AMO-Advanced Modeling and Optimization, Volume 9, Number 1, 2007

A Decision Support System for the Facility Location Problem under Time Constraints

Jason Papathanasiou[1]

Aristotle University of Thessaloniki, e-mail: jasonp@gen.auth.gr

Anastasia Paparrizou

Aristotle University of Thessaloniki, e-mail: paparriz@auth.gr

Abstract

The facility location problem in a market environment where other similar facilities are already activated has been studied extensively in the international bibliography. What has not been materialized however, with only a few incomplete exceptions, is software specialized on the specific problem. Given the huge number of models of the problem, the even greater number of their solution methods and its frequent appearance in the operational planning of both public and private organisations, this lack becomes noticeable. In the present paper, a decision support system about the simultaneous location of multiple competitive facilities in a network environment under strict time restrictions is presented.

Keywords: multiple facility location problem, decision support systems, linear programming, time restrictions.

[1] corresponding author, address: Prox. Koromila 34, 55236, Panorama, Thessaloniki, Greece

1. Introduction

Most probably, the major aim in the strategic planning of any manufacturing unit is finding the most suitable facility location. The unit should be located where it will be the most competitive; be able to satisfy the production demands within the time restrictions imposed by the market; has to succeed in attaining the highest economic benefit and all these in order for the investment to be successful. The problem becomes even more complex when a number of new competitive units, cooperating or not, are simultaneously in search of the best possible facility location in an area with other already existing competitive units.

In this paper a Decision Support System is presented. The system is based on a model of linear integer programming that is solved, depending on the size of the underlying network with algorithms that find the most optimal or an approximate solution. It was programmed in Python and a range of other programs and software libraries were also used, such as gnuplot and graphviz, both free and open source software. The system is still in phase of intense development, it has however the complete capability to produce concrete results.

2. Problem description and model specification

A number of competitive manufacturing units, producing certain products (e.g. pasteurized milk, computer parts, etc) operate in a particular area of the market. This market requires a specific quantity of this product within a determined time period while in the existing situation the market covers its needs to the greatest degree. A set of new enterprises, cooperating among themselves, each seeks the best location for the establishment of a unit for itself, where they will produce the same products as existing units (Bresnahan and Reiss, 1991, Drezner T, Drezner Z and Salhi, 2002, Plastria, 2001, Eiselt, Laporte and Thisse, 1993). Their aim is to obtain the largest possible share of the market, as well as to avoid any overlapping among themselves of the market surfaces they will serve. The above aim implies that although the new enterprises will come into direct confrontation with existing ones, they will attempt to decrease any possible conflict which may

arise among them in the future, even if in the present they are cooperating. For this reason, the location chosen for each facility must ensure their survival with the least possible interchanges between them. Put simply, the new units will have to be assumed as being competitive among them; if not now, most certainly in the near future.

For each of the new units there must be optimum investment of the initial capital outlay in such a way that the greatest quantity of the product is produced at the lowest possible cost during their first stage of operation and at the same time there must not be any surplus stock of unsold products for any of the units. In accordance with the strategic planning of the cooperating enterprises, the objective function of the problem expresses the optimum production. This case aims at extracting the highest possible share of the market and is considered to be aggressive (high risk policy) towards the existing units since its objective is to decrease their sales as much as possible, which will have as a consequence their steady disappearance from the market.

The determined time within which the market demands a set quantity of a product depends on the parameters that constitute the specific problem of location. In the case of dairy enterprises for example, as well as in other cases such as the fuel producing industries and their byproducts, this time period is from one to several days.

Each new unit should occupy a share of the market so that its production is higher than the sales threshold level, above that which the unit is economically viable (Berry and Garrison, 1958 in Shonwiller and Harris 1996). Needless to say, the same goes for the existing units. Should they fall below the sales threshold level, they embark on a course of economic decline (Balakrishnan and Storbeck, 1991, Drezner T and Drezner Z, 2001, Serra, Revelle and Rosing, 1999, Serra and Revelle, 1995, Shiode and Drezner, 2003).

