OPTIMAL ALGORITHM FOR THE DEMAND ROUTING PROBLEM IN MULTICOMMODITY FLOW DISTRIBUTION NETWORKS WITH DIVERSIFICATION CONSTRAINTS AND CONCAVE COSTS

Abstract.-
Distribution problems are of high relevance within the supply chain system. In real life situations various different commodities may flow in the distribution process. Furthermore, the connection between production and demand centres makes use of complex mesh networks that can include diversification constraints to avoid overcharged paths. In addition, the consideration in certain situations of economies of scale gives rise to non-linear cost functions that make it difficult to deal with an optimal routing scheme. This problem is well represented by the multicommodity flow distribution networks with diversification constraints and concave costs (MFDCC) problem. Here we present an optimal algorithm based on the Kuhn-Tucker optimality conditions of the problem and capable of supplying optimal distribution routes in such complex networks. The algorithm follows an iterative procedure. Each iteration constructive solutions are checked with respect to the Kuhn-Tucker optimality conditions. Solutions consider a set of paths transporting all the demand allowed by its diversification constraint (saturated paths), a set of empty paths, and an indicator path transporting the remaining demand to satisfy the demand equation. The algorithm reduces the total cost in the network in a monotonic sequence to the optimum. The algorithm was tested in a trial library and the optimum was reached for all the instances. The algorithm showed a major dependency with respect to the number of nodes and arcs of the graph, as well as the density of arcs in the graph.

Keywords: demand routing; concave cost; distribution network; Kuhn Tucker conditions

1. Introduction

Distribution problems are of high relevance within the supply chain system. One of the principal root problems in the family is the transportation problem that was introduced in 1941 by Hitchcock (Hitchcock, 1941). This has been researched extensively due to its importance and the wide range of applications it has. The transportation problem involves a network with a group of production centres willing to send their commodity to a group of demand centres through a set of arcs with linear costs.

However, in real life situations various different commodities may flow instead of just one commodity (e.g. Kengpol *et al.,* 2011). Furthermore, each production centre is not connected with each demand centre using a single arc but rather with a mesh network of different arcs (Ubeda *et al*., 2010). Certain cases cannot be correctly modelled using a linear cost function and require a more complex way of modelling (Tsao *et al*. 2010). Sometimes more accurate results can be achieved with concave cost functions. This situation arises when dealing with economies of scale which are very common in many fields such as freight distribution and demand routing systems, and even other technological fields such as telecommunication networks.

In this paper, we consider a mesh network with a set of nodes as the origin of demand and a set of nodes as the demand destination. Each pair of nodes constitutes an origin-destination pair which is known as a specific commodity, converting the problem into a multicommodity flow problem. Additionally, the costs in arcs are modelled by concave costs. Lastly, we propose the diversification constraints needed to prevent the arcs from becoming extremely loaded which can produce malfunction effects on the network and may arise when concave costs are involved as a result of the benefits offered by economies of scale. This type of constraint increases the survivability and reliability of the networks.

Distribution problems with concave costs are NP-Hard and their optimal solution cannot be found in polynomial time. Several authors have proposed approximate methods to tackle this problem, such as the linear approximation by Thach (1992), the lagrangian relaxation by Larsson *et al.* (1994), or the dynamic programming approach by Zangwill (1968), Florian and Klein (1971) and Burkard *et al.* (2001). Metaheuristic approaches have also been tested although mainly in telecommunication network problems (Kapsalis *et al*, 1993; Dengiz *et al*, 1997; Altiparmak *et al*, 2003; Zhou and Gen, 2003). In a more general case, Yan and Luo (1999) developed a heuristic based on simulated annealing and threshold acceptance, and Altiparmark and Karaoglan (2008) proposed a tabu approach to solve the transportation problem with concave costs. Here, we have developed an optimal approach based on the optimality Kuhn-Tucker conditions of the mathematical formulation of the problem.

The paper then defines the problem and its mathematical formulation in Section 2. Section 3 presents the Kuhn-Tucker optimality conditions for the demand routing problem in multicommodity flow networks with diversification constraints and concave costs that has been described in the previous section. Next, the solution based on the optimality constraints demand routing algorithm is proposed is Section 4. The results for a trial library with 27 generated instances are presented in Section 5 and the detailed procedure for an example network is developed. Finally, the main conclusions are presented in Section 6.

2. Problem definition and formulation

Given a graph $G=(N,E)$ where N is the set of nodes and E the set of arcs, a set of commodities K is considered and represents every origin-destination pair of demand. We define each arc of the network as $e \in E$ and the set of paths connecting every origin-destination pair is given by $P(k)$. A subset of this set is given by the arc-disjoint paths, $P^d(k)$.

Let γ_k be the demand volume for each origin-destination pair, k, and δ_k the diversification parameter for each pair k, so that the demand of pair k is routed along $\lceil 1/\delta_k \rceil$ different paths. As paths and arcs are pre-computed, φ_{eh} is a binary parameter that determines if arc e is on path h.

The variables of the problem are the demand fraction related to the origin-destination pair k that is routed on path h, which is a continuous variable given by p_{kh}, and the total

flow transported on arc *e*, including the total amount of flow from all the commodities on the arc, which is a continuous variable given by x_e.

Owing to the nature of the problem described, numerous concave cost functions can be considered. We used the objective function given by $c_e(x_e) = d_e\sqrt{x_e}$ for arc *e*, where x_e has been previously explained and d_e means the per unit variable cost corresponding to the arc. We selected this function because it provides a widely acknowledged basis of analysis of scale economies' effect in transportation problems, and because it is taken as a representative cost function very similar to many transportation cost functions in practice (see detailed arguments in LeBlanc 1976; Larsson et al. 1994; Yan and Luo 1999; and Altiparmak and Karaoglan 2008). The cost concave function is, therefore, a differentiable function. Note that the objective function is used to provide comparable numerical examples in the subsequent results section. The selection of this function does not condition the methodology and developed algorithm, because the objective function only affects the procedure for the calculation of the derivative providing numerical data.

As a result, the demand routing problem in multicommodity flow networks with diversification constraints and concave costs (MFDCC problem) can be formulated as:

$$\text{Minimize} \quad \sum_{e\in E} c_e(x_e)$$

s.t.:

$$x_e = \sum_{k\in K}\sum_{h\in \mathrm{P}(k)} \varphi_{eh} p_{kh} \ , \ \forall e \in E \qquad (1)$$

$$\sum_{h\in \mathrm{P}(k)} p_{kh} = \gamma_k \ , \ \forall k \in K \qquad (2)$$

$$p_{kh} \le \delta_k \gamma_k \ , \ \forall h \in \mathrm{P}^d(k) \ , \ \forall k \in K \qquad (3)$$

$$x_e \ge 0 \ , \ p_{kh} \ge 0$$

Constraint (1) calculates the total amount of flow on the arc *e*; constraint (2) ensures that the demand for every origin-destination pair is met; and constraint (3) ensures that the demand is routed along $\lceil 1/\delta_k \rceil$ alternative paths. Set $P^d(k)$ represents all the feasible disjoint paths for each origin-destination pair. The "disjoint" concept is important and can be weakly imposed on the arcs or strongly imposed on the nodes. Here, we have chosen the arc diversification concept because in most cases it is normally only necessary to take this weak constraint into consideration. Using an example from telecommunications, disjoint arcs can be used to model reality, since the optical cross-connect (OXC) in an all-optical WDM mesh network is seldom broke. The OXC is a device which switches an optical signal from an incoming fibre to an outgoing fibre on the same wavelength. So, constraint (3) is equivalent to:

- $p_{kh} \le \delta_k \ , \ \forall (i,j) \in h \ , \ h \in P^d(k) : \ k \in K$, constraint for arc-disjoint paths
- $p_{kh} \le \delta_k \ , \ \forall j \in h \ , \ h \in P^d(k) : \ k \in K$, constraint for node-disjoint paths

depending on what is being considered and as explained above, we are looking at

constraints for arc-disjoint paths.

3. Kuhn Tucker conditions for the demand routing problem in multicommodity flow networks with diversification constraints and concave costs

Every solution verifying the necessary and sufficient Kuhn-Tucker conditions of any problem will be the optimal solution of problem. This fact is proven for linear or non-linear problems since they are global optimality conditions. Additionally, duality theory can be regarded as a particular case of the Kuhn-Tucker conditions for linear problems. See classic theory such as Hillier and Lieberman (1977), Nocedal and Wright (1999), Avriel (2003) for detailed explanations on Kuhn-Tucker conditions. Appendix 1 depicts a general overview on the Kuhn-Tucker necessary optimality conditions and their detailed corresponding application to MFDCC problem, and appendix 2 depicts the equivalent general overview and detailed application of the Kuhn-Tucker sufficient optimality conditions to the MFDCC problem.

