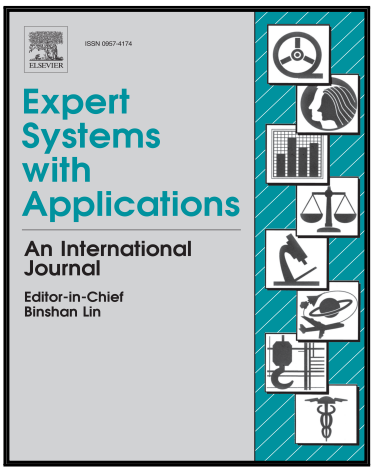
Accepted Manuscript



Genetic Local Search Algorithm for a New Bi-Objective Arc Routing Problem with Profit Collection and Dispersion of Vehicles

Guilherme Dhein, Olinto César Bassi de Araújo,
Ghendy Cardoso Junior

PII: S0957-4174(17)30654-1
DOI: 10.1016/j.eswa.2017.09.050
Reference: ESWA 11571

To appear in: *Expert Systems With Applications*

Received date: 5 January 2017
Revised date: 21 September 2017
Accepted date: 22 September 2017

Please cite this article as: Guilherme Dhein, Olinto César Bassi de Araújo, Ghendy Cardoso Junior, Genetic Local Search Algorithm for a New Bi-Objective Arc Routing Problem with Profit Collection and Dispersion of Vehicles, *Expert Systems With Applications* (2017), doi: 10.1016/j.eswa.2017.09.050

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Highlights

- The proposed problem has two objectives: profit collection and vehicles dispersion
- Vehicles collect rewards for arcs traversed while traveling scattered in environment
- We propose a Multi-objective Genetic Local Search method to find approximation sets
- It uses specialized chromosome, genetic operators and local search strategy
- Our MOGLS presents better approximation sets than a NSGA II implementation

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Genetic Local Search Algorithm for a New Bi-Objective Arc Routing Problem with Profit Collection and Dispersion of Vehicles

Guilherme Dhein[a,*], Olinto César Bassi de Araújo[b], Ghendy Cardoso Junior[a]

[a] *Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Maria, Santa Maria, RS, 97105-900, Brazil*
[b] *Colégio Técnico Industrial de Santa Maria, Universidade Federal de Santa Maria, Santa Maria, RS, 97105-900, Brazil*

Abstract

We present a new bi-objective arc routing problem in which routes must be constructed in order to maximize collected profit and a non linear dispersion metric. A dispersion metric calculated based on instantaneous positions, suitable to capture routing characteristics found when vehicles have to travel in hostile environments, is a novelty in the routing literature. The inherent combinatorial nature of this problem makes it difficult to solve using exact methods. We propose a Multi-objective Genetic Local Search Algorithm to solve the problem and compare the results with those obtained by a well known multi-objective evolutionary algorithm. Computational experiments were performed on a new set of benchmark instances, and the results evidence that local search plays an important role in providing good approximation sets. The proposed method can be adapted to other multi-objective problems in which the exploitation provided by local search may improve the evolutionary procedures usually adopted.

Keywords: bi-objective problem, profit, dispersion metric, genetic algorithm, local search, synchronized routes

*Corresponding author
Email addresses: gdhein@redes.ufsm.br (Guilherme Dhein), olinto@ctism.ufsm.br (Olinto César Bassi de Araújo), ghendy@ufsm.br (Ghendy Cardoso Junior)

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

1. Introduction

Arc routing is an operational research area that addresses several problems of transport management. Traditionally, human planners tackle these problems, but over the past decades a considerable effort has been put into developing computer systems to assist or even replace those planners. There are numerous practical issues to accomplish this goal, especially as it relates to difficulty in dealing with real complexity in algorithmic solutions. We propose a bi-objective arc routing problem that considers both the total collected profit and a nonlinear dispersion metric. To overcome the difficulties imposed by the characteristics of the problem, we resorted to an intelligent method capable of dealing with the trade-off between two conflicting objectives. It is based on the hybridization of a genetic algorithm with a local search procedure.

The work in (Malandraki and Daskin, 1993) is usually referred to as the first proposal of an arc routing problem with profit or prize collection. Before this work, arc routing problems focused on costs reduction, while benefit (traversing all edges or arcs) was fixed. Profit objective inverts this logic. Benefit becomes the objective function to be maximized, while resource or costs limitations are fixed. We are particularly interested in multi-vehicles variations of problems with profits, as those in (Feillet et al., 2005), (Archetti et al., 2010) and (Zachariadis and Kiranoudis, 2011). A summary of arc routing problems with profit is presented in (Black et al., 2013), along with a time dependent variation of the problem.

Our second objective is conceived in order to keep the vehicles away from each other, and to the best of our knowledge, it is new in routing problems literature. The metric proposed to calculate the dispersion is described in details in section 2.

Arc routing problems are associated to a vast array of real problems, such as waste collection, street cleaning, mail and newspapers delivery, meter reading and road network surveillance. Details about those and other applications can be found in surveys (Eiselt et al., 1995a,b; Assad and Golden, 1995) or in book (Corberán and Laporte, 2015).

Our bi-objective approach takes into account two other applications of arc routing. The first one is humanitarian aid distribution in disaster or conflict areas (Sharif and Salari, 2015). In this case, the dispersion is directly related to efficacy. As pointed by Vitoriano et al. (2009), traveling in convoys increases the safety of the cargo, but scattering the vehicles increases the ro-

ACCEPTED MANUSCRIPT

bustness of the delivering system. As some routes may be blocked because of severe damage to transport infrastructure (in disaster areas) or hostile actions (in conflict areas), the probability that most vehicles will finish their routes is increased when they travel apart. Also, it is easier for single vehicles to travel unnoticed. In this scenario, the profit objective is related to efficiency, since it can be used to direct the routes to the most affected sites.

The second application considered is surveillance, especially the problem of planning routes for police or security guards patrol (Shafahi and Haghani, 2015)(Willemse and Joubert, 2012). Such problems can benefit from both profit and dispersion objectives. Profit can be used to increase the importance of some streets or areas, according to their crime incidence or other characteristics (vicinity of banks, jewelry stores and railroad stations, for example). Also, if we consider that a patrol watching over a certain area is able to serve occurrences close to its location, dispersion is important to increase the total area covered by multiple patrols.

The search for possible nuclear threats, as described in (Hochbaum et al., 2014), is also an interesting surveillance application. In this problem, a set of trucks equipped with highly sensitive equipment is used to patrol an area and detect unusual radiation. In such application, profit can be used to favor the search in more suspicious or more populated sites. Dispersion reduces the chance of intersection among the areas covered by two trucks, thus increasing the total area covered. Also, vehicles traveling apart would reduce the chance of being perceived as an unusual behavior.

A dispersion objective imposes synchronization among vehicles, since it is related to their relative distances. Arc routing problems with synchronization characteristics occur in applications like waste collection, in cases when waste collected by smaller vehicles is transferred to larger ones during trips (Del Pia and Filippi, 2006; Rosa et al., 2002), or road marking, when painting vehicles meet tank trucks to refill (Amaya et al., 2007, 2010; Salazar-Aguilar et al., 2013). Those problems present synchronization on meeting points. There are also problems in which arcs must be traversed by two or more vehicles concomitantly, as in routes involving nonautonomous vehicles (like trailers) that must be pulled by autonomous ones (Drexl, 2013), or in snow plowing operations on streets with multiple lanes (Salazar-Aguilar et al., 2012).

It is interesting to note that routing problems with synchronization usually have the interdependence among routes imposed by constraints, like ordered (Morais et al., 2014) and/or simultaneous (Haddadene et al., 2016) visits to nodes or the already referred concomitant traversal of edges. In our

ACCEPTED MANUSCRIPT

problem, the dispersion is enforced by an objective function direction.

Drexl (2012) presents a discussion about synchronized routing. Despite focusing on vertices routing, the difficulties he reports can also be associated to arc routing problems. Any kind of space-time synchronization results in some degree of interdependence among routes. Thus, even small changes in a route may cause quality degeneration or infeasibility to other routes.

Our dispersion objective does not imply an absolute synchronization, with vehicles present concomitantly at the same node or arc, thus it will never be a cause of infeasibility. But synchronization is present during the entire solution, and it is verified among all vehicles. Spatial and temporal interdependence is amplified, and the difficulty is still perceived.

The research about multi-objective problems resulted in methods such as PAES (Knowles and Corne, 2000), SPEA2 (Zitzler et al., 2002), NSGA II (Deb et al., 2002), MOEA/D (Zhang and Li, 2007) and AbYSS (Nebro et al., 2008). Those methods are all population-based metaheuristics, and this characteristic is appropriate since multi-objective solutions are approximations sets. There are also multi-objective adaptations of trajectory methods like Tabu Search (Jaffrès-Runser et al., 2008), Simulated Annealing (Liu et al., 2014) and GRASP (Martí et al., 2015).

Tests focusing on the dispersion objective using a GRASP implementation presented a good exploitation, as the local search generated solutions of satisfactory quality. But the time consumed by each local search execution reduced the number of possible iterations, and this resulted in a poor exploration of the search space. On the other hand, tests with a genetic algorithm resulted in an opposite behavior, as the execution of several evolutionary iterations exhibited exploratory qualities. Some individuals of good quality were generated, and the use of those already evolved solutions as initial points shortened the trajectory of the local search process. This indicated the hybridization of genetic algorithm with local search, like the one present in the method described in (Arroyo and Armentano, 2005), as a promising alternative. Modifications on this method and the adoption of specialized features, a constructive algorithm and a crossover operator, resulted in the Multi-objective Genetic Local Search Algorithm (MOGLS) we implemented to obtain our best results.

