International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 8 Issue 6 Ser. I ǁ 2020 ǁ PP. 01-10

Comparing Metaheuristic Procedures to Solve the Single-Source Transportation Problems

Mata Crespo, R.[1]
[1]*Department of Statistics and Operational Research, School of Industrial Engineering Valladolid University 47011 Valladolid, España*

Abstract
In transport networks, it is generally required that all demand points must be supplied from a single source. The formulation of the model is presented and randomized versions of two well-known constructive heuristics are applied: On the one hand, the maximum regret method and on the other hand the maximum demand method. In addition, two other heuristic improvements are applied with local search techniques: "shift" and "swap" improvements. With all elements listed above, the GRASP method to solve single-source transportation problems is presented. However this work mainly provides simple tools for the optimization of the logistics (supply chains) of food and forestry products.
Keywords: *Transportation Problems, Integer Programming, Heuristic Procedures, Combinatorial Optimization, Statistical Inference, Local Search.*

Date of Submission: 23-06-2020 Date of acceptance: 11-07-2020

I. INTRODUCTION

In general, the transportation problem refers to the shipment of quantities, products, etc. from its origin to its destination. However, logistics must lead the way: First and foremost, to establish a transport model and also to be optimal. Consecuently the modelling of the problem and its optimization are two essential subjects. The basic notions on necessary operational research can be found in [16].

In the formulation of a transportation model, m sources of a certain amount s_i of units supplies of a product, $i = 1, ..., m$ are considered. In addition, we must consider the n destination points that have demands or orders of d_j units of that product. The cost of sending a unit of the product from origin i to destination j, what it is known as unit transport cost, is $c_{ij} > 0$, $i = 1, ..., m$, $j = 1, ..., n$.

It is necessary to limit transport costs, $j = 1, ..., n$ and, obviously, the purpose is to minimize the total cost. It is normally assumed that the cost of transportation is directly proportional to the amount transported. Thus the problem can be handled as a linear programming problem. The data can be organized in a double entry table: supplies (s_i) vs. demands (d_j), the costs (c_{ij}) being the table coefficients.

Alternatively, a graph could be used since it allows not only a quick and intuitive but also qualitative visualization of transport routes, formulating it as a network optimization problem [3].

With regard to a transportation problem, a demand point could be supplied from a number of origins, but it might be quite expensive. Due to this reason, the condition that any demand point is supplied by a single origin can be imposed. We will call this condition the single-source constraint. Nevertheless, considering that it is necessary to analyze how much the total cost would increase. Therefore just knowing what the total cost is if the condition of single-source is required. With these purposes two formulations for the problem of single-source transportation (SSTP) are presented. In spite of its importance, this topic has not received a great deal of attention from the 70's ([1], [15] and [11]).

2 Model Formulation

2.1 First Model

As to demand point j, the variables that indicate how the demand is covered d_j are z_{1j}; z_{2j}; ... z_{mj}, and the single-source constraint assumes that only one of these m variables can have a positive value in the optimal solution. This constitutes a special case of the logical constraint that we call "maximum number of variables", which is reduced to the fact that at most a single variable of m can be positive. Modelling this condition there are considered binary variables $y_{ij} \in \{0,1\}$, $i = 1, ...m; j = 1, ..., n$, $z_{ij} \leq u_{ij} \cdot y_{ij}$ and the logical constraint $\sum_{i=1}^{m} y_{ij} \leq 1$, $j = 1, ..., n$. The upper bound can be taken as $u_{ij} = \min\{s_i, d_j\}$. The complete formulation is an example of **Mixed Integer Programming (MIP)**:

$$
\begin{array}{ll}
\text{Minimize} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}z_{ij} \\
\text{subject to} & \sum_{j=1}^{n} z_{ij} \leq s_i \ , \forall i = 1, ...m \\
& \sum_{i=1}^{m} z_{ij} \geq d_j, \forall j = 1, ...n \\
& z_{ij} \leq u_{ij}y_{ij}, \forall i = 1, ...m \ \forall j = 1, ...n \\
& \sum_{i=1}^{m} y_{ij} \leq 1, \forall j = 1, ...n \\
& z_{ij} \geq 0, \forall i = 1, ...m \ \forall j = 1, ...n \\
& y_{ij} \in \{0,1\}, \forall i = 1, ...m \ \forall j = 1, ...n
\end{array}
$$

