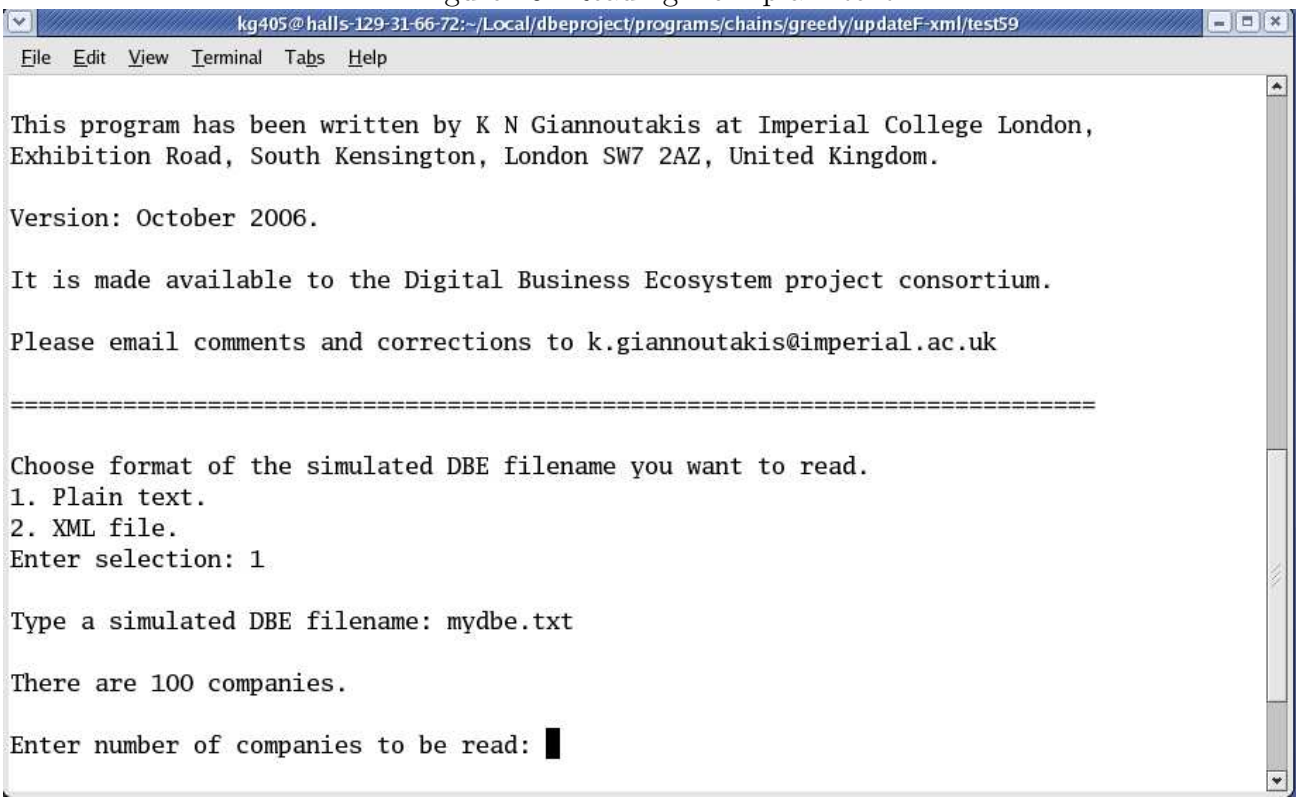


Figure 11: Set target length