General Variable Neighborhood Search for the Data Mule Scheduling Problem

Pablo Luiz Araújo Munhoz [1,3] Pedro Henrique González [2]
Uéverton dos Santos Souza [1] Luiz Satoru Ochi [1]
Philippe Michelon [3] Lúcia Maria de A. Drummond [1]

Abstract

A usual way to collect data in a Wireless Sensor Network (WSN) is by the support of a special agent, called data mule, that moves among sensor nodes and performs all communication between them. This article dealt with the Data Mule Scheduling Problem (DMSP), where in addition to the data mule routing, it is necessary to plan the speed that this data mule will use and also to schedule the attendance of the sensors in this route. Mixed integer linear programming and heuristics based on the GRASP and GVNS metaheuristics were proposed. Besides