Next, we re-formulate MFDCC problem to specify the objective function in terms of p_{kh} variables. This can be done because p_{kh} and x_e are inter-related by constraint (2). Therefore, constraint (2) is included in the objective function to re-formulate the problem and subsequently formulate the Kuhn-Tucker necessary optimality conditions:

$$Minimize \sum_{e\in E} c_e\left(x_e\right) \equiv Minimize\ C\left(x\right) \equiv Minimize\ C\left(p\right)$$

s.t.:

$$\gamma_k - \sum_{h\in P(k)} p_{kh} = 0\ ,\ \forall k \in K \qquad \leftarrow \mu_k \text{ (free multipliers)}$$

$$p_{kh} - \delta_k \cdot \gamma_k \le 0\ ,\ \forall h \in P^d\left(k\right)\ ,\ \forall k \in K \qquad \leftarrow v_{kh} \text{ (non-negative multipliers)}$$

$$-p_{kh} \le 0\ ,\ \forall h \in P^d\left(k\right)\ ,\ \forall k \in K \qquad \leftarrow u_{kh} \text{ (non-negative multipliers)}$$

Kuhn-Tucker conditions state that all $h \in P^d\left(k\right)$ and $k \in K$ can be written as:

$$\begin{aligned} \frac{\partial C(p)}{\partial p_{kh}} - \mu_k &= u_{kh} - v_{kh} \\ (\,p_{kh} - \delta_k \cdot \gamma_k\,) \cdot v_{kh} &= 0 \\ p_{kh} \cdot u_{kh} &= 0 \end{aligned} \qquad (4)$$

By calculating the cost function variation related to the demand fraction routed along path *h*, we replace the expression of the total cost in the network $C(p)$ with the sum of the costs of the flows in the arcs, and then we substitute the value of the flow x_e accordingly with equation (1):

$$\frac{\partial C(p)}{\partial p_{kh}} = \frac{\partial \sum_{e \in E} c_e(x_e)}{\partial p_{kh}} = \sum_{e \in E} \frac{\partial c_e(x_e)}{\partial p_{kh}} = \sum_{e \in E} \frac{\partial c_e(x_e)}{\partial x_e} \frac{\partial x_e}{\partial p_{kh}} =$$
$$= \sum_{e \in E} \left[c'_e(x_e) \cdot \frac{\partial}{\partial p_{kh}} \left(\sum_{k \in K} \sum_{h \in P(k)} p_{kh} \varphi_{eh} \right) \right] = \sum_{e \in E} c'_e(x_e) \varphi_{eh} \qquad (5)$$

that corresponds to the sum of cost variations on the arcs belonging to path *h*. The optimality conditions are thus enforced on the increase of the cost and not on its absolute value. By analysing these conditions, we can differentiate the meaning of the conditions and three different types of paths are found:

- Case 1: the origin-destination pair does not use path *h* to establish a communication.

$$\begin{aligned} & p_{kh} = 0 \Rightarrow u_{kh} \geq 0 \text{ and } v_{kh} = 0 \\ & \frac{\partial C(p)}{\partial p_{kh}} - \mu_k = u_{kh} \geq 0 \\ & \sum_{e \in E} c'_e(x_e) \varphi_{eh} - \mu_k = u_{kh} \geq 0 \\ & \sum_{e \in E} c'_e(x_e) \varphi_{eh} \geq \mu_k \end{aligned} \qquad (6)$$

  Therefore, the sum of the cost variations corresponding to the arcs on the *empty or void paths* that are not used by the origin-destination pair *k* is greater than μ_k.

- Case 2: the origin-destination pair uses path *h* to establish a communication. Here, two possibilities arise:
  - Case 2.a: the arc transports all the flow allowed by the diversification constraint. There are $\left(\left\lceil \frac{1}{\delta_k} \right\rceil - 1 \right)$ paths in this case. In this case, the Kuhn-Tucker conditions are reduced to:

$$\begin{aligned} & p_{kh} = \delta_k \Rightarrow u_{kh} = 0 \text{ and } v_{kh} \geq 0 \\ & \frac{\partial C(p)}{\partial p_{kh}} - \mu_k = - v_{kh} \leq 0 \\ & \sum_{e \in E} c'_e(x_e) \varphi_{eh} - \mu_k = - v_{kh} \leq 0 \\ & \sum_{e \in E} c'_e(x_e) \varphi_{eh} \leq \mu_k \end{aligned} \qquad (7)$$

    The sum of the cost variations corresponding to the arcs on the *saturated paths* used by the origin-destination pair *k*, i.e. $h \in P^d(k)$, is upper bounded by μ_k.

- Case 2.b: the arc transports less flow than the limitation imposed by the diversification constraint. There is only one path that verifies this situation, which is:

$$\begin{aligned} & p_{kh} < \delta_k \Rightarrow u_{kh} = 0 \text{ and } v_{kh} = 0 \\ & \frac{\partial C(p)}{\partial p_{kh}} - \mu_k = 0 \\ & \sum_{e \in E} c'(x_e)\varphi_{eh} = \mu_k \end{aligned} \qquad (8)$$

  The sum of the cost variations corresponding to the arcs on the path that transports a positive amount of flow but below the maximum value imposed by the diversification constraint related to the origin-destination pair *k* is equal to μ_k. This path will be known as the *indicator path*. This is a concept that appears in similar terms in the traditional transportation problem (when links in the graph are grouped into three sets; a set of slacks equal to zero; a set of slacks equal to the capacity, that is empty arcs; and one link with an amount of flow equal to the difference between the demand and the sum of all the flow in arcs with slack equal to zero). We originally introduced this term in Cortes *et al*. (2006).

The existence of only one indicator path is also intuitive. Note that if there was a cheaper path, that path would be full, and if there was a more expensive path it should be empty, and so the path in the middle separating cheap paths from expensive paths should transport the rest of demand to satisfy the demand equation. There could be only two particular cases: (i) the rest of demand is equal to the capacity, so it would be full, (ii) there is a draw between two paths, corresponding to a case of alternative solutions, but the solution cost would be the same, and these two cases do not condition the algorithm procedure that will be later detailed in section 4, because either of the drawing paths could act as indicator as a matter of fact.

In summary:

- For every origin-destination pair, *k*, cheap, expensive and indicator paths can be considered.
  The cheap paths are saturated and transport as much flow as possible, i.e.: $p_{kh} = \delta_k \cdot \gamma_k$.
  The expensive paths are empty and do not transport flow, i.e.: $p_{kh} = 0$
  The indicator path transports the necessary amount of flow to satisfy the demand equation (eq. 2), which is less than $\delta_k \cdot \gamma_k$ and more than 0.

- The transported product that flows through the communication paths in order to satisfy the demand of all origin-destination pairs will be optimum if the "separator cost concept" is given by $\mu_k = \sum_{e \in E} c'(x_e)\varphi_{eh}$ in the indicator path.

In this way, the Kuhn-Tucker multipliers include the following meanings:

$\mu_k \sim c'_{h_i}$ It is a concept associated with the value of the derivative of the cost function along the indicator path.

$v_{kh} \sim \left(c'_{h_i} - c'_{hs}\right)$ It is a concept associated with the deviation between the values of the derivative of the cost function of the indicator path and each saturated path. The Kuhn-Tucker optimality conditions determine that this deviation must be positive for every $h \in P^d(k)$ and $k \in K$.

$u_{kh} \sim \left(c'_{h_v} - c'_{h_i}\right)$ It is a concept associated with the deviation between the values of the derivative of the cost function of each empty path and the indicator path. The Kuhn-Tucker optimality conditions determine that this deviation must be positive for every $h \in P^d(k)$ and $k \in K$.

4. Demand routing algorithm

After having explained the Kuhn-Tucker optimality conditions above, this section describes an optimal iterative algorithm that can be used to solve the demand routing problem in multicommodity flow networks with diversification constraints and concave costs. The algorithm converges to the optimal process by reducing the non-feasibility after every iteration.

The algorithm needs to pre-compute all the paths of the graph between a pair of nodes. This is done with an algorithm based on a depth first search routine that is shown in appendix 4. Although the problem of finding all the paths between every pair of nodes in a graph is a NP-Hard problem, we consider sparse networks, so that results can be obtained quickly as the result section shows next.

Step 1. Initialisation. Greedy heuristic

The procedure is based on the *k* shortest disjoint path algorithm provided by Kleinberg (1996).

1. Order the origin-destination pairs according to the demand volume, γ_k. Subsequently, following that order:
2. For each origin-destination pair, *k*:
    a. For a number of times equal to $\lceil 1/\delta_k \rceil$-1
        i. Calculate the shortest path between the origin $o(k)$ and the destination $d(k)$.This is a saturated path $h_s \in P^d(k)$.
        ii. Assign a volume of demand equal to $\delta_k \cdot \gamma_k$.
        iii. Erase the path to search for a new disjoint path.
    b. Calculate the shortest path between the origin $o(k)$ and the destination $d(k)$, and assign the remaining demand to satisfy equation (2). This is the candidate indicator path $h_i \in P^d(k)$.

In the greedy heuristic the costs of the arcs are simplified to $c_e(x_e) = d_e$ in order to calculate the shortest paths. Dijkstra's algorithm can, therefore, be applied. This initial flow load provides a determined level of flow along the paths, p_{kh}^0, corresponding to a

specific level of flow load on the arcs of the network, x_e^0.

We used this greedy heuristic with the objective of providing an initial set of disjoint paths. It starts selecting a set of edge-disjoint paths that provides an initial set that is only considered as the seed of the algorithm. After that, the iterative algorithm (next steps of the iterative routine that forces to accomplish the Kuhn-Tucker conditions) guides the flow in the paths from that seed to optimality by testing the optimality conditions for all the paths of the graph connecting such pair *k*. The main reason to use this shortest path based initialization algorithm is to generate a seed quickly. Furthermore, as the shortest path tries to minimise the cost between a pair of nodes, this can be taken as a reasonably good initial solution following the arguments detailed in Kleinberg (1996).