of chains

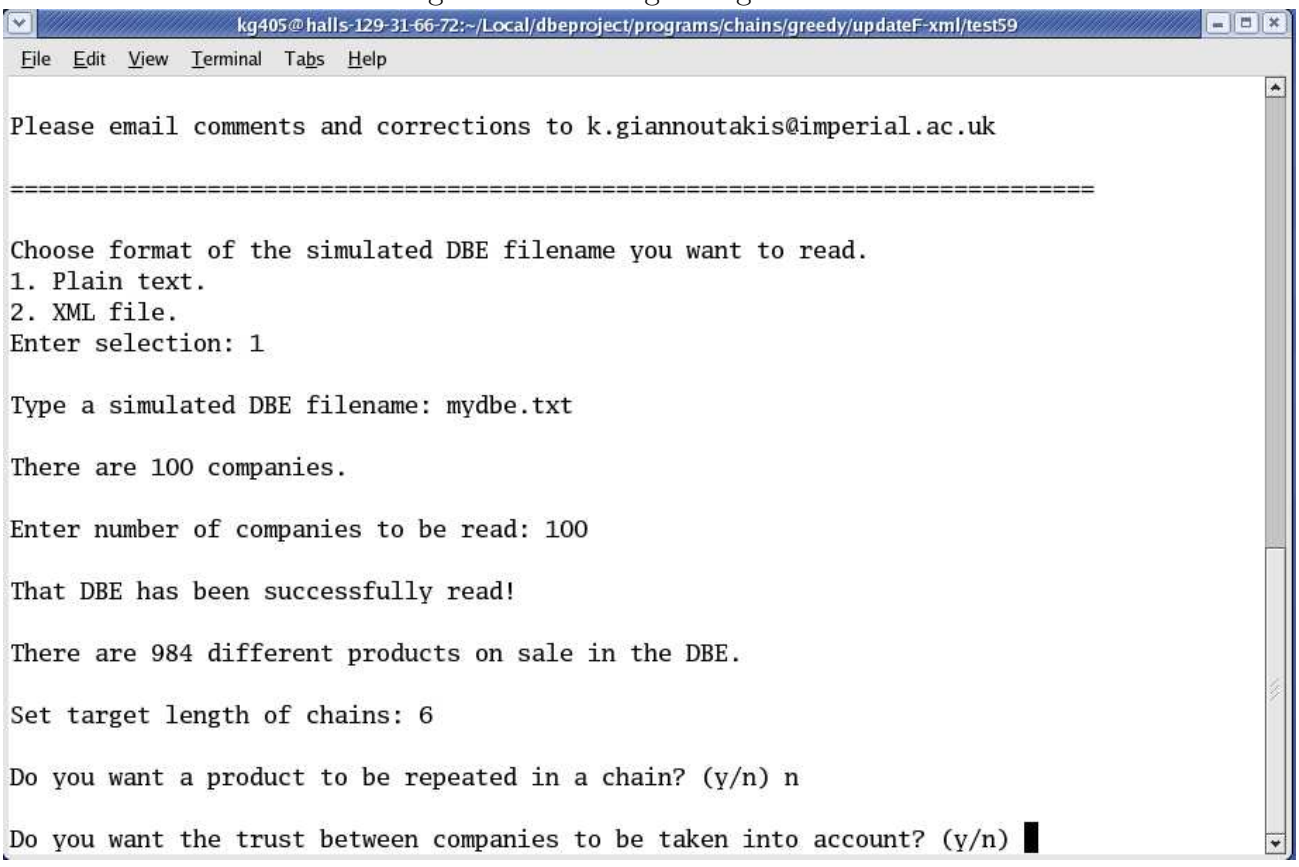


Figure 12: The