2.2 Second Model

Variables are redefined as $y_{ij} = \frac{z_{ij}}{d_j}$. Those results in a transportation model equivalent to the original, however with variables $0 \leq yij \leq 1$ representing the fraction of the demand for point j covered or served from the source i. Formally, the model is as it follows:

$$
\begin{array}{ll}
\text{Minimize} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}d_jy_{ij} \\
\text{subject to} & \sum_{j=1}^{n} d_jy_{ij} \leq s_i \ , \forall i = 1, ...m \\
& \sum_{i=1}^{m} y_{ij} = 1, \forall j = 1, ...n \\
& 0 \leq yij \leq 1, \forall i = 1, ...m \ \forall j = 1, ...n
\end{array}
$$

Now the constraints $0 \leq yij \leq 1$ are replaced by $y_{ij} \in \{0,1\}$ in agreement with the single-source constraint. Hence, the problem of single-source transportation can be modelled as a problem with all binary variables, providing an example of **Pure Binary Programming (PBP)**, which can be formulated this way:

$$
\begin{array}{ll}
\text{Minimize} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}d_jy_{ij} \\
\text{subject to} & \sum_{j=1}^{n} d_jy_{ij} \leq s_i \ , \forall i = 1, ...m \\
& \sum_{i=1}^{m} y_{ij} = 1, \forall j = 1, ...n \\
& y_{ij} \in \{0,1\}, \forall i = 1, ...m \ \forall j = 1, ...n
\end{array}
$$

3 Bounds and Heuristic Procedures

As it is well known, these methods allow to solve complicated problems in an approximate way, providing feasible solutions that, although they do not optimize the objective function, it is sufficiently close to the optimum value by using a reasonable time.

While the original problem is difficult to solve, the relaxed problem may be easier to work out. The most common way to achieve a bound is the so-called linear relaxation where the $y_{ij} \in \{0,1\}$ constraint is replaced by $0 \leq yij \leq 1$.

Another bound used in this work is the called Relax and Fix (see [18]). It consists in the relaxation of the objective function, previously establishing a tolerance (in our case tole = 0.0001 has been established) so that if $y_{ij} \geq$ 1-tole then $y_{ij} = 1$ is fixed and it is enough to impose the condition that the lower bound of y_{ij} be 1 (setlb $(y_{ij}) := 1$). Similarly, if $y_{ij} \leq$ tole then set $y_{ij} = 0$, and it is enough to impose the condition that the upper bound of y_{ij} be 0 (setub $(y_{ij}) := 0$).

As for the heuristic procedures in the greedy method a feasible solution is built step by step. On the other hand, heuristic improvement methods start from a feasible solution to the problem and try to improve it. It is likely to combine both heuristics. For instance, a two-stage heuristic by using first a greedy method to obtain an initial solution, secondly try to improve it by a local search method.

Alternatively, the method can be both a single-step and multistart procedure. In the latter case, some type of randomization has to be used to obtain a different solution in each step. There are several ways to introduce randomness to the algorithm. One of the best known procedures use a RCL (Restricted Candidate List). This list consists of the elements candidates that offer the best values of the greedy criterion. The next candidate added to the solution is chosen randomly from the restricted list of candidates. Such a list may have a fixed number of elements (cardinality constraint).

In this work, randomized versions of two well-known constructive heuristics are applied: the maximum regret method and the maximum demand method. Both cover two interesting aspects of transport logistics with a single source, reinforcing the scope and applicability of our results to specific cases of interest to potential users. The greedy randomized code with cardinality-based RCL has been taken from [13] (Figure 2) and [14].

In the local search procedures Algorithm 1 and Algorithm 2, the environment or neighborhood of the feasible solution is explored through a basic operation called movement that, applied to the solution, provides the solutions of its environment. Concretely, the method known as VNS (Variable Neighboord Search), can be found in [10]. In this work, two local search techniques were used: "shift" and "swap" improvements. In the first one, the movements consist of, given a solution, changing the assignment of a destination to a different origin. However in the second case the movements consist in the exchange of assignments to two destinations, Algorithm 3 and Algorithm 4.