The market surface has the form of a network with nodes, which are the points where the already existing units are located, the points of demand and the candidate points of the location of the new units. The axes of transport comprise the arcs of the network and movement along them is allowed in both directions.

The symbols used in the model description and the DSS screenshots are as follows:

P: the set of new units [$p_n \in P=\{p_1, p_2, \ldots, p_k\}$, n=1,…, k].

M: the set of the already existing units [$m_f \in M=\{m_1,m_2,\ldots,m_h\}$, f=1,…, h].

I: the set of the possible locations (candidate) nodes of the new units [$i_s \in I=\{i_1, i_2,\ldots, i_q\}$, s=1,…, q].

J: the set of demand nodes [$j_r \in J=\{j_1, j_2,\ldots, j_b\}$, r=1,…, b].

T: the time within which the market demands a specific quantity of the product in question (in time units, e.g. hours).

DP_{ip}: the production capacity in time T of the new unit p established in node i (product units/ T).

DP_{ipmax}: the maximum production capacity in time T of the new unit p established in node i (product units / T).

DP_{ipmin}: the minimum acceptable production capacity (production threshold) in time T of the new unit p established in node i (product units / T).

HP^p_{ij}: the fraction of demand in node j, which is serviced by node i where the new unit p has been located (product units / T)

$$\mathbf{X_i} = \begin{cases} 1 & \text{if the new unit is located at the candidate point i} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{Y_{ij}} = \begin{cases} 1 & \text{if the demand in node j is serviced by a new unit in node i} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{UP_{ij}} = \begin{cases} 1 & \text{if node j is within the range of i in time T} \\ 0 & \text{otherwise} \end{cases}$$

UM_{mj} : corresponds to UPij for existing facilities

When X is a set, |**X**| denotes the number of the elements of set X.

In accordance with the above symbols the total number of nodes of the network is |I|+|M|+|J|.

The mathematical form of the problem described above, is as follows:

$$\textbf{max} \qquad \sum_i \sum_p DP_{ip} X_i \tag{1}$$

Under the constraints:

$$DP_{ipmin} \leq DP_{ip} \leq DP_{ipmax} \tag{2}$$

$$\sum_i X_i = P \tag{3}$$

$$Y_{ij} - X_i \leq 0 \tag{4}$$

$$\left.\begin{aligned} X_i &= 0, 1 \\ Y_{ij} &= 0, 1 \end{aligned}\right\} \tag{5}$$

$$UP_{ij} = 0, 1$$

$$UM_{mj} = 0, 1$$

$$\sum_p DP_{ip} = \sum_p \sum_i \sum_j H^p{}_{ij} Y_{ij} UP_{ij} \tag{6}$$

The objective function (1) concerns the maximization of the product produced. Constraint (2) refers to the range of prices which the quantity of production can obtain for each p_n within the given time T, (3) requires that precise |P| units are established, the relationship (4) allows the

service only from nodes where units have been established and constraints (5) show that these variables are integers to the values of zero and one. In addition, relationship (6) shows that each new unit's entire production is consumed; otherwise surplus stock of unsold products will be created. Finally, all the variables of the problem are non-negative numbers.

Observed is that the model does not require that demand will be fully met, however, as mentioned before, both old and new units, already actually cover the needs of the market to the greatest possible degree. In addition, the condition 'all or nothing' does not exist; in other words, a consumer can be served simultaneously by two or more enterprises.

As the range of a facility located on the node of a network, we take the greatest distance lengthwise of each already existing transport axis, which has its source in the node on which the unit in question is established and where the particular unit can transport a specific amount of the product within a set time period determined by market needs under conditions of favorable transportation costs. Interaction with other related manufacturing units is assumed to be non-existent. The axes are considered to have a specific capacity in time T.

It is obvious that with the range we can determine the exact size of the network under study. Of the total number of the demand nodes of a market, we are interested in only those that are within the range of the existing facilities, as well as the range of the new facilities positioned on each one of the candidate location nodes, since it is evidently impossible for the remainder to affect production allocation. The concept of range helps us to precisely determine the network under analysis, i.e., in effect, time T strictly determines the size of the network.