Since the problem is new, we used an NSGA II implementation to validate our results. This method is widely used and is established as a good method to generate approximations sets to multi-objective problems. The same specialized constructive algorithm and crossover operator developed

ACCEPTED MANUSCRIPT

for the MOGLS method were incorporated to NSGA II to allow a fair comparison. As those features are non-canonical, the resulting method will be referred to in the remaining of the paper as MOGA (from Multi-objective Genetic Algorithm). Computational experiments disclosed that the hybridization is beneficial, and incorporating local search is fundamental to overcome the difficulties of the problem.

The remaining of the paper is organized as follows. In section 2, we describe the problem in details, especially the dispersion metric. The implemented methods are described in section 3. In section 4, we describe the tests performed and present the results obtained. We close the paper with a short conclusion in section 5.

2. Problem Definition

The routes are constructed over a connected graph $G = (N, A)$, where $N = \{0, 1, \ldots, n\}$ is the set of $n + 1$ nodes and A is a set of arcs connecting nodes in N. Node 0 is the depot, and the location of each node $i \in N$ is given by an ordered pair of coordinates (x_i, y_i). The m identical vehicles in $K = \{1, 2, ..., m\}$ depart from node 0 to perform m open routes. As the vehicles are identical, they have the same velocity magnitude. A cost c_{ij} and a profit p_{ij} are associated with every arc $a_{ij} \in A$, where the cost represents the travel time necessary to completely traverse the respective arc and the profit represents a gain obtained if the arc is included in a route. Profit may be zero in some arcs, but it is always non-negative. There is a predefined maximum length s for the routes, established to represent the work shift of the driving staff. All vehicles depart from the depot at time 0 and collect profits during the entire shift. Usually the shift ends while a vehicle is between two nodes, i.e., the last arc is not completed. In such cases, the collected profit is proportional to the length already traversed when time s is reached. Finally, an arc may be traversed several times in a solution, but its profit will be collected only once.

2.1. Dispersion Objective Calculation

We defined a dispersion metric based on a continuous tracking of the instantaneous distances between pairs of vehicles, and the dispersion objective function is conceived to favor solutions in which vehicles travel apart from each other.

When a vehicle $k \in K$ travels along an arc a_{ij}, its velocity vector $\vec{\nu_k} = \langle \nu_{xk}, \nu_{yk} \rangle$ has the same direction as the vector $\langle x_j - x_i, y_j - y_i \rangle$, and the displacement along a_{ij} can be represented by the following equation, parameterized by time t,

$$X_k(t) = x_i + t\ \nu_{xk} \quad , \quad Y_k(t) = y_i + t\ \nu_{yk}, \quad t \in [0, c_{ij}] \tag{1}$$

So, if we consider a pair of vehicles k and k' traveling along arcs a_{ij} and $a_{i'j'}$, as depicted in Figure 1, the instantaneous Euclidean distance between them at time t is calculated as follows:

$$d_{kk'}(t) = \sqrt{[X_k(t) - X_{k'}(t)]^2 + [Y_k(t) - Y_{k'}(t)]^2} \tag{2}$$

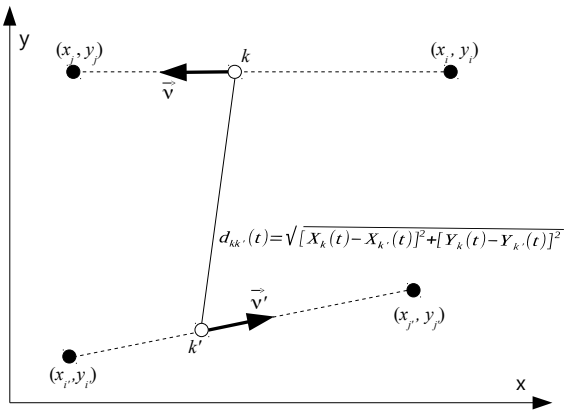


Figure 1: Diagram representing the instantaneous distances between two vehicles.

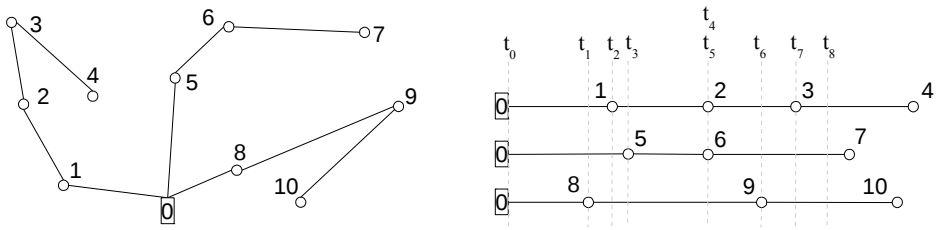
Every time a vehicle changes direction, its velocity vector also changes and the respective displacement equation must be updated. Observe that in a solution with g arcs and m vehicles there are $g-m$ timestamps corresponding to the instants in which at least one vehicle changes direction. It is easy to see that there is a direction change when each arc's traverse is completed, except for those m completed after the shift is finished. There are also two timestamps representing the departure from the depot at time zero and the end of the shift at time s, totalizing $g - m + 2$ timestamps.

We refer to each interval in which the displacement equations remain unchanged for all m vehicles as *time slice*. The ordered sequence of timestamps

ACCEPTED MANUSCRIPT

$\mathcal{S} = \{t_\tau : \tau = 0, \ldots, g - m + 1\}$ defines all time slices, numbered from 1 to $g - m + 1$, in which each one is delimited by two consecutive timestamps $[t_{\tau-1}, t_\tau]$ with a length $T_\tau = t_\tau - t_{\tau-1}$.

An example that illustrates time slices for a solution with three vehicles and 10 arcs is depicted in Figure 2, where timestamps t_4 and t_5 define a time slice with length zero. Timestamp t_0 represents the departure from the depot and t_8 represents the end of the shift at time s.



(a) Geographic representation (b) Temporal representation

Figure 2: Routes and the respective time slices. The diagrams in (a) and (b) present the same routes depicted according to two different representation, geographic and temporal, respectively. In (b), the vertical dashed lines identify the time slices.

If we consider an individual time slice τ, the displacement of a vehicle k traversing an arc a_{ij} during the time slice interval $[t_{\tau-1}, t_\tau]$ is given by

$$X_k(t) = X_k(t_{\tau-1}) + t\,\nu_{xk\tau}\ ,\ Y_k(t) = Y_k(t_{\tau-1}) + t\,\nu_{yk\tau}\ ,\ t \in [t_{\tau-1}, t_\tau] \quad (3)$$

where $\langle \nu_{xk\tau}, \nu_{yk\tau} \rangle$ represents the velocity vector of vehicle k, with direction $\langle x_j - x_i, y_j - y_i \rangle$, and $(X_k(t_{\tau-1}), Y_k(t_{\tau-1}))$ is the vehicle location at the beginning of the time slice.

Based on instantaneous positions of all vehicles, we can calculate the instantaneous Euclidean distance $d_{kk'}(t)$ between each pair k and k'. If we integrate $d_{kk'}(t)$ over the length T_τ of the slice τ, we have the local dispersion between those vehicles within the slice. The dispersion of a slice is given by the pair of vehicles with smallest dispersion, as stated in Equation (4).

$$c_\tau = \min_{\substack{k\,=\,1,\ldots,m \\ k'\,=\,k+1,\ldots,m}} \int_0^{T_\tau} d_{kk'}(t)dt \quad (4)$$

ACCEPTED MANUSCRIPT

Equation (5) represents the total dispersion of a solution, given by the sum of dispersions of all time slices. This is the second objective function to be maximized using the Pareto domination relations.

$$C = \sum_{\tau=1}^{g-m+1} c_\tau \tag{5}$$

The integral in Equation (4) is a definite integral of square root of a quadratic polynomial and an analytic solution can be found applying trigonometric substitutions.

3. Implemented Methods

We implemented two methods to generate approximation sets. The remaining of this section is devoted to detail them. Section 3.1 describes the MOGA implementation, which provides approximation sets to be compared with those generated by the MOGLS described in section 3.2. The MOGA is essentially an NSGA II implementation, but to allow a fair comparison it incorporates the same specialized procedures to generate initial population and new individuals, developed for our MOGLS method and described in sections 3.3 to 3.5. Finally, the local search procedure used in MOGLS is described in section 3.6.

3.1. Main procedure of the Multi-objective Genetic Algorithm

Algorithm 1 presents the pseudocode of the proposed Multi-objective Genetic Algorithm (MOGA).

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Algorithm 1 Multi-objective genetic algorithm

$curPop \leftarrow InitialPopulation(nRCL, ps_1)$
$fastNondominatedSort(curPop)$
$crowdingDistanceComputation(curPop)$
while $currentTime < maxTime$ **do**
  $newPop \leftarrow makeNewPopulation(curPop, ps_1)$
  $newPop \leftarrow newPop \cup curPop$
  $FastNondominatedSort(newPop)$ comment: F_0 to F_f nondominated frontiers
  $f' \leftarrow f$
  while $|newPop| - |F_{f'}| > ps_1$ **do**
    $newPop \leftarrow newPop \setminus F_{f'}$
    $f' \leftarrow f' - 1$
  end while
  $CrowdingDistanceComputation(newPop)$
  Remove from $newPop$ the $|newPop| - ps_1$ individuals $\in F_{f'}$ with smaller crowding distance
  $curPop \leftarrow newPop$
end while
return the approximation set in frontier F_0

We use the constructive algorithm described in section 3.3 to generate an initial population $curPop$, with ps_1 individuals. This population is then organized and evaluated as in NSGA II (Deb et al., 2002). A fast nondominated sorting procedure, implemented according to the improved algorithm presented in (Jensen, 2003), is applied to rank and to organize the individuals in sets (or frontiers) according to different nondomination levels. Those solutions with lower rank constitute the frontier F_0 of nondominated solutions. Crowding distance is computed for all individuals right after the sorting. Rank and crowding distance are used to define the comparison operator that guides the selection procedure, favoring individuals with lower rank and, when the rank is the same, those with higher crowding distance.

