Tarea 8
Experimentos con transiciones de fase

Pamela Jocelyn Palomo Martínez

Nota: La tarea 8 la hice sobre un algoritmo que estoy trabajando para un artículo. El análisis está escrito en inglés para evitar la tarea de escribirlo dos veces. También incluyo algunas partes del artículo que está en proceso de escritura para dejar en claro cuál es el problema y el algoritmo que trabajó. Aunque en realidad la parte relativa a la tarea es la sección 4.1. No incluyo código en el repositorio porque este es un trabajo en proceso.

1 Introduction

We study a logistic problem arising from a real-life situation faced by a local company. This company has a team of technicians located in a depot to serve several branch offices. In a working day, the team has to carry out activities in some branch offices and each activity requires certain materials in order to be performed. Due to the company constraints, usually the existing stock is not large enough to perform all the scheduled activities. As a consequence, the team has to purchase the required materials and perform the activities in the same day. The company wishes to minimize the transportation and purchasing costs and also wants the activities to be performed as soon as possible.

This problem is modeled as a generalization of the well-known Traveling Purchaser Problem (TPP). In the TPP there is a set of markets that offer different products. Each product can be purchased in more than one market, but the offer and the price vary. A demand of products is given and the objective is to design a route that visits some markets in order to satisfy the demand, meanwhile the sum of purchasing and traveling costs is minimized.

In the problem addressed in this work the demand is given by a set of customers (branch offices). There is a stock kept in a depot, but it is not sufficient to satisfy the demand; in consequence, some of the products must be purchased in the markets and then, delivered to the customers. The objective is to design a route in which all the customers demands are satisfied and both the total cost (traveling plus purchasing costs) and the waiting time of the customers are minimized, simultaneously. This problem is called the bi-objective Traveling Purchaser Problem with Deliveries (2-TPPD). Note that all customers must be served, but only a subset of markets have to be included in the route, this means that the 2-TPPD involves both selecting and routing decisions. To the best of our knowledge, this problem has never been addressed before.

2 Problem description and mathematical model

Let C be the set of customers and P_i the set of products required by customer $i \in C$. Specifically, d_{pi} units of product p are demanded by customer i. To simplify notation, the set $\bigcup_{i \in C} P_i$ is denoted as P. A quantity of s_p units of product p are stored at depot, but it is assumed that it is not sufficient to satisfy the demand. Then, for every product p, there is a set of markets M_p in which it can be purchased. Each market $i \in M_p$ offers q_{pi} units of product p at an unitary cost c_{pi}. From now, we will consider that $M = \bigcup_{p \in P} M_p$.

In the 2-TPPD a complete graph $G = (N, A)$ is given, where $N = \{0\} \cup C \cup M \cup \{n+1\}$ is the node set and A, the arc set. Nodes 0 and $n+1$ are the same depot, where $n = |C| + |M|$. The cost and the travel time for every arc $(i, j) \in A$ are denoted as d_{ij} and t_{ij}, respectively. Besides, the service time for each location $i \in N$ is denoted as a_i.

The objective is to design a route that minimizes the travel and purchasing cost and the customers waiting time, subject to the following constraints:

- the route starts at node 0 and ends at node $n+1$;
- the customers demand is satisfied; and
- the quantity of product purchased in a market cannot be larger than its offer.

The following decision variables are used to model the 2-TPPD .

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i,j) \text{ is traversed}; (i,j) \in A \\ 0 & \text{otherwise;} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if location } i \text{ is visited}; i \in N \\ 0 & \text{otherwise;} \end{cases}$$

$$u_{ij} = \begin{cases} 1 & \text{if location } i \text{ is visited before customer } j; i \in N, j \in C, i \neq j \\ 0 & \text{otherwise;} \end{cases}$$

v_i arrival time at location $i; i \in N$;

w_{pi} quantity of product p purchased in market $i; p \in P, i \in M_p$.