trust issue

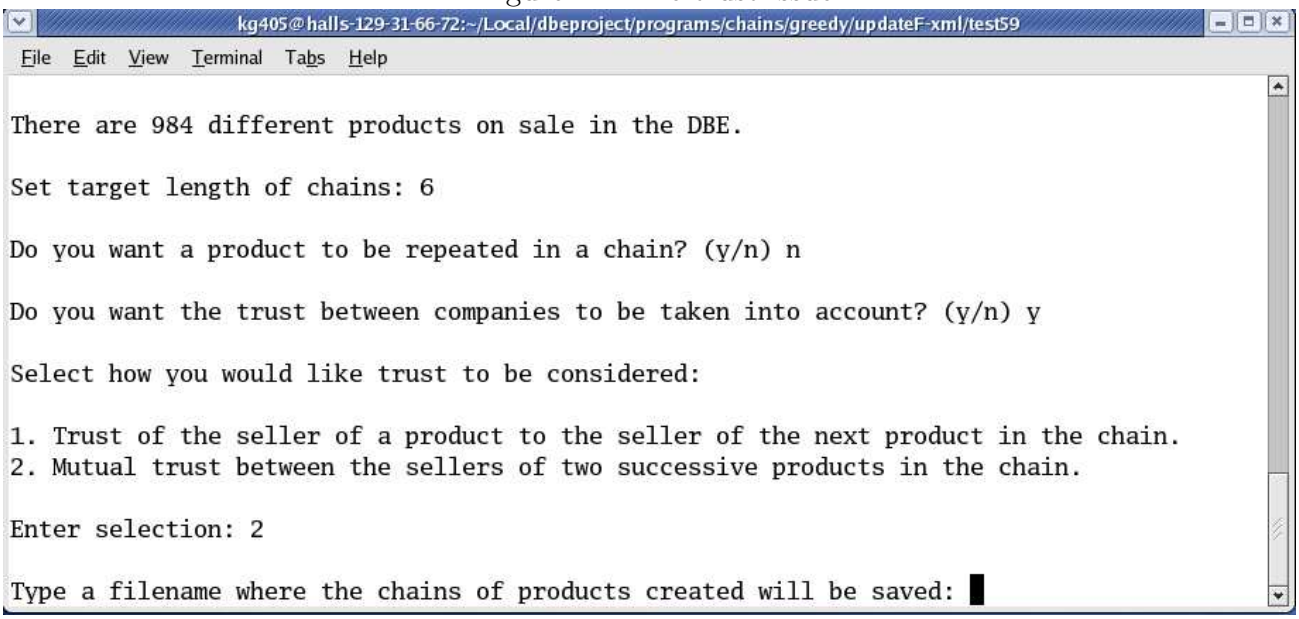
