Hybridization of GRASP algorithm with genetic algorithm.

Christian Perez, Miguel A. Salido
[1]Universitat Politécnica de Valencia
Camí de Vera s/n
Valencia, Spain
cripeber@doctor.upv.es, msalido@dsic.upv.es

Abstract

(Por escribir)

Introduction

(Por Escribir)

Problem specification

This section features the most relevant aspects of the problem specification proposed by the company with the intention of providing a better comprehension of the presented algorithm. The objective of the problem to solve is to minimize the transport cost and the stock cost by assigning each travel to a warehouse, taking into account every characteristic of each travel and warehouse. The minimum cost of transportation is obtained by finding the trip with the minimum value of transportation, which makes this process relatively easy to perform. However, a problem arises when looking for the minimum stock cost due to the fact that there must be a balance between the stock of each type of container. In order to solve this problem, it is needed to stablish a multi-objective function by assigning weights to the stock cost and the transport cost in order to determine the fitness function. The objective of this function is to maintain a stable transport cost while distributing homogeneously the stock between the warehouses. Thus, the most important variables in the problem are taken into account.

- **Transport cost** (TC): Cost asociated with an order and based on the distance between the warehouse and the starting point. This is one of the values that is taken into account in the calculation of the final cost in the fitness function.
- **Stock cost** (SC): If an order is assigned to be loaded at a warehouse where the current stock is less than the stock requested, the negative stock remaining given by this load operation must be replaced. Thus, the cost of the stock is proportional to the product of the cost of replacing each of the items and the negative stock.
- **Delay of delivery** (DD): Each warehouse that can be associated with an order is located in a different place, such that the distance between each warehouse and the starting point is different. Therefore, the delay of delivery value depends on the warehouse chosen by the order. This delay affects how much stock is reduced from the warehouse from the scheduled date to the actual delivery date.
- **Single load point** (SLP): For each order there is a number of possible warehouses where it can be loaded. All the items in an order must be loaded in the same location, creating the constraint that each order can only be loaded at one warehouse.

The problem is formed by a dataset (DS) that is based on the combination of three main elements. This three elements contain all of the information necessary to represent an instance of the problem (see Equation 1). Hence, each problem instance is formed by a group of orders (O) each of which represent a trip to transport items, all of the warehouses that have every item requested by the order (W), and a set of prices (P) for each item.

$$DS = \{O, W, P\} \tag{1}$$

The set $O = \{o_1, o_2, \ldots, o_N\}$ contains all orders, where N is the number of orders. Each order $o_i \in O$ is formed by three parameters $o_i = [D_{o_i}, Ir_{o_i}, W_{are_{o_i}}]$:

- D_{o_i} is the delivery date of o_i.
- Ir_{o_i} is the set of items requested in the order o_i (see Equation 2). Each element $ir_{o_ij} \in Ir_{o_i}$ represents the number of items that have to be loaded.

$$Ir_{o_i} = \{ir_{o_i1}, ir_{o_i2} \ldots, ir_{o_iP}\} \tag{2}$$

- W_{areo_i} is the set of possible warehouses that have every item requested in order o_i (see Equation 3).

$$Ware_{o_i} = \{Ware_{o_i1}, Ware_{o_i2}, \ldots Ware_{o_iQ}\} \tag{3}$$

Each W_{o_ij} is formed by three values:

- ava_{o_ij} represents if the warehouse j is available to load all of the items requested by the order o_i. Therefore, $ava_{o_ij} = 1$ means that the warehouse is available, and $ava_{o_ij} = 0$ means that it is not.

- $pr_{o_i j}$ contains the transport cost of order o_i from warehouse j. If $ava_{o_i j} = 0$, this value is null.
- $dl_{o_i j}$ is the delay to load the order o_i from warehouse j. If $ava_{o_i j} = 0$, this value is null.

At the beginning of each week and in order to control the amount of stock, the current status of each warehouse is provided by the company through the use of a three-dimensional matrix W (see Equation 4).

$$W = \{w_1, w_2, w_3, \dots w_P\} \tag{4}$$

where each $wst_i \in W$ is composed of a bidirectional matrix (item x week days). Each of the logistic centers (warehouses) has the ability to replace the lack of items by shipping them. This allows the company to generate negative stock in the warehouses asociated to an order $wst_{ijk} \in \mathbb{N}$.

