Multiobjective Optimization of the Transport in Oil Pipelines Networks

J.M.de la Cruz, B.de Andres-Toro, A.Herrán, E.Besada Porta, P.Fernandez Blanco

Dpt. Computer Architecture and Automatic Control
Universidad Complutense de Madrid
28040- Madrid
SPAIN

***Abstract* - Heuristic methods are specially well suited to solve combinatorial problems. One of this problem is the distribution of petroleum products through oil pipelines networks. In this paper the problem is stated and solved using a multiobjective and constraint evolutionary optimization algorithm. Several objective functions has been defined to express the goals of the solutions as well as the preferences among them. Some constraints are included as hard objective functions and some has been evaluated through a repairing function to avoid infeasible solutions. An example of working is given.**

I. INTRODUCTION

Distribution of petroleum product through oil pipeline networks is a very important problem since it is an activity of economic importance in every country. Usually the products are taken from refineries, ports or storage centers and transported to the destination points. Usually the pipes are unidirectional but in some special cases there can be bidirectional pipes.

The main goal is to satisfy the demanded products at the destination points in due time, but other important goal is to avoid sending consecutive products of different kinds because they may contaminate each other. Moreover, a number of constrains must be satisfied, as the limits in capacity of delivering of sources and of receiving in destination, the limits in the capacity of the transportation network, and the limits in the storage capacity.

This is a problem of combinatorial type. This kind of problems are well suited for evolutionary algorithms (EA) [1], [2]. Although originally developed to optimize a cost function or fitness function, algorithms have been proposed to solve multiobjective optimization and constrained problems [3],[4],[5].

In this paper we present a solution to a simplified problem of the optimal distribution of petroleum products through oil pipelines networks using an evolutionary multiobjective constrained optimization algorithm. In section 2 we study the model of the problem. In section 3 the model coding and representation used by the evolutionary algorithm is given. The objective functions, constraints and priorities used in the algorithm of the problem are stated in section 4. An example is presented in section 5 and the conclusions are stated in section 6.

To the autors knowledge this problem has not been aborded before.

II. MODEL OF THE OIL PIPELINE NETWORK

We consider a simplified model of an actual network. The network has a set of nodes made up of a set of sources, a set of sinks or receiving terminals, such as delivery points or storage terminals, and a set of intermediate connections that actuate as receiving and delivering points with storage capacity. Every source and intermediate connections may have different pipes to different nodes and can deliver different products in different pipes simultaneously. We consider that the different products or oils are delivered as discrete packets. There might be as many different types of packets as number of different products. A unit packet is the minimum fluid volume delivered by a source or intermediate node in a unit time, that is, the minimum volume of the pipe filled by a fluid. Every sink and intermediate node have as many tanks as products he can receive, to store the different products. Also we can assume that the sources take the fluids from tanks.

In order to simplify the problem we assume that all pipes have the same diameter and characteristics. We also assume that all packets flow with the same speed and that they occupy a similar volume in the pipe. If two packets of different fluids follow one another there exist the possibility of both products to become contaminated. In a number of pipes the fluids may flow in both directions from one node to the other.

A simple network can be seen in figure 1. This network has two sources (nodes 1 and 2), three sinks (nodes 5, 6 and 7) and two intermediate nodes (4 and 5). In pipe joining nodes 3 and 4 the fluid can flow in both directions. Numbers in links joining two nodes give the normalized distance in terms of units of time needed by a given packet to cover the whole pipe. For instance, number 4 in pipe linking nodes 1 and 3 means that one packet spends four periods of unit time to go from node 1 to node 3, or that the pipe may contain four packets.