3. Software

The structure of the DSS can be seen in diagram 1. From the very beginning of the DSS development the intention was to use strictly free and open source software. Moreover, it was desirable to integrate already existing tools at the highest possible level. The operating system in

which the development took place was SUSE Linux 10.1 and the software components (both programs and software libraries) used were as below:

- Python

An interpreted, interactive, object-oriented, extensible programming language (Langtangen, 2004).

- Numarray

Numarray is a set of extensions to the Python programming language which allows Python programmers to efficiently manipulate large sets of objects organized in grid-like fashion. This is a reimplementation of the earlier Numeric module (aka numpy).

- Pmw

Pmw is a toolkit for building high-level compound widgets in Python using the Tkinter module. It consists of a set of base classes and a library of flexible and extensible megawidgets built on this foundation. These megawidgets include notebooks, comboboxes, selection widgets, paned widgets, scrolled widgets, dialog windows, etc.

- BLT

BLT Graph is a highly configurable plotting library for 2D graphs written in Tcl.

- NetworkX

NetworkX is a Python-based package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks.

- Graphviz

Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. Graphviz is an open source graph visualization software.

- Gnuplot

Gnuplot is a command-driven interactive function and data plotting program.

- Gnuplot.py

A pipe-based interface to the gnuplot plotting program.

- Pygraphviz

Pygraphviz is a wrapper to the graph data structure of the graphviz graph layout and visualization package.

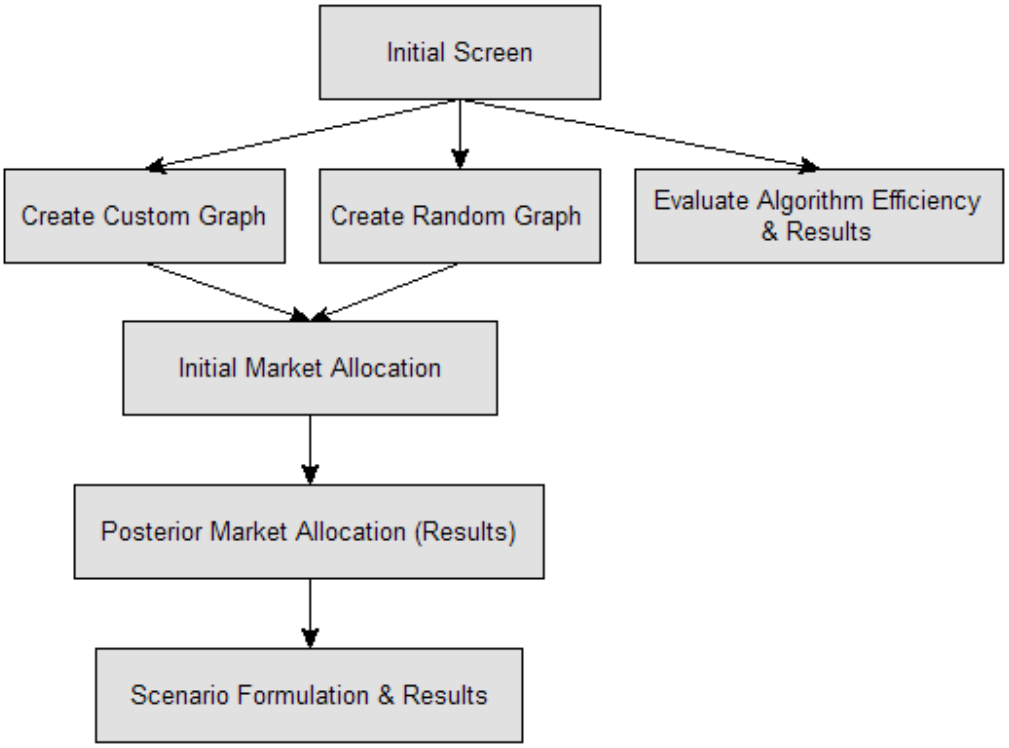

Diagram 1

- Pydot

Graphviz's dot language Python interface. This module provides a full interface to create, handle, modify and process graphs in Graphviz's dot language.

