Combining Tabu Search and Genetic Algorithms to Solve the Capacitated Multicommodity Network Flow Problem

Carolina LAGOS[1], Broderick CRAWFORD[1], Enrique CABRERA[2], Ricardo SOTO[1], Jose-Miguel RUBIO[1], Fernando PAREDES[3]

[1] Pontificia Universidad Católica de Valparaíso, 2340025 Valparaíso, CHILE. carolina.lagos@mail.pucv.cl

[2] CIMFAV, Universidad de Valparaíso, Valparaíso, CHILE.

[3] Escuela de Ingeniería Industrial, Universidad Diego Portales, 8370179 Santiago, CHILE.

Abstract: Network design has been an important issue in logistics during the last century. This is due to the significant impact that an efficient distribution network design can have over both costs and service level. In this article, we present a heuristic solution approach for the well-known capacitated multicommodity network flow problem. The heuristic approach combines two well-known algorithms namely Tabu Search and Genetic Algorithms. While the main algorithm is Tabu Search, the Genetic Algorithm is used to select the best option among the neighbours of the current solution. To be able to do that some well-known evolutionary operators such as cross-over and mutation are made use of. This hybrid approach obtains important improvements when compared to the ones presented previously in the literature.

Keywords: Multicommodity network flow problem, network design, probabilistic neighbour selection criterion, tabu search, genetic algorithms.

1 Introduction

Network design has been an important issue in logistics during the last century. This is due to the impact that an efficient distribution network design can have over both cost and service level. One specific problem arising in network design is the problem of moving different commodities over a network from different sources to several destinations. This problem is called Multicommodity Network Flow (MNF) problem and is mostly encountered in telecommunications and transportation network planning. Several models have been proposed in literature to represent different real life situations where this kind of problems can be found (see [19]).

In this article, we consider a transportation problem of multiple products (commodities) through a network with capacity constraint, fixed demands and both fixed and variables costs in edges construction and flows, respectively. The capacity constraint considered in this article is an important element for the complexity of the capacitated MNF (CMNF) problem. In fact, several articles in literature confirm that this problem is NP-Hard [19, 6, 7, 20]. Thus, when this problem becomes bigger in terms of the number of decision variables, mathematical programming techniques are not able to find the optimal solution within reasonable time. Therefore other techniques such as heuristic procedures must be considered to find an acceptable approximation of the actual optimal solution within some time limits. In this article we consider a hybrid algorithm of Tabu Search (TS) and Genetic Algorithms (GA) to approximately solve this problem. We chose these two strategies based on the good performance that they have shown when applied on other complex combinatorial optimization problems such as routing, facility location, set covering, among others.

The hybrid algorithm is mainly based on TS. We use the GA in order to increase the algorithm diversification degree. Increasing the algorithm diversification degree allows the algorithm to increase its exploration level over the set of feasible solutions, raising the likelihood of finding good quality neighbourhoods. A large number of techniques have been applied to this problem previously in literature; in [7] the authors present a multilevel cooperative TS algorithm for the CMNF problem, using different levels of cooperation, as well as a set of new cooperation operators. In [17] the authors propose a Lagrangean heuristic within a branch-and-bound framework. The Lagrangean heuristic uses a Lagrangean

relaxation to obtain easily handled sub-problems and solves the Lagrangean dual by sub-gradient optimization. The Lagrangean heuristic is then embedded into a branch-and-bound scheme that yields further primal improvements. In [11] authors propose a path re-linking procedure.

Cycle-based neighbourhoods are used to move along paths between elite solutions and to generate the elite candidate set, like in local search procedures. In [2] the authors present an efficient procedure using a scatter search framework, which is modified in [3] through an embedded greedy random adaptive search procedure (GRASP). More recently, [26] have presented a hybrid simulated annealing and column generation approach to solve this problem. In their paper, simulated annealing manages open and closed arcs while column generation is used to solve (exactly) the resulting sub-problem.

In [24] the authors present a three-phase strategy that combines TS with path re-linking and exact techniques. While heuristics are used to explore the solution set, the exact algorithm is used to intensify the search in a specific part of the solution set.

In this article we present a TS-based algorithm which selects at each iteration a candidate from a list through a probabilistic criterion based on evolutionary algorithms. TS algorithms have been applied on a wide range of different combinatorial optimization problems [1, 4, 5, 15, 22]. Different implementations of TS have been also used to solve the CMNF problem (see e.g. [7]). Regarding the hybridization proposed in this work, the literature reports many different hybrid approaches of TS and evolutionary algorithms [8, 9, 13, 14, 18, 21, 23, 27, 28, 29]. In the majority of these works, authors rely on the exploration capabilities of GA while using TS to refine and exploit the neighbourhood of the best solution obtained by the GA. Unlike these approaches, in this article we make use of the evolutionary algorithm only to choose the next solution among the set of neighbours candidates of the current solution in the TS algorithm. Moreover, to the best of our knowledge, no hybrid algorithm of TS and GA to solve the CMNF problem, such as the one presented in this article, has been reported in the literature.