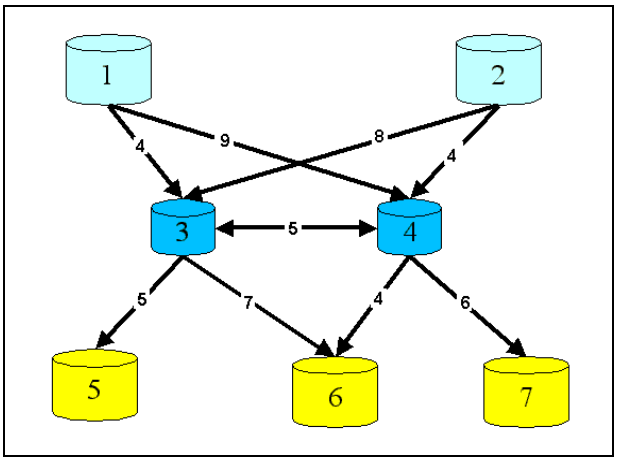

Figure 1: Simple network model

To code the topology of the network we use a matrix having as entries the distance among the nodes. An entry of zero says that there is no connection between these nodes. A connection is represented as a pair (*i*,*j*) where *i* is the output node and *j* the input node. A bidirectional links is acknowledge because we have a pair of symmetric connections (*i*, *j*) and (*j*, *i*).

III. MODEL REPRESENTATION

We describe the data structure, operators and functions that characterize the solution space. Firs we have the solution coding and then we pass to describe the most important genetic operators used over the population.

A. Coding a solution.

A solution to the problem is given by the kind of packet sent by every source or interconnection node at every instant. To code a solution we use a matrix in which the rows are the network connections and the columns the time. Every entry represents the kind of product that is delivered to the link at this time instant. Every product is coded as an integer and the value 0 is used to represents that no packet is delivered at this time. Figure 2 shows a solution for the network of figure 1. In order to use this coding, we need a prior knowledge of a maximum value of the time needed to find a solution. This estimation is made from the network parameters and the constraints. In figure 2 we have taken this value as *Ninst* = 15 and we have suppose that there are three kind of products.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CONECTION 1	1	1	2	0	0	2	0	3	0	2	2	3	3	0	0
CONECTION 2	1	2	2	2	3	2	1	1	3	1	2	0	2	0	1
CONECTION 3	2	3	0	2	1	2	3	0	2	2	2	2	0	2	3
CONECTION 4	2	1	1	3	3	2	1	1	3	1	1	1	3	0	1
CONECTION 5	2	3	0	2	1	1	2	0	2	2	3	0	2	2	3
CONECTION 6	2	1	1	3	3	2	2	2	0	1	1	1	3	0	1
CONECTION 7	2	3	0	2	1	1	2	0	2	2	3	0	2	2	3
CONECTION 8	1	2	2	2	3	2	2	2	0	1	2	0	2	0	1
CONECTION 9	2	0	3	1	1	1	2	0	2	2	2	2	0	2	3
CONECTION 10	1	2	2	2	3	2	1	1	3	1	2	0	2	0	1

Figure 2: Matrix solution coding.

Not only will this coding let the objective function reproduce a solution and evaluate it properly, but also it favours the implementation of quickly initialization, crossover and mutation operators: initial values are random integers in the range [0, number of oil types], the crossing points will be instant of times and the crossover performed dividing the matrix in groups of columns, and the mutation will be carried out modifying randomly the value of some of the elements of the matrix.

However, in order to store a whole population in a no structured variable is more suitable to represent each individual as a vector and the population as a matrix, with a row for individual. The above coding can be rearranged in a row vector, by placing consecutively each of the columns of the matrix (instant of times), in such a way that the first *n* elements store the types of package units sent through the *n* connections in the first instant of time, the following *n* elements the types of package units sent during the second instant of time, and so on. The new coding of the same solution to the problem as the one presented in figure 2, is shown in figure 3:

Instant 1	.....	Instant 15
1.....10	.....	1.....10
1 1 2 2 2 2 2 1 2 1	.....	0 3 1 3 1 3 1 3 1 3

Figure 3: Vector solution coding

Implementing the objective function for this coding is as easy as for the original one and as all the population can be stored in a matrix, quicker genetic operators that work over the whole population (and not individual by individual) can be developed straightforward.

However, this codification doesn't necessary fulfil the problem statement conditions yet, because the types of oils produced for each source could be different, and all the elements in the row vector solution are in the range [0, number of oil types]. The easiest way of handling this aspect of the problem is to code, for each of the sources, the types of oils from 1 to the number of different oils sent by that source (leaving the 0 for the case of not sending any product at that instant of time). This coding will not overload the genetic operators and the real type of oil coded in the individual can be obtained, before evaluating the individuals to obtain its fitness, by means of the Package_Type variable.