The process enters a loop where the current population is improved. A new population is created with genetic operators, in the procedure described in section 3.4. The new population is combined with $curPop$, resulting in a population $newPop$ with $2ps_1$ individuals.

This population is reduced to ps_1 individuals with the strategy defined in NSGA II, discarding individuals based on their rank and crowding distance values. The reduced population enters a new iteration as the current population, and the process is repeated until the execution time limit is reached.

ACCEPTED MANUSCRIPT

3.2. Main procedure of the Multi-objective Genetic Local Search Algorithm

We also implemented a Multi-objective Genetic Local Search Algorithm (MOGLS) to generate a set of solutions approximating the Pareto frontier of our bi-objective problem. A pseudocode describing the method is presented in Algorithm 2.

Algorithm 2 Multi-objective genetic local search algorithm

```
Archive ← ∅
curPop ← InitialPopulation(nRCL, ps2)
g ← 1
while currentTime < maxTime do
    FastNondominatedSort(curPop)  comment: F0 to Ff nondominated frontiers
    CrowdingDistanceComputation(curPop)
    Archive ← nondominated solutions of (Archive ∪ F0)
    curPop ← MakeNewPopulation(curPop, ps2)
    Elite ← SelectElite(es, Archive)
    if (g mod β = 1) then
        LocalSearch(curPop, Elite)
    else
        curPop ← curPop ∪ Elite
    end if
    g ← g + 1
end while
return the approximation set in Archive
```

The approximation set is kept apart from population in an *Archive* set, which contains all nondominated solutions found during the entire process. The process starts with an empty *Archive* set and a population with ps_2 individuals generated using the constructive algorithm described in section 3.3. Counter g is initialized to 1. Successive populations are iteratively generated in the main loop, limited by a maximum duration time.

Starting each iteration, the population is organized in nondominated frontiers by the fast nondominated sorting procedure. The crowding distance is computed for all individuals right after the sorting. The fast nondominated sorting procedure, the crowding distance calculation, and the comparison operator used in selection are the same used in MOGA as described in section 3.1. *Archive* is then updated with the solutions in frontier F_0.

A new population is generated with the procedure described in section 3.4, and replaces the current one. *Elite* set is created as a subset of *Archive*,

ACCEPTED MANUSCRIPT

with solutions chosen by binary tournament. Between two randomly selected solutions, the one with higher crowding distance in the current approximation frontier is favored.

At each β iterations, a local search procedure is executed in order to improve the frontiers in *Elite* and F_0. The local search method and the ways the new improving solutions are included in population are described in section 3.6. In iterations without local search, *Elite* solutions are directly included in population.

3.3. Initial Population

Algorithm 3 presents the greedy randomized constructive algorithm used to create the initial population. Each iteration of the outer loop provides an individual I, generated as follows.

In order to avoid including the same arc twice, we define the set *Out* to keep record of the arcs not included in the solution under construction. Initially, it contains all arcs in A.

A route starts with an arc from the depot to another node. An arc a_{0i_k} is randomly selected among those in *Out* if there is at least one arc starting at node 0. Otherwise, it is selected among those in A, which means it will be duplicated in the solution. At this point, we do not favor arcs with profit, in order to enable different starting arcs and promote diversity in the initial population. The selected arc is removed from *Out* and included in the solution. The node i_k is included in set LN, whose elements are the last nodes of each partial route, and the estimated arrival time at this node (t_{i_k}) is calculated. This procedure is repeated m times, defining a starting arc for each route.

We identify, among the m partial routes, the one that is lagging behind the others, and we denote the last node of this route as $j' \in LN$. This route is chosen to be the next one to incorporate a new arc selected from a restricted candidate list RCL constructed as follows.

Initially, we consider as candidates those arcs in *Out* starting in j' and with positive profit in order to generate initial solutions with a good collected profit objective. If there is no arc with such characteristics, the candidates are all arcs in A starting in j'. The RCL is defined with $nRCL$ arcs $a_{j'h} \in Candidates$ with highest evaluation according to the minimum distance from node h to all nodes in $LN \setminus \{j'\}$. The goal is to include in the solution an arc among those with a high minimum distance to other routes, which is coherent with the dispersion objective function. An arc $a_{j'l} \in RCL$ is

ACCEPTED MANUSCRIPT

randomly selected to be included in the lagging route. The arc is removed from Out, l replaces j' in LN as the new final node in their partial route, and t_l is calculated.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Algorithm 3 Initial population construction

Input:
$nRCL$ – number of candidates to consider in the random selection
ps – population size

$Pop \leftarrow \emptyset$
while $|Pop| < ps$ **do**
  Create a new individual I
  $Out \leftarrow A$
  $LN \leftarrow \emptyset$
  for k from 1 to m **do**
    $Candidates \leftarrow \{a_{0i} \in Out : \forall i \in N\}$
    if $Candidates = \emptyset$ **then**
      $Candidates \leftarrow \{a_{0i} \in A : \forall i \in N\}$
    end if
    Randomly select an arc $a_{0i_k} \in Candidates$
    $Out \leftarrow Out \setminus \{a_{0i_k}\}$
    Start a new route in individual I inserting arc a_{0i_k}
    $LN \leftarrow LN \cup \{i_k\}$
    $t_i \leftarrow c_{0i_k}$
  end for
  $j' \leftarrow \underset{i \in LN}{\text{argmin}} \{t_i\}$
  while $t_{j'} < s$ **do**
    $Candidates \leftarrow \{a_{j'h} \in Out : \forall h \in N \wedge p_{j'h} > 0\}$
    if $Candidates = \emptyset$ **then**
      $Candidates \leftarrow \{a_{j'h} \in A : \forall h \in N\}$
    end if
    $eval_{j'h} \leftarrow \min_{j \in LN \setminus \{j'\}} \{c_{hj}\}, \quad \forall a_{j'h} \in Candidates$
    Create the RCL from the $nRCL$ arcs in $Candidates$ with highest $eval$
    Randomly select an arc $a_{j'l^*} \in RCL$
    Insert arc $a_{j'l^*}$ in individual I
    $Out \leftarrow Out \setminus \{a_{j'l^*}\}$
    $LN \leftarrow LN \setminus \{j'\}$
    $LN \leftarrow LN \cup \{l^*\}$
    $t_l \leftarrow t_{j'} + c_{j'l^*}$
    $j' \leftarrow \underset{i \in LN}{\text{argmin}} \{t_i\}$
  end while
  $Pop \leftarrow Pop \cup \{I\}$
end while
return Pop

ACCEPTED MANUSCRIPT

An individual is considered complete and included in population if all routes have the length equal to or greater than the shift duration s.

3.4. New Population Generation

New population generation is described in Algorithm 4. Two different individuals are selected by binary tournament, using the previously described comparison operator. Offspring is generated through crossover or asexual reproduction, according to a crossover probability cp. The crossover operator is described in section 3.5 and asexual reproduction only duplicates each parent in one individual of the offspring. A mutation operator is applied to each offspring individual according to a probability mpC if the individual is the result of a crossover, or to a probability mpD if the individual is the result of an asexual reproduction. The selection-reproduction-mutation sequence is repeated to complete the new population with ps individuals, and the new population is returned.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Algorithm 4 Method to generate a new population

Input:
$curPop$ – current Population
ps – population size

$newPop \leftarrow \emptyset$
while $|newPop| < ps$ **do**
  $I1 \leftarrow SelectIndividual(curPop)$
  repeat
    $I2 \leftarrow SelectIndividual(curPop)$
  until $I2 \neq I1$
  if $GenerateRandomNumber(\in [0,1[) < cp$ **then**
    $Offspring \leftarrow crossover(I1, I2)$ comment: $Offspring = \{O1, O2\}$
    if $GenerateRandomNumber(\in [0,1[) < mpC$ **then**
      $Mutate(O1)$
    end if
    if $GenerateRandomNumber(\in [0,1[) < mpC$ **then**
      $Mutate(O2)$
    end if
  else
    $Offspring \leftarrow duplicate(I1, I2)$ comment: $Offspring = \{ O1, O2 \}$
    if $GenerateRandomNumber(\in [0,1[) < mpD$ **then**
      $Mutate(O1)$
    end if
    if $GenerateRandomNumber(\in [0,1[) < mpD$ **then**
      $Mutate(O2)$
    end if
  end if
  $newPop \leftarrow newPop \cup \{ O1, O2 \}$
end while
return $newPop$

3.5. Chromosome and genetic operators

Usually, genetic algorithms applied to vehicle routing problems use chromosomes represented as chains of integers to encode routes. In this kind of structure, the integers represent nodes and crossing points are based on index positions. This procedure is consistent with objectives related to total routes length, since sequences of short arcs are naturally propagated by genetic operators.