Then, the 2-TPPD is modeled as follows:

$$\text{minimize } z_1 = \sum_{(i,j)\in A} d_{ij}x_{ij} + \sum_{p\in P}\sum_{i\in M_p} c_{pi}w_{pi} \tag{1}$$

$$\text{minimize } z_2 = \sum_{i\in C} v_i \tag{2}$$

subject to:

$$\sum_{i\in N:(0,i)\in A} x_{0i} = 1 \tag{3}$$

$$\sum_{i\in N:(i,n+1)\in A} x_{in+1} = 1 \tag{4}$$

$$\sum_{j\in N:(j,i)\in A} x_{ji} = y_i \qquad i \in N\backslash\{0\} \tag{5}$$

$$\sum_{j\in N:(i,j)\in A} x_{ij} = y_i \qquad i \in N\backslash\{n+1\} \tag{6}$$

$$y_i = 1 \qquad i \in C \tag{7}$$

$$v_i + a_i + t_{ij} \leq v_j + T(1 - x_{ij}) \qquad (i,j) \in A \tag{8}$$

$$u_{ij} \leq y_i \qquad i \in N, j \in C \tag{9}$$

$$T(u_{ij} - 1) \leq v_j - v_i \leq T u_{ij} \qquad i \in N\backslash\{n+1\}, j \in C \tag{10}$$

$$w_{pi} \leq q_{pi} y_i \qquad p \in P, i \in M_p \tag{11}$$

$$s_p + \sum_{j\in M_p} w_{pj}u_{ji} - \sum_{j\in C\backslash\{i\}} d_{pj}u_{ji} \geq d_{pi} \qquad i \in C, p \in P_i \tag{12}$$

$$v_i \geq 0 \qquad i \in N \tag{13}$$

$$w_{pi} \geq 0, \text{integer} \qquad p \in P, i \in M_p \tag{14}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{15}$$

$$y_i \in \{0,1\} \qquad i \in N \tag{16}$$

$$u_{ij} \in \{0,1\} \qquad j \in C, i \in N\backslash\{j\} \tag{17}$$

The objective function (1) seeks to minimize the total cost and the objective function (2) minimizes the customers waiting time. Constraints (3) and (4) assure that the route starts and ends at depot, while flow conservation is guaranteed by constraints (5) and (6). Equations (7) ensure that all customers must be served. Time consistency is assured by constraints (8), where T is a sufficiently large constant. Constraints (9) and (10) guarantee that if the arrival time to location i is smaller than arrival time to location j, then i is visited before j. Constraints (11) ensure that the quantity of product that is purchased in a market does not exceed the offer. Demand satisfaction is assured by constraints (12). Finally, constraints (13)–(17) define the domain of the decision variables.

3 Relinked variable neighborhood search

For a proper understanding of our algorithm we now define the merit functions used in the procedure, as well as the concept of dominance. Given a solution X, we consider the merit functions $z_1(X)$ and $z_2(X)$ defined through (1) and (2), respectively. On the other hand, we say that a solution X_i dominates another solution X_j if and only if

$$z_1(X_i) \leq z_1(X_j), \tag{18}$$
$$z_2(X_i) \leq z_2(X_j), \tag{19}$$

and at least one of the inequalities (18) and (19) is strict.

Our RNVS is outlined in Algorithm 1. First, a solution X_0 from the approximated Pareto front $\mathscr{P}$ is selected. In our implementation, we considered three different selection criteria that are discussed in Section 4.1.

After that, we relink three VNS schemes as follows: a VNS procedure tries to minimize z_1 starting from the disturbed solution X_0. The resulting solution is denoted by X_1. Then, another VNS is performed in order to minimize z_2 using X_1 as initial solution. As before, the solution found through this procedure is denoted by X_2. Finally, the cycle is completed with a VNS that minimizes z_1 starting from X_2 which is the last solution found by the previous search. It is noteworthy that different results may be achieved by first minimizing z_2, then z_1, and finally, z_2 again. Therefore, in our procedure, in each iteration we select at random which objective will be minimized first.

The Pareto front $\mathscr{P}$ is updated every time that a new local optimal is found during a VNS execution: a solution is included in $\mathscr{P}$ if it is not dominated by any other solution already included in $\mathscr{P}$. Moreover, when a new solution is included in $\mathscr{P}$, the solutions dominated by this one are removed. The RVNS stops when a cycle is completed and $\mathscr{P}$ is not updated.

The following subsections describe the construction method used to find the initial Pareto front and the subroutines called in each VNS procedure, including the shaking and the local searches.