Finally, P is the set of unitary price for each items (see Equation 5).

$$P = \{p_1, p_2, p_3, \dots, p_i, \dots p_P\} \tag{5}$$

Solving techniques

This section explains the different algorithms being used in the evaluation of the solving of the problem proposed by the company, pointing out the the final method used, which is an hybridization of the GRASP algorithm and a Genetic Algorithm (GA). The resolution techniques presented are the Greedy algorithm, the GRASP algorithm and the Genetic Algorithm.

The Greedy algorithm

Currently, the purpose of the company is to plan orders each week that are based on the demand of customers, considering the expected available stock for each day. The solution for this purpose is based on a greedy technique. The availability of each item is obtained from a given loading plan that its updated each week. This update consists on the stock remaining from the week before, the stock obtained by shipments, etc. This algorithm is determined by different constraints, imposed by factors such as customers, warehouses and geographical areas:

- There are clients who only allow loading at specific warehouses due to convenience factors and distance.
- The assingments of warehouses to orders have a priority that is dependent on the geographical area or zone of influence.
- The order is restricted by the type of items it contains, thus it can not load at a warehouse that does not have every item requested.
- Theq quantity of products (vegetables, fruits, fishes) is not constant thorughout the year, therefore the warehouses have different delays for each product depending on the time of the year.

The current algorithm developed by the company is composed of two layers. First, the algorithm uses a heuristic called closeness centrality that obtains a measure of centrality of a transport network by adding the distance between the different nodes, obtaining the central nodes (warehouses) as those that have minimum distance to the other nodes. Then, all orders are classified according to the item with the largest part of the order.

In the second layer, if an order does not meet the restrictions of the first layer it is not assigned to a warehouse. These orders are assigned warehouses with larger stock that are less central. The orders that are still unassigned can be reordered using the customers in order to improve the warehouse system or to make up for the lack of stock. However, there may be still unassigned orders which are assigned after an expert human looks for combinations that improve the global transport cost withouth increising stock cost. It is necessary for an expert to review all of the trips in order to obtain a better solution for the company.

GRASP

A GRASP algorithm has been developed taking into account the drawbacks of the Greedy algorithm used by the company, in order to obtain in terms of stock balance and travel cost a balanced solution.

Algorithm 1 GRASP

```
1: input: All orders $O$, All stock matrix $W$, Item price vector $P$, Size of LCR list $n$
2: output: Optimized solution $s^*$
3: $i \longleftarrow 0$
4: $t \longleftarrow [1, 2, 3, \dots, |O|]$
5: $LCRordering(\alpha)$
6:
7: $s \longleftarrow [\ \ ]$
8: $LCR \longleftarrow [o_1, \dots, o_n]$
9: while $|s| \neq |LCR|$ do:
10:     $j \longleftarrow 0$
11:     $lcr \longleftarrow [\ \ ]$
12:     while $j < |LCR|$ do:
13:         $lcr \longleftarrow lcr \cup$ LocalSearch$(LCR_j, \alpha, |LCR|)$
14:         $j \longleftarrow j + 1$
15:     end while
16:     $s \longleftarrow s \cup lcr*$
17:     $LCR \longleftarrow LCR \setminus lcr*$
18:     $LCR \longleftarrow LCR \cup o_{n+1}$
19: end while
```

The GRASP Algorithm 1 is composed of two parts. First, the set of orders is sorted with purpose of improving the efficiency of the algorithm (line 5). This type of sorting is based on the idea of constrainedness, meaning giving the algorithm more decision power in the last iterations by analyzing the most restrictive orders in the beginning. Second, in the iterative part of GRASP (from line 9 to 19), a list (called LCR) that contains the candidates to assign a warehouse given by the algorithm is obtained. Later, through the use of local search the best warehouse stablished. When the best warehouses for all the trips are known, the warehouse with the lowest cost is selected, deleting the order from the LCR list and adding the next one.