Step 2.Checking the Kuhn-Tucker optimality conditions

The Kuhn-Tucker conditions are checked for every origin-destination pair, *k*. The condition is given by (9):

$$\begin{aligned} & c'_{h_s}\left(p_{kh}^0\right) \le c'_{h_i}\left(p_{kh}^0\right) \le c'_{h_v}\left(p_{kh}^0\right) \\ & \text{for every } h = \left\{h_s\left(\text{saturated path}\right), h_i\left(\text{indicator path}\right), h_v\left(\text{empty or void path}\right)\right\} \in P^d\left(k\right) \\ & \text{where } c'_h\left(p_{kh}\right) = \sum_{e \in E} c'_e\left(x_e^0\right) \cdot \varphi_{eh} \end{aligned} \qquad (9)$$

After the saturated and indicator paths have been obtained (as a result of step 1 or the iterative process), we have to check the conditions for all the empty paths of the graph (it has to be taken into account that the condition must be verified for all the empty paths in the graph as well. These do not necessarily have to be disjoint paths).

The iterative process comes to an end if the optimality conditions are checked; otherwise, the procedure follows step 3.

Step 3. Iteration

All the origin-destination pairs that do not check the optimality conditions shape the set $\bar{K}$ (we say that $k \in \bar{K}$ if pair *k* does not verify the Kuhn-Tucker conditions, i.e. eq. 9). For these pairs, the set *H*(*k*) is defined as the paths that do not check the condition of such pair, $k \in \bar{K}$. Next procedure is followed:

1. For every $k \in \bar{K}$, paths in *H*(*k*) are arranged in increasing order according to the derivative of the cost function assessed for the previous (step 2) demand routing pattern, p_{kh}^0. That is, $c'_h\left(p_{kh}^0\right) = \sum_{e \in E} c'_e\left(x_e^0\right) \cdot \varphi_{eh}$ is calculated for every $h \in$ *H*(*k*), for every $k \in \bar{K}$.

2. Origin-destination pair k^* is selected as it is the pair that provides the highest absolute deviation with respect to the optimality conditions, as calculated in (10).

$$k^* = \left\{ k \in \bar{K} : \max_{h \in H(k)} \left\{ \max_{h_s} \left\{ \left| c'_{h_s}\left(p^0_{kh}\right) - c'_{h_i}\left(p^0_{kh}\right) \right| \right\}, \max_{h_v} \left\{ \left| c'_{h_i}\left(p^0_{kh}\right) - c'_{h_v}\left(p^0_{kh}\right) \right| \right\} \right\} \right\} \qquad (10)$$

3. For $k^* \in \bar{K}$, the maximum demand allowed by constraint (3), $\delta_k \cdot \gamma_k$, is assigned to the "$\left(\lceil 1/\delta_k \rceil - 1\right)$" disjoint paths with lower value of $c'_h\left(p^0_{k^*h}\right)$. These are saturated paths for the new iteration.

4. The next disjoint path, following the order determined by the derivative, transports the rest of the required demand to satisfy constraint (2). This is the indicator path for the new iteration.

Subsequently, the procedure follows step 2 to check the optimality of the new demand routing pattern.

Although we have tackled for this algorithm only the Kuhn-Tucker optimality necessary conditions, the sufficient condition is always checked for every intermediate feasible (non-optimal) solution of the algorithm given the special characteristics of the problem. In fact, the sufficiency condition is always checked independently from its optimality condition due to every solution composed of an indicator path, a number of $\left(\lceil 1/\delta_k \rceil - 1\right)$ saturated paths and a set of empty paths checks it (see appendix 2 for a detailed argument and proof). Therefore, a feasible solution $\left[p_{kh}\right]^*$ which also checks the necessary optimality conditions previously described in equation (9) will be the global optimum of the MFDCC problem. See Hanson and Mond (1987) for detailed arguments and explanations for necessary and sufficient conditions in optimisation problems.

In addition, the procedure guarantees that the algorithm follows a monotonic sequence that reduces the cost of the objective function step-by-step. Proof and arguments are shown in appendix 3 arguing how the total cost is reduced after each iteration.

5. Computational results

5.1. Application example

We present here the next case as an application example to demonstrate the performance of the algorithm. Consider the eight nodes, and eleven arcs network in Figure 1. Data over the arcs represent the value of d_e in the cost function $c_e\left(x_e\right) = d_e\sqrt{x_e}$. The remaining data appear in Table 1.

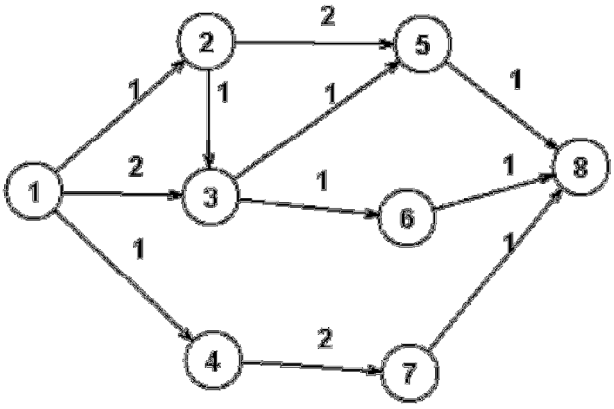

Figure 1. Example network

Table 1. Origin-destination data

$k \in K$	origin-destination	γ_k	δ_k
1	1-2	46	1.00
2	1-3	12	0.53
3	1-5	31	0.89
4	1-8	51	0.51
5	1-6	3	1.00
6	1-7	36	1.00
7	2-5	33	0.8
8	2-8	16	0.95
9	2-6	3	1.00
10	3-8	44	0.38
11	3-6	6	1.00
12	4-8	33	1.00
13	7-8	34	1.00

Although this example network is a simple case, the size of the problem leads to 37 constraints (11 corresponding to the flow evaluation in the arc, eq. [1]; 13 corresponding to the demand satisfaction equation, eq. [2]; and 13 corresponding to the diversification constraint, eq. [3]), and 154 variables (143 path variables, p_{kh}, and 11 flow variables, x_e). The example provides an idea of how complex the problem is in real cases.

We initialise the demand routing pattern in the network with the values shown in Table 2:

Table 2. Greedy heuristic: first demand routing pattern

$k \in K$	Origin-destination	p_{kh}	Path	
1	1-2	46.00	(1,2)	
2	1-3	6.31	(1,3)	h_s
		5.69	(1,2);(2,3)	h_i
3	1-5	27.47	(1,2);(2,5)	h_s
		3.53	(1,3);(3,5)	h_i
4	1-8	26.01	(1,2);(2,5);(5,8)	h_s
		24.99	(1,3);(3,6);(6,8)	h_i
		0.00	(1,4);(4,7);(7,8)	h_v
5	1-6	3.00	(1,3);(3,6)	h_s
		0.00	(1,2);(2,3);(3,6)	h_v
6	1-7	36.00	(1,4);(4,7)	
7	2-5	26.37	(2,5)	h_s
		6.63	(2;3);(3,5)	h_i
8	2-8	15.12	(2,5);(5,8)	h_s
		0.88	(2,3);(3,6);(6,8)	h_i
9	2-6	3.00	(2,3);(3,6)	
10	3-8	36.55	(3,5);(5,8)	h_s
		7.45	(3,6);(6,8)	h_i
11	3-6	6.00	(3,6)	
12	4-8	33.00	(4,7);(7,8)	
13	7-8	34.00	(7,8)	

This assignment provided a total cost given by (11) and equal to 105.02.

$$CT = \sum_{e \in E} c_e(x_e) = \sum_{e \in E} d_e \sqrt{x_e} = \sum_{e \in E} d_e \sqrt{\sum_{k \in E} \sum_{h \in P(k)} p_{kh} \varphi_{eh}} \tag{11}$$

The second step was to check the optimality conditions in pairs 2; 3; 4; 5; 7; 8; and 10. To do so, we needed to calculate the value of the derivatives, $c'_h(p_{kh})$, for every path connecting such pairs (eq. 12):

$$c'_h(p_{kh}) = \sum_{e \in E} c'_e\left(x_e^0\right) \cdot \varphi_{eh} = \sum_{e \in E} \left(\frac{d_e}{2\sqrt{x_e}} \cdot \varphi_{eh} \right) = \sum_{e \in E} \left(\frac{d_e}{2\sqrt{\sum_{k \in K} \sum_{h \in P(k)} p_{kh} \delta_e^{kh}}} \cdot \varphi_{eh} \right) \tag{12}$$

By computing the values for those origin-destination pairs, the following results were obtained (Table 3):

Table 3. Checking the optimality conditions. 1st iteration

$k \in \mathbf{K}$	Origin-destination	p_{kh}	Path	Type	$c'_h(p_{kh})$
2	1-3	6.31	(1,3)	h_s	0.163
		5.69	(1,2);(2,3)	h_i	0.173
3	1-5	27.47	(1,2);(2,5)	h_s	0.151
		3.53	(1,3);(3,5)	h_i	0.236
4	1-8	26.01	(1,2);(2,5);(5,8)	h_s	0.208
		24.99	(1,3);(3,6);(6,8)	***h_i***	***0.323***
		0.00	(1,4);(4,7);(7,8)	***h_v***	***0.265***
5	1-6	3.00	(1,3);(3,6)	h_s	0.237
		0.00	(1,2);(2,3);(3,6)	h_v	0.247
7	2-5	26.37	(2,5)	h_s	0.103
		6.63	(2;3);(3,5)	h_i	0.197
8	2-8	15.12	(2,5);(5,8)	h_s	0.159
		0.88	(2,3);(3,6);(6,8)	h_i	0.285
10	3-8	36.55	(3,5);(5,8)	h_s	0.130
		7.45	(3,6);(6,8)	h_i	0.161

The optimality conditions as stated in (9) were not checked for pair 4, specifically, it was the comparison value between the indicator and empty paths of pair k = 4 that was not checked. Values are shown in bold and italic figures. So, in this case $\bar{K} = \{4\}$ with k^* = 4 as well. The saturated path stayed the same (it had the lowest value for the derivative) and the indicator and empty paths had to be swapped. So, according to the algorithm 24.99 units of flow followed path (1,4);(4,7);(7,8) and path (1,3);(3,6);(6,8) remained empty.