In our case, only movements in the current solution environment are allowed to improve the value of the objective function. As a consequence, we have the so-called descent method that always ends at a local minimum; but a local optimum will not always be a global optimum. By starting with a given solution x, k = 1 is taken and repeated until k = 2, since two types of environment are considered. The environment is explored to find the best solution x' of the k-th environment NK (x) and a decision is made to move or not. If the solution obtained x' is better than x, x = x' and k = 1 are taken. Otherwise, k = k + 1 is taken. This scheme has been taken following [6].

In order to find better solutions than the ones we described before, the GRASP (Greedy Randomized Adaptative Search Procedures) were used where in each iteration of the multistart (randomized) method a local search procedure is applied to the greedy solution. This method incorporates the two main features that are required of a metaheuristic: diversification (by randomization and repetition) and intensification through local search. In each iteration one obtains a local minimum and, if randomization works correctly, multiple regions will been examined so that the best solution will be close to the global optimum. To be more precise, this paper presents hybrid algorithms that are based on constructive greedy heuristics with local search. First, the solution is built by a construtive heuristic. Second, the local search is used to improve that initial solution, and the last phase consists of updating the parameters that govern the process of construction in the first phase. The three phases are repeated until the stop criterion is reached.

The methodology described can be easily adapted to other combinatorial optimization problems, since it is in fact a particular case of the GAP (Generalized Assignment Problem) described in [8]. Also related problems are considered in [12]. Certainly the heuristics that have inspirated this work has been taken from [7]. Moreover, other heuristics found in [2] [9] and [17] have been useful in this way.

4 Algorithm

This section includes main algorithms used, being M a sufficiently large quanty:

Algorithm 1 An algorithm for regret heuristic improvement using local search techniques

```
Initializations nasign=0 ;zheur=0 ;iter=0 ;feas=1
while (nasign < n and feas = 1) do
  iter=iter+1;
  while (j in destinations / y(j)=0) do
    nk=0;
    for (i in origins) do
      if (demand(j) ≤ ofres(i)) then
        nk=nk+1;
      end if
    end for
    if (nk = 0) then
      feas=0;
    else
      cmin1=M;
      for (i in origins / demand(j) ≤ ofres(i)) do
        if (cost(i,j) < cmin1) then
          imin(j)=i; cmin1=cost(i,j);
        end if
      end for
      if (nk = 1) then
```

```
          regret(j)=M;
        else
          cmin2=M;
          for (i in origins / demand(j)≤ ofres(i) and i≠imin(j)) do
            if (cost(i,j) < cmin2) then
              i2=i; cmin2=cost(i,j);
            end if
          end for
          regret(j)= cmin2 - cmin1;
        end if
      end if
    end while
    if (feas =1) then
      nK=0;
      for (j in destinations / y(j)=0) do
        maxreg=-M;
        for (j in destinations / y(j)=0) do
          if (regret(j) > maxreg) then
            jmax=j; maxreg=regret(j);
          end if
        end for
        nK=nK+1; ind(nK)=jmax; mark(jmax)=1; y(jmax)=imin(jmax);
        zheur=zheur+cost(imin(jmax),jmax)*demand(jmax);
        ofres(imin(jmax))=ofres(imin(jmax))-demand(jmax); nasign=nasign+1;
      end for
    end if
  end while
  if (feas = 1) then
    m1=swapimprovement; m2=shiftimprovement;
  end if
```

Algorithm 2 An algorithm for maximum demand heuristic improvement using local search techniques

```
  Initializations nasign=0 ;zheur=0 ;feas=1
  while (nasign < n and feas = 1) do
    nK=0;
    dmax=-M;
    for (j in destinations/ y(j)=0) do
      if (demand(j) > dmax) then
        jmax=j ;
        dmax=demand(j) ;
      end if
    end for
    if (dmax > -M) then
      nK=nK+1;
      ind(nK)=jmax;
      mark(jmax)=1;
    end if
    cmin=M;
    nk=0;
    for (i in origins / demand(jmax)≤ ofres(i)) do
      if (cost(i,jmax)<cmin) then
        cmin=cost(i,jmax);
        omin=i;
        nk=nk+1;
      end if
```

```
    end for
    if (nk > 0) then
      y(jmax)=omin;
      zheur=zheur+cost(omin,jmax)*demand(jmax);
      ofres(omin)=ofres(omin)-demand(jmax);
      nasign=nasign+1;
    else
      feas=0;
    end if
  end while
if (feas = 1) then
  z1 = zheur;
  m1=swapimprovement; m2=shiftimprovement;
  if (zheur < zimprove) then
    zimprove = zheur;
    yimprove = y;
  end if
end if
```