LRC ordering: In order to provide more flexibility and to test the algorithm with different data inputs, the LCR list is sorted. This ordering consists in different steps. First as a pre-processing, every order that only has one available warehouse due to restrictions and availability is automatically allocated. Second, the order of the list is based on the number of possible warehouses, positioning in the first order the trips with fewer possible warehouses, allowing the algorithm to employ less decisions in this assignations. The trips with the same number of warehouses are sorted by the number of items requested by each trip, positioning the ones with fewer items first.

Local search

With the LCR list sorted, a local search is executed with the intention of obtaining the best assignation. This is performed by obtaining the objetive function for each warehouse in a trip. This function consists on the calculation of the transport cost and the stock cost, giving more weight to the transport cost by means of a parameter called alpha, because the transport generates more costs.

Algorithm 2 Local Search

```
1: *input:* A order $o$, List of item price $P$, Alpha value for objective function $\alpha$, Length of LCR list $nLCR$
2: *output:* Index of warehouse chosen $j$, Minimum cost of warehouse chosen $Q*$
3: $Q \longleftarrow [\quad]$
4: $I \longleftarrow Ir_o$
5: $M \longleftarrow W_o$
6: **For** $j$ **in** $\{0, \ldots, |M|\}$:
7:     $Cs \longleftarrow []$
8:     $Ct \longleftarrow$ **getTransportCost**$(M_j)$
9:     **For** $i$ **in** $\{0, \ldots, |I|\}$:
10:        $Cs_i \longleftarrow P_i * \sum_{k=D_o}^{|wst_{ij}|} wst_{ijk} \mid wst_{ijk} < 0$
11:        $Cs \longleftarrow Cs \cup Cs_i$
12:    **end for**
13:    $Q_j \longleftarrow \alpha * Ct + (1-\alpha) * \sum Cs$
14:    $Q \longleftarrow Q \cup Q_j$
15:    $F_\alpha \longleftarrow (1-\alpha)/nLCR$
16:    **if** $\sum Cs < 0$ **then**:
17:        $\alpha \longleftarrow \alpha + F_\alpha$
18:    **else:**
19:        $\alpha \longleftarrow \alpha - F_\alpha$
20:    **end if**
21: **end for**
```

The transport cost is calculated by the addition of the price of traveling to a warehouse for each order. The stock cost is calculated by the subtraction of the quantity of an item required by a warehouse at a certain time, and then multiplying the cost of manufacturing this item if the quantity remaining is negative. If the stock is positive, the cost will be zero. Finally the warehouse that provides a better value of the obtjective function is chosen.

Improvements

There have appeared several problems regarding the optimization of the solution during the developement of the algorithm, in order to resolve them, some sections of the code have been imiproved:

- **LCR sorting:** ADesgranstics, there have been a sorting of the LCR list, leaving the orders with more solutions at the end of this list.
- **Shuffle:** It has been obeserved that in each one of the iterations of the GRASP algorithm, the same order of trips in the LCR list is generated. By using the same criteria employed in the LCR ordering, a shuffle method that randomizes the LCR list has been developed. This shuffle allows to avoid identical solutions, by generating different LCR lists.
- **Scheduler:** The objective function applies a value alpha to the transport cost and the stock cost with the intention of finding the optimal point. During the execution of the tests there is a polarization of the objective function due to the amount of items with negative stock in the warehouses, losing the optimal value. The reason behind this is that a fixed alpha value worsens the results by losing the dynamism of the algorithm. In order to solve this problem, a *scheduler* has been developed, which changes the alpha value in each iteration. The alpha value decreases or increases whether the order has been assigned a warehouse with negative or positive stock. The factor used is obtained by dividing the rest of alpha by the number of orders that are not assigned yet. Through this process the stock cost is minimized in an easier way by the *scheduler*.

Genetic Algorithm

In order to improve the solutions given by the GRASP algorithm previously explained, a Genetic Algorithm (GA) is developed. The input of this algorithm is the solutions provided by the GRASP, allowing an hybridization of both algorithms through this process. This hybridization allows an improvement of the solutions that have converged in the GRASP algorithm.