- Pyparsing

An object-oriented approach to text processing in Python.

In future versions of the system it is intended to switch to pure C++ libraries with the use of SWIG (Simplified Wrapper Interface Generator), which is a software development tool that simplifies the task of interfacing different languages to C and C++ programs. That is, in order to (vastly) improve the running time of the algorithms. Another possible way to continue the development is to implement the system, using large parts of the already existing code, with the JPython (Jython) development environment. That is, in order to produce browser accessed Java applets, without actually involving the complexity of Java itself.

One of the main objectives of this paper and the presentation of the particular DSS is to prove that a fully functional system can be implemented with the use of already existing free and open source tools. It is then just a matter of combining, converting and writing relatively little code, leading to short development times and robust products. Script languages with a very clean syntax, as Python, provide an ideal tool for combining code written in more efficient languages (like C/C++), and at the same time they provide shorter code, ease of use and maintenance and large future development capabilities.

4. DSS description

4.1 Initial Screen (figure 1)

Three options are available:

- Create custom graph

  Allows the user to create a specific graph (directed or undirected).

- Create random graph

  Allows the user to create random graphs (directed or undirected), following the users specifications.

- Evaluate algorithm efficiency

  Allows the user to evaluate in a number of ways the efficiency of the approximation algorithm used by the system.

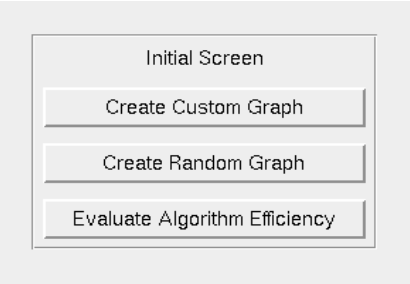

Figure 1

4.2 Creation of a custom graph (figure 2)

There are four frames in this window. From left to right, top to bottom:

- This frame is about the construction of the graph. Options to save or load a previously made graph are provided.
- The constructed graph is displayed here.
- The number of each kind of node is entered here. The user must start from this frame.
- Graphviz is called from this frame. The graph image is saved in the path defined by the user.

4.3 Creation of a random graph (figure 3)

Like before, from left to right, top to bottom:

- Firstly the user provides the number of each kind of node. Connection probability is the percentage that each node is connected to an adjacent one. Seed is optionally used if the user needs to test the same randomly created graph continually. The limits are the boundaries the values of economical and time distances and link capacities can be. The

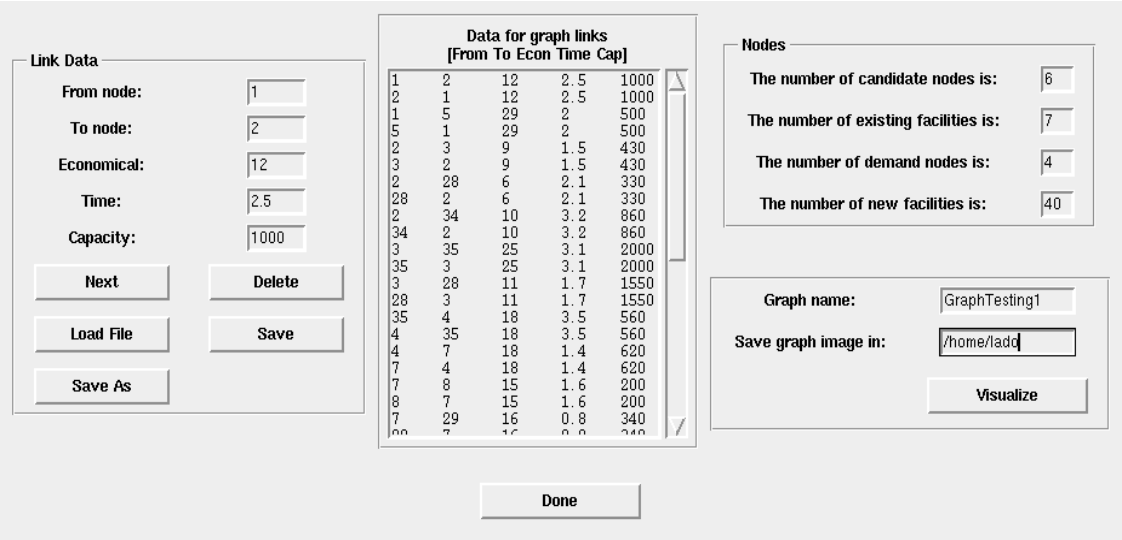

Figure 2

button generates the graph and as a measurement of the system capabilities the time needed to generate the graph is shown.