Algorithm 3 An algorithm for shift function

```
zini=sum(j in destinations)demand(j)*cost(y(j),j);
final=0;
while (final=0) do
  maximprovement=-M;
  for (j in destinations) do
    i1=y(j);
    nk=0;
    for (i in origins / i ≠ i1 and demand(j)≤ ofres(i)) do
      nk=nk+1
      if (nk > 0) then
        cmin=M;
        for (i in origins / i ≠ i1 and demand(j)≤ofres(i)) do
          if (cost(i,j)<cmin) then
            cmin=cost(i,j);
            i2=i;
          end if
        end for
        improvement=(cost(i1,j)-cost(i2,j))*demand(j);
        if (improvement > maximprovement) then
          maximprovement=improvement;
          jmax=j;
          i1max=i1;
          i2max=i2;
        end if
      else
        improvement=-M;
      end if
    end for
  end for
  if (maximprovement ≤ 0) then
    final=1;
  else
    y(jmax)=i2max;
    zheur=zheur+maximprovement;
    ofres(i1max)=ofres(i1max)+demand(jmax);
    ofres(i2max)=ofres(i2max)-demand(jmax);
    z3 = zheur;
  end if
end while
if (zheur < zini) then
  returned=1;
else
  returned=0;
end if
```

Algorithm 4 An algorithm for swap function

```
Initializations iter=0; final=0
zini=sum(j in destinations)demand(j)*cost(y(j),j);
while (final=0) do
  maximprovement=-M;
  iter=iter+1;
  for (j1,j2 in destinations / j1 ≠ j2) do
    i1=y(j1);i2=y(j2);
    if ofres(i1)-demand(j1)+demand(j2)≤0 and ofres(i2)-demand(j2)+demand(j1)≤0) then
      improvement=cost(i1,j1)+cost(i2,j2)-cost(i1,j2)-cost(i2,j1);
      if (improvement > maximprovement) then
        maximprovement=improvement;
        j1max=j1;
        j2max=j2;
      end if
    end if
  end for
  if (maximprovement ≤ 0) then
    final=1;
    zheur=sum(j in destinations)demand(j)*cost(y(j),j);
    z2 = zheur;
  else
    i1=y(j1max);
    i2=y(j2max);
    y(j1max)=i2;
    y(j2max)=i1;
    ofres(i1)=ofres(i1)-demand(j1max)+demand(j2max);
    ofres(i2)=ofres(i2)-demand(j2max)+demand(j1max);
    zheur=sum(j in destinations)demand(j)*cost(y(j),j);
    z2 = zheur;
  end if
end while
if (zheur < zini) then
  returned=1;
else
  returned=0;
end if
```

5 Experiments and computational results

This section describes the computational experiments that were made so as to evaluate the proposed metaheuristics, the computational results and a comparison between the different methods.

For experiments, we have developed our own "dataset", generating the following uniform probability distributions with the professional version of Xpress Mosel since the number of rows and columns exceeded the maximum allowed for the free license that is 5000. It is convenient to know the program interface and the basic optimization environment [5].

Uniform probability distributions have been considered for: Offers $\sim U(75, 95)$, Demands $\sim U(55, 85)$, Costs $\sim U(100, 100)$. The set of worked situations (different m $\times$ n scenaries) are collected in table 1 together with the integer optimal solution which was obtained for every one.

m x n	Optimal
40x40	295438
40x100	280369
40x150	273387
40x200	278303
60x100	430804
60x200	416723
100x100	708686
100x120	719392
100x200	719130

Table 1: Optimal solution for each scenary

The solution quality of each method was appraised by the percentage of variation with respect to the optimum, so we look for a small percentage gap to have a high quality solution. Also it is interesting to gather in Table 2 the percentage gap for bounds with all heuristic procedures.