In the context of synchronized routing as imposed by our dispersion met-

ACCEPTED MANUSCRIPT

ric, the spatial proximity among arcs traversed concomitantly by the m vehicles is relevant. In order to propagate good sequences of concomitant arcs we represent each route by a linked list and the crossing point is based on a timestamp that determines the instant at which the routes are sectioned. As the crossing point is the same for all routes, we favor the propagation of good spatial and temporal relations among concomitant arcs, which is crucial for the dispersion objective, as long as it does not have impact on the profit objective.

As routes are implemented as linked lists, there are no coding and decoding procedures to be performed. This also has an impact on local search performance, because movements are applied over the same structure and there is no need for decoding procedure in order to evaluate a solution.

We implemented a simple mutation operator, in which a route is randomly chosen and a pair of adjacent arcs a_{il} and a_{lj} is replaced in this route by another pair a_{ik} and a_{kj}. The choice of node k is done randomly within all nodes in $N \setminus \{l\}$.

The implemented crossover combines two parent individuals, p_1 and p_2, to create an offspring of two new individuals, o_1 and o_2.

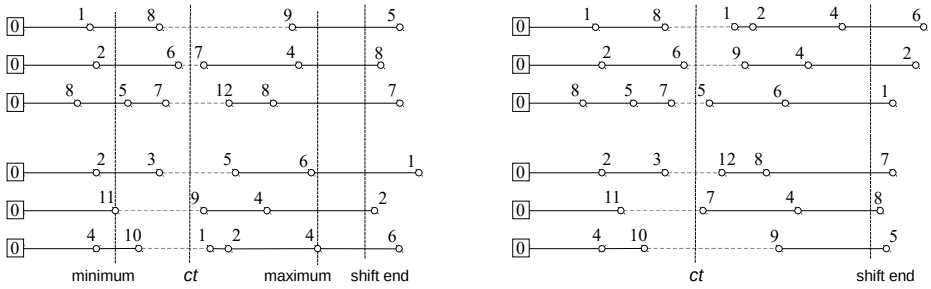
The first step in the crossover procedure is the definition of a time instant ct, randomly chosen among the duration of the routes, to be used as crossing point. We then remove from both parents all arcs planned to be traversed by some vehicle at time ct. This means that each route in both parents is broken in two segments, the first one with arcs to be traversed prior to ct, and the second with arcs to be traversed after ct.

Each route in offspring $o1$ combines the first segment of a route in parent p_1 with the second segment of a route in p_2. This implies the insertion of a new arc, in order to link the final node of the first segment with the first node of the second one. There is no prior definition about the pairs of routes from different parents that should exchange their segments, so this definition is done randomly each time a crossover is performed.

It is interesting to highlight that there is only one crossing time, but there are m crossing points since the crossover occurs in all routes. This is the main reason for the use of linked lists to represent the chromosome: the crossover redefines all routes.

If the first arc in each route is removed due to a small value for ct, the crossover operation will result in an offspring that represents the same solutions provided by its the parents. On the other hand, ct values close to the routes length limit may result in the removal of the last arcs of the

routes, and there is only one node positioned after shift time to be exchanged in each route. To avoid such situations, we define minimum and maximum limits for the crossing time, calculated considering the characteristics of all routes in both parents. The minimum limit is the latest instant any vehicle finishes its first arc, and the maximum limit is the latest instant any vehicle starts its last arc. The crossover procedure is depicted in Figure 3.



(a) Parents (b) Offspring

Figure 3: Crossover applied to two parents with three routes each. The vertical lines in (a) represent *ct* between its maximum and minimum possible values, and the end of the shift. The arcs represented by dashed lines in (a) are those removed from the routes, and in (b) are those included to connect the two segments in each new route in offspring solutions.

Finally, it is important to notice that some routes in offspring solutions may result longer than the routes in parents, and some arcs may be dislocated in time to be traversed after the shift time. Those arcs are removed from their routes. On the other hand, some routes may become shorter than the limit. In those cases, arcs among those starting in the last node of the partial route are randomly selected to be included, until the shift limit s is reached. Usually, it implies the insertions of only one or two random arcs. This feasibility issue occurs also in mutation or neighborhood movements, and the treatment is the same for all those operations.

3.6. Local Search

Local search (LS) methods designed for single objective problems explore neighborhood of an incumbent solution to find an improving solution. In bi-objective problems, there is no incumbent solution but a current nondominated frontier, and the search is for new solutions that improve this frontier. The remaining of this section describes how we implemented the local search procedure.

ACCEPTED MANUSCRIPT

3.6.1. Frontier Improvement Procedure

Our local search method is based on the frontier improvement procedure described in Algorithm 5, which is similar to the Multi-Objective Local Search procedure described in (Arroyo and Armentano, 2005).

Algorithm 5 Frontier improvement

Input:
ND – set of nondominated solutions
$nIter$ - number of iterations
$nMax$ - number of solutions to evaluate in each iteration
L - set of all solutions whose neighborhood was explored

$S \leftarrow ND$
while $nIter > 0 \wedge |S| > 0$ **do**
  if $|S| > nMax$ **then**
    $Snot \leftarrow$ set of $|S| - nMax$ randomly selected solutions in S
    $S \leftarrow S \setminus \{Snot\}$
  end if
  $S' \leftarrow \emptyset$
  for all $x \in S$ **do**
    if $x \notin L$ **then**
      $N(x) \leftarrow generateNeighborhood(x)$
      $L \leftarrow L \cup \{x\}$
      $ND \leftarrow$ nondominated solutions of $(ND \cup N(x))$
      $S' \leftarrow$ nondominated solutions of $(S' \cup \{N(x) \cap ND\})$
    end if
  end for
  $S \leftarrow$ nondominated solutions of $(S' \cup Snot)$
  $nIter \leftarrow nIter - 1$
end while
return ND

Algorithm 5 receives as input a frontier of nondominated solutions ND to be improved, and it is duplicated in S. The main loop is executed until set S becomes empty or the maximum number of iterations $nIter$ is reached. Inside the loop, if set S has more than $nMax$ solutions, it is reduced to the cardinality defined by this parameter. Parameters $nIter$ and $nMax$ establish limits due to the high computational cost of neighborhood exploration.

ACCEPTED MANUSCRIPT

Solutions removed from S are stored in the *Snot* set and neighborhood $N(x)$ is then generated for every solution x in set S and not in set L. Set L contains all solutions already explored in local search executions, and it is used to avoid exploring the same solutions several times.

If $N(x)$ is generated, solution x is included in set L. Updating L at this point of the algorithm is an important difference of our method when compared to the seminal one. In the method described in (Arroyo and Armentano, 2005), all solutions that are part of the initial nondominated set (our starting ND) are included in L. However, as our algorithm performs a search with breadth limited by $nMax$ and depth limited by *nIter*, it is not guaranteed that all solutions in the initial ND will actually be explored in the frontier improvement procedure.

A new set of nondominated solutions is constructed after the union of $N(x)$ and ND and the removal of the dominated solutions from the resulting set. Also, the new solutions in $N(x)$ that were not dominated by any solution in ND are included in set S'. At the end of the *for loop*, solutions in S' are new and nondominated solutions. These solutions, combined with those left apart and stored in *Snot*, constitute the new set S for the next iteration.

3.6.2. Frontiers improved and insertion of resulting solutions in population

The frontier improvement method, as described in Algorithm 5, is executed twice every time the local search is performed. The first execution improves the subset of the *Archive* that in generations without local search is directly included in population as the *Elite*. The resulting improved nondominated set is referred to in the sequence as $ND1$.

The second execution improves the set with nondominated solutions of the current population, identified as frontier with rank 0 (or F_0) after the nondominated sorting procedure. The resulting improved set is referred to as $ND2$. In order to avoid waste of computation time, we try to improve *Elite* and F_0 sets only after the removal of solutions already in L.

$ND1$ and $ND2$ sets represent the result of the local search procedure. In order to take part in future selection-reproduction procedure, their solutions must be included in the current population. This occurs according to the cardinalities of original *Elite* and F_0 sets and $ND1$ and $ND2$ sets.

If $|Elite| + |F_0| = |ND1| + |ND2|$, all individuals in *Elite* and F_0 are discarded and those in $ND1$ and $ND2$ are included directly in the current population.

Another situation occurs if $|Elite|+|F_0| > |ND1|+|ND2|$, i.e., the number of solutions in $ND1$ and $ND2$ is not enough to fill up the population size. In this case, $Elite$ and F_0 sets are not totally discarded, but reduced to fit the difference $\Delta = (|Elite| + |F_0|) - (|ND1| + |ND2|)$. We first eliminate from the $Elite$ set all solutions that remained in $ND1$ or have descendant solutions in $ND1$, and from F_0 all solutions that remained in $ND2$ or have descendant solutions in $ND2$. The idea is to maintain the population's diversity. A solution x' is said descendant of x if it was generated as a neighbor of x, or so on recursively. If the number of solutions in $Elite$ and F_0 sets is greater than Δ after this removal, solutions are randomly removed from both sets, until $|Elite| + |F_0| = \Delta$. Finally, solutions in $ND1$ and $ND2$ and the remaining solutions in $Elite$ and F_0 are included in the current population. This procedure is depicted in Figure 4.