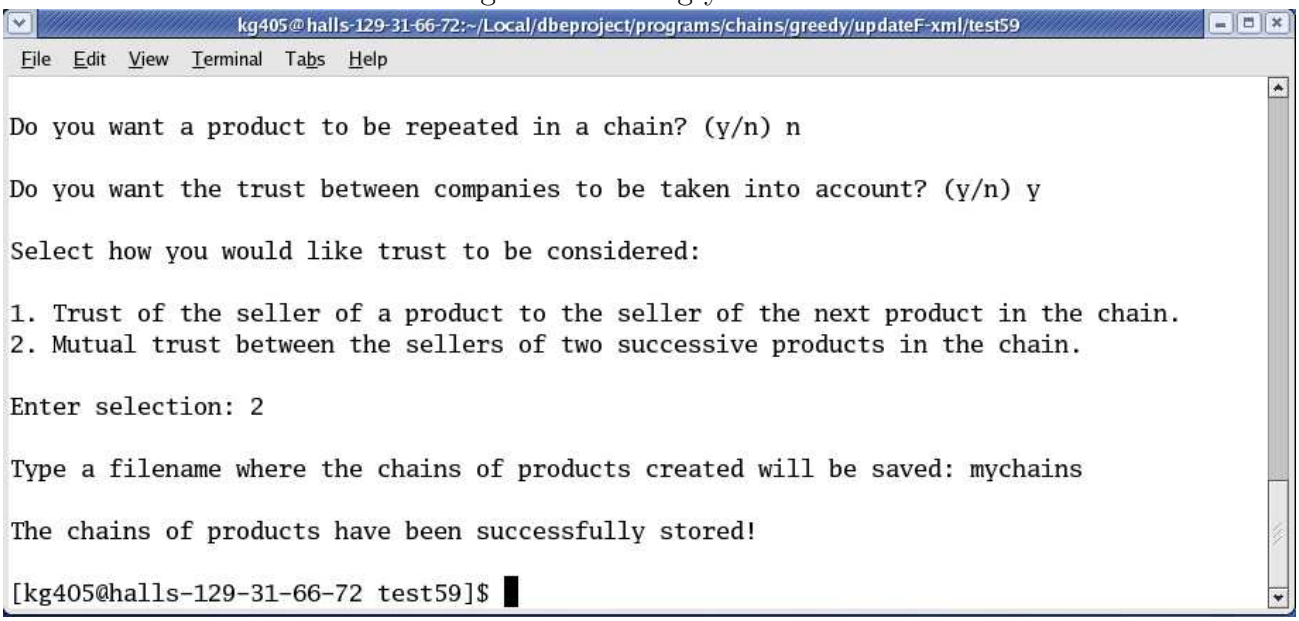
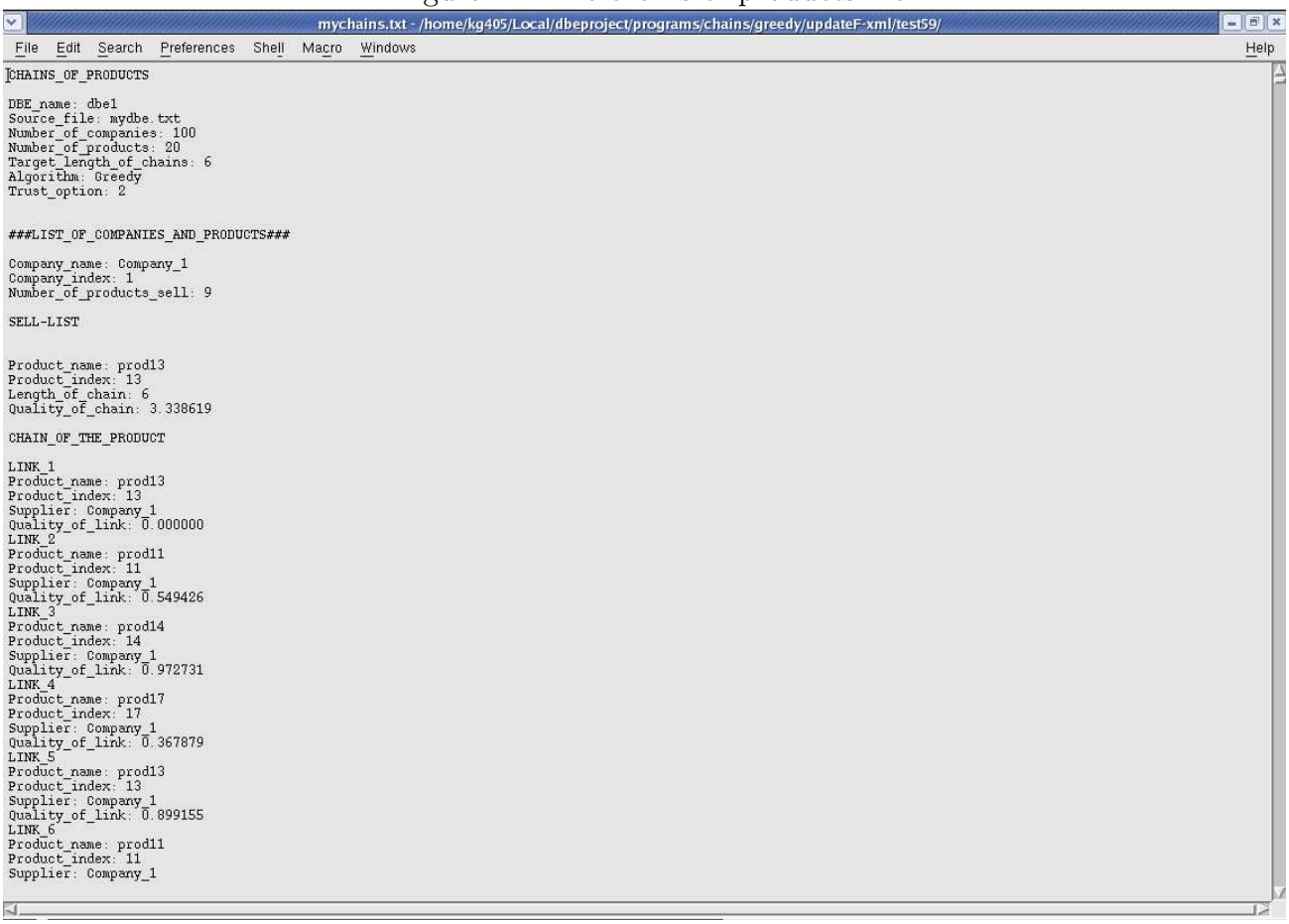