2. Mathematical Formulation

In this paper we consider the mathematical model of the CMNF problem proposed by [10]. This model is as follows:

Let $G = (N, A)$ be a network where N corresponds to the set of nodes and A is the set of directed arcs. Without loss of generality, we assume that all $(i,j) \in A$ are directed arcs with i, j $\in N$. We denote f_{ij} and u_{ij} the fixed cost and the capacity of arc (i,j), respectively. Let P denote the set of commodities. For each commodity $p \in P$, one must move w^p units of flow from its (unique) origin $O(p)$ to its (unique) destination $S(p)$. Thus, c_{ij}^p denotes the variable cost of moving one unit flow of commodity p from node i to node j or, equivalently, the cost of moving one unit flow of commodity p using arc (i,j).

Two sets of decision variables are defined. Design variables $y_{ij} \in \{0,1\}$ with $(i,j) \in A$, will be equal to 1 if arc (i,j) is selected in the final network design and 0 otherwise; distribution variables $x_{ij}^p \in \mathbb{R}_0^+$ indicates the amount of flow of commodity $p \in P$ on arc (i,j). The arc-based formulation of the capacitated multicommodity network flow (CMNF) problem is then:

$$\min_{x\geq 0, y\in\{0,1\}} z(x,y) = \sum_{(i,j)\in A} f_{ij} y_{ij} + \sum_{p\in P}\sum_{(i,j)\in A} c_{ij}^p x_{ij}^p \quad (1)$$

s.t.

$$\sum_{j\in N^+(i)} x_{ij}^p - \sum_{j\in N^-(i)} x_{ij}^p = \begin{cases} w^p, if\ i = o(p) \\ -w^p, if\ i = o(p), \\ 0\ otherwise \end{cases} \forall i \in N, \forall p \in P \quad (2)$$

$$\sum_{p\in P} x_{ij}^p \leq u_{ij} y_{ij} \ \forall (i,j) \in A \quad (3)$$

$$\sum_{p\in P} x_{ij}^p \geq 0 \ \forall (i,j) \in A \quad (4)$$

Where $N^+(i)$ and $N^-(i)$ represent the set of successor and predecessor nodes of node i, respectively. The objective function (1) accounts for the total system cost computed as the sum of the fixed costs of the arcs included in the design plus the cost of routing the product demand on the resulting network. Constraint (2) represents the network flow conservation relations. The linking constraint (3) states that the total flow (of all commodities) on an open arc (y_{ij}) cannot

exceed its capacity, while it must be 0 if the arc is closed ($y_{ij} = 0$).

3. Solution Approach

This section presents our proposed technique to solve the CMNF problem. The next sub-section presents an overview of the implemented TS algorithm and shows some tests for adjusting the input parameters of the algorithm. Finally, the GA-based neighbour selection technique is presented.

3.1 Tabu search-based algorithm

The TS is essentially a local search algorithm, i.e. it needs to "exchange information" with its neighbours. To do that, firstly the neighbourhood must be defined to determine whether a solution is neighbour of another or not. In this paper a neighbourhood movement corresponds to a change in a set of edges within the same commodity. Therefore, the neighbourhood of a solution s will be defined as the set of solutions $NS = \{s'_1, \ldots, s'_{ns}\}$ such that solutions in NS differ from s in the set of edges belonging to only one commodity $p \in P$. ns corresponds to the neighbourhood size and it is a parameter of our TS-based heuristic, which is fixed throughout the execution of the algorithm.

One distinctive feature of TS is that it can make use of both short- and long-term memory. The main mechanism for implementing the short-term memory function is the tabu list. This list contains all the neighbourhood moves that are prohibited at some iteration during the algorithm execution. Moreover, the main mechanism for implementing the long-term memory function is the diversification criterion. Usually, diversification criterion is implemented using a simple memory structure which once reached a pre-established threshold (number of iterations without improvements) generates a new random solution to re-start the algorithm. This allows the algorithm to get out from neighbourhoods where the local optimum (or another solution close to it in terms of objective function value) has been already found.

Making use of the short- and long-term memories in our TS algorithm implies we need to set its parameters. Particularly we need to fix the length of the tabu list as well as the threshold for the diversification criterion.

The length of the tabu list is a parameter and corresponds to the number of iterations during which a specific neighbourhood movement is not allowed. Many authors claim that the tabu list size should be fixed in 7, but there is not a logical explanation to do that. More recently, authors take values based on either the size of the problem or using a dynamic short-term memory size. Either way, the length of Tabu list is an important parameter whose influence must be analysed. Our experiments showed that with a constant short term memory size of 7 we obtained the best results for most of the instances, which is consistent with [12, 25], among others. Figure 1 depicts the results for different instances of the problem using different values for the length of the Tabu list.