m x n	Relaxed	Relax - Fix	VNS Regret	VNS Dem.	GRASP Regret	GRASP Dem.
40x40	1.17 %	1.4 %	1.8%	2.5 %	0.59 %	0.61 %
40x100	0.15 %	0.06 %	0.008%	0.11 %	0.0001 %	0.04 %
40x150	0.08 %	0.0001 %	0.0001%	0.16 %	0.0001 %	0.16 %
40x200	0.079 %	0.03 %	0.02%	0.05 %	0.005 %	0.0001 %
60x100	0.23 %	0.01 %	0.03%	0.33 %	0.03%	0.30 %
60x200	0.12%	0.02 %	0.001%	0.04 %	0.001 %	0.04 %
100x100	0.76 %	0.23 %	0.25%	1.09 %	0.26 %	0.74 %
100x120	0.21 %	0.07 %	0.057%	0.54 %	0.025 %	0.37 %
100x200	0.093%	0.058 %	0.02%	0.07 %	0.01 %	0.05 %
Average	**0.321333%**	**0.208668 %**	**0.24289%**	**0.543333%**	**0.102336 %**	**0.256668 %**

Table 2: Percentage gap for the different bounds and heuristic procedures

Moreover, computational time (in seconds) is reflected in Table 3 for each scenary and also its average values were calculated for the different methods that were used.

m x n	Relaxed	Relax - Fix	VNS Regret	VNS Dem.	GRASP Regret	GRASP Dem.
40x40	0.047	0.016	0.016	0.016	0.281	0.094
40x100	0.125	0.031	0.032	0.015	0.578	0.063
40x150	0.187	0.032	0.046	0.014	0.842	0.062
40x200	0.218	0.062	0.047	0.011	1.139	0.093
60x100	0.203	0.031	0.062	0.016	1.31	0.109
60x200	0.343	0.078	0.124	0.010	2.449	0.156
100x100	0.359	0.062	0.187	0.047	3.448	0.78
100x120	0.359	0.062	0.187	0.047	3.448	0.78
100x200	0.603	0.11	0.328	0.016	6.505	0.297
Average	**0.278444**	**0.0555556**	**0.116111**	**0.0195556**	**2.27244**	**0.227111**

Table 3: Computational times for the different bounds and heuristic procedures

The percentage gap versus the computational time is displayed in Figure 1. The average percentage gap versus the average computational time for bounds and heuristic procedures is displayed in Figure 2.