For example, for the network of the figure 1, if the source 1 can handle packages of type 1 and 2, and the source 2 can handle packages of type 3 and 4, the valid values in the individual row vector for the connections 1, 2, 3 and 4 will be in the range [0,2] and for the others in the range [0-4]. In the two first connections, the values 1 and 2 will code types 1 and 2, while in the two following connections the same values will code the types 3 and 4. For all the other connections, the values [1,4] will code

types [1,4]. The Package_Type for the example is presented in table 1 (the coding values in first row are mapped to the valid types for each node in the same column), while the final coding for an individual is shown in figure 4 (at the top, the vector individual; at the bottom the individual after the mapping to the real oil types):

coding values	0	1	2	3	4
Types for Source 1	0	1	2		
Types for Source 2	0	3	4		
Types for Inter. node 1	0	1	2	3	4
Types for Inter. node 2	0	1	2	3	4

Table 1: Package_Type variable.

Instant 1	.....	Instant 15
1.....10	.....	1.....10
1 1 1 1 2 2 2 1 2 1	.....	0 1 1 2 1 3 1 3 1 3

Instant 1	.....	Instant 15
1.....10	.....	1.....10
1 1 3 3 2 2 2 1 2 1	.....	0 1 3 4 1 3 1 3 1 3

Figure 4: Vector individual coding (top) and real type mapped individual (bottom).

The information in table 1 can be kept easily in a structure where every row is a cell associated to a node, and ,within every row, there are as many columns as connections departing from this node. So, the value of the gene acts as entry to get the product associated to it. In Matlab notation the representation of the information in table 1 would be as follows:

```
ProductType={[1 2] ;        % node 1
             [3 4] ;        % node 2
             [1 2 3 4] ;    % node 3
             [1 2 3 4] }; % node 4
```

With this representation the products associated to the genes of the first node are:

```
» ProductType {1}(1)=1
» ProductType {1}(2)=2
```

and those of the second node are

```
» ProductType {2}(1)=3
» ProductType {2}(2)=4
```

and so on.

B. Genetic operators.

For the *generating law*, we can buid a mask in order to keep each gen of our individual structure (type of package sent by each conection in each time instant) in his corresponding value range. The mask is codified as in chromosome of figure 3. The example corresponding to table 1 is given in table 2:

	node 1		node 2		node 3			node 4		
Conection	1,3	1,4	2,3	2,4	3,4	3,5	3,6	4,3	4,6	4,7
Nº conect	1	2	3	4	5	6	7	8	9	10
Nº products	2	2	2	2	4	4	4	4	4	4

Table 2: Mask (grey colour) needed for the mapping process.

Then, if we have *Nind=20* individuals (with 10 conex and 15 time instants) we can use the above mask to generate the initial population as follows (using Matlab notation):

```
Nconex=10;
Ninst=15;
Ngenes=Ninst*Nconex;
Nind=20;
mask=[2 2 2 2 4 4 4 4 4 4];
pop_aux=rand(Nind,Ngenes);
mask=repmat(mask,Nind,Ninst);
pop=round(pop_aux.*mascara);
```

Now, we pass to explain the genetic operators used in the reprodution process:

Crossover operator: We can see a sechematic of the cross process in figure 5. Althrougth our final codification is the showed in figure 4, the efectc of the cross operator is more esaily explained making use of our initial codification showed in figure 2. Then, if we have two individuals with 4 conections and 15 instants the following cross is implemented with one cross point:

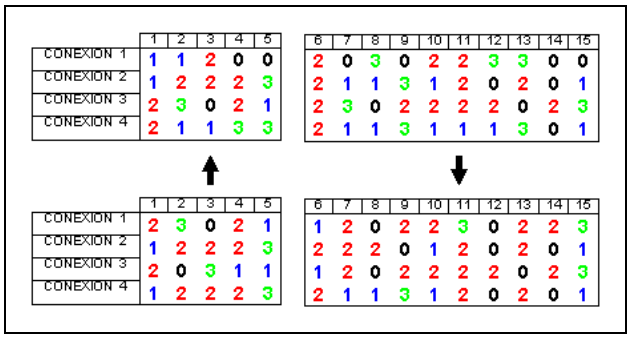

Figure 5: Crossover operator

Mutaton operator: Again we can make use of the mask for implement the mutation operator. Continuing with the above example the implementation of this operator with Matlab notation is:

```
probMut=0.08;
mutp=(rand(Nind,Ngenes)<probMut);
pop_aux=rand(Nind,Ngenes);
mutated=round(mask.*pop_aux).*mutp;
new_pop=pop.*(~mutp) + mutated;
```