Algorithm 1 Relinked variable neighborhood descent

Input:
X ▷ Initial solution
$\mathscr{N}_i$ for $i = 1, 2, 3$ ▷ Set of neighborhood structures used to minimize z_i
Set $\mathscr{P} \leftarrow \{X\}$ ▷ Pareto front
Set $\mathscr{P}' \leftarrow \emptyset$
while $\mathscr{P} \neq \mathscr{P}'$ **do** ▷ The process stops when a cycle is completed and $\mathscr{P}$ is not updated
    Set $\mathscr{P}' \leftarrow \mathscr{P}$
    Select a solution X_0 from $\mathscr{P}$
    Randomly set z_1 and z_2 as (1) or (2), respectively
    for $i \leftarrow 1$ **to** $i = 3$ **do**
        $X_i \leftarrow X_{i-1}$
        Set $j \leftarrow 1$
        Variable Neighborhood Search
        repeat
            $X' \leftarrow$Shaking(X_i)
            $X^* \leftarrow$LocalSearch(X', N_j), $N_j \in \mathscr{N}_i$
            $\mathscr{P} \leftarrow$UpdateParetoFront$(\mathscr{P}, X^*)$
            if X^* is better than X_i with respect to z_i **then** ▷ Consider that $z_3 = z_1$
                Set $X_i \leftarrow X^*$
                Set $j \leftarrow 1$
            else
                Set $j \leftarrow j + 1$
            end if
        until $j = |\mathscr{N}_i| + 1$
    end for
end while

3.1 Construction method

The construction of a single solution is made in two stages. The first one builds a solution including only customers and the second one adds markets to this solution.

At the beginning of the first phase, we have a solution containing only the initial and ending locations (depot) and an arbitrary customer. Then, iteratively, the remaining customers are added one by one in the position in which the increment in the sum of the customer waiting times is minimum.

In the second stage, for every customer in the route the algorithm checks if it is possible to satisfy its demand with the visits included in the route so far. If it is not possible, for every product which demand is not satisfied, a random market offering it is routed before the customer in the position in which the increment in the travel cost is minimized. The process stops when it is possible to satisfy the demand of all the customers.

This procedure is repeated $|C|$ times, every time the initial solution includes the depots and the customer $i \in C, i = 1, 2, \ldots, |C|$. The, the initial Pareto front contains all the non-dominated solutions found through out this method.

3.2 Variable neighborhood search

Two VNS approaches are needed in our RVNS, one seeks to minimize the total cost and the other one, the customers waiting time. The shaking procedure described in Section 3.2.1 is used in both VNS algorithms, and the local search algorithms are discussed in Section 3.2.2.

3.2.1 Shaking procedure

The shaking procedure is performed in order to find disperse non-dominated solutions. First, a small percentage of the locations visited in the route is randomly chosen and then relocated at any random position. If it is possible to satisfy the demand with this new order, the shaken solution is kept; otherwise, the perturbation is not accepted.

3.2.2 Local search algorithms for VNS

The local search algorithms used to minimize the total cost and the waiting time are adapted from the literature of the TSP, the TPP, and the minimum latency problem. In order to define the algorithms, we use the concept of block which is a sequence of consecutive visited locations satisfying that either all of them are customers or all of them are markets, and the locations visited before the beginning and after the ending of the sequence are of a different type of the locations of the sequence.

In order to reduce the traveling cost and the total waiting time, we use local search algorithms whose changes in the solution are accepted if the respective objective is reduced. Otherwise, the search continues. Such algorithms are defined as follows:

- *Intra-block relocate (IntraR).* For every block, each visit belonging to it is relocated at a random position of the same block.
- *Intra-block swap (IntraS).* For every block and for every visit of it, a different random location in the same block is selected and their positions are exchanged.
- *Intra-block 2-opt (Intra2opt).* For every block and for every visit included in it, another random location in the same block is selected. then, the classical 2-opt move is performed.
- *Inter-block relocate (InterR).* For every location in the current route, it is checked whether it is a customer or a market. If the location is a customer, it is relocated at a random position of a posterior random block . If it is a market, a random previous block is chosen and the market is relocated at a random position of this block.
- *Market remove (MR).* If this operator is performed for the minimization of the cost, routed markets are removed when the demand can be satisfied by other routed markets and the decrease in the travel cost compensates the increase in the purchasing cost. On the other hand, when it is used to minimize the waiting time, a market is removed if the demand can be satisfied by the remaining markets.