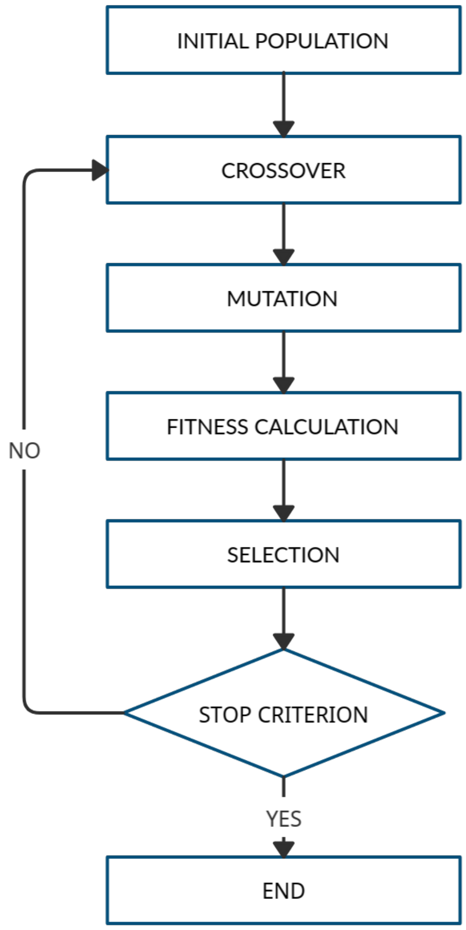

Figure 1: Genetic algorithm structure.

This algorithm is composed of different parts, all presented in the Figure 1. Each of this parts, except the initial population, are repeated until the stop criterion is satisfied. This stop criterion consists on whether the algorithm has performed an stablished number of iterations.

Initial population: The complete set of initial solutions or initial population is composed of two main groups. First, 60% of the initial population are final solutions from the previously implemented GRASP algorithm, with the intention of stacking the GRASP algorithm and the Genetic algorithm (GA) in order to obtain better results. Second, the rest of the initial population (40%) are randomized solutions.

These random solutions are composed of the same orders of the proposed problem but assigning each order a random warehouse. However, this warehouse must be one of the possible platforms to which the order can be assigned, thus complying with the limitations of the order.

Before starting the iterating part of the algorithm, the fitness of each solution is calculated, in order to execute the crossover part.

Crossover: Once the fitness of each solution from the population is obtained, the list of all solutions is sorted in ascending order, placing the solutions with better fitness at the first positions. A list composed of parent pairs is then formed by selecting the two best solutions that have not yet been assigned in order to form each parent pair.

The objetive of the crossover or reproduction is to obtain a number of new solutions (or children) from each parent pair. To achieve this goal, each child is formed by a random fraction of the orders that compose one of the parents, and the rest of the orders are taken from the remaining parent. The number of childs that are breeded is variable, and it is passed to the algorithm as a paramater. The result of this process is a list that contains sets of solutions called families, each one formed by a parent pair and a number of children.

Mutation: In order to avoid local minima during the finding of solutions that minimize the fitness function, the process of mutation is performed. This step of the algorithm is composed of two parts, and consists of randomly changing the assigned warehouses of a set of orders in a solution.

First, a random value is obtained for each member of a family.Then, this value is compared to a parameter k in order to determine whether this member will mutate or not. Second, for each solution or member selected for mutating, 1% of its orders are obtained, to which one of the possible warehouses will be assigned randomly.

Fitness calculation:

Selection:

Evaluation

Conclusions and future work

Transport companies need to optimize their logistics infrastructures and strategies in order to be more efficient in a more competitive world. This work tries to merge two different problems, the warehouse stock management problem and the routing problem in order to minimize both the negative stock and the transport cost. To do this, a GRASP-based metaheuristic has been developed to improve the greedy algorithm that is currently being used by the company. The results in several case studies show that the greedy algorithm had a better behavior in the transport cost, since it is specially guided by a heuristic. However, the proposed GRASP algorithm overcomes the results obtained by the greedy algorithm in stock balancing, improving the negative average stock by up to 82%.

In future works, we will improve the proposed algorithm by combining the proposed GRASP algorithm with a genetic algorithm (GA). Thus, the solutions obtained by the GRASP algorithm will participate as a subset of the initial population for the genetic algorithm. This could improve the quality of the solutions due to the high capability of the GA to combine previous solutions and avoid blockage in the local optimal.

Acknowledgements

The paper has been partially supported by the Spanish research project TIN2016-80856-R.

References