V. DESIGN PROBLEM

In addition to satisfying the demands in time, the main objective is to reduce to a minimum the number of different kind of packets through every pipe, so that to avoid, as much as possible, the possibilities of contaminating products, while verifying a number of constraints. Table 3 gives the nomenclature of the parameters and variables used in the model.

Parameters (provided by the problem specification)	
Nf	number of source nodes
Ni	number of intermediate nodes
Nd	number of sink nodes
Nc	number of connections in the network
Nt_i	number of tanks in node i.
LCm_{ij}	lower limit in number of packets of tank j of node i
LCM_{ij}	upper limit in number of packets of tank j of node i
P_{ij}	minimum number of packets of product j to be sent from source i
D_{ij}	number of packets of product j demanded by destination i
LTm_j	lower limit in the arrival time of packets to destination i
LTM_j	upper limit in the arrival time of packets to destination i
$types$	number of different products.
Decision variables	
E_{ij}	number of packets of product j that has been sent by source i
R_{ij}	number of packets of product j received at destination i
NC	number of colisions in bidirectional pipes
C_{ij}	number of packets in tank j of node i
T_i	time of arrival of a packet of any product to destination i
F_i	number of changes in the type of products sent through conexión i (fragmentation)

Table 3: Parameters and decision variables used in the model.

The model is subject to the following constraints:

C1. A minimum production must be fulfilled:

$$P_{ij} \le E_{ij} \le LCM_{ij}$$
$$i = 1, \ldots.., Nf$$
$$j = 1, \ldots.., Nt_i$$

C2. Every destination must received the amount of demanded packets:

$$R_{ij} = D_{ij}$$
$$i = 1, \ldots.., Nd$$
$$j = 1, \ldots.., Nt_i$$

C3. There must be no collisions of packets through a bidirectional pipe:

$$NC = 0$$

C4. The tank capacity must not be violated:

$$LCm_{ij} \le C_{ij} \le LCM_{ij}$$
$$i = 1, \ldots.., (Nf + Ni)$$
$$j = 1, \ldots., Nt_i$$

C5. The arrival of a packet to a node must be at due time:

$$LTm_i \le T_i \le LTM_i$$
$$i = 1, \ldots.., Nd$$

And the objectives are the following ones:

O1. Minimize the time that takes in verifying the demand in each destination:

$$\min(T_i)$$
$$i = 1, \ldots.., Nd$$

O2. Minimize the sum of the previous times:

$$\min\left(\sum_{i=1}^{Nd} T_i\right)$$

O3. Minimize the fragmentation of the products in every connection:

$$\min(F_i)$$
$$i = 1, \ldots.., Nc$$

O4. Minimize the whole fragmentation:

$$\min\left(\sum_{i=1}^{Nf+Nd} F_i\right)$$

To solve the problem we use a multiobjective and multiple constraint optimization evolutionary algorithms (MOEA) as proposed in [4]. Constraint satisfaction can be seen as hard objective. In that way, constraints C1, C2, C3 are implemented as high priority objective functions, and solutions are penalized on the extent to which they violate the constraints. On the other hand, constraints C4 and C5 are handled by a repairing function. This function test each individual to find if every delivered packet meet the constraints, if so the individual is kept, or else each wrong packet is removed from the chromosome. Therefore this function assures us that constraints C4 and C5 are always fulfilled. Hence, the repairing operator is inserted in the EA implemented for this problem, which is schematized in the following figure, just before the evaluation step.