This modification affected not only pair 4, but also every pair whose communicating paths contained some of the arcs whose relative demand fraction had been modified. Subsequently, we recalculated the values as shown in Table 4:

Table 4. Checking the optimality conditions. 2nd iteration

$k \in \mathbf{K}$	Origin-destination	p_{kh}	Path	Type	$c'_h(p_{kh})$
2	1-3	6.31	(1,3)	***h_s***	***0.279***
		5.69	(1,2);(2,3)	***h_i***	***0.173***
3	1-5	27.47	(1,2);(2,5)	h_s	0.151

		3.53	(1,3);(3,5)	h_i	0.352
4	1-8	26.01	(1,2);(2,5);(5,8)	h_s	0.208
		24.99	(1,4);(4,7);(7,8)	h_i	0.219
		0.00	(1,3);(3,6);(6,8)	h_v	0.563
5	1-6	3.00	(1,3);(3,6)	$\boldsymbol{h_s}$	***0.390***
		0.00	(1,2);(2,3);(3,6)	$\boldsymbol{h_v}$	***0.284***
7	2-5	26.37	(2,5)	h_s	0.103
		6.63	(2;3);(3,5)	h_i	0.197
8	2-8	15.12	(2,5);(5,8)	h_s	0.159
		0.88	(2,3);(3,6);(6,8)	h_i	0.408
10	3-8	36.55	(3,5);(5,8)	h_s	0.130
		7.45	(3,6);(6,8)	h_i	0.284

This new assignment provided a total cost given by (11) and equal to 100.77.

The optimality conditions were not checked by pair 2 (where the derivative of the saturated path was greater than the derivative of the indicator path) and pair 5 (where the derivative of the saturated path was greater than the derivative of the empty path) as can be viewed in Table 4 (values are shown in bold and italic figures) being $\overline{K} = \{2,5\}$. The difference (equation 10) is the same for both cases and equal to 0.106, and any of them can be chosen. We selected for this iteration $k^* = 2$. By performing the corresponding re-arrangement for pair 2, the following results were obtained (Table 5):

Table 5. Checking the optimality conditions. 3rd iteration

$k \in \mathbf{K}$	Origin-destination	p_{kh}	Path	Type	$c'_h(p_{kh})$
2	1-3	6.31	(1,2);(2,3)	h_s	0.171
		5.69	(1,3)	h_i	0.286
3	1-5	27.47	(1,2);(2,5)	h_s	0.151
		3.53	(1,3);(3,5)	h_i	0.359
4	1-8	26.01	(1,2);(2,5);(5,8)	h_s	0.208
		24.99	(1,4);(4,7);(7,8)	h_i	0.219
		0.00	(1,3);(3,6);(6,8)	h_v	0.570
5	1-6	3.00	(1,3);(3,6)	$\boldsymbol{h_s}$	***0.397***
		0.00	(1,2);(2,3);(3,6)	$\boldsymbol{h_v}$	***0.281***
7	2-5	26.37	(2,5)	h_s	0.103
		6.63	(2;3);(3,5)	h_i	0.195
8	2-8	15.12	(2,5);(5,8)	h_s	0.159
		0.88	(2,3);(3,6);(6,8)	h_i	0.406
10	3-8	36.55	(3,5);(5,8)	h_s	0.130
		7.45	(3,6);(6,8)	h_i	0.284

This new assignment provided a total cost given by (11) and equal to 100.70.

Again, the optimality conditions were not checked by pair 5 (where the derivative of the saturated path was greater than the derivative of the empty path) as Table 5 depicts. Therefore, the flow was recirculated between both paths and (1,2);(2,3);(3,6) turned into the saturated path meanwhile (1,3);(3,6) remained as the empty path. Calculating again the values for the fourth iteration (Table 6):

Table 6. Checking the optimality conditions. 4rth iteration

$k \in \mathbf{K}$	Origin-destination	p_{kh}	Path	Type	$c'_h(p_{kh})$
2	1-3	6.31	(1,2);(2,3)	h_s	0.160
		5.69	(1,3)	h_i	0.329

3	1-5	27.47	(1,2);(2,5)	h_s	0.151
		3.53	(1,3);(3,5)	h_i	0.402
4	1-8	26.01	(1,2);(2,5);(5,8)	h_s	0.207
		24.99	(1,4);(4,7);(7,8)	h_i	0.219
		0.00	(1,3);(3,6);(6,8)	h_v	0.613
5	1-6	3.00	(1,2);(2,3);(3,6)	h_s	0.271
		0.00	(1,3);(3,6)	h_v	0.440
7	2-5	26.37	(2,5)	h_s	0.103
		6.63	(2;3);(3,5)	h_i	0.185
8	2-8	15.12	(2,5);(5,8)	h_s	0.159
		0.88	(2,3);(3,6);(6,8)	h_i	0.396
10	3-8	36.55	(3,5);(5,8)	h_s	0.130
		7.45	(3,6);(6,8)	h_i	0.284

This demand routing provided a total cost equal to 100.28. In addition, after this iteration, every path now checked the optimality conditions, being optimal the solution provided, and ending the procedure.

Next Table 7 shows the flow in the links through the different iterations of the algorithm showing the cost evolution to the optimum value.

Table 7. Flow in links and network cost through the iterations

Link	d_e	***1st iteration***		***2nd iteration***		***3rd iteration***		***4th iteration***	
		x_e	$C_e(x_e)$	x_e	$C_e(x_e)$	x_e	$C_e(x_e)$	x_e	$C_e(x_e)$
(1,2)	1	105,17	10,26	105,17	10,26	105,79	10,29	108,79	10,43
(1,3)	2	37,83	12,30	12,84	7,17	12,22	6,99	9,22	6,07
(1,4)	1	36,00	6,00	60,99	7,81	60,99	7,81	60,99	7,81
(2,3)	1	16,20	4,02	16,20	4,02	16,82	4,10	19,82	4,45
(2,5)	2	94,97	19,49	94,97	19,49	94,97	19,49	94,97	19,49
(3,5)	1	46,71	6,83	46,71	6,83	46,71	6,83	46,71	6,83
(3,6)	1	45,32	6,73	20,33	4,51	20,33	4,51	20,33	4,51
(4,7)	2	69,00	16,61	93,99	19,39	93,99	19,39	93,99	19,39
(5,8)	1	77,68	8,81	77,68	8,81	77,68	8,81	77,68	8,81
(6,8)	1	33,32	5,77	8,33	2,89	8,33	2,89	8,33	2,89
(7,8)	1	67,00	8,19	91,99	9,59	91,99	9,59	91,99	9,59
		Total cost	105,02	***Total cost***	100,77	***Total cost***	100,70	***Total cost***	100,28
				Improvement	-4,25	***Improvement***	-0,069	***Improvement***	-0,423

5.2. Computational results

We constructed a trial library with 30 instances to carry out a detailed analysis of the algorithm, including convergence to the optimum, time consumption, and the analysis of the main algorithm parameters. This trial library can be accessed through the website http://io.us.es/RedTeltrials/ConcaveOptimize/. In the library, the first 26 trials correspond to sparse and medium density graphs while the four last problems correspond to fully connected dense graphs (density ratio equal to one). These two different set of trial graphs are separated by a dot line in Table 8. These two sets allow analysing the behaviour of the algorithm when increasing the connectivity among nodes in the graph. The dimension of the problems depending on the parameters arcs, origin-

destination pairs, and number of paths communicating such pairs varies from around 500 constraints and 200 variables (instances 6 or 13) to more than 2,000 constraints and 1,000 variables (instances 1, 2 8, 16, 18, 29 or 30). This variety provides a good basis for the analysis.

The considered parameters are detailed in Table 8. First, we included the number of nodes, *N*, arcs, *E*, and origin-destination pairs, *K*. We have, also, analysed the volume of demand in the network noted as $\sum \gamma_k$, and the level of alternative paths forced in the demand routing that we have called diversification and is given by $\sum \delta_k$ (a lower result from the sum requires a higher degree of alternative routing paths).

The main output values from our analysis are the cost, the algorithm computational time and the required number of iterations, and they are compared with respect to the main design parameter. The considered parameters were: (i) the number of nodes; (ii) the number of arcs; (iii) the density of the graph that is calculated as the ratio between the arcs and the maximum number of arcs considering possible connections among all the nodes. So, the ratio is calculated as: $2A/\big(N \cdot (N-1)\big)$; (iv) the number of origin-destination pairs; (v) the total demand volume in the network; and (vi) the diversification parameter.

Table 8 provides the results obtained for the 30 instance library. Data related to the cost and the required computational time and number of iterations to obtain the optimum are provided together with data related to preparation of the problem. That is, the required time for finding the paths in the graph connecting every origin-destination pair, as well as the number of paths that was found.