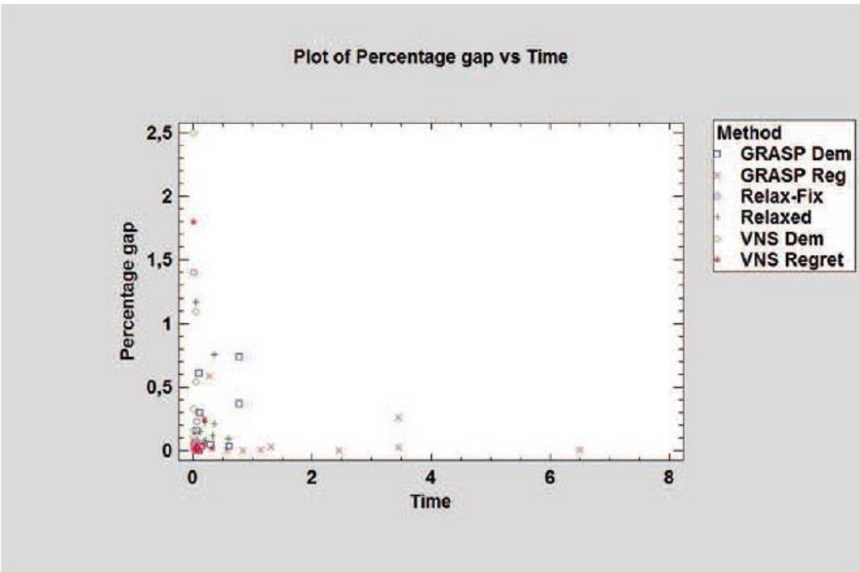

Figure 1: Percentage gap vs. Computational time

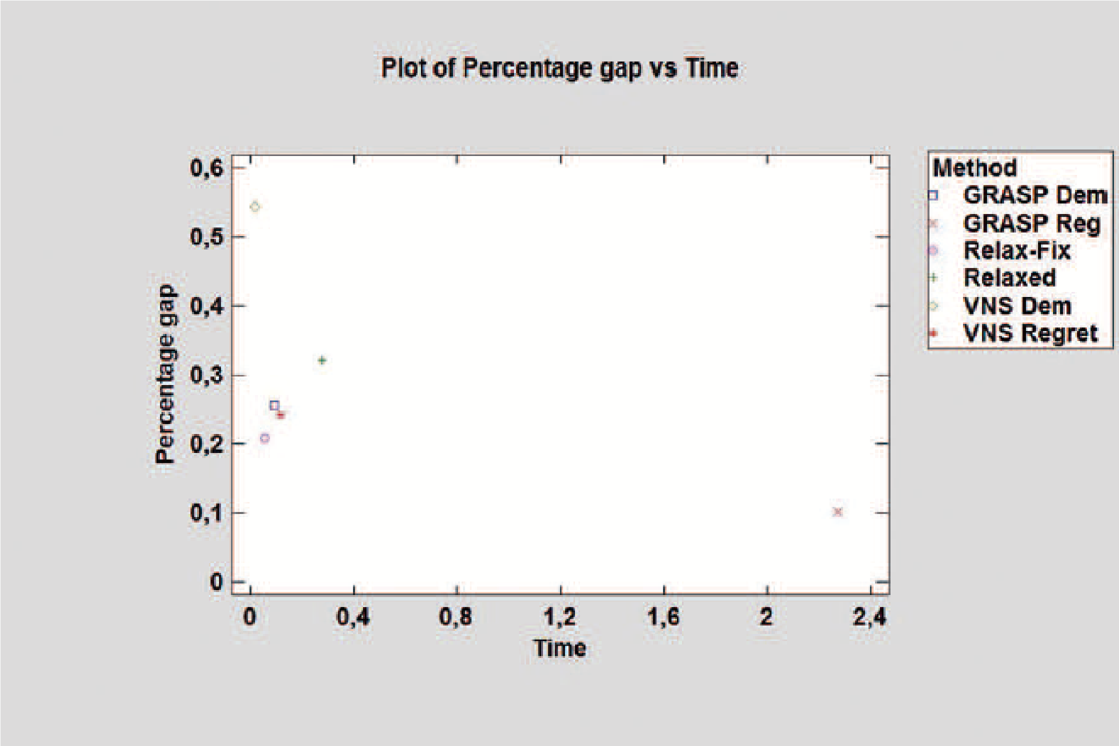

Figure 2: Average percentage gap vs. Average computational time

Besides, it is interesting to compare the average percentage gap versus the average computational time only for heuristic procedures as it is shown in Figure 3.