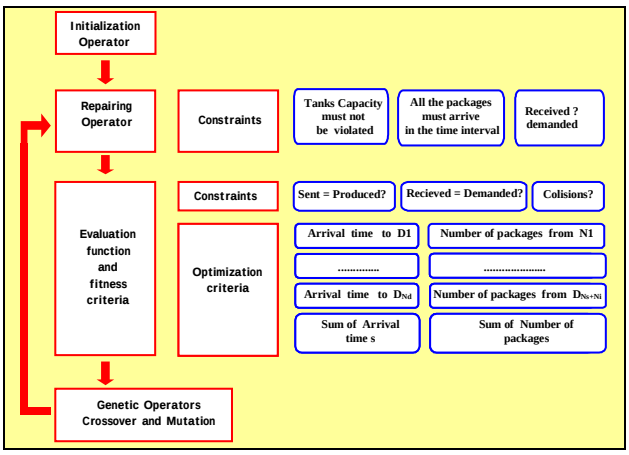

Figure 6. Constraints and objective functions in the MOEA.

The following objective functions are considered:

- J_1 : penalty for the infeasibility of C1. As the minimal production of each product of each source is constrained and must be sent, there is an objective **value J_1 that measures the part of the minimal production which has not been sent, i.e. the** total number of package units that each source should have sent but haven't. High values of J_1 mean that the constraint is far to be fulfilled, while when the value is 0 all the minimal production will be consumed by the evaluated working strategy.

- J_2 : penalty for the infeasibility of C2. The number of package units received by each destiny for each of the product types is fixed in the problem statement, therefore there is an objective value **J_2 which measures the total (for all the products and all the destinies) number of package units that the destinies should and have not received**. Individuals with more receptions than requests don't exist because the repairing operator solves that problem. High values of J_2 mean that the requests are far to be fulfilled, while when the value is 0 the number of package unit receptions will be exactly the number of package units demanded.

- J_3 : penalty for the infeasibility of C3. As there must not be collisions of packages in the pipes where the products can be sent in both directions, there is an objective value **J_3 which measures the number of package units that have collided**. The higher the value of J_3, the bigger the number of collisions. If the value is 0, the constraint is fulfilled.

- J_4, ... J_{3+j} , .. J_{Nd+3} : for every destination *j*, time of arrival of the last packet (O_1). For each destiny node the time of arrival of the products must be inside the correct arrival time interval and once inside the interval minimized. So, the best arrival time will be the minimal arrival time. **There is an arrival time objective value $J_{4:Nd+3}$ for each of the destiny nodes**.

- J_{Nd+4} : sum of J_4, ... J_{3+j} , .. J_{Nd+3} (O_2). **The total arrival time J_{Nd+4} which is obtained as the sum of all the arrival time objective values,** is also used. The arrival times are used to let the EA minimize independently the time of arrival in each destiny and the total arrival time is used to make the EA minimize all the arrival times simultaneously.

- J_{Nd+5}, ... J_{Nd+4+j} , .. $J_{Nd+Nc+4}$: fragmentation at the output of every connection *j* (O_3). The number of packages sent through each pipeline must be minimized, i.e. the package units of each product must be sent as continuously as possible. Instead of having a different objective in charge of accounting the number of packages sent through each pipe, **there is a different objective $J_{Nd+5:Nd+4+Ns+Ni}$ for counting the number of packages sent by each source and intermediate node** to maintain a manageable number of objectives in complex functions.

- $J_{Nd+Nc+4}$: sum of J_{Nd+5}, ... J_{Nd+4+j} , .. $J_{Nd+Nc+4}$ (O_4). **The total number of packages sent through the whole network $J_{Nd+5+Np+Ni}$, obtained as the sum of all the number of packages sent by each source and intermediate node**, is also used. The number of packages sent by each source and intermediate node is used to let the EA minimize independently the number of packages sent by the pipelines departing each distributing node and the total number of packages to make the EA minimize the total number of packages simultaneously.

For each population the objective functions are evaluated and the population is ranked using the preferability relation given in figure 7. Now a dominance matrix is built. This matrix keeps the dominance relation

between each pair of individuals. Fitness is assigned over the population using Goldberg method [1]. In this method the population is divided in several groups. First, it is formed a group with the individuals that are not dominated; next group is obtained by eliminating of the population the individuals from the first group and taking the individuals that, now, are not dominated, and so on. The individuals in a group are given the same fitness value. The worst group is given a fitness value of 1, the second worst group is given a fitness value of 1 + Δ, the third worst group is given a fitness value of 1+2Δ, and so on. Where, Δ is a design parameter. The fitness value given to every group with the objectives values obtained by each individual gives the final value used for selecting the parents.