Figure 13: Saving your chains



After setting up all parameters, the algorithm creates chains of products of the specified target length and you can finally save your results (figure 13) in a **.txt** file by typing its name, eg mychains.txt (you do not need to type the extension .txt, as it is added automatically by the program). Figure 14 shows the produced file with the chains of products.

Figure 14: The chains of products file



C.2 Reading a simulated DBE from XML

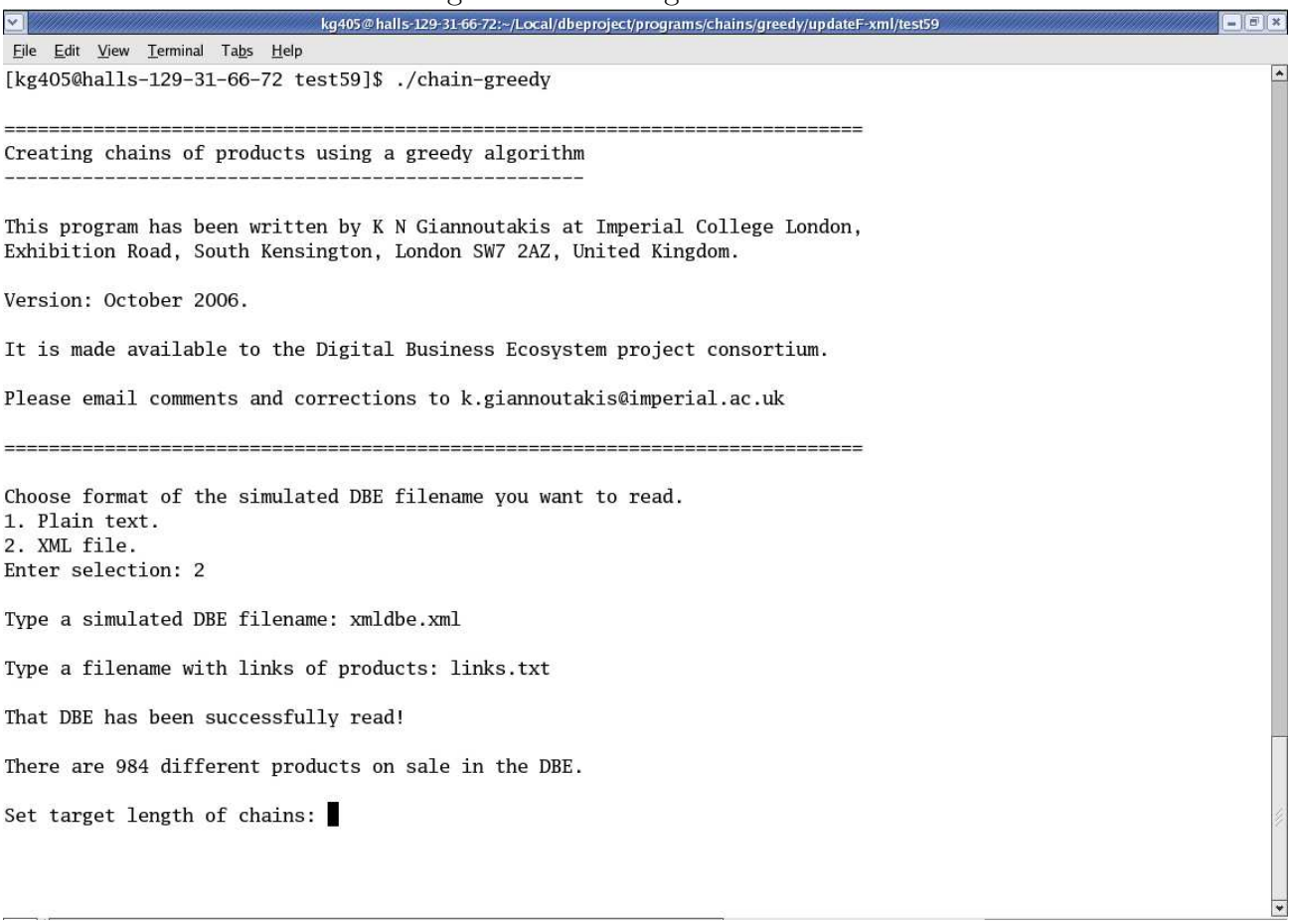
The process is very similar to the one with plain text. As mentioned above, the main difference is that you have to type the filename with the links of products, and also, you do not specify the number of companies to be read; all companies are read. Figure 15 shows how to start creating chains of products with a DBE from an XML file. After that, the process is the same as before.

D Reading the chains of products

As mentioned above, the software for reading chains of products can be found in the package **readchains.tar**. To run the program, untar the package in its directory, copy the produced file of chains in there and type **./readchains**[2]. That is the executable for running the program. You will find nine other files in the same package:

[2]Type "gcc -lm -o readchains Main.c" if you need to compile the program.

Figure 15: Reading from XML file



1. **Get_routines.c**, as in section 1.
2. **InitFunctions.c**, as in section 1.
3. **Main.c** the file with the **main()** function of the program.
4. **mydbe.txt** the file with the simulated DBE in plain text.
5. **ReadChains.c** the file with the routine **ReadChains()**, the function that reads the chain of each product and links them to the simulated DBE.
6. **ReadText.c**, as in section 1.
7. **ReadXML.c** as in section 1.
8. **Structs.h**, as in section 1.
9. **xmldbe.xml** the file with the simulated DBE in XML format.