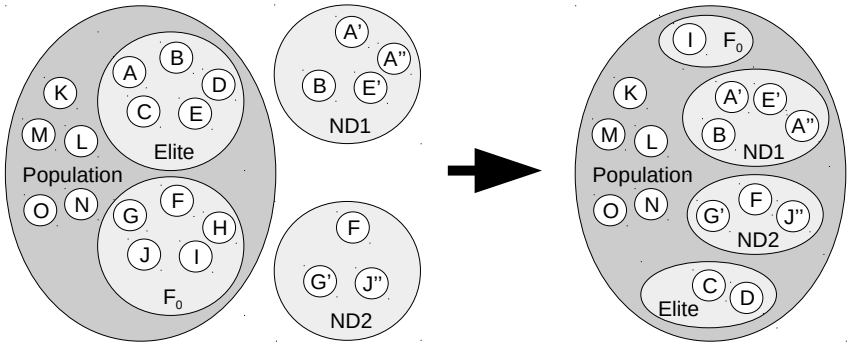


Figure 4: $ND1$ and $ND2$ do not fill up the population. Not all solutions in $Elite$ and F_0 are discarded. We discard those in $Elite \cap ND1$ and in $F_0 \cap ND2$, B and F, and solutions with descendants, A, E, G and J. Another solution must be discarded, and H is randomly selected. The 15 remaining solutions compose the population.

A third situation is presented in Figure 5 and occurs if $|Elite| + |F_0| < |ND1| + |ND2|$. In order to favor diversity, we first reduce $ND1$ discarding individuals in $Elite \cap ND1$, since those are already stored in $Archive$. If this reduction is not enough, individuals are randomly selected from population and discarded until the insertion of $ND1$ and $ND2$ does not exceed the population size.

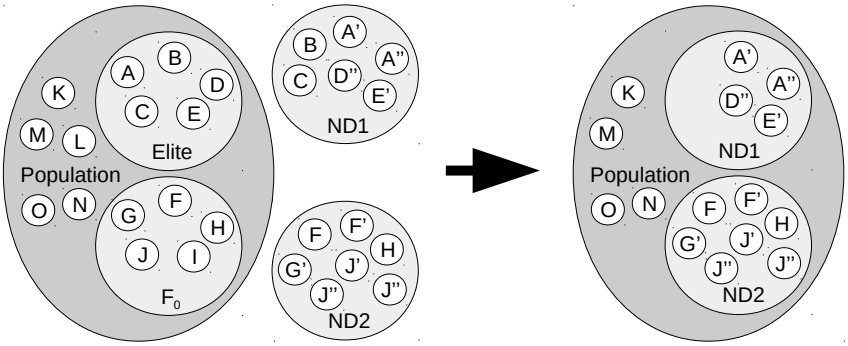


Figure 5: $ND1$ and $ND2$ together have more individuals than *Elite* and $F0$. *Elite* and $F0$ are fully discarded, along with individuals in $ND1 \cap Elite$, B and C. Another individual must be removed from original population, and L is randomly selected.

3.6.3. Neighborhood

The local search process explores four neighborhoods defined by the following movements: (1) an arc a_{ij} is replaced by two arcs a_{ik} and a_{kj}; (2) two arcs a_{il} and a_{lj} are replaced by two arcs a_{ik} and a_{kj}; (3) two arcs a_{il} and a_{lj} are replaced by one arc a_{ij}; (4) two arcs a_{ij} and a_{kl}, placed in different routes and at least partially concomitant, are replaced by arcs a_{il} and a_{kj}. The fourth movement represents the exchange of routes segments and is important to avoid routes crossing.

There is a high computational cost associated to the neighborhoods exploration, since it implies in a large number of possible movements to be considered. To reduce computational cost, we establish some conditions to decide when to apply a movement. The first and second movements are executed if the arcs exchange represents a profit gain, or if the movement involves at least one arc that is determinant for at least one time slice dispersion. The fourth movement is executed if the arcs exchange represents a profit gain or if $c_{ij} + c_{kl} > c_{il} + c_{kj}$. The third movement is always executed, since the number of generated neighbors is small.

Finally, it is necessary to define when an arc is determinant for a time slice dispersion. Consider Figure 6, where a time slice and four routes segments are depicted. The closest arcs are a_{kl} and a_{mn}, so those arcs are determinant for the local cost associated to the time slice. It is interesting to notice that

ACCEPTED MANUS

ACCEPTED MANUSCRIPT

arc a_{ij} does not contribute to the dispersion of any time slice.

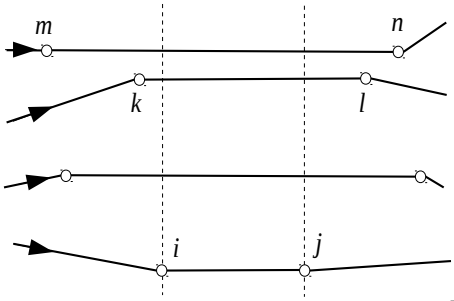


Figure 6: Arcs determinant for a time slice dispersion. Dashed vertical lines represent the limits of the time slice.

4. Results and Discussion

Computational experiments were conducted on a dual chip machine with two Intel E5-2680 processors, 64 GB of RAM and clock of 2.7 GHz. The algorithms were implemented in Java and run on Oracle Hotspot implementation, version 1.8.0 and revision 72, over Linux O.S.. No kind of parallelism was implemented, so each run used one processor only.

4.1. Instances and parameters

Usually, instances for Arc Routing Problems with profit present only information about arcs sets. Each arc is represented by a pair of nodes and its associated values, like profit, cost or demand. However, the dispersion metric introduced in this paper is calculated based on instantaneous positions of the vehicles, and those positions depend on the location of the arcs. Thus, we had to generate instances, specific for the new problem, that list the required coordinates of all nodes in N.

The first set of instances was generated according to (Feillet et al., 2005): the graph is complete; all points were randomly generated with coordinates inside a square with 500 distance units width; each cost c_{ij} corresponds to the Euclidean distance between nodes i and j; each arc has a probability of 50% of having a profit $p_{ij} \neq 0$, which is randomly generated in the range $[c_{ij}, 1.5c_{ij}]$; and the limit to the routes length (shift time) is set to 1500.

ACCEPTED MANUSCRIPT

There are also noteworthy differences between our instances and those created in (Feillet et al., 2005): we do not have capacity associated to arcs, since each arc's profit is collected only once, and we do not round to a 10^{-1} precision the values of c_{ij}. Also, contrary to the problem addressed in (Feillet et al., 2005), that does not impose limit to the number of vehicles, we used configurations with 2, 3 and 4 vehicles as in (Archetti et al., 2010), and added instances with 5 vehicles. Our instances were generated with 900, 1600, 2500 and 3600 arcs, or 30, 40, 50 and 60 nodes, thus the first set has 16 different instances.

Three other instances sets, all with 16 instances each, were also generated. They differ in the shift time, increasing the time to 2000, or in the range in which the profits are generated, allowing even p_{ij} values smaller than c_{ij}. Table 1 summarizes the characteristics of the instance sets.

Table 1: Characteristics of the instance sets

Set type	Profit Range	Shift Time
0	$[c_{ij}, 1.5c_{ij}]$	1500
1	$[c_{ij}, 1.5c_{ij}]$	2000
2	$[0.1c_{ij}, 2.0c_{ij}]$	1500
3	$[0.1c_{ij}, 2.0c_{ij}]$	2000

All instances are available at https://github.com/gdhein/arcRouting.

The parameters' values for MOGA and MOGLS were empirically adjusted through several experiments and are summarized in Table 2. The value given by $maxTime$ represents the maximum runtime (in minutes) spent to solve each instance configuration. It is important to notice that population sizes ps_1 and ps_2 were obtained independently for each method. We attribute the difference to the existence of an *Archive* set in MOGLS, since MOGA keeps all nondominated individuals found within the population.

4.2. Computational results

In our experiments, we ran the MOGA and MOGLS methods ten times each for every instance. The MOGA method implements the NSGA II evolutionary strategy and allows a comparison between the specialized method represented by our MOGLS and a popular multi-objective method. Our first evaluation concerned the impact of the local search method. Considering all 640 executions of each method, MOGLS method presented less than 26% of

ACCEPTED MANUSCRIPT

Table 2: Parameters and values used in MOGA and MOGLS

parameter	description	value
ps_1	population size (MOGA)	1300
ps_2	population size (MOGLS)	800
cp	crossover probability	90%
mpC	mutation probability after crossover	5%
mpD	mutation probability after parents duplication	30%
es	elite size	40
β	number of generations between LS executions	50
$nIter$	number of LS iterations	5
$nMax$	number of solutions exploited in each LS iteration	20
$maxTime$	execution time limit	$\lceil n \cdot m/10 \rceil$

the overall number of generations presented by MOGA. This result makes evident the computational cost related to the local search procedure.

A second evaluation considered the absolute number of solutions in approximation sets generated by each method. MOGLS presented approximation sets with 719.7 solutions on average, while MOGA presented approximation sets with 382.2 solutions only. Also, there is no instance in which MOGA had more solutions than MOGLS, considering the average results of all ten executions.

The difference is more expressive if we consider reference frontiers obtained for each instance with the union of all 20 resulting approximation sets followed by the removal of all dominated solutions. The approximation sets generated by MOGLS presented, on average, 132.8 solutions that appear in reference frontiers, while those sets generated by MOGA presented only 15.21 reference solutions on average.

We also used two performance metrics to analyze the resulting approximation sets. The first metric is usually referred as $D1_R$ (Chang et al., 2007; Knowles and Corne, 2002) and was proposed in (Czyzak and Jaszkiewicz, 1998).