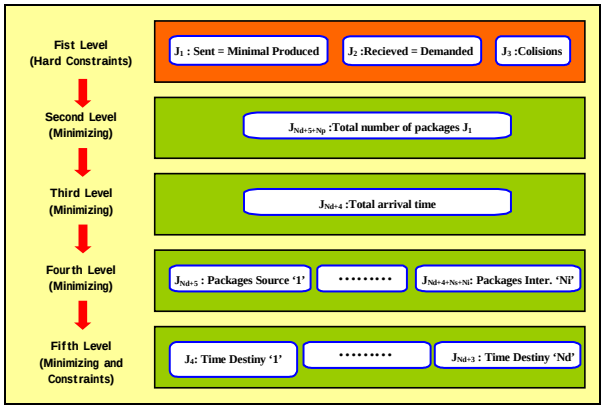

Figure 7. Scheme of priorities.

V. EXAMPLE

To solve the problem we have used the toolbox EVOCOM [6]. We show the results obtained when applying the algorithm to the network of figure 8. This network has 3 source nodes, 4 sink nodes and 5 intermediate nodes. There are 20 unidirectional links and one bidirectional link, and, therefore, 22 connection points. We suppose that three kind of products can flow through the network: A, B and C.

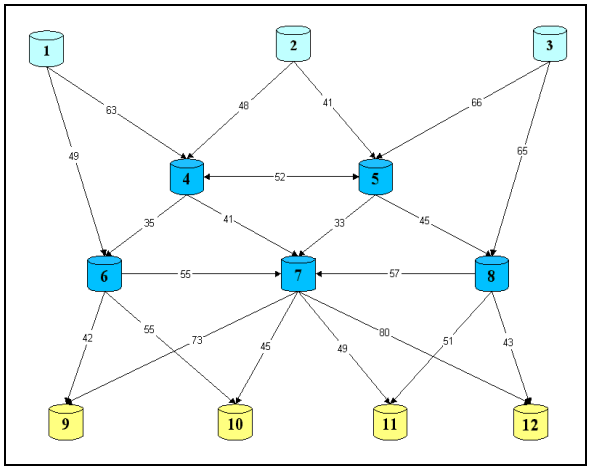

Figure 8: Network model.

Table 4 shows the minimum, maximum and initial content of the tanks of the source and intermediate nodes, table 5 shows the minimum production in the source nodes, and table 6 shows for every product the minimum and maximum time allowed to fulfill the demand, as well as the demanded amount.

	Type A			Type B			Type C		
NODE	Min	Cont	Max	Min	Cont	Max	Min	Cont	Max
Source 1	0	100	100	0	100	100	0	100	100
Source 2	0	100	100	0	100	100	0	100	100
Source 3	0	100	100	0	100	100	0	100	100
Inter. 1	0	50	50	0	50	50	0	50	50
Inter. 2	0	50	50	0	50	50	0	50	50
Inter. 3	0	30	50	0	30	50	0	30	50
Inter. 4	0	30	50	0	30	50	0	30	50
Inter. 5	0	30	50	0	30	50	0	30	50

Table 4: Capacity and content of the tanks of source and intermediate nodes

	Type A	Type B	Type C
Source 1	10	10	10
Source 2	10	10	10
Source 3	10	10	10

Table 5: Minimum production of packets of every source

	Type A			Type B			Type C		
NODO	T.Min	T.Max	Nºpaq	T.Min	T.Max	Nºpaq	T.Min	T.Max	Nºpaq
Destiny 1	0	100	21	0	100	20	0	100	20
Destiny 2	0	100	20	0	100	18	0	100	19
Destiny 3	0	100	22	0	100	22	0	100	21
Destiny 4	0	100	15	0	100	16	0	100	18

Table 6: Number of demanded packets and allowed arrival time

The following parameters are chosen in the algorithm:

Parameter	Value
Number of individua	21
Number of sustitutions	7
Number of generations	12000
Crossover probability	0.8
Number of cross points	2
Probability of mutation	0.008

Table 7: The most relevant EA parameters

For the recombination process we took one of the recombination methods provided by the Toolbox EVOCOM [6]. The tried method was *rec_subgen* which keep constant the population size. This method operates in the following form: given the old and new population (with *n* individuals), it substitutes the last *n* individuals of the old population by the individuals of the new population.