Figure 16: Reading the file with the chains of products

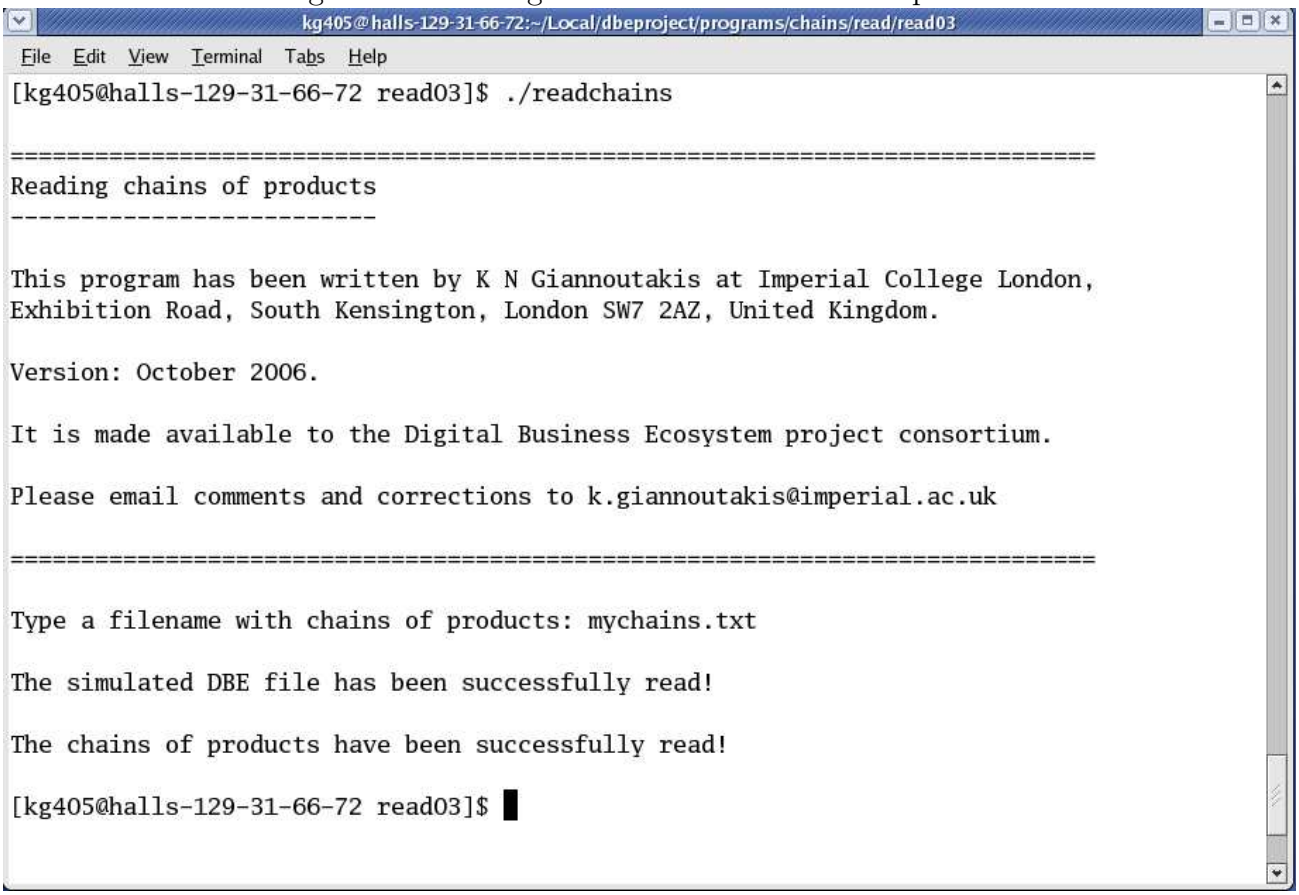


Figure 16 shows how the program reads the chains. The only thing you have to do is to type in the filename of the chains. The program opens the file, reads the file of the simulated DBE, where the chains refer to, it opens the simulated DBE file automatically and reads it and finally, links the chains of products to the corresponding product where they start from. It is necessary to keep the

file of the simulated DBE in the same directory, so that the program copies in the memory all data about your simulated DBE. If any of those files is missing, the program returns an error message and terminates.