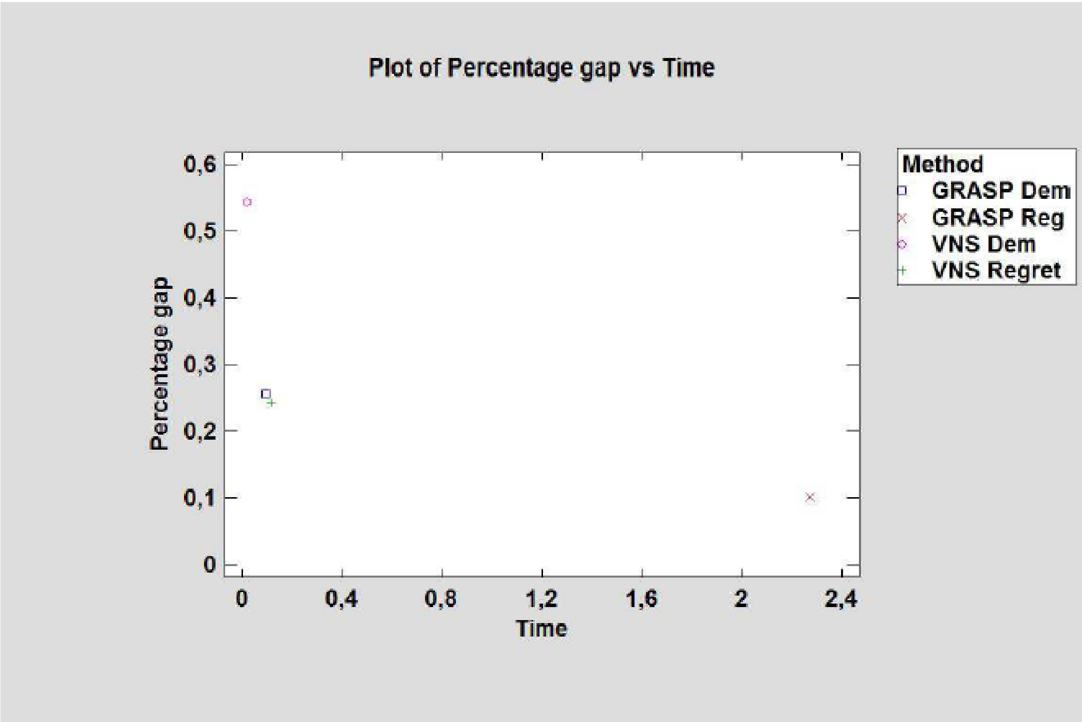

Figure 3: Heuristic procedures: Average percentage gap vs. Average computational time

From the statistical point of view, it appears to be relevant a comparative study of both percentage gap and time for the different methods. For that purpose the multifactor analysis of variance, ANOVA ([4] Chapter 5) was used dealing with the data given in tables 2 and 3. In our case, the null hypothesis H_0 states that the average values of the six methods are the same and as an alternative hypothesis H_1 states that the average values of the six methods are different. In this analysis, we considered the confidence level at 95 % and therefore the nominal level $\alpha = 0.05$. Tables 4 and 5 summarize the ANOVAs of the variables quality and time, respectively.

	SS	Df	F	p-value
Method	0.9866	5	0.8082	0.5496
Residuals	11.7197	48		

Table 4: Analysis of Variance for percentage gap

Taking into account the result obtained (Table 4), the quality variable shows a p-value greater than the nominal level $\alpha = 0.05$, therefore the null hypothesis of equality between the six averages is not rejected and we cannot affirm that there is some difference between them. The six methods can be considered equivalent in terms of quality.

	SS	Df	F	p-value
Method	34.566	5	9.805	0.000001678 ***
Residuals	33.844	48		

Table 5: Analysis of Variance for time

The time p-value is small enough (as indicated by the lower significance coded with the asterisks in Table 5, the p-values are < 0.05) to reject the null hypothesis of equality between the six averages and there is some difference between different methods. The next step is to determine that difference. For this, the Tukey multiple comparison was used. The confidence level has also been established at 95 % and therefore the nominal level $\alpha = 0.05$.

The results which were obtained are given in Table 6.

Methods	Difference	Lower bound	Upper bound	p-value
GRASP Reg-GRASP Dem	2.04533333	0.8705436	3.2201231	0.0000643
Relaxed-GRASP Dem	0.05133333	-1.1234564	1.2261231	0.9999945
Relax Fix-GRASP Dem	-0.17155556	-1.3463453	1.0032342	0.9979388
VNS Dem-GRASP Dem	-0.20755556	-1.3823453	0.9672342	0.9949288
VNS Regret-GRASP Dem	-0.11100000	-1.2857897	1.0637897	0.9997503
Relaxed-GRASP Reg	-1.99400000	-3.1687897	-0.8192103	0.0000996
Relax Fix-GRASP Reg	-2.21688889	-3.3916786	-1.0420991	0.0000146
VNS Dem-GRASP Reg	-2.25288889	-3.4276786	-1.0780991	0.0000106
VNS Regret-GRASP Reg	-2.15633333	-3.3311231	-0.9815436	0.0000247
Relax Fix-Relaxed	-0.22288889	-1.3976786	0.9519009	0.9929369
VNS Dem-Relaxed	-0.25888889	-1.4336786	0.9159009	0.9860127
VNS Pen-Relaxed	-0.16233333	-1.3371231	1.0124564	0.9984177
VNS Dem-Relax Fix	-0.03600000	-1.2107897	1.1387897	0.9999991
VNS Reg-Relax Fix	0.06055556	-1.1142342	1.2353453	0.9999875
VNS Reg-VNS Dem	0.09655556	-1.0782342	1.2713453	0.9998742

Table 6: The Tukey multiple comparison test for pairwise time averages among the different methods

The Tukey multiple comparison of means allows us to state that the differences between the times of the GRASP method based on regret are clearly significatives with respect to the other methods since the p-values are the smallest ones.