Analysed instances vary in the number of commodities to consider: 10, 50 and 100. We perform a total of 30 runs of the algorithm for

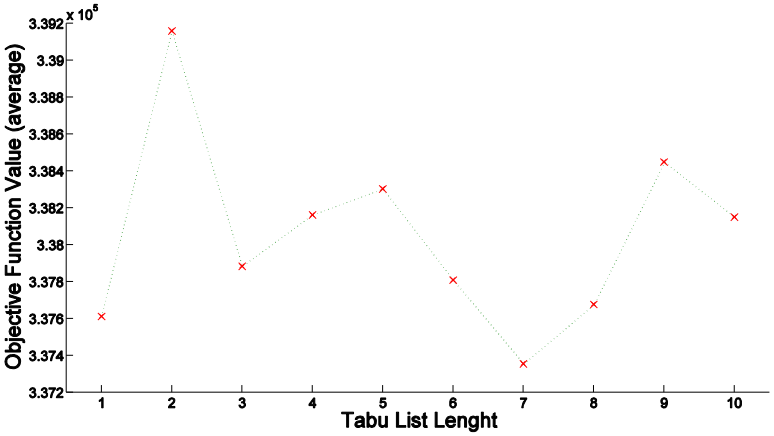

Figure 1. Objective function values (in average) obtained when different tabu list sizes (between 1 and 10) are applied over instances with 10 commodities.

each class of instances. Number seven was consistently one of the best values for all of the analysed instances. Despite it was not the best one for some of the instances, we decided to use this value as it makes the algorithm performs faster and it requires less objective function evaluations. Figure 1 shows that the best average value is obtained when the tabu list size is seven. Other values have been used, but they had a poor performance. Figure 2 shows the best average value obtained when the size of tabu list is 45. However, we can see how the objective function value obtained by number seven still is quite competitive. In terms of CPU time, the algorithm performance using number seven is about five times faster than when 45 is considered as the size of the tabu list.

Finally, Figure 3 confirms the trend of the previous two figures, i.e. number seven is among the best w.r.t. it average objective function value and it is the faster one.

Making use of these concepts we test two types of diversification strategies which are explained below:

- Radical Diversification (radDiv): The algorithm is completely initialised. Only the best solution found so far (s^{best}) is considered in the new cycle.
- Random Diversification (randDiv): The algorithm is partially initialised. Only a fix number of paths from the current solution are modified.

We perform some tests on a subset of instances using these two criteria of diversification and

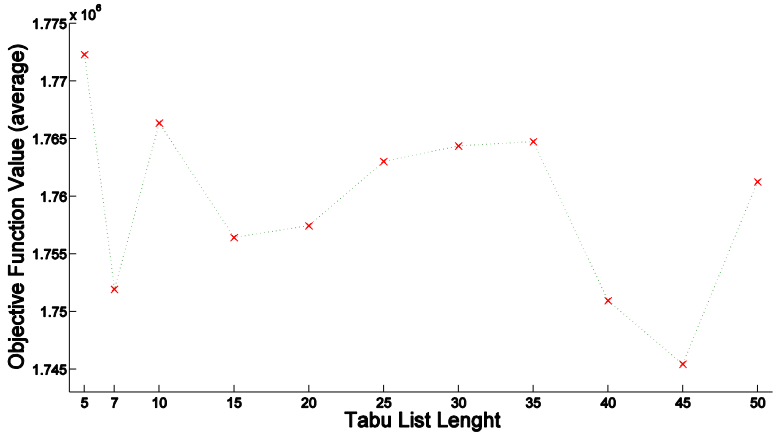

Figure 2. Objective function values (in average) obtained when different tabu list sizes (between 1 and 10) are applied over instances with 50 commodities.

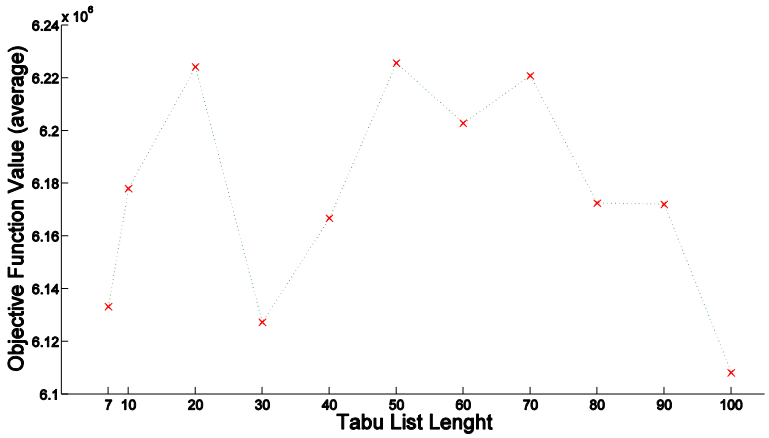

Figure 3. Objective function values (in average) obtained when different tabu list sizes (between 1 and 10) are applied over instances with 50 commodities.

the results of those tests are presented below in Table 1.

Table 1. Comparison of two different strategies of diversification. Tests were applied on a set of instances with 30 Nodes.