- The graph is visible in the center frame.
- Changes in the graph, saving it or recalling another one, is possible in the right-top frame.
- And again, the graph maybe visualized by Graphviz.

4.4 Initial market allocation (figure 4)

This window is about the allocation of the demand before the new facilities enter the market. The only acting facilities are the already existing ones. In the example presented in corresponding figure 4 the market demand is covered by 93.28%, thus leaving a small margin for new facilities.

$\mathbf{GM_{mj}}$ stands for the final sale price per product unit on node j from existing facility m (currency units / product unit).

$\mathbf{H_j}$ stands for demand in demand node j (product units / T).

D: stands for demand allocation per existing facility (product units / T).

4.5 Posterior market allocation (figure 5)

The allocation of the market demand after the locating of new facilities has taken place is pictured in this window. The percentage of the demand covered by the new facilities and the old ones is shown, as well as the value of the objective function. If the location of the new facilities is not possible, a relevant message is shown. There are a couple of buttons; one is to save the results and the other to proceed to the next window.

$\mathbf{GP^{p}_{ij}}$: stands for the final sale price per product unit on node j from new facility p located on node i (currency units / product unit).

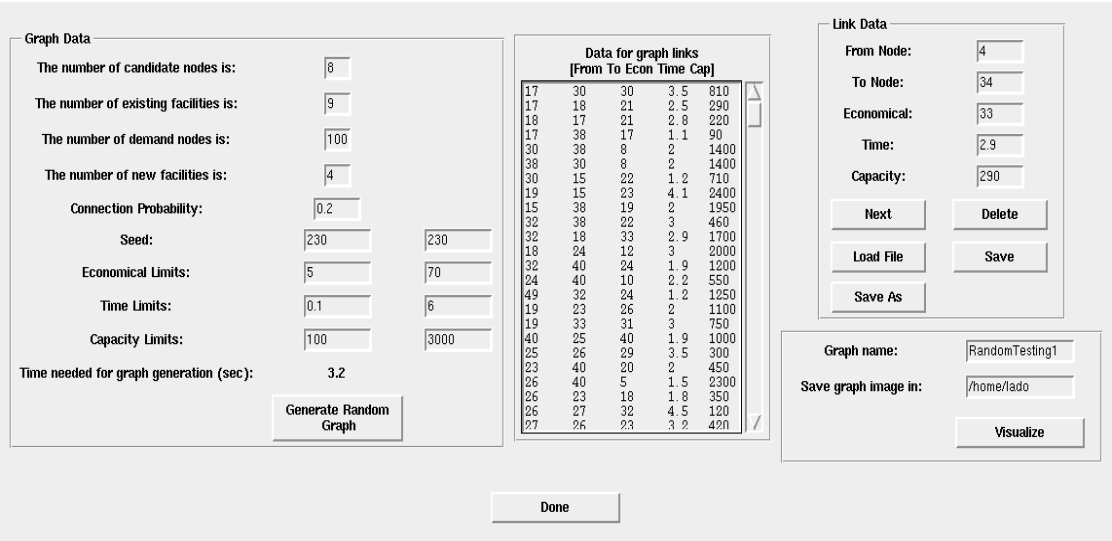

Figure 3

4.6 Scenario formulation and results (figure 6)

This window allows the user to simulate different scenarios for the future development of the market. That is, after the location of the new facilities has taken place. Time interval number refer to time T periods where the demand of each demand node is changing, and the production capabilities of each facility (both existing and new) are changing as well. Each scenario can be saved or loaded and the evolution of the demand for any demand node or the market share of any facility can be shown for each time period in the diagram. The diagram can be animated and can