Table 8. Summary of the trial library result

No. Graph	Nodes (N)	Arcs (E)	Density	Origin-destination pairs (K)	$\sum \gamma_k$	$\sum \delta_k$	Pre-computing[1]		Number of iterations	Algorithm time (seconds)	Cost
							Time for finding paths	Number of paths			
1	29	76	0.19	325	7500	243.60	0.093	8284	51	1415	2330.99
2	16	40	0.33	101	2470	74.52	0.015	653	19	52	867.90
3	17	44	0.32	134	2748	81.30	0.062	2832	20	128	929.25
4	28	64	0.17	349	6040	175.14	0.078	7221	23	1544	1976.86
5	24	65	0.24	235	5699	167.73	0.063	6290	42	756	1815.33
6	11	22	0.40	48	1184	37.31	0.063	171	8	11	363.51
7	21	49	0.23	165	3114	97.17	0.016	866	15	87	1163.39
8	26	65	0.20	284	5366	169.77	0.015	14950	34	2705	1846.75
9	10	14	0.31	37	671	22.44	0.001	91	2	1	274.93
10	19	47	0.27	162	4143	122.84	0.001	5442	19	215	1096.81
11	26	64	0.20	287	6984	217.22	0.079	15327	39	1312	2012.19
12	23	56	0.22	206	3915	125.53	0.001	3519	12	171	1236.76
13	18	31	0.20	137	2381	76.76	0.156	921	4	14	832.22
14	10	18	0.40	38	661	22.21	0.046	113	3	13	292.31
15	13	27	0.35	62	1440	47.23	0.047	227	5	11	559.51
16	30	73	0.17	346	6988	206.28	0.001	16882	28	2498	2284.20
17	14	28	0.31	66	1315	38.93	0.016	219	4	24	478.74
18	24	66	0.24	252	6353	179.64	0.001	18071	38	2767	1831.47
19	20	52	0.27	162	3597	100.10	0.001	2094	23	144	1148.47
20	21	56	0.27	173	3690	104.53	0.001	3050	22	160	1203.23
21	28	68	0.18	322	7438	244.62	0.094	6412	31	563	2337.84
22	15	37	0.35	94	1625	47.08	0.063	933	18	51	605.38
23	21	53	0.25	174	3890	128.89	0.063	3870	46	489	1270.06
24	17	37	0.27	111	2573	84.58	0.001	648	13	27	891.24
25	12	26	0.39	53	1069	31.22	0.001	181	7	28	448.28
26	15	31	0.30	78	1418	44.49	0.015	333	9	16	528.53
27	10	45	1.00	45	1099	44.01	0.005	1013	14	23	265.39
28	12	66	1.00	66	1695	64,01	0.005	4083	21	49	387.04
29	15	105	1.00	105	2638	103,01	0.005	32752	49	2890	618.72
30	16	120	1.00	120	3102	109	0.015	65519	51	3801	745.69

[1] See appendix 4

In next sections, we make a graphical analysis of the results in Table 8 (done in figures 2 to 6) in order to appreciate tendencies, dominances, etc. Firstly, we provide an analysis for the set of sparse and medium density graphs, and secondly an analysis for the four dense graphs is provided. The reason for separating them in the analysis is for avoiding possible interferences between different conceptual graphs with a very different density of arcs.

5.3. Analysis of the results for sparse and medium density graphs

First, we analyse tendencies and dependencies for sparse and medium density graphs. Starting with the objective function dependencies with respect to the main parameters, we can say that the cost of the problem maintains an approximately linear tendency with respect to the number of nodes, the number of origin-destination pairs, the total demand volume in the network, and the diversification parameter. On the other hand, the relation with respect to the number of arcs in the graph slightly shows a higher than

linear relation that shoots up the distribution cost in the network, being the parameter with a more significant repercussion (see Figure 2).

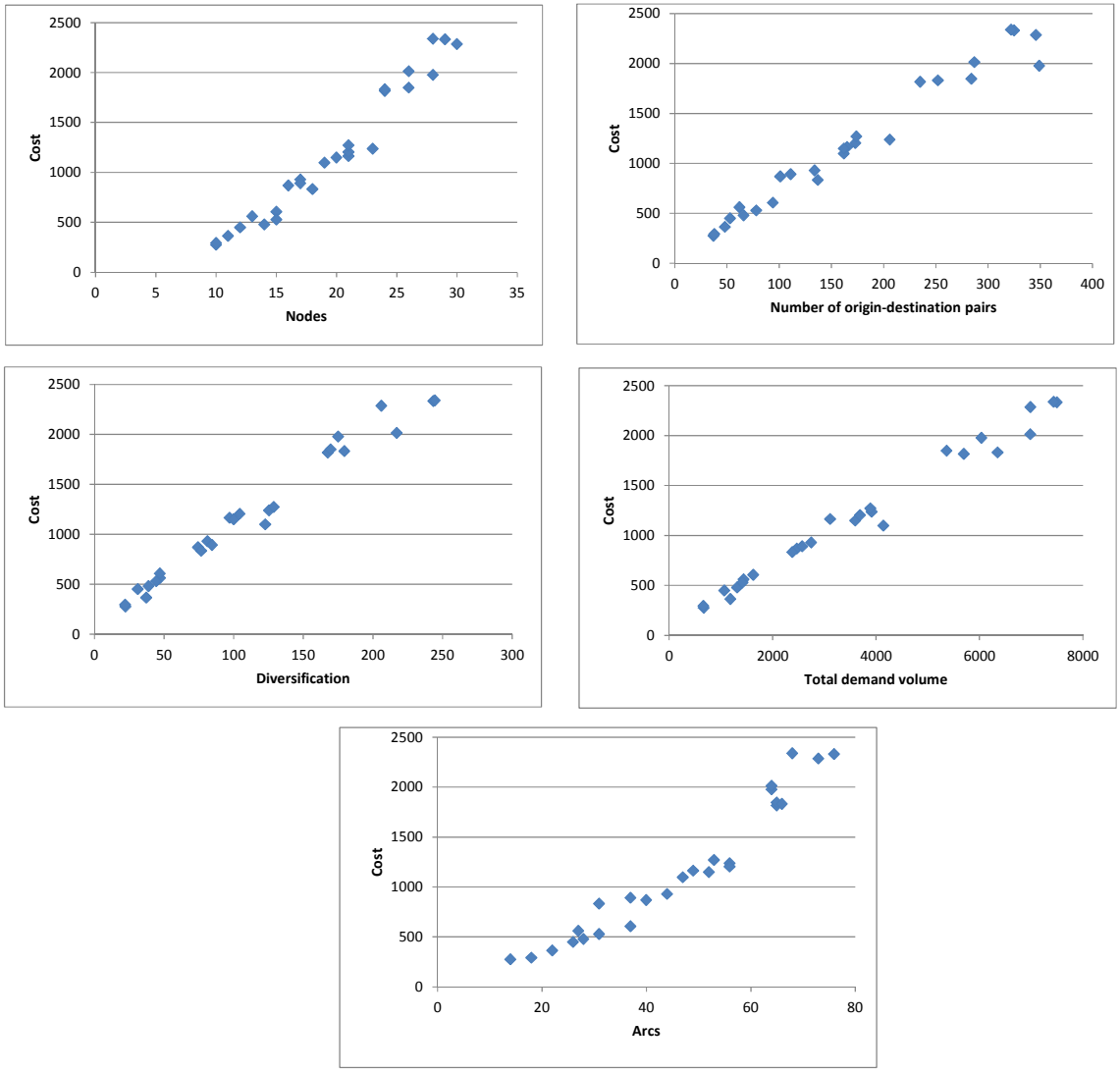

Figure 2. Relation between cost and main design parameters for instances 1 to 26

The time required by the algorithm to get the optimal solution was mainly dependent on the number of arcs and nodes (Figure 3). The time consumption increased at a greater rate than the size of the problem as expected for a NP-Hard problem such as the analysed.

Instances 8, 16 and 18 showed the higher times to reach the optimum of the problem because they represent the major dimension problems. So, worse results were obtained for graphs which combined a large number of nodes, arcs, origin-destination pairs, and demand.