Instance	*randDiv*	*radDiv*
p3010fl1	80,303	80,530
p3010ft1	164,994	165,124
p3010vl1	28,273	28,282
p3010vt1	51,789	51,967
p3050fl1	620,900	626,609
p3050ft1	716,467	709,002
p3050vl1	195,756	195,328
p3050vt1	213,410	214,659
p30100fl1	2,259,040	2,224,325
p30100ft1	2,380,726	2,418,749
p30100vl1	686,501	681,571
p30100vt1	693,245	701,388

As Table 1 shows, the best results are consistently obtained when *randDiv* operator is applied exchanging a number of paths randomly. The number of routes exchanged that had the best performance was two. Then, we need to determine after how many iterations without improvement the diversification strategy must be applied. To do that we perform several tests over an instance of 30 nodes and 10 commodities. The idea is to determine, in average, how many iterations the algorithm needs to find a solution better than the current one. We perform these experiments without any diversification criterion. Results shows that, in general, the number of iterations required to find a new solution better than the current one are quite similar among the tested instances. The average value was 790 which means that, in average, the algorithm needs around that number of iterations (maximum) to find a new solution better than the current one. Thus, the threshold to invoke the diversification criterion must be larger than this value. However it must not be much larger as we should take into account that the larger the value the longer the algorithm will take searching around a (possibly) local optimal solution, a non-desirable behaviour for a local search algorithm. Finally, the algorithm stops once the maximum number of iterations is reached. Algorithm 1 shows a general framework for the implementation of a TS algorithm.

Algorithm 1: Algorithmic Frame for TS

begin
$k = 0$; Generate initialSolution s_0;
$s^{best} = s_0$;
While ($k < maxIterations$) **do**
NS_k = getNeighbourhood (s_k);
$NS_k^{best} = findBest(NS_k)$;
While(NS_k^{best} *is Tabu*) **do**
If($NS_k^{best} < s^{best}$) **then**
$aspirationCriterion(NS_k^{best})$;
end If
end while
If($NS_k^{best} < s^{best}$) **then**
$s^{best} = NS_k^{best}$;
noImproveIter = 0;
else
noImproveIter++;
end if
$k = k + 1$; $s_k = NS_k^{best}$;
$update(tabuList)$;
$check$ $(diversificationCriterion)$;
end while
end

Parameters for our algorithm are demand vectors per node (d_i), transport cost vectors per edge (c_j^i), set-up cost vectors per edge (f_j^i) and total capacity vectors per edge (v_{ij}).

Solution found by the heuristic corresponds to two vectors: x_{ij}^k for all $k \in K, (i,j) \in A$ and $y_{ij} \in \{0,1\}$ for all $(i,j) \in A$. Where K corresponds to the set of commodities to be transported.

As indicated by the Algorithm 1 in row 3, it is necessary to generate an initial solution for the heuristic. The following explains how the

current solution is generated for the general solution presented in this paper.

For each commodity, we need to build one or more paths. A path is defined as a set of edges which a commodity is transported through. Each path starts from a source node and ends in a destination node.

Figure 4 shows all the feasible paths between node i and node j. Please note that only dotted lines correspond to a path.

Generally, capacity c_k of a path is not enough to transport the full amount (K) of commodities k. Because of that, it is necessary to generate other paths to increase capacity c_k of the network. This path generation should be done making sure to satisfy capacity constraints at the edges and the flow conservation constraints in the nodes, ensuring the feasibility of the initial solution. The initial node will be the source node of the flow for a commodity k. We then analyse the nodes adjacent to the source node. One of those adjacent nodes must be selected according to the probability function $f_{prob}(i)$. This function calculates the probability of occurrence of node i . This function makes use of the following criteria:

- Available Capacity: The probability is directly proportional to the available capacity on the road tested
- Cost: The probability is inversely proportional to the cost of evaluated path.
- Cost / Capacity: The probability is indirectly proportional to the expression *Cost/Capacity* of the evaluated road.

Table 2. Paths Assignment Results.

Criterion	**Objective Function Value**
Available Capacity	693,047.182
Cost	777,910.909
Cost/Capacity	715,670.636

Once implemented, tests were performed on a subset of instances. The results of these tests are presented in Table (2). As we can see, all the three criteria were tested. When the allocation of paths is made based on the available capacity, the average cost obtained was 693,047.182. When the criterion for allocation of paths corresponds to the cost, the best average result obtained is 777,910.909 cost units. Finally when the allocation is made using the *Cost/Capacity* criterion, the best average result obtained is 715,670.636 cost units. Thus, we found that the best performance was obtained when the generation of paths was made using a criterion based on the available capacity of the network. Consequently, in this paper we consider this criterion for our selection function f_{prob} . In case that the destination node is within the set of adjacent (and feasible) nodes, the algorithm will select it independently of the probability to be assigned. If during construction of the initial solution a node (different from the destination node) does not have adjacent nodes (either due to the configuration of the network or due to capacity constraints) the process restarts.