This metric measures the convergence of an approximation set ND towards a reference frontier Ref. If we consider solutions as points in the bi-objective space, the metric can be considered as the average Euclidean distance between each solution in Ref to its closest solution in ND. Thus, smaller $D1_R$ values indicate better solutions sets. The $D1_R$ value is defined

ACCEPTED MANUSCRIPT

in Equation 6.

$$D1_R = \frac{1}{|Ref|} \cdot \sum_{x' \in Ref} \min\{d(x'x) | x \in ND\} \quad (6)$$

The distance between two solutions is given by

$$d(x', x) = \sqrt{[f_1^*(x') - f_1^*(x)]^2 + [f_2^*(x') - f_2^*(x)]^2}$$

where f_i^* is the normalized value of the objective function i. We apply the same normalization used on $D1_R$ metric in (Ishibuchi et al., 2003; Framinan and Leisten, 2008; Leiss et al., 2011), defined by the following equation.

$$f_i^*(x) = \frac{f_i(x) - f_i^{min}}{f_i^{max} - f_i^{min}} \times 100$$

The values f_i^{min} and f_i^{max} represent, respectively, the minimum and maximum values found for objective function i among those solutions in Ref.

Our second metric is the widely used hypervolume (HV) indicator, introduced by (Zitzler and Thiele, 1999). This metric calculates the total hypervolume between a nondominated frontier and a reference point. The value represents the amount of the objectives space that is dominated by solutions in the frontier, so better solutions present larger values. When calculated for a bi-objective problem, the hypervolume indicator can be interpreted as an area. The reference point we used in calculation is given by the minimum values of both objectives in the reference frontier obtained for the instance, or (f_1^{min}, f_2^{min}).

Table 3 summarizes the results for those metrics. It presents the results for the best approximation set found and the average results for each instance, considering the ten executions of MOGA and MOGLS.

All solutions are available at https://github.com/gdhein/arcRouting.

Table 3: Computational Results

$\|A\|$	m	Type	Best approximation set				Arithmetic mean			
			Hypervolume(10^6)		$D1_R$		Hypervolume(10^6)		$D1_R$	
			MOGA	MOGLS	MOGA	MOGLS	MOGA	MOGLS	MOGA	MOGLS
900	2	0	1025.79	1037.60	0.51	0.11	1020.30	1036.31	1.08	0.22
900	2	1	1821.76	1836.53	0.67	0.15	1810.11	1830.80	1.01	0.34
900	2	2	1904.48	1929.48	0.67	0.15	1891.39	1926.53	0.93	0.29
900	2	3	3442.30	3526.71	1.08	0.17	3418.13	3506.32	1.47	0.41
900	3	0	1011.51	1060.75	3.90	0.60	973.72	1049.78	5.48	1.19

Continue on next page

ACCEPTED MANUSCRIPT

Table 3 – *Continued from previous page*

$\|A\|$	m	Type	Best approximation set				Arithmetic mean			
			Hypervolume(10^6)		$D1_R$		Hypervolume(10^6)		$D1_R$	
			MOGA	MOGLS	MOGA	MOGLS	MOGA	MOGLS	MOGA	MOGLS
900	3	1	2061.30	2139.27	4.73	1.05	2002.76	2114.08	5.76	1.77
900	3	2	1729.18	1837.10	3.98	0.48	1695.14	1796.41	4.90	1.91
900	3	3	2621.24	2827.30	4.20	0.38	2591.20	2742.27	4.91	2.21
900	4	0	1071.97	1139.26	3.21	0.38	1021.45	1088.80	6.19	2.74
900	4	1	2565.94	2805.18	5.80	0.36	2492.34	2655.15	7.46	4.07
900	4	2	1691.24	1800.53	4.13	1.28	1611.87	1748.92	6.76	2.86
900	4	3	3464.81	3659.62	5.99	1.38	3345.24	3476.68	7.61	5.87
900	5	0	1126.88	1209.35	4.00	0.61	1029.14	1123.78	10.27	4.98
900	5	1	1003.56	1064.79	4.99	0.90	907.49	962.25	8.16	4.14
900	5	2	1296.46	1449.83	6.31	1.08	1180.73	1355.32	10.38	4.68
900	5	3	2202.86	2403.01	6.87	2.67	2132.98	2294.34	9.12	4.30
1600	2	0	1508.48	1541.04	4.21	0.06	1505.09	1540.14	5.27	0.21
1600	2	1	3137.16	3197.56	0.61	0.19	3084.18	3168.98	4.20	1.19
1600	2	2	1856.19	1898.53	0.93	0.04	1770.29	1890.80	4.82	0.09
1600	2	3	3499.64	3684.67	4.02	0.05	3465.68	3667.18	4.56	0.14
1600	3	0	1808.16	1857.79	2.20	0.72	1770.76	1829.20	4.27	1.62
1600	3	1	3239.75	3313.44	1.45	0.52	3186.68	3284.60	3.99	1.42
1600	3	2	1592.40	1665.73	2.44	0.66	1566.52	1650.29	3.72	1.22
1600	3	3	2475.56	2563.65	3.00	1.32	2435.77	2529.71	3.83	2.30
1600	4	0	1419.98	1497.40	4.04	0.64	1378.24	1423.21	6.44	3.69
1600	4	1	1605.93	1653.97	3.02	0.74	1499.68	1541.47	5.32	3.11
1600	4	2	1786.28	1910.61	3.79	0.64	1729.60	1850.61	5.71	2.67
1600	4	3	3316.82	3408.57	2.28	1.57	3225.74	3360.96	4.11	2.30
1600	5	0	925.72	971.71	3.33	1.01	860.20	941.49	8.16	2.78
1600	5	1	1714.19	1859.25	4.34	0.28	1640.35	1691.07	8.38	5.72
1600	5	2	1105.58	1205.27	6.66	1.07	1038.72	1151.63	9.39	4.20
1600	5	3	1807.72	1946.78	4.02	1.05	1724.79	1864.93	6.96	3.89
2500	2	0	1540.81	1546.47	0.26	0.11	1533.26	1540.68	0.64	0.37
2500	2	1	3072.21	3093.08	0.48	0.17	3041.73	3075.68	1.35	0.62
2500	2	2	2593.78	2653.86	1.29	0.21	2574.45	2645.69	1.61	0.39
2500	2	3	3253.25	3345.09	1.08	0.21	3223.94	3324.09	2.42	0.50
2500	3	0	1466.81	1519.55	3.90	0.75	1428.99	1501.55	5.86	2.06
2500	3	1	2745.67	2798.20	2.66	0.68	2666.40	2758.90	5.35	1.93
2500	3	2	2291.92	2375.81	2.63	0.60	2254.77	2332.61	3.35	2.11
2500	3	3	3798.12	3958.78	2.24	0.62	3708.60	3900.97	3.48	1.55
2500	4	0	1084.83	1119.27	2.13	0.47	1064.63	1085.24	3.39	2.44
2500	4	1	1903.11	1960.30	2.49	1.01	1824.68	1858.24	4.62	3.70
2500	4	2	1383.88	1394.57	2.01	1.97	1326.16	1356.58	4.79	3.78
2500	4	3	3249.46	3377.10	3.41	0.96	3169.83	3215.89	4.96	4.03
2500	5	0	1252.38	1319.76	4.66	1.43	1188.07	1235.14	8.51	5.99
2500	5	1	907.29	924.23	4.48	1.91	856.34	880.06	6.31	4.22
2500	5	2	2203.36	2390.35	7.34	0.45	2046.37	2204.68	10.93	6.13
2500	5	3	2192.23	2478.62	6.17	0.91	2098.84	2255.43	8.31	5.74
3600	2	0	1776.32	1817.96	1.35	0.20	1762.41	1803.13	1.77	0.66
3600	2	1	2884.59	2957.20	1.87	0.21	2859.81	2930.41	3.58	0.81
3600	2	2	1615.36	1637.46	0.30	0.01	1593.92	1621.20	1.53	0.63
3600	2	3	3710.54	3763.74	0.92	0.20	3653.68	3744.32	1.54	0.38
3600	3	0	1015.88	1033.69	1.22	0.36	992.05	1017.87	2.87	1.41
3600	3	1	1582.84	1619.71	1.49	0.70	1548.79	1571.94	3.11	2.15
3600	3	2	1785.62	1850.93	2.35	0.64	1715.60	1823.29	5.20	1.88

Continue on next page

ACCEPTED MANUSCRIPT

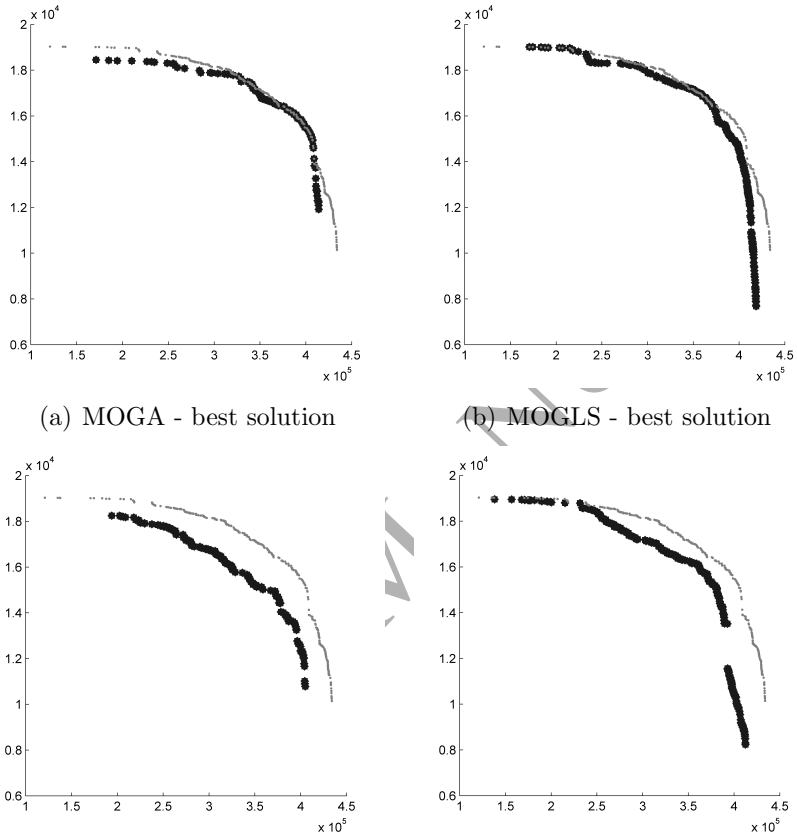
Table 3 – *Continued from previous page*