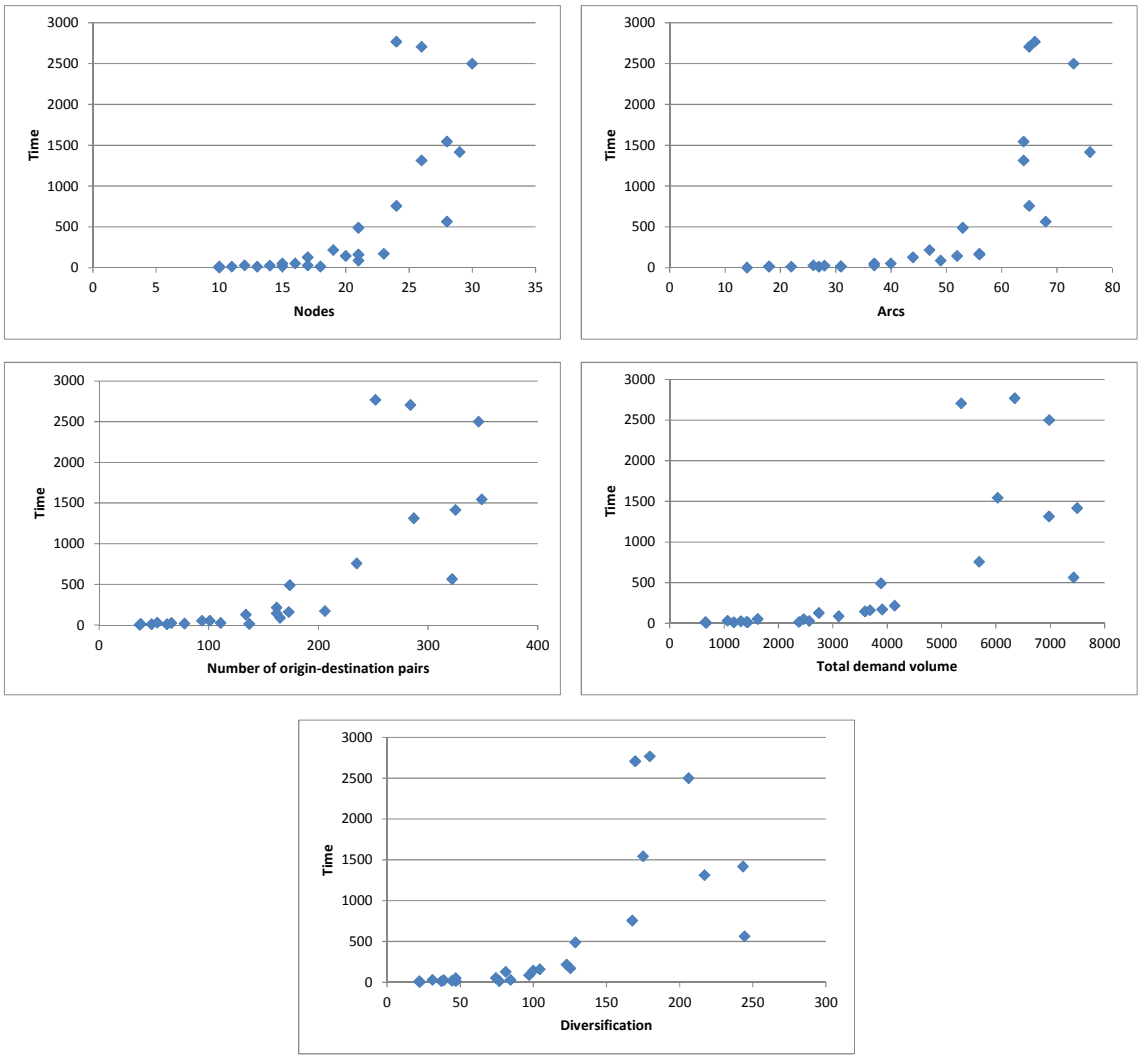

Figure 3. Relation between computational time and main design parameters for instances 1 to 26

The number of iterations of the algorithm is strongly linked to the computational time of the algorithm, and it is mainly affected by the density of the graph, as well as the number of arcs and nodes too. So, the problems that take a greater time in the optimum convergence were the larger problems and also those problems with a higher number of pre-computed paths, which required a greater number of iterations (Table 8).

Following this later aspect, Figure 4 shows the relation between the number of paths that were pre-computed for each graph, and the computational time and the number of iterations of the algorithm. The figure shows that the increase in terms of computational time and number of iterations follows an increasing tendency with a rate lower than when it is analysed with respect to other design parameters (see how the increasing rate is higher when the computational time is analysed with respect to other design parameters in Figure 3). This fact contributes to validate and enhance the real potential applicability of the approach.

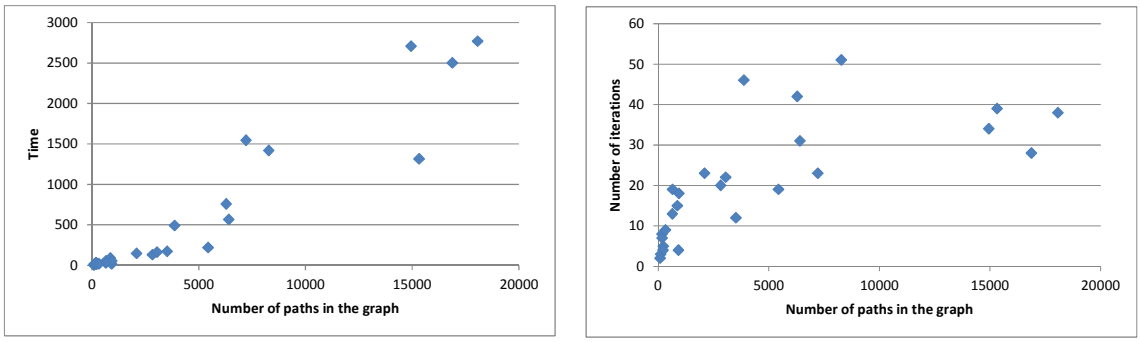

Figure 4. Time and number of iterations evolution with respect to the number of paths in the graph for instances 1 to 26

Finally, Figure 5 provides an interesting analysis related to the algorithm convergence. The optimising curve depicts a monotonic decrease of the total cost value that the algorithm achieved, as shown in the figure. We have selected instances 1, 4, 8 and 18 as representative examples to appreciate the algorithm convergence. It is interesting to note how this phenomenon that is argued in appendix 3 is empirically checked when solving the instances of the trial library.

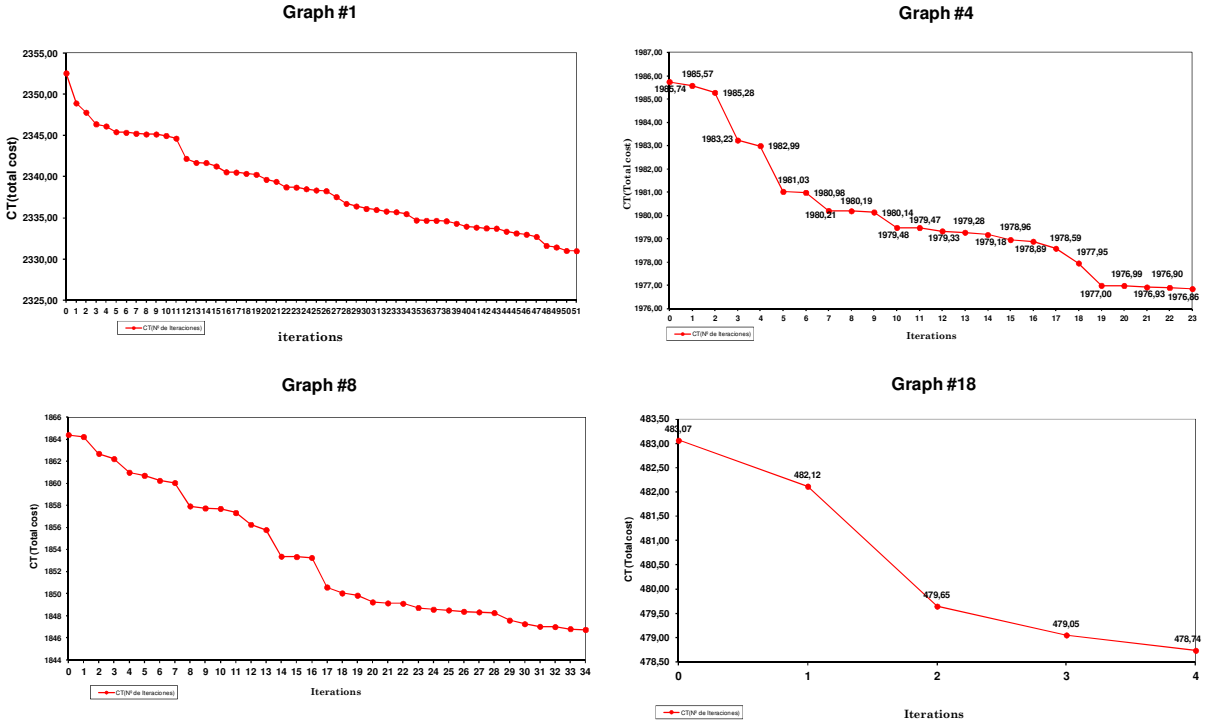

Figure 5. Convergence to the optimum curves for graph instances 1, 4, 8 and 18

5.4. Analysis of the results for dense graphs

Finally, we analyse the main parameter dependencies related to dense graphs. Starting with the objective function, the cost analysis in dense graph follows a very similar structure to the analysis in sparse or medium density graphs, depicting a clearly linear tendency. However, the linear dependency is stronger with respect to the nodes than the arcs (see left side of Figure 6). Furthermore, the dependency with respect to the diversification parameter ($\sum \delta_k$) than with respect to the number of origin-destination

pairs (see right side of Figure 6).

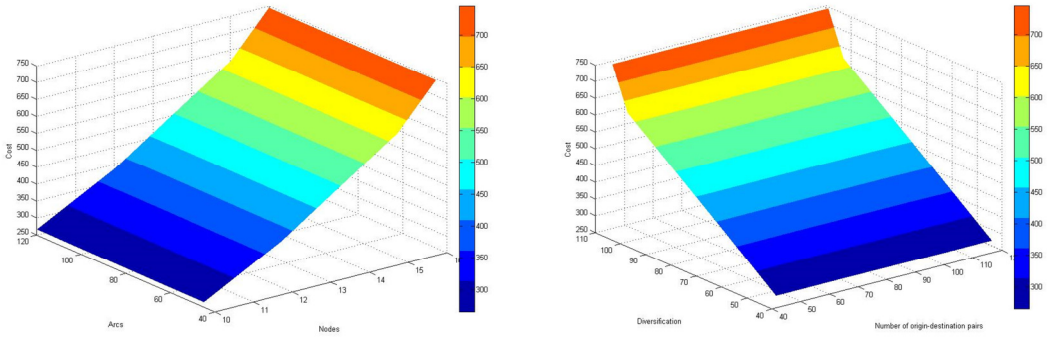

Figure 6. Relation between cost and main design parameters for instances 27 to 30

The analysis of the computational time against the main parameters is also interesting and relevant. Here again major dependencies appear for nodes and diversification. Also, a higher than linear tendency is appreciated with respect to number of nodes (left side of Figure 7), and especially with respect to the diversification parameter (right side of Figure 7).