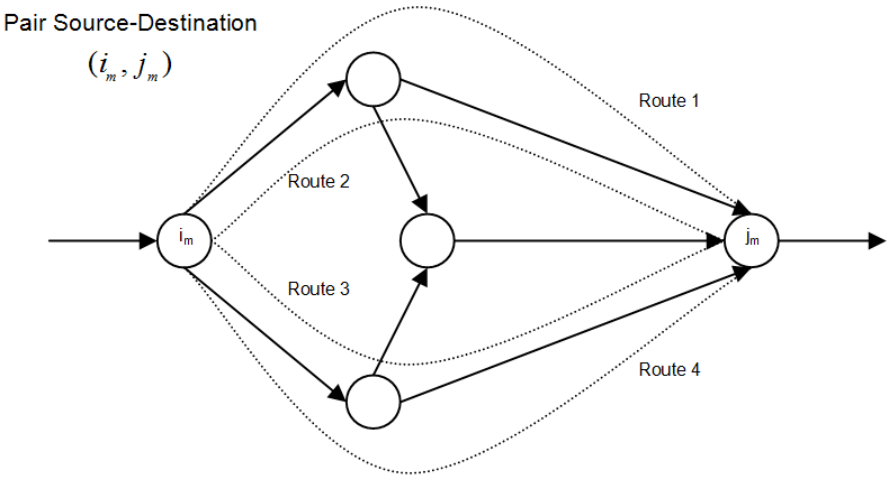

Figure 4. All possible flows from a source i_m to a destination j_m divided into paths. The paths of the same flow (i_m, j_m) interact sharing edges and their capacities.

As stated before, the implementation of the TS algorithm done in this paper is a variant of the general TS framework, since rather than select the best candidate deterministically selection is based on an evolutionary strategy. In next section this variation is presented.

3.2 Probabilistic neighbour selection criterion

As mentioned above, TS-based algorithm moves across the neighbourhood using a predefined neighbourhood-movement. Such movement generates a set of non-Tabu neighbour solutions which could be chosen to perform the movement. However, a problem arises when trying to define the criteria for this choice. On the one hand we have a completely random strategy to choice among the neighbours. This strategy should increase the algorithm exploration level. On the other hand we have several strategies to make the algorithm tends to perform in the way we desire, increasing algorithm exploitation level in detriment of its exploration level. Pure TS algorithm always chooses the best non-Tabu search alternative, at least there exists a tabu alternative with an objective function value better than the best solution found so far (aspiration criterion).

Unlike pure TS and other elitist strategies, in this article we propose a probabilistic neighbours selection criterion (PNSC) inspired in selection criteria used by evolutionary algorithms, particularly by genetic algorithms (GA). PNSC firstly increases the number of candidates by mean of the well-known evolutionary operators namely cross-over and mutation. New candidates are generated starting from the original neighbourhood. After the entire set of candidates is generated we need to select one of these candidates as the new current solution. The selection criterion is the well-known roulette wheel. The roulette wheel assigns a proportion of the wheel to each candidate according to their objective-function value. Then a random selection is made similar to how the roulette wheel is rotated. Naturally, good candidate solutions will be less likely to be eliminated [16]. This way PNSC increases the exploration level of our hybrid algorithm. However, if any candidate improves the best objective function found so far by the algorithm, that candidate will have selection probability equal to 1 (aspiration criterion). This is done, firstly, to keep those very good candidate solutions and, secondly, to balance the relation between exploration and exploitation of the algorithm.

4. Experiments and Computational Results

The hybrid TS-GA heuristic presented in this paper was implemented in Java, on a Ubuntu 12.04 operative system. The tests were performed on a computer with an Intel i5 processor and 4 GB of RAM. In next section, the used benchmark is presented. After that, the results obtained by the TS-GA heuristic and compares them with results obtained in [2] are presented.

4.1 Benchmark

In [2], the author design a random instance generator in order to generate instances classified according to the following:

- Class I: High Fixed Costs, Loosely Capacitated Networks
- Class II: High Fixed Costs, Tightly Capacitated Networks
- Class III: High Variable Costs, Loosely Capacitated Networks
- Class IV: High Variable Costs, Tightly Capacitated Networks

The instances generator is the same used in [2]. It creates a text file (ASCII) with different records that will contain the information and characteristics of each instance of the network. The name of the file is defined according to the characteristics that the network it represents (number of nodes, number of products, type of network and type of costs). The generated file is then used as input for our TS-GA heuristic, which will read the file and store the data in the classes that represent the system. The instances generator has an additional option that allows us to obtain networks with different capacities and fixed costs weights. We used this option to obtain, from a given network instance, some additional instances with the same number of products and variable costs but with different characteristics of capacity and fixed costs.

4.2 Results

The algorithm was tested in a set of instances presented in [2, 6]. We present in Tables 3 to 5 obtained results for each instance type. In each

Table the instance name (*Instance* column), cost obtained by our TS-GA algorithm (*TS-PNSC* column), cost obtained by the Scatter Search algorithm in [2] (*SS* column) and the difference between them (Δ column) is presented.