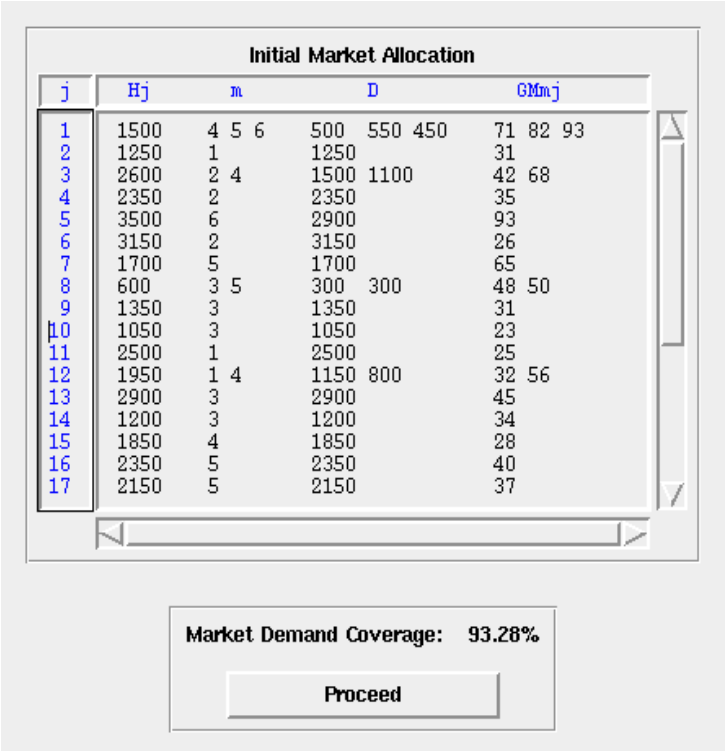

Figure 4

be saved at postscript format. This is probably the most useful tool of the DSS. It is also possible for the DSS to allow for function style input for the demand and the production, e.g. H_1=aT+b, whereas H_1 is the demand of node 1 and a, b are some constants. In that case the time intervals must be the total number of intervals and all demand and production input must be functions of T.

4.7 Evaluation of algorithm efficiency (figure 7)

There are four variables that can change their values, the number of candidate nodes, of existing facilities, of new facilities and of demand nodes. For each variable an initial value and a final one is needed and in addition the step used to change the values. For each set of variable values a random graph is generated and a solution is reached (regardless of successfully placing the new facilities or not). Noted is that graph density should be the same in all graphs and that only three out of four variables can change their value, the other or others being a constant. The output is