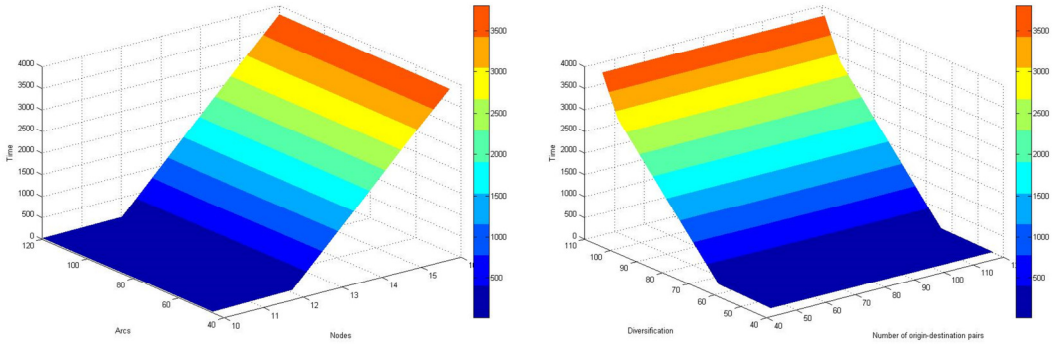

Figure 7. Relation between time and main design parameters for instances 27 to 30

Finally, similar comments to those provided for sparse and medium density graphs attending to the computational time and number of iterations with respect to the number of paths is also valid for dense graphs. Here again, the increase both in computational time and number of iterations is not so high with respect to the number of paths in the graphs as with respect to other design parameters as can be viewed in Figure 8.

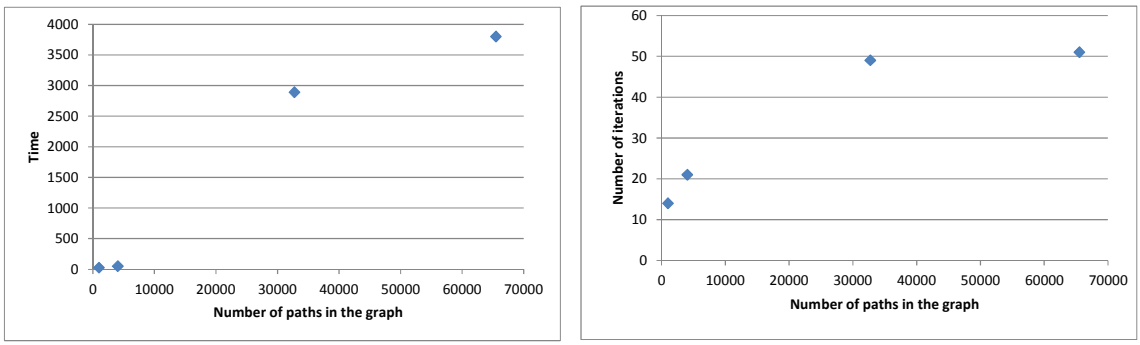

Figure 8. Time and number of iterations evolution with respect to the number of paths in the graph for instances 27to 30

6. Conclusions

In this paper we have proposed a novel optimal algorithm based on the Kuhn-Tucker optimality conditions to determine the optimum of the demand routing problem in multicommodity flow networks with diversification constraints and concave costs. The aforementioned problem is considered NP-Hard because of its concave cost function. In addition, it is an ambitious problem that takes most of the elements from traditional concave cost transportation problems literature such as the objective function and the multicommodity constraints, but also includes new elements as the diversification constraints that have not been considered in concave cost paper literature, depicting a more complex problem that those tackled in most of the concave cost transportation papers. Such diversification constraints are required many times to model situations that require diversifying elements in alternative routes of a distribution network, such as freight in transportation networks or cells in telecommunication networks, modelling a fact that appear in a number of real practice situations.

We have presented an iterative algorithm based on the Kuhn-Tucker optimality conditions of the problem. Feasible solutions composed of an indicator path, a number of $\left(\lceil 1/\delta_k \rceil - 1\right)$ saturated paths and a set of empty paths are analysed for checking the Kuhn-Tucker necessary conditions each iteration, since the sufficient condition is always checked for every intermediate feasible solution defined as the previous one and independently from its optimality. The algorithm enables the optimum to be reached after a number of iterations. The method is a constructive step-by-step method because after each iteration the total cost in the network is reduced in a monotonic sequence. Therefore, given a MFDCC problem, a sufficiently high number of iterations of the algorithm will allow reaching the optimum of the problem. It is clear that as the MFDCC problem is NP-Hard, the number of iterations of the algorithm increases in a non-polinomial way when the size of the problem increases. For very great size problems this exact algorithm could not be efficient.

The algorithm showed a major dependency with respect to the number of nodes and arcs of the problem as well as the level of diversification, which define the size of each instance. Also the density of the graph depicted a clear influence for problems of the same size increasing significantly the computational time. The number of origin-destination pairs, as well as the number of alternative paths for connecting each origin-destination pair revealed a not so relevant dependency levelling off its tendency. Also, it is relevant to note that tendencies were appreciated in a similar way both for sparse, medium density and dense graphs.

The optimum was obtained for all the instances studied, and the time consumption was feasible when analysed from a non-real-time perspective, which is normal in network design problems. Although in real time situations, the algorithm may not be suitable depending on the time response required, alternatives such as stop criteria after a number of iterations or time consumption could be considered because the method guarantees a monotonic approach to the optimum.

Acknowledgements

The authors acknowledge an anonymous referee for his/her helpful comments on earlier versions of this manuscript that have significantly contributed to enrich the quality of the paper in its final form.

Appendix 1

Appendix 1 states the Kuhn-Tucker optimality necessary conditions for every optimisation problem, and its application to the MFDCC problem.

Consider the next generic optimization problem:

$$Min\ f(x) \tag{13}$$

s.t.:

$$g_k(x) \leq 0\ ,\forall k \tag{14}$$

$$h_i(x) = 0\ ,\forall i \tag{15}$$

$$x \in \mathcal{R}^n$$

The Kuhn-Tucker conditions are given by:

$$\nabla f(x) + \sum_i u_k \nabla g_k(x) + \sum_i \mu_i \nabla h_i(x) = 0 \tag{16}$$

$$g_k(x) \leq 0\ ,\forall k$$

$$h_i(x) = 0\ ,\forall i \tag{17}$$

$$u_k \cdot g_k(x) = 0\ ,\forall k \tag{18}$$

$$u_k \geq 0\ \ \forall k \ \text{ and } \ \mu_i\ free \tag{19}$$

The Kuhn-Tucker conditions for the MFDCC problem are given by:

$$\frac{\partial C(p)}{\partial p_{kh}} - \mu_k = u_{kh} - v_{kh} \tag{20}$$

$$(p_{kh} - \delta_k \cdot \gamma_k) \cdot v_{kh} = 0 \tag{21}$$

$$p_{kh} \cdot u_{kh} = 0 \tag{22}$$

$$u_{kh}, v_{kh} \geq 0 \text{ and } \mu_k\ free \tag{23}$$

A solution verifying the necessary condition is only optimum if it verifies also the sufficiency condition in problems with non-convex objective functions.

Appendix 2

Appendix 2 deals with the Kuhn-Tucker optimality sufficient conditions. In addition, here is proven that every solution for the communication of K commodities, each of them with a demand γ_k and forced to a diversification given by δ_k, that is composed by a set of $\left(\lceil 1/\delta_k \rceil - 1\right)$ saturated paths, an indicator path and being the rest void paths verifies the Kuhn-Tucker sufficient conditions. Thus, every solution associated to each iteration of the algorithm described in section 4 will verify the Kuhn-Tucker sufficient condition.

Let consider the generic problem given by equations (13-15). The sufficiency condition for such generic problem is given by (24):

$$y^T \nabla_{xx} L(x,u,\mu) y \geq 0 \;\; \forall y \in V(x) \tag{24}$$

Where:

$$\nabla_x L(x,u,\mu) = \nabla f(x) + \sum_i u_k \nabla g_k(x) + \sum_i \mu_i \nabla h_i(x)$$

and:

$$V(x) = \{y \in \mathcal{R}^n : \nabla h_i(x)^T \cdot y = 0\,, \forall i\,;\; \nabla g_k(x)^T \cdot y = 0\,, \forall k \in A(x)\}$$

where $A(x)$ is the set of active constraints (that is, verifies constraint $g_k(x) \leq 0$ with zero slack).

The sufficiency condition for the MFDCC problem is given by (25):

$$y^T \left[\frac{\partial^2 C(p)}{\partial p_{kh}^2}\right] y \geq 0, \forall y \in V(x) \tag{25}$$

And this expression must be satisfied in set $V(x)$ that is defined as:

$$V(x) = \left\{ y \in \mathcal{R}^n : \sum_{h \in P(k)} y_{kh} = 0, \forall k \in K \text{ and } y_{kh} = 0 \text{ for those paths } h \in P(k) \text{ so that } p_{kh} = \delta_k \text{ (saturated paths) or so that } p_{kh} = 0 \text{ (empty paths)} \right\}$$

which is verified by every feasible solution associated to the corresponding iteration of the algorithm described in section 4. In effect, equation (25) is verified in every saturated and empty path because $y_{kh} = 0$ for such paths, and the indicator path verifies also $y_{kh} = 0$ because $\sum_{h \in P(k)} y_{kh} = 0, \forall k \in K$ and there is only one indicator path in each iteration, so being the other paths (saturated and empty) equal to zero, the indicator path must be equal to zero too.