Table 3 shows the results for the instances consisting of 30 customers and 10 commodities. On average, the savings for these instances reached 1.54% with a peak of 6.58% (*p3010ft4* instance, class II). Moreover, the lowest performance is obtained for instance *p3010vt5* with increased costs of 1.83%.

Similarly, Table 4 shows the results obtained for instances of 30 customers and 50 commodities. Our algorithm presents a better performance than the Scatter Search algorithm from [2] for the majority of the instances. However, in average this difference corresponds to 0.69% only, with a peak of 5.5% (*p3050fl4* instance, class I). Moreover, the average of those instances where there is no saving is 0.83%.

Table 3. Results for instances with 30 Nodes and 10 commodities.

Instance	TS-PNSC	SS	Δ
p3010fl1	80,303	80,742	-0,54%
p3010fl2	68,114	71,288	-4,45%
p3010fl3	111,136	112,483	-1,20%
p3010fl4	86,466	88,035	-1,78%
p3010fl5	70,976	71,792	-1,14%
p3010ft1	160,950	167,019	-3,63%
p3010ft2	146,564	149,824	-2,18%
p3010ft3	198,960	207,554	-4,14%
p3010ft4	150,400	160,999	-6,58%
p3010ft5	143,150	149,995	-4,56%
p3010vl1	28,167	28,281	-0,40%
p3010vl2	22,965	23,589	-2,65%
p3010vl3	37,691	37,601	0,24%
p3010vl4	29,029	29,154	-0,43%
p3010vl5	24,267	23,997	1,13%
p3010vt1	50,113	50,061	0,10%
p3010vt2	44,471	44,361	0,25%
p3010vt3	62,665	62,272	0,63%
p3010vt4	48,153	48,812	-1,35%
p3010vt5	44,610	43,807	1,83%

Table 4. Results for instances with 30 Nodes and 50 commodities.

Instance	TS-PNSC	SS	Δ
p3050fl1	600,826	594,872	1,00%
p3050fl2	658,782	664,774	-0,90%
p3050fl3	870,069	862,578	0,87%
p3050fl4	712,844	754,294	-5,50%
p3050fl5	713,464	697,016	2,36%
p3050ft1	691,217	715,559	-3,40%
p3050ft2	725,156	732,436	-0,99%
p3050ft3	978,242	971,473	0,70%
p3050ft4	827,636	845,984	-2,17%
p3050ft5	817,905	816,569	0,16%
p3050vl1	186,979	191,260	-2,24%
p3050vl2	214,047	213,792	0,12%
p3050vl3	270,508	269,559	0,35%
p3050vl4	235,758	240,052	-1,79%
p3050vl5	220,967	224,295	-1,48%
p3050vt1	209,752	207,560	1,06%
p3050vt2	232,811	233,405	-0,25%
p3050vt3	294,363	295,885	-0,51%
p3050vt4	254,949	255,879	-0,36%
p3050vt5	245,178	247,309	-0,86%

Table 5 shows a very similar behaviour between both algorithms, being our algorithm slightly better on average (0.13%). The best performance of our TS-GA algorithm is obtained at the instance *p30100vl1* class III (5.6%). Unlike the results shown in Tables 3 and 4, in this case we have a significant increment in the average cost of those instances where there is no saving. This increment is 2.45% on average, with a peak of 5.09% (*p30100fl3* instance, class I)

Table 5. Results for instances with 30 Nodes and 100 commodities.

Instance	TS-PNSC	SS	Δ
p30100fl1	1,836,807	1,777,644	3,33%
p30100fl2	2,134,616	2,063,516	3,45%
p30100fl3	1,812,369	1,724,508	5,09%
p30100fl4	2,183,939	2,160,174	1,10%
p30100fl5	1,681,758	1,711,917	-1,76%
p30100ft1	2,019,340	2,073,628	-2,62%
p30100ft2	2,307,337	2,361,424	-2,29%
p30100ft3	1,902,447	2,010,868	-5,39%
p30100ft4	2,378,418	2,443,325	-2,66%
p30100ft5	1,883,920	1,875,002	0,48%
p30100vl1	577,123	611,379	-5,60%
p30100vl2	699,374	700,556	-0,17%
p30100vl3	577,334	572,004	0,93%
p30100vl4	710,446	726,896	-2,26%
p30100vl5	594,903	575,406	3,39%
p30100vt1	608,137	627,080	-3,02%
p30100vt2	715,723	708,939	0,96%
p30100vt3	573,291	581,496	-1,41%
p30100vt4	736,563	709,097	3,87%
p30100vt5	603,449	592,142	1,91%

Tables 6 to 8 show an analysis of the results grouped by class, cost, type of network and number of commodities, respectively. Our algorithm presents, on average, better results than the Scatter Search in [2] in almost all classes (I to III) with the only exception of class IV.

Table 6. Comparison of instances grouped by class.