$\|A\|$	m	Type	Best approximation set				Arithmetic mean			
			Hypervolume(10^6)		$D1_R$		Hypervolume(10^6)		$D1_R$	
			MOGA	MOGLS	MOGA	MOGLS	MOGA	MOGLS	MOGA	MOGLS
3600	3	3	2526.46	2620.71	2.78	0.84	2437.27	2555.11	4.51	2.34
3600	4	0	1366.99	1405.38	2.11	0.70	1326.54	1379.11	4.51	1.86
3600	4	1	2404.26	2475.43	2.67	1.15	2358.48	2401.01	4.19	2.89
3600	4	2	2324.70	2484.61	3.69	0.58	2259.51	2418.74	6.25	2.19
3600	4	3	2551.06	2674.13	2.36	0.30	2479.19	2540.29	3.98	3.17
3600	5	0	692.21	758.69	6.02	0.46	664.86	707.76	8.33	4.98
3600	5	1	1054.42	1072.54	2.20	1.96	981.13	1005.24	6.65	4.97
3600	5	2	1671.98	1732.74	3.29	1.25	1555.44	1630.79	6.43	3.80
3600	5	3	2199.77	2236.13	3.40	2.77	2033.00	2153.28	7.19	4.76

These results demonstrate the superiority of MOGLS method over MOGA, as the first one presents better results on both metrics for all instances. To compare HV values of approximation sets obtained by each method, we used a gap given by $gap = (sol_a/sol_b - 1) \cdot 100$, where sol_a corresponds to MOGLS solutions and sol_b to MOGA solutions. We calculated the gap for each instance, and obtained a total average gap of 4.31% when considering the best approximation set found (values in columns 4 and 5), and total average gap of 4.54% when considering arithmetic mean of ten runs (values in columns 8 and 9).

$D1_R$ is used to estimate the distance between the obtained approximation set and a reference frontier and, unlike HV metric, lower values indicate better solutions. So, for $D1_R$ metric, we calculated for each instance the ratio given by $ratio = (sol_a/sol_b) \cdot 100$ to compare the methods. When we use the best approximation sets found (columns 6 and 7), we have a total average ratio of 26.26%. This indicates that the best frontiers obtained with MOGA are almost four times as distant from the reference frontier as those obtained with MOGLS (on average). If we consider the arithmetic mean obtained for each instance after ten runs (columns 10 and 11), the difference between the methods is reduced, and we have a ratio of 46.92%. But it is still impressive, since MOGLS $D1_R$ values are less than half of MOGA $D1_R$ values. We point out that local search, despite reducing the diversity by limiting the number of generations, is important to overcome the difficulties imposed by the synchronization characteristics.

Figures 7(a)-7(d) present four approximations sets obtained from different executions on the same instance, and allow us to compare the best and worst approximations sets generated by each method considering $D1_R$ metric. It is possible to identify that solutions generated by MOGLS are considerably better if we observe frontier extension and proximity to reference set.



(a) MOGA - best solution

(b) MOGLS - best solution

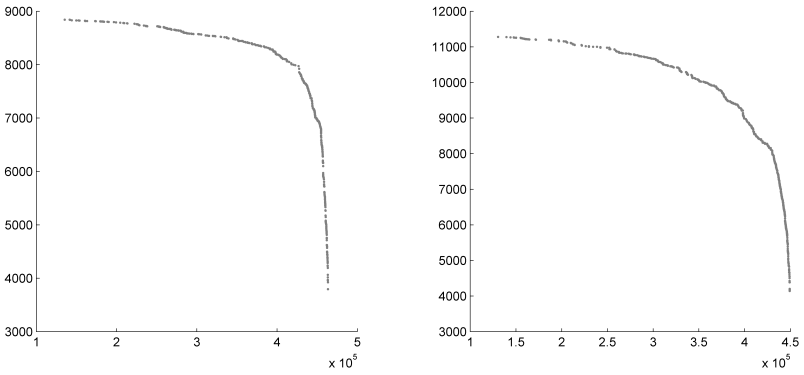
(c) MOGA - worst solution

(d) MOGLS - worst solution

Figure 7: Best and worst approximation frontiers for MOGA and MOGLS according to $D1_R$ metric. All frontiers are related to an instance of type 3 with $|A| = 3600$ and $m = 5$. Gray small dots represent solutions in reference frontier and large black dots represent solutions in an approximation frontier. The vertical axis represents the profit objective, and the horizontal axis represents the dispersion objective.

ACCEPTED MANUSCRIPT

Observing such reference frontiers' representations, we could identify a characteristic that is related to instance types, more specifically to the range in which profits were generated. Frontiers related to instances of type 0 or 1 presented a prominent knee, illustrated in Figure 8(a), while those related to instances of type 2 and 3 were more rounded, as in Figure 8(b).



(a) Type 0 (b) Type 2

Figure 8: Comparing two reference frontiers for instances with different profit ranges. Both frontiers are related to instances with $|A| = 1600$ and $m = 4$.

As indicated in (Branke et al., 2004), those *knees* are related to regions in the solutions space where small improvements in one objective result in large (relative) deterioration in the other. Concerning the set of instances proposed in our work, we associate this characteristic to the variation in the ratio p_{ij}/c_{ij}. When the variation in this ratio is small, swapping a profitable arc for another, also profitable, may have significant impact on dispersion objective, but a small one on profit objective.

Once solutions with high profit are saturated, improvements in dispersion are achieved by swapping profitable arcs for zero profit arcs, and dispersion objective improvements result in degradation of profit objective.

To evaluate this interpretation, we ran a test with an instance with $p_{ij} = c_{ij}$. The resulting reference frontier is presented in Figure 9. It is possible to see a large segment in frontier where only small changes in profit are verified, and also a very prominent knee.

On the other hand, when the ratio p_{ij}/c_{ij} presents a higher variation, like in instances of type 2 or 3, changes in arcs tend to always represent a

ACCEPTED MANUSCRIPT

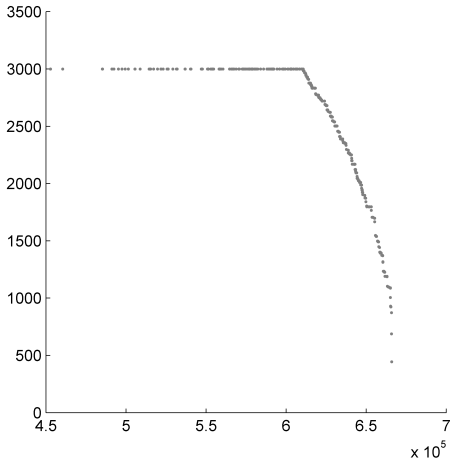


Figure 9: Reference frontier for an instance with $p_{ij} = c_{ij}$.

considerable impact for both dispersion and profit objectives.

Finally, as dispersion objective and dispersion metric are new in literature, it is important to graphically analyze the planned routes and the temporal disposition of vehicles.

Extreme solutions, where one objective is largely favored over the other, made clear the bi-objective nature of our problem. In solutions with very high dispersion, it was possible to observe vehicles traversing repeatedly the same few arcs, concentrated in different corners of the square environment. In highly profitable solutions it was possible to observe two or more vehicles close together, even traversing symmetric arcs a_{ij} and a_{ji} concomitantly.

Figure 10 exhibits a sequence of frames for a solution with 5 vehicles. This solution was chosen among those located in the *center* of the found frontier, i.e., it presents a balance between the two objectives. It is possible to observe profit collection while vehicles are kept apart.

5. Conclusions

We propose a new bi-objective arc routing problem to maximize both the collected profit and a nonlinear dispersion metric. Dispersion is relevant if vehicles have to travel across hostile environments and the robustness of the transportation system can be increased if they travel scattered. Continuous dispersion evaluation induces synchronization by imposing a spatial and

ACCEPTED MANUSCRIPT

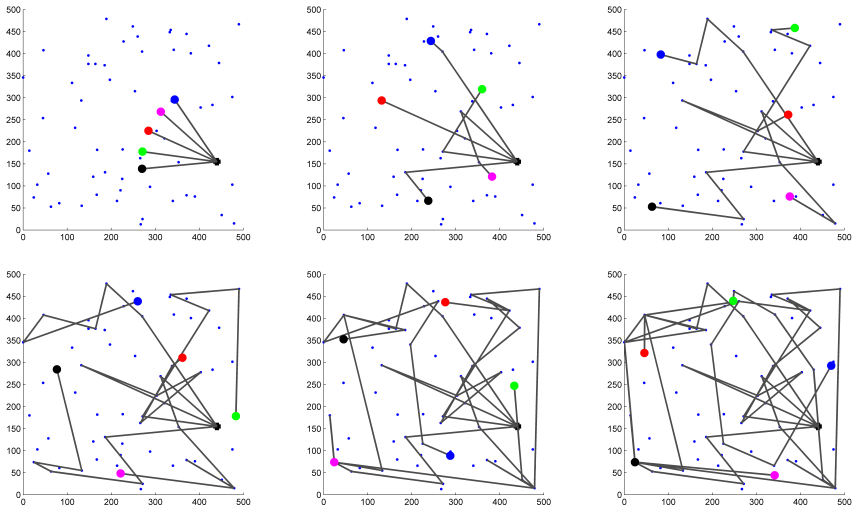


Figure 10: Time progress of a solution. Vehicles are represented as large dots, and only arcs with non zero profit are painted when traversed. An animation with the continuous progress of this solution is available at https://github.com/gdhein/arcRouting

temporal interdependence among vehicles. We see this idea as an important contribution of this paper, since it is new for routing problems.