Class	Capacity	Customer service time
1	Capacitated	High
2	Capacitated	Low
3	Uncapacitated	High
4	Uncapacitated	Low

Table 1: Instance classes description

The following algorithm is performed to reduce the total purchasing cost:

- *Market insert (MI).* Non-routed markets are inserted in the route when the decrement in the purchasing cost offsets the increment in the travel cost. If after an insertion, there are some markets in which no products are purchased, they are removed.

4 Computational experiments

Our RVNS was coded in C++ and compiled in GNU on a 2.1 GHz Intel Xeon(R) CPU E5-2620 v2 under Ubuntu 16.04 operating system. The RVNS was tested over a set of 64 artificial instances of the problem divided into four classes. Each class contains the same graphs, demands, and service times. The graph sizes vary between 50 and 200 nodes (depot, markets, and customers), and the number of products goes from 50 to 200. The set of instances contains both capacitated and uncapacitated cases, meaning that in some cases if a market offers a product, it is able to fully satisfy the demand of such product (uncapacitated or unrestricted), and, in some others, the offer of a product cannot be fulfilled by any market (capacitated or restricted). Also, we considered both high and low customer service times. Low service times resemble the cases in which only deliveries are performed; on the other hand, high service times are set to simulate cases in which activities are carried out in customers locations. Table 1 summarizes the characteristics of each instance class.

4.1 Selection criteria

As mentioned in Section 3, we tested three different criteria in order to select a non-dominated solution as starting point for a cycle of our RVNS. The first one selects the solution that has been part the front for the longest number of iterations. The second one selects the solution for which (20) is maximum, where $ED(X, \bar{X})$ represents the Euclidean distance between solutions X and $\bar{X}$ in the objectives space. Since the configuration of our instances may lead to large costs and waiting times, the Euclidean distances are calculated over the points in logarithm base 2 scale. Finally, the third criterion chooses a random solution.

$$X' = \arg\max_{X \in \mathscr{P}} \left\{ \min_{\bar{X} \in \mathscr{P} \setminus \{X\}} \left\{ ED(X, \bar{X}) \right\} \right\}. \tag{20}$$

From now, the RVNS that uses the first, second, or third criteria, will be called RVNS-LNI, RVNS-MD, or RVNS-R, respectively.

In order to test the efficiency of each criterion, we ran the algorithms ten times per instance and evaluated the results considering three different metrics used in the literature to asses the quality of multi-objective algorithms:

- *Number of points in the Pareto front:* It is important to find fronts with a large number of points, since the decision maker has a wide range of options to make the final decision.

- *k-distance:* This metric, proposed by Zitzler et al. [1], evaluates the dispersion of the non-dominated solutions. For each point it measures the Euclidean distance to the k-th nearest point (we considered a logarithm base 2 scale). Notice that this metric is defined for each single point individually, then we considered the average k-distance to evaluate the whole front. In our experiments, we detected that the Pareto front with the least number of points contains only three solutions; therefore, we set $k = 2$ in order to be able to compute this metric for every front.

Algorithm version	Class	Number of points	K-distance	Hypervolume
RVNS-LNI	1	8.42	**9.59**	15.68
	2	14.04	17.51	11.22
	3	10.80	13.50	5.96
	4	12.48	**9.96**	8.27
RVNS-MD	1	**6.04**	11.03	10.77
	2	**5.47**	**9.30**	**6.79**
	3	**7.17**	**11.28**	**5.50**
	4	**7.17**	11.28	**5.30**
RVNS-R	1	7.96	15.43	**8.52**
	2	10.23	14.18	10.67
	3	10.23	13.37	10.67
	4	10.11	13.33	10.76

Table 2: Comparison among the average percentage gaps obtained by using different versions of RVNS

- *Hypervolume:* The hypervolume metric, also known as *size of space covered*, was proposed by Zitzler and Thiele [2]. In the 2-dimensional case, this metric measures the area of the dominated portion of the rectangle $(0, 0)$ and $(\max\{z_1\}, \max\{z_2\})$. A largest hypervolume corresponds to a better front. Notice that, the calculation of $(\max\{z_1\}, \max\{z_2\})$ is not trivial for the 2-TPPD; then, for each instance we set $\max\{z_1\}$ to the largest cost found during the experiments (in 30 runs, 10 per selection criterion). Analogously, $\max\{z_2\}$ was set to the largest waiting time found through the algorithms execution. As before, we calculated the areas considering the points in a logarithm base 2 scale.