Class	TS-PNSC	SS	Δ
I	2,977,749.60	2,687,126.60	0.00%
II	3,285,099.00	3,136,331.80	-2.62%
III	902,116.00	893,564.20	-0.72%
IV	959,674.80	941,621.00	0.19%

Table 7. Comparison of instances grouped by Cost.

Cost	TS-PNSC	SS	Δ
Fix	3,131,424.30	2,911,729.20	-1.31%
Var.	930,895.40	917,592.60	-0.27%

Based on the results presented above, it is possible to state that our algorithm shows a satisfactory performance for most of the instances.

Table 8. Comparison of instances grouped by type of network (Loosely and Tightly Capacitated).

Network	TS-PNSC	SS	Δ
LC	1,926,666.10	1,790,345.40	-0.36%
TC	2,109,424.40	2,038,976.40	-1,21%

However, it can be noted that as the problem increases in size or complexity the performance of our algorithm tends to fall. Because of that, new strategies must be implemented to improve the performance of the algorithm in these more complex (bigger) instances. New strategies to select the best candidate solution as well as adding dynamic selection strategies could be an option to improve the algorithm performance.

Table 9 shows a summary of the results sorted by the number of commodities.

Table 9. Comparison of instances grouped by number of commodities (#p).

#p	TS-PNSC	SS	Δ
10	80,644.10	82,583.30	-2,57%
50	498,072.65	501,727.55	-0,73%
100	1,326,881.30	1,330,350.05	-0,26%

From these results, some important remarks can be given: The algorithm performs better in those instances with predominant fixed costs, as seen in Tables 6 and 7. This is because of the strategy used in the generation of paths, which is preferentially guided to the destination node as long as this is in the list of candidate nodes. Our TS-GA algorithm performs better when applied to networks with restricted capacity. This may be due to another criterion used in the generation of paths: preference is given to those paths with greater available capacity. While this strategy aims to create a smaller number of paths-per-flow, might not be taking into account the creation of a greater number of new edges, which would translate into higher fixed costs.

Table 9 shows that for the instances of 10 and 50 commodities a non-negligible improvement exists compared to the benchmark. Finally, for instances with 100 commodities it can be seen a small deterioration in performance in relation to total costs obtained, which can be attributed to two situations: The method used to diversify is not the best one we can use and, therefore, we should look for strategies that increase the solution space and therefore increase the ability of our TS-GA algorithm of finding a solution better than the one obtained by the Scatter Search algorithm. Also it might be that the neighbourhood definition or movement definition is not the adequate for this problem.

5. Conclusions and Future Work

In this paper we have presented a hybrid algorithm of TS and GA. While TS guides the search, GA balances the relation between exploration and exploitation levels of the algorithm by means of evolutionary operators, such as cross-over and mutation, used during the selection process. We described in detail how we made decisions about parameter values of the algorithm as well as the different criteria used by our algorithm. The parameters in a meta-heuristic are a key factor in its performance. In the early stages of the implementation up to 300% difference were obtained in results compared to the best solution due to poor parameters calibration in the tabu list size and diversification threshold. Obtained results shown that our algorithm is able to find very competitive solutions especially for small instances or with non-too-hard restrictions. However, it is necessary to try different strategies within the algorithm to keep the good performance obtained for small instances on those large instances with hard restrictions.

Finally, the implementation of a number of strategies remains as future work. Among them, new probabilistic selection criteria, new neighbourhood definitions as well as the implementation of other path generation strategies could lead to improve results obtained by our hybrid algorithm.

REFERENCES

1. Al-SULTAN, K., M. Al-FAWZAN, **A Tabu Search Approach to the Uncapacitated Facility Location Problem**, Annals of Operation Research, vol. 86, 1999, pp. 91-103.
2. ALVAREZ, A., J. GONZALEZ-VELARDE, K. DE-ALBA, **GRASP Embedded Scatter Search for the Multicommodity Capacitated Network Design Problem**, Journal of Heuristics, vol. 11, 2005, pp. 233-257.
3. ALVAREZ, A., J. GONZALEZ-VELARDE, K. DE-ALBA, **Scatter Search for Network Design Problem**, Annals of Operation Research, vol. 138, 2005, pp. 159-178.
4. BRANDÃO, J., A. MERCER, **A Tabu Search Algorithm for the Multi-Trip Vehicle Routing and Scheduling Problem**, European Journal of Operational Research, vol. 100, 1997, pp. 180-191.
5. CABRERA, G., P. A. MIRANDA, E. CABRERA, R. SOTO, B. CRAWFORD, J. M. RUBIO, F. PAREDES, **Solving A Novel Inventory Location Model with**

Stochastic Constraints and (*R, s, S*) Inventory Control Policy, Mathematical Problems in Engineering, vol. 2013, Article ID 670528, 12 pages, 2013.