Because of its complexity, the problem is not amenable to be solved by methods that search for optimal configurations. We made a choice for evolutionary strategies, usual for bi-objective problems. A MOGLS method was implemented to solve the problem and it is compared with a MOGA method.

We propoe a set of 64 benchmark instances to asses the performance of the MOGA and MOGLS approaches. The MOGA method, although being based on the well known and widely used NSGA II method, presented approximation sets of poor quality. It was possible to observe that the genetic algorithm had an important role in the exploration of different regions of the solutions space, providing populations with diversity. In turn, the exploitation provided by local search had a major importance as an improvement procedure, providing new dominant solutions. Frontiers obtained with the MOGLS method were consistently and remarkably closer to reference frontiers. Also, it was possible to observe the bi-objective nature of the problem.

A promising research line to be followed as a future work is to develop

ACCEPTED MANUSCRIPT

an alternative exploitation process that improves the canonical local search. There are intelligent methods in literature, like Iterated Local Search and Variable Neighborhood Search, that could be hybridized with genetic algorithm. The second of those methods is based on different neighborhood structures, and the design of specialized movements, focused on profit collections or synchronized routing, is also a future endeavor. We also intend to use the proposed MOGLS to obtain approximation sets for other multi-objective problems. The evolutionary strategy and the local search procedures are not specific for our new problem, and the particular features (constructive algorithm to generate initial solutions, specialized crossover operators) are encapsulated and can be replaced by others, designed for different problems. This would allow us to observe the behavior of the MOGLS method facing problems with different characteristics.

Future research could also involve the implementation of alternative approaches to solve the new problem proposed in this paper, in order to allow a deeper perception of the search space and a comparison of a broader variety of solutions (approximations sets). Finally, the idea behind our dispersion metric can be adapted to a wide range of routing applications that can benefit from the continuous tracking of the relative vehicles positions.

Acknowledgements

The last author would like to thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for the partial financial support.

References

Amaya, A., Langevin, A., Trépanier, M., 2007. The capacitated arc routing problem with refill points. Operations Research Letters 35 (1), 45 – 53.

Amaya, C., Langevin, A., Trépanier, M., 2010. A heuristic method for the capacitated arc routing problem with refill points and multiple loads. JORS 61 (7), 1095–1103.

Archetti, C., Feillet, D., Hertz, A., Speranza, M. G., Nov. 2010. The undirected capacitated arc routing problem with profits. Comput. Oper. Res. 37 (11), 1860–1869.

ACCEPTED MANUSCRIPT

Arroyo, J. E. C., Armentano, V. A., 2005. Genetic local search for multi-objective flowshop scheduling problems. European Journal of Operational Research 167 (3), 717–738.

Assad, A. A., Golden, B. L., 1995. Arc routing methods and applications. In: Network Routing. Vol. 8 of Handbooks in Operations Research and Management Science. Elsevier, pp. 375 – 483.

Black, D., Eglese, R., Wøhlk, S., 2013. The time-dependent prize-collecting arc routing problem. Computers & Operations Research 40 (2), 526 – 535.

Branke, J., Deb, K., Dierolf, H., Osswald, M., 2004. Finding Knees in Multi-objective Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 722–731.

Chang, P.-C., Chen, S.-H., Liu, C.-H., 2007. Sub-population genetic algorithm with mining gene structures for multiobjective flowshop scheduling problems. Expert Systems with Applications 33 (3), 762 – 771.

Corberán, A., Laporte, G., 2015. Arc Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Czyzak, P., Jaszkiewicz, A., 1998. Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis 7 (1), 34–47.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., Apr. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. Trans. Evol. Comp 6 (2), 182–197.

Del Pia, A., Filippi, C., 2006. A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. International Transactions in Operational Research 13 (2), 125–141.

Drexl, M., 2012. Synchronization in vehicle routing - A survey of VRPs with multiple synchronization constraints. Transportation Science 46 (3), 297–316.

Drexl, M., 2013. Applications of the vehicle routing problem with trailers and transshipments. European Journal of Operational Research 227 (2), 275 – 283.

ACCEPTED MANUSCRIPT

Eiselt, H. A., Gendreau, M., Laporte, G., 1995a. Arc routing problems, part I: the chinese postman problem. Operations Research 43 (2), 231–242.

Eiselt, H. A., Gendreau, M., Laporte, G., 1995b. Arc routing problems, part II: the rural postman problem. Operations Research 43 (3), 399–414.

Feillet, D., Dejax, P., Gendreau, M., 2005. The profitable arc tour problem: Solution with a branch-and-price algorithm. Transportation Science 39 (4), 539–552.

Framinan, J. M., Leisten, R., 2008. A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. OR Spectrum 30 (4), 787–804.

Haddadene, S. R. A., Labadie, N., Prodhon, C., 2016. A GRASP x ILS for the vehicle routing problem with time windows, synchronization and precedence constraints. Expert Systems with Applications 66 (Supplement C), 274–294.

Hochbaum, D. S., Lyu, C., Ordóñez, F., Oct. 2014. Security routing games with multivehicle chinese postman problem. Networks 64 (3), 181–191.

Ishibuchi, H., Yoshida, T., Murata, T., April 2003. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation 7 (2), 204–223.

Jaffrès-Runser, K., Gorce, J., Comaniciu, C., 2008. A multiobjective Tabu framework for the optimization and evaluation of wireless systems. I-Tech, Vienna, AUT, p. 29–54.

Jensen, M. T., Oct. 2003. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. Trans. Evol. Comp 7 (5), 503–515.

Knowles, J., Corne, D., May 2002. On metrics for comparing nondominated sets. In: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. Vol. 1. pp. 711–716.

Knowles, J. D., Corne, D. W., Jun. 2000. Approximating the nondominated front using the pareto archived evolution strategy. Evol. Comput. 8 (2), 149–172.

ACCEPTED MANUSCRIPT

Leiss, E., Santos, R. M., Arroyo, J. E. C., dos Santos Ottoni, R., de Paiva Oliveira, A., 2011. Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. Electronic Notes in Theoretical Computer Science - Proceedings of the 2011 Latin American Conference in Informatics (CLEI) 281, 5 – 19.

Liu, L., Mu, H., Yang, J., Li, X., Wu, F., 2014. A simulated annealing for multi-criteria optimization problem: DBMOSA. Swarm and Evolutionary Computation 14, 48 – 65.

Malandraki, C., Daskin, M. S., 1993. The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem. European Journal of Operational Research 65 (2), 218–234.

Martí, R., Campos, V., Resende, M. G., Duarte, A., 2015. Multiobjective GRASP with path relinking. European Journal of Operational Research 240 (1), 54 – 71.

Morais, V. W., Mateus, G. R., Noronha, T. F., 2014. Iterated local search heuristics for the vehicle routing problem with cross-docking. Expert Systems with Applications 41 (16), 7495 – 7506.

Nebro, A. J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J. J., Beham, A., Aug 2008. Abyss: Adapting scatter search to multiobjective optimization. IEEE Transactions on Evolutionary Computation 12 (4), 439–457.

Rosa, B. D., Improta, G., Ghiani, G., Musmanno, R., 2002. The arc routing and scheduling problem with transshipment. Transportation Science 36 (3), 301–313.

Salazar-Aguilar, M. A., Langevin, A., Laporte, G., 2012. Synchronized arc routing for snow plowing operations. Computers & Operations Research 39 (7), 1432 – 1440.

Salazar-Aguilar, M. A., Langevin, A., Laporte, G., 2013. The synchronized arc and node routing problem: Application to road marking. Computers & OR 40 (7), 1708–1715.

Shafahi, A., Haghani, A., 2015. Generalized maximum benefit multiple chinese postman problem. Transportation Research Part C: Emerging Technologies 55, 261 – 272.

ACCEPTED MANUSCRIPT

Sharif, M. T., Salari, M., 2015. A GRASP algorithm for a humanitarian relief transportation problem. Engineering Applications of Artificial Intelligence 41, 259 – 269.

Vitoriano, B., Ortuño, T., Tirado, G., 2009. HADS, a goal programming-based humanitarian aid distribution system. Journal of Multi-Criteria Decision Analysis 16 (1-2), 55–64.

Willemse, E. J., Joubert, J. W., 2012. Applying min-max k postmen problems to the routing of security guards. The Journal of the Operational Research Society 63 (2), 245–260.

Zachariadis, E., Kiranoudis, C., 2011. Local search for the undirected capacitated arc routing problem with profits. European Journal of Operational Research 210 (2), 358 – 367.

Zhang, Q., Li, H., Dec 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11 (6), 712–731.

Zitzler, E., Laumanns, M., Thiele, L., 2002. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation, and Control. CIMNE, Barcelona, Spain, pp. 95–100.

Zitzler, E., Thiele, L., Nov 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3 (4), 257–271.

ACCEPTED MANUSCRIPT