Therefore, any flow pattern verifying equations (20 to 23) verifies the sufficient condition too, and therefore the necessary conditions becomes the global optimality conditions for the specific structure of the analysed problem.

Appendix 3

Appendix 3 proves that the algorithm described in section 4, guarantees the reduction of the cost of the objective function after each step following a monotonic sequence to optimality.

Suppose that after iterating, the Kuhn-Tucker conditions cannot be satisfied for every origin-destination pairs. Therefore, the origin-destination pair k^* is given by (9). For such pair three options can appear:

a) $c'_{h_s} > c'_{h_i}$ the derivative is higher in a saturated path than in the indicator path

b) $c'_{h_i} > c'_{h_v}$ the derivative is higher in the indicator path than in a void path
c) Both situations occur but one of the expressions: $\left(c'_{h_s} - c'_{h_i}\right)$ and $\left(c'_{h_i} - c'_{h_v}\right)$ is higher than the other, so we select the corresponding case.

So we need to analyse the recirculation of flow for the two cases given below as option A and option B.

Option A:

It corresponds to recirculate the flow from the saturated path to the indicator path for cases a) or c) when $\left(c'_{h_s} - c'_{h_i}\right) > \left(c'_{h_i} - c'_{h_v}\right)$.

Total cost is given for current situation (26) and, after recirculating, new situation (27):

$$C^c(P) = \sum_{e \notin \{h_s; h_i\}} c_e(x_e) + \sum_{e \in h_s} c_e(x_e) + \sum_{e \in h_i} c_e(x_e) \tag{26}$$

$$C^n(P) = \sum_{e \notin \{h_s; h_i\}} c_e(x_e) + \sum_{e \in h_s} c_e(x_e - \Delta x) + \sum_{e \in h_i} c_e(x_e + \Delta x) \tag{27}$$

So, there is an unchanging part given by the flown on the links for every link not belonging to the saturated or indicator path, and a changing part given by the links in the saturated and indicator paths.

Calculating the variation of cost between new and current situations, equation (28) is reached.

$$C^n(P) - C^c(P) = \left[\sum_{e \in h_s} c_e(x_e) + \sum_{e \in h_s} c_e(x_e - \Delta x)\right] + \left[\sum_{e \in h_i} c_e(x_e) + \sum_{e \in h_i} c_e(x_e + \Delta x)\right] \tag{28}$$

From the checking of the Kuhn-Tucker conditions we know that $c'_{h_s} > c'_{h_i}$ (that was the reason for non-optimality). Therefore, the decrease in the cost experimented when reducing flow in the saturated path corresponding to the evaluation of the first term of the sum is higher than the increase in the cost experimented when increasing flow in the indicator path corresponding to the evaluation of the second term of the sum (this argument is based on $c'_{h_s} > c'_{h_i}$). And thus, $C^n(P) - C^c(P) < 0$ producing a decrease of the total cost after recirculating the flow consequently.

Option B

It corresponds to recirculate the flow from the indicator path to the void path for cases b) or c) when $\left(c'_{h_s} - c'_{h_i}\right) < \left(c'_{h_i} - c'_{h_v}\right)$.

Total cost is given for current situation (29) and, after recirculating, new situation (30):

$$C^{current}(P) = \sum_{e \notin \{h_i; h_v\}} c_e(x_e) + \sum_{e \in h_i} c_e(x_e) + \sum_{e \in h_v} c_e(x_e) \tag{29}$$

$$C^{new}(P) = \sum_{e \notin \{h_i; h_v\}} c_e(x_e) + \sum_{e \in h_i} c_e(x_e - \Delta x) + \sum_{e \in h_v} c_e(x_e + \Delta x) \quad (30)$$

So, there is an unchanging part given by the flown on the links for every link not belonging to the indicator or void path, and a changing part given by the links in the indicator and void paths.

Calculating the variation of cost between new and current situations, equation (31) is reached.

$$C^n(P) - C^c(P) = \left[\sum_{e \in h_i} c_e(x_e) + \sum_{e \in h_i} c_e(x_e - \Delta x)\right] + \left[\sum_{e \in h_v} c_e(x_e) + \sum_{e \in h_v} c_e(x_e + \Delta x)\right] \quad (31)$$

From the checking of the Kuhn-Tucker conditions we know that $c'_{h_i} > c'_{h_v}$ (that was the reason for non-optimality). Therefore, the decrease in the cost experimented when reducing flow in the indicator path corresponding to the evaluation of the first term of the sum is higher than the increase in the cost experimented when increasing flow in the void path corresponding to the evaluation of the second term of the sum (this argument is based on $c'_{h_i} > c'_{h_v}$). And thus, $C^n(P) - C^c(P) < 0$ producing a decrease of the total cost after recirculating the flow consequently.

These arguments prove that after each iteration the total cost in the network is reduced. Therefore, given a MFDCC problem, a sufficiently high number of iterations of the algorithm allow reaching the optimum of the problem. It is clear that as this problem is NP-Hard, the number of iterations of the algorithm increases in a non-polinomial way when the size of the problem increases.

Appendix 4

This appendix includes the procedure followed to find all the paths in a graph. There are several alternatives to find all the paths between a pair of nodes in a graph. Most of them follow Depth-First Search (DFS) or Breadth-First-Search (BFS) structures. We followed a procedure based in the DFS algorithm. This is a well-known algorithm that can be described as next:

```
Input: A graph G and a vertex v of G
Output: A labelling of the edges in the connected component of v as
discovery edges and back edges

procedure DFS(G,v):
    label v as explored
    for all edges e in G.adjacentEdges(v) do
         if edge e is unexplored then
            w ← G.adjacentVertex(v,e)
             if vertex w is unexplored then
                 label e as a discovery edge
                 recursively call DFS(G,w)
```

```
else
    label e as a back edge
```

References

1. Altiparmak F, Dengiz B, Smith AE., 2003. Optimal design of reliable computer networks: A comparison of metaheuristics. Journal of Heuristics 9, 471-487.

2. Altiparmak F, Karaoglan I., 2008. An adaptive tabu-simulated annealing for concave cost transportation problems. Journal of the Operational Research Society 59, 331-341.

3. Avriel, M., 2003. Nonlinear Programming: Analysis and Methods. Dover Publishing.

4. Burkard RE, Dollani H, Thach PT., 2001. Linear approximations in a dynamic programming approach for the uncapacitated single source minimum concave cost network flow problem in acyclic networks. Journal of Global Optimization 19, 121-139.

5. Dengiz B, Altiparmak F, Smith AE., 1997. Local search genetic algorithm for optimal design of reliable networks. IEEE Transactions on Evolutionary Computation 1; 179–188.

6. Cortes, P., Muñuzuri, J., Onieva, L., Larrañeta, J., Vozmediano, J.M., Alarcon, J.C., 2006. Andalucía assesses the investment needed to deploy a fiber-optic network. Interfaces 36 (2), 105-117

7. Florian M, Klein M., 1971. Deterministic production planning with concave cost and capacity constraints. Journal of Management Science 18, 12–20.

8. Hanson MA, Mond B., 1987. Necessary and sufficient conditions in constrained optimization. Mathematical programming 37; 51-58.

9. Hillier, F.S., Lieberman, G.J., 1974. Operations Research. Holden-Day Inc. (San Francisco, California, US)

10. Hitchcock, F.L., 1941. The distribution of a product from several sources to numerous locations. Journal of Mathematical Physics 20, 224–230.

11. Kapsalis A, Rayward-Smith VJ, Smith GD., 1993. Solving the graphical Steiner tree problem using genetic algorithms. Journal of the Operational Research Society 44, 397–406.

12. Kengpol, A., Meethom, W., Tuominen, M., 2012. The development of a decision support system in multimodal transportation routing within Greater Mekong sub-region countries. International Journal of Production Economics 140(2), 691–701.

13. Kleinberg, J. 1996. Approximation Algorithms for Disjoint Paths Problems. Ph.D Thesis, Department of EECS, MIT.

14. Larsson T, Migdalas A, Ronnqvist M., 1994. A lagrangian heuristic for the capacitated concave minimum cost network flow problem. European Journal of Operational Research 78, 116–129.

15. LeBlanc, L.J., 1976. Global solutions for a nonconvex, nonconcave rail network model. Management Science 23, 131-139.

16. Nocedal, J., Wright, S. J., 2006. Numerical Optimization (Second Edition) by Numerical Optimization, Springer.

17. Thach PT., 1992. A decomposition method using a pricing mechanism for minimum concave cost flow problems with a hierarchical structure. Journal of Mathematical Programming 53, 339–359.

18. Tsao, Y-C., Sheen, G-J., 2012. A multi-item supply chain with credit periods and weight freight cost discounts. International Journal of Production Economics 135 (1), 106-115.

19. Ubeda, S., Arcelus, F.J., Faulin, J., 2011. Green logistics at Eroski: A case study. International Journal of Production Economics 131(1), 44-51.

20. Yan S, Luo SC., 1999. Probabilistic local search algorithms for concave cost transportation network problems. European Journal of Operational Research 117, 511–521.

21. Zangwill WI., 1968. Minimum concave cost flows in certain networks. Management Science 14, 429–450.

22. Zhou G, Gen M., 2003. A genetic algorithm approach on tree like telecommunication network design problem. Journal of the Operational Research Society 54, 248-254.