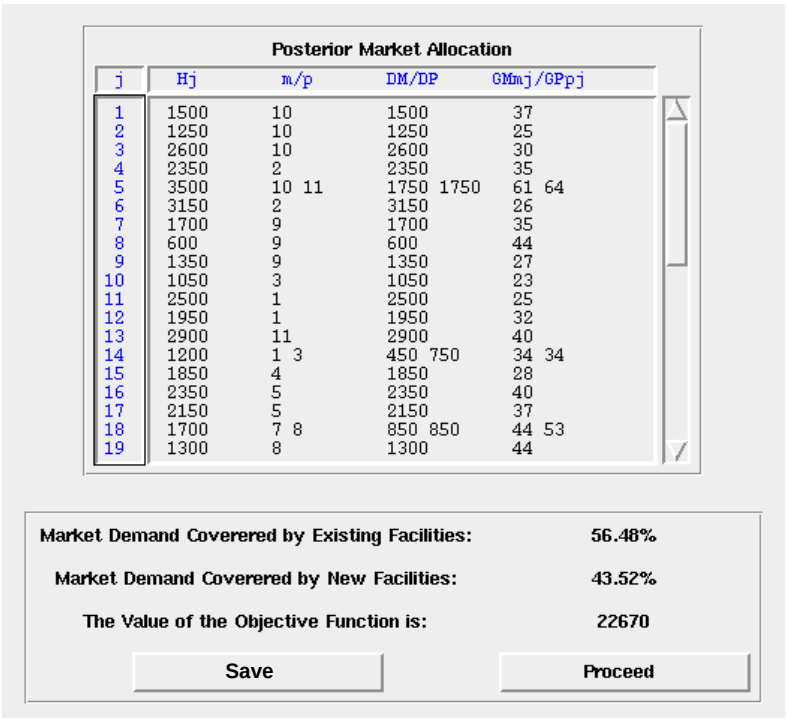

Figure 5

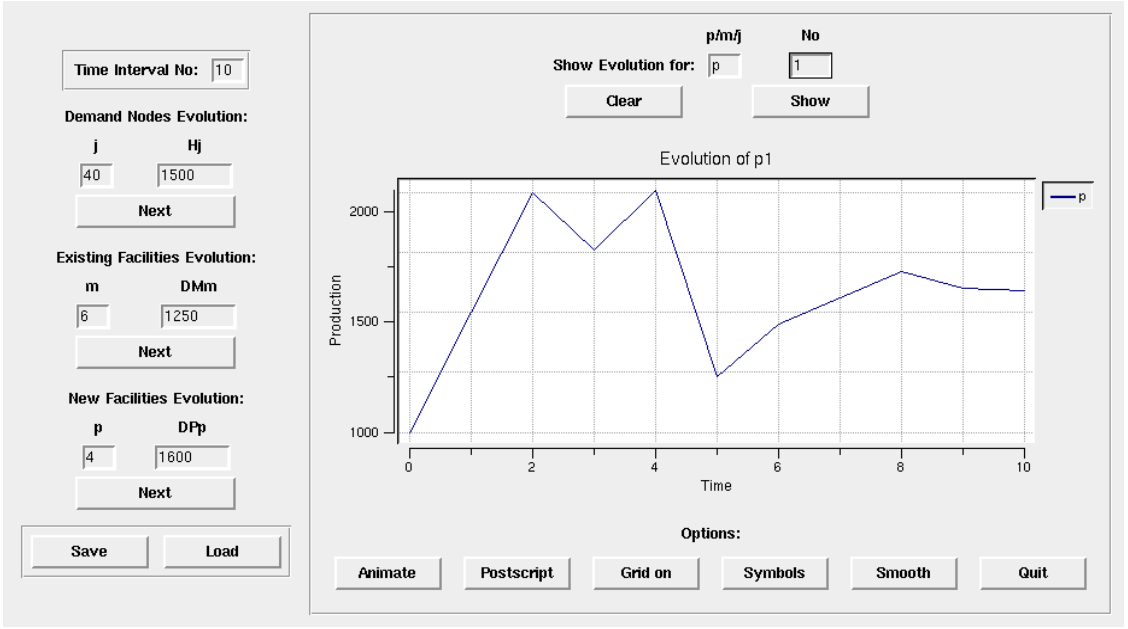

Figure 6

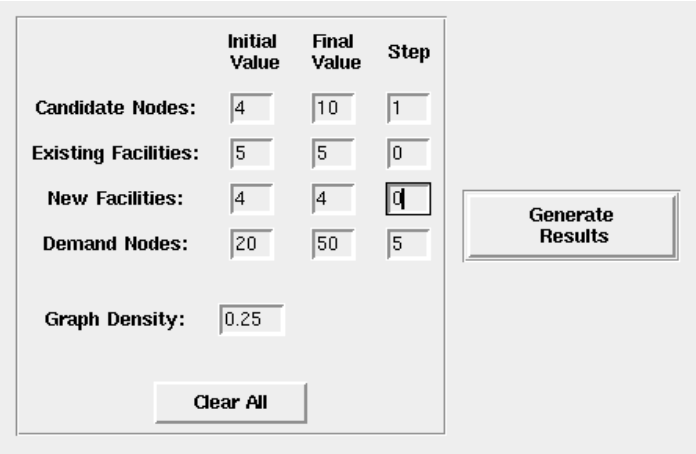

Figure 7

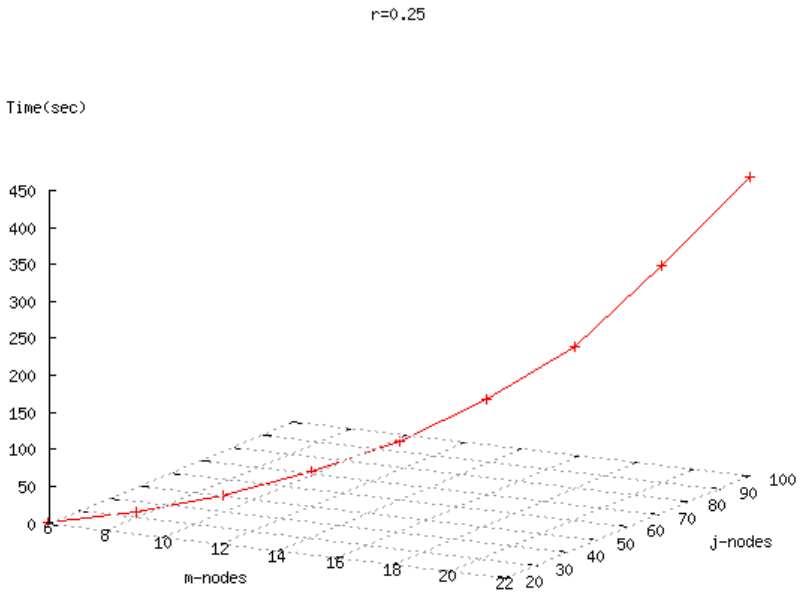

Figure 8

channeled to Gnuplot and an image like the one on figure 8 is stored at the home folder of the user.

5. Conclusions

The described DSS allows not only considering the possibility of locating a number of new units in a competitive environment, but for the formulation of scenarios as well. Thus, the interested parties can check the viability of their investment long-term. The DSS provides also detailed enough diagrams to find out whether under a certain scenario the production of a unit falls under the threshold and for how long. More over, the underlying heuristics (Papathanasiou and Manos, 2007) can be evaluated. The interface for heuristic selection is however under development at the time of the writing of this paper.

Finally the DSS is fully based on free and open source software, cutting down drastically the development time and on the same time proving that the design and implementation, with this kind of software, of such systems is more than possible.

References

Balakrishnan P and Storbeck J (1991): *Mctresh: Modeling Maximum Coverage with Threshold Constraints* - Enviromental & Planning B, 18, 459-472.

Bresnahan T and Reiss P (1991): *Entry and competition in concentrated markets* - Journal of Political Economy, 99, 977-1009.