While running the EA, the values of the different objectives of the best solution in each generation were stored and are presented in the following figures to show the evolution of the algorithm. The results obtained after

12000 generations are given in the following figures. Figure 9 shows the values of the objective functions in the first 10 generations. As can be seen, the initial population fulfills constraints C1 and C3, and these constraints keep on fulfilling. Initially only two individuals violates constraint C2, but as soon as two generations later no violation occurs.

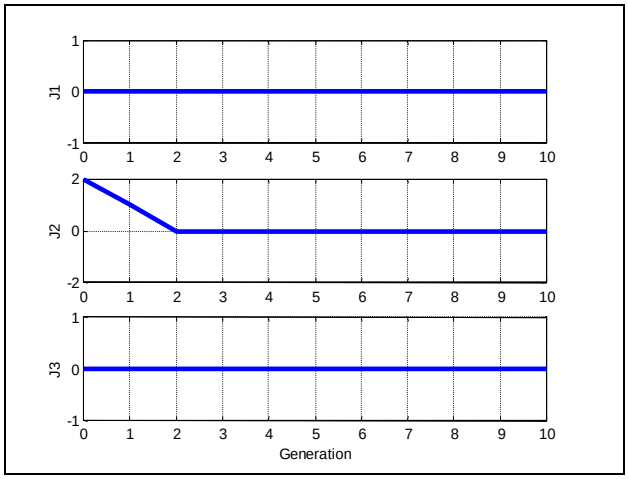

Figure 9: Evolution of the cost functions of constraints C1, C2 C3

Figure 10 shows the cost functions for objective O1, that is the maximum time of arrivals of packets to the sink nodes.

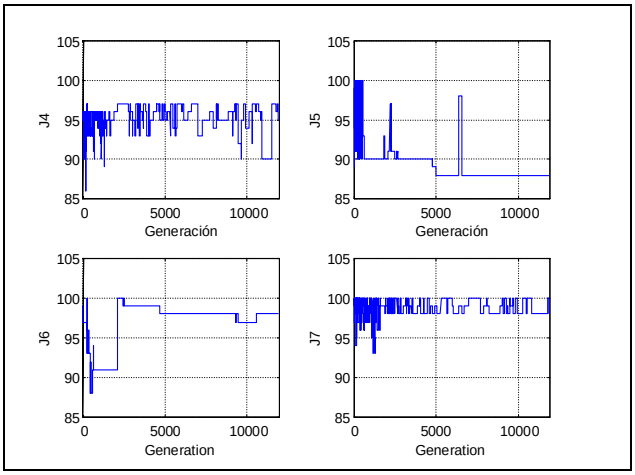

Figure 10: Time spend in satisfying demand at the sink nodes (objective O1)

Figure 11 shows the cost function for objective O2. We can appreciate that about the generation 2000 there is an increase in the cost function, followed by a trend to decrease. As we can see in figures 11 and 12, this is due to an increase in the rate of reduction of objectives O3 and O4, that is in reducing the number of different packets sent through the delivering nodes.

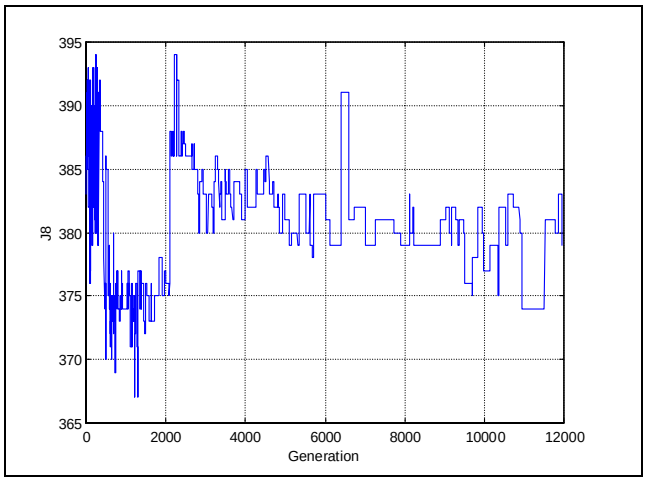

Figure 11: Sum of times in figure 10 (objective O2).