6. COBOS, N., A. ALVAREZ, **Tabu Search-Based Algorithm for Capacitated Multicommodity Network Design Problem**, 14th International Conference on Electrical, Communication & Computing - ICECC, 2004, p. 144.
7. CRAINIC, T. G., Y. LI, M. TOULOUSE, **A First Multilevel Cooperative Algorithm for Capacitated Multicommodity Network Design**, Computers & Operations Research, vol. 33, 2006, pp. 2602-2622.
8. DAVIDOVIC, T., P. HANSEN, N. MLADENOVIC, **Permutation-based Genetic, Tabu, and Variable Neighborhood Search Heuristics for Multiprocessor Scheduling with Communication Delays**, Asia-Pacific Journal of Operational Research, vol. 22,, 2005, pp. 297-326.
9. GARAI, G., B. CHAUDHURII, **A Novel Hybrid Genetic Algorithm with Tabu Search for Optimizing Multi-dimensional Functions and Point Pattern Recognition**, Information Sciences, vol. 221, 2013, pp. 28-48.
10. GENDRON, B., T. CRAINIC, A. FRANGIONI, **Multicommodity Capacitated Network Design**, Telecommunications Network Planning, 1999, pp. 1-19.
11. GHAMLOUCHE, I., T. CRAINIC, M. GENDREAU, **Path Relinking, Cycle-based Neighbourhoods and Capacitated Multicomodity Network Design**, Annals of Operation Research, vol. 131, 2004, pp. 109-133.
12. GLOVER, F., B. MELIÁN, **Tabu Search**, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial vol. 19, 2003, pp. 29-48.
13. GONZALEZ, M., C. VELA, R. VARELA, **Genetic Algorithm Combined with Tabu Search for the Job Shop Scheduling Problem with Setup Times, Methods and Models in Artificial and Natural Computation**, 2009, pp. 265-274.
14. El FERCHICHI, S., K. LAABIDI, S. ZIDI, **Genetic Algorithm and Tabu Search for Feature Selection**, Studies in Informatics and Control, 2009, vol. 18(2), pp. 181-187.
15. HANAFI, S., A. FREVILLE, **An Efficient Tabu Search Approach for the 0 - 1 Multidimensional Knapsack Problem**, European Journal of Operational Research, vol. 106, 1998, pp. 659-675.
16. HOLLAND, J., **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**, University of Michigan Press, 1975.
17. HOLMBERG, K., D. YUAN, **A Lagrangian Heuristic based Branch-and-Bound Approach for the Capacitated Network Design Problem**, Operations Research, vol. 48, 2000, pp. 461-481.
18. CABRERA, G., S. RONCAGLIOLO, J.P. RIQUELME, C. CUBILLOS, R. SOTO. **A Hybrid Particle Swarm Optimization - Simulated Annealing Algorithm for the Probabilistic Travelling Salesman Problem**, Studies in Informatics and Control, 2012, vol. 21(1), pp. 49-58.
19. MAGNANTI, T., R. WONG, **Network Design, and Transportation Planning: Models and Algorithms**, Transportation Science, vol. 1, 1984, pp. 1-55.
20. MINOUX, M., **Networks Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications**, Networks, vol. 19, 1988, pp. 313-360.
21. TING, C. K., C. F. KO, C. H. HUANG, **Selecting Survivors in Genetic Algorithm using Tabu Search Strategies**, Memetic Computing vol. 1, 2009, pp. 191-203.
22. VALLS, V., M. PEREZ, M. QUINTANILLA, **A Tabu Search Approach to Machine Scheduling**, European Journal of Operational Research, vol. 106, 1998, pp. 277-300.
23. VILCOT, G., J. C. BILLAUT, **A Tabu Search and a Genetic Algorithm for Solving a Bicriteria General Job Shop Scheduling Problem**, European Journal of Operational Research, vol. 190, 2008, pp. 398-411.

24. VU, D., T. CRAINIC, M. TOULOUSE, **A Three-phase Matheuristic for Capacitated Multi-Commodity Fixed-Cost Network Design with Design-Balance Constraints**, Journal of Heuristics vol. 19, 2013, pp. 757-795.

25. WESLEY BARNES, J., M. LAGUNA, **A Tabu Search Experience in Production Scheduling**, Annals of Operation Research, vol. 41, 1993, pp. 139-156.

26. YAGHINI, M., M. RAHBAR, M. KARIMI, **Hybrid Simulated Annealing and Column Generation Approach**, Journal of the Operational Research Society, vol. 64, 2013, pp. 1010-1020.

27. HAGEMAN, J., R. WEHRENS, H. van SPRANG, L. BUYDENS, **Hybrid Genetic Algorithm-Tabu Search Approach for Optimising Multilayer Optical Coatings**, Analytica Chimica Acta vol. 490, 2003, pp. 211-222.

28. JAT, S. N., S. YANG, **A Hybrid Genetic Algorithm and Tabu Search Approach for Post Enrolment Course Timetabling**, Journal of Scheduling vol. 14, 2011, pp. 617-637.

29. DURAN, O., L. PEREZ, **Solution of the Spare Parts Joint Replenishment Problem with Quantity Discounts using a Discrete Particle Swarm Optimization Technique**, Studies in Informatics and Control, 2013, vol. 22(4), pp. 319-328.