Drezner T, Drezner Z, Salhi S (2002): *Solving the multiple competitive facilities location problem* - European Journal of Operational Research, 142, 138-151.

Drezner T and Drezner Z (2001): *A threshold – satisfying competitive location model*, - Proceedings of the Decision Sciences Institute.

Eiselt HA, Laporte G, Thisse JF (1993): *Competitive location models: A framework and bibliography* - Transportation Science, 27(1), 44-54.

Papathanasiou J and Manos B D (2007): *An Approximation Algorithm for the Location of Dairy Enterprises under Time Constraints* - to appear in European Journal of Operational Research.

Plastria F (2001): *Static competitive facility location: An overview of optimization approaches* - European Journal of Operational Research, 129, 461- 470.

Langtangen H.P.: *Python Scripting for Computational Science* - Berlin: Springer-Verlag Heidelberg 2004.

Serra D, Revelle C, Rosing K (1999): *Surviving in a competitive spatial market: The threshold Capture Model* - Journal of Regional Science, 4(39), 637-652.

Serra D. and Revelle C (1995): *Competitive location in discrete space* - in: Z., Drezner, ed. Facility Location: A survey of applications and methods, Berlin: Springer , p.367-386.

Shiode S. and Drezner Z (2003): *A competitive facility location problem on a tree network with stochastic weights* - European Journal of Operational Research, 149, 47-52.

Shonwiller J. and Harris T (1996): *Rural Retail Business Thresholds and Interdependencies*, - Journal of Regional Science, 21, 189-198.