Figure 12 shows the evolution of the fragmentation at the output of sources and intermediate nodes (objectives O3).

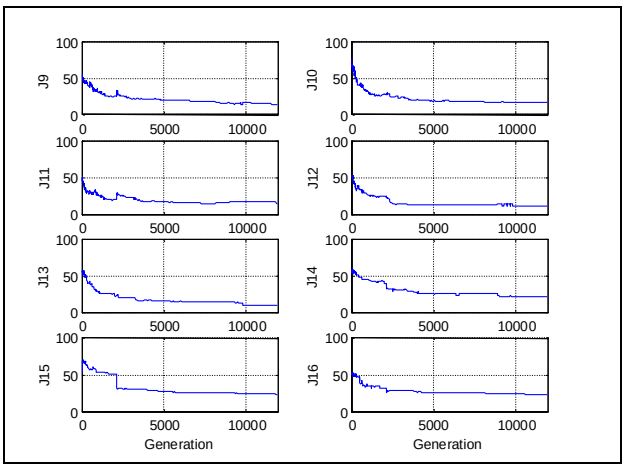

Figure 12: Fragmentation at the output of the delivery nodes. Objective O3.

We can see a continuous decreasing trend. This trend is more clearly appreciated in figure 13 where the sum of the previous objective functions is shown.

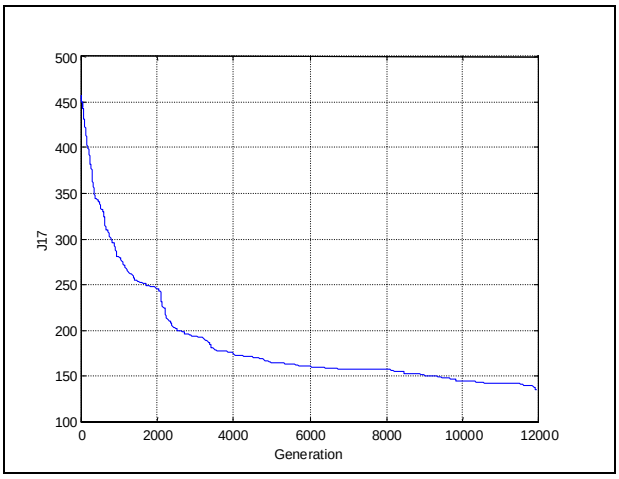

Figure 13: Total fragmentation. Objective O4

VI. CONCLUSIONS

We have presented a simplified problem about the continuous distribution of products through an oil-pipeline network. The main objective of the network is satisfy demands in a minimum time and to avoid fragmentation as much as possible in order to reduced contamination of products. Moreover, a number of constraints must be fulfilled. Although the problem belongs to the kind of NP-complete problems has been solved using an evolutionary multiobjective and constraint optimization algorithm. The problem solved can be used for networks with every number of sources, sinks and intermediate nodes, and with any number of products.

VII. REFERENCES

[1] Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.

[2] Michalewicz, Z.Genetic Algorithms +Data Structures=Evolution Programs. 3rd ed. Springer-Verlag, Berlin

[3] Coello Coello, C.A. (2000). An Updated Survey of GA-Based Multiobjective Optimization Techniques. ACM Computing Surveys. Vol. 32. nº 2. June 2000, 109-143.

[4] Fonseca, C. M. and P.J. Fleming (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithm-Part I: Unified Formulation. IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans. Vol. 28, nº 1. January 1998, 26-37.

[5] Michalewicz, Z. and Schoenauer, M. Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation 4(1): 1-32, 1996.

[6] Besada-Portas, E. López-Orozco, J.A., Andrés-Toro. B. (2001). A versatile toolbox for solving industrial problems with several evolutionary techniques. In Evolutionary Methods for Design, Optimization and Control. Ed. International Centre for Numerical Methods in Engenieering (CIMNE). Barcelona. Spain, March 2002. ISBN 84-89925-97-6.