For each instance we computed the average number of non-dominated solutions found in 10 runs of RVNS-LNI, RVNS-MD, and RVNS-R. Then, we found the best of those averages and calculated the percentage gap of each reported average with respect to the best one. We proceeded similarly for the k-distance and for the hypervolume. Table 2 shows the average gap for each instance class and for each criterion. Numbers in bold indicate the best average gap per instance class and per performance metric.

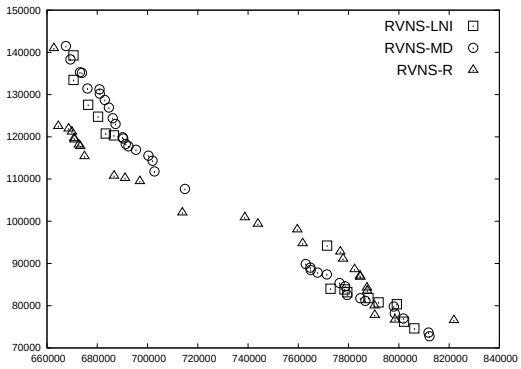
It is worth noting that in almost all cases, the best results were achieved by RVNS-MD, except for the k-distance in Classes 1 and 4 in which the best gaps were reported by RVNS-LNI, and for the hypervolume in Class 1 in which the best result was obtained by RVNS-R. Nonetheless, the differences between the best results and the ones obtained by RVNS-MD in these cases are not significant (1.44%, 1.32%, and 2.51%). Therefore, we conclude that, in our instances, RVNS-MD retrieves the best evaluated fronts, meaning that it is better to select the most disperse solution in the Pareto front to start a new cycle of the RVNS.

A visual example of the previously discussed results is given in Figure 1 in which the fronts obtained by means of RVNS-LNI, RVNS-MD, and RVND-R for an instance with 44 markets, 5 customers, and 50 products are shown. Figures 1a-1b represent the approximated Pareto fronts for the described instance belonging to Class 1 to 4, respectively; meaning that the graph and the demands are the same, but the customer service times and the markets capacity vary.

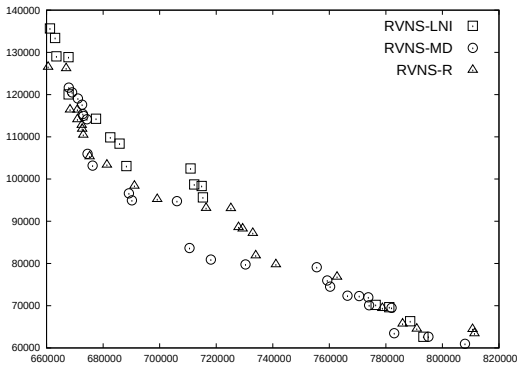
References

[1] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical report, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

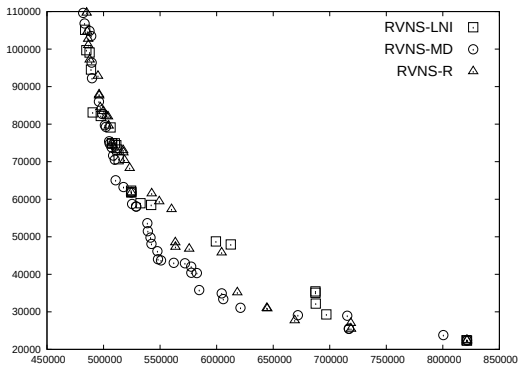
[2] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Trans. Evol. Comp*, 3(4):257–271, 1999.



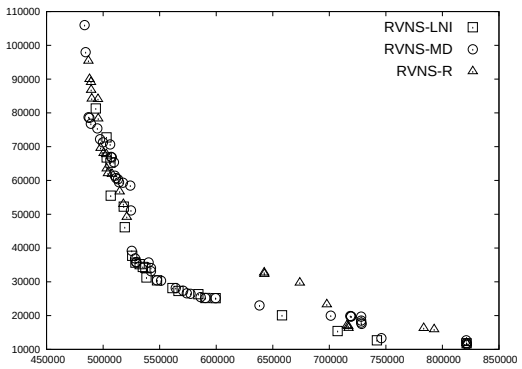
(a) Class 1



(b) Class 2



(c) Class 3



(d) Class 4

Figure 1: Approximated Pareto fronts for instances from different classes