6 Final conclusions

The main contribution of this work is the application of the GRASP heuristic for the single-source transport problem based on the maximum regret criterion or the maximum demand criterion. The computational experiment showed that *the GRASP method based on the maximum regret criterion obtains the best results*, even in problems considered as NP-hard within reasonable times. From the results in the comparison of the methods considered we can conclude that the worst option is the VNS method based on maximum demand in terms of quality, taking into account the percentage gap of variation with respect to the optimal value.

As for the bounds, the relaxed problem and the Relax-Fix get quick solutions, providing better solution in terms of quality the Relax-Fix. For quick solutions with heuristic methods, the VNS method based on maximum regret and the GRASP method based on the maximum demand criteria are also fast methods.

It is interesting to highlight the efficiency of the heuristic procedures that were used, both GRASP and VNS, based on maximum regret against the maximum demand criteria. The results favorably compare the GRASP method based on the maximum regret criteria in terms of cpu times and quality of the solution.

References

[1] A. De Maio and C. Roveda, *An All Zero-One Algorithm for a Certain Class of Transportation Problems*, Operations Research, **19**(1971), 1406-1418.

[2] L. Deroussi, *Metaheuristics for Logistics*, Wiley-ISTE, London, 2016.

[3] A. Díaz, F. Glover , H.M.Ghaziri, J.L. González, M. Laguna, P. Moscato and F.T. Tseng, *Optimización Heurística y Redes Neuronales*, Paraninfo, Madrid, 1996.

[4] A. García Pérez, *Métodos avanzados de estadística aplicada. Métodos robustos y de remuestreo*, UNED, Madrid, 2012.

[5] C. Guéret, C. Prins and M. Sevaux, *Applications of Optimization with Xpress-MP*, Eyrolles, Paris, 2002.

[6] P. Hansen, N. Mladenovic and J. A. M.Pérez, *Búsqueda de entorno variable*, Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial, **7**(2003), 77-92.

[7] H.R. Lourenço, *Heurísticas adaptativas para el problema de asignación generalizada*, Proceedings of The First Spanish Congress in Evolutive and Bioinspired Algorithms, Mérida (Spain), 2002, pp. 267-275.

[8] S. Martello and P. Toth, *Generalized assignment problems*, Proceedings on the Third International Symposium on Algorithms and Computation, London, 1992, pp. 16-18.

[9] R. Martí, *Procedimientos metaheurísticos en optimización combinatoria*, Matemátiques, **1**(2003), 3-62. Valencia (Spain).

[10] J.A Moreno, *Búsqueda por Entornos Variables para Planificación Logística*, XII Congreso Español de Metaheurísticas Algoritmos Evolutivos y Bioinspirados & XII Metaheuristic International Conference, Barcelona, 2017.

[11] R.V. Nagelhout and G.L. Thompson, *A single source transportation algorithm*, Computers & OR, **7**(1980), 185-198.

[12] T. Öncan, *A Survey of the Generalized Assingnment Problem and Its Applications*, INFOR Journal, **45**(2007), 123-141.

[13] M.G.C. Resende and J.L. González, *GRASP: Procedimientos de Búsqueda miopes aleatorizados y adaptativos*, Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial, **7**(2003), 61-76.

[14] M.G.C. Resende and C.C. Ribeiro, *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*, Springer, Berlin, 2016.

[15] V. Srinivasan and G.L. Thompson, *An Algorithm for Assigning Uses to Sources in a Special Class of Transportation Problems*, Operations Research, **21**(1973), 284-295.

[16] H.A. Taha, *Operations Research*, Pearson, Mexico, 2012.

[17] E. Talbi, *Metaheuristics: From Design to Implementation*, Wiley, USA, 2009.

[18] M.A. Trick, *A linear relaxation heuristic for the generalized assignment problem*, Naval Researh Logistic (NRL), **39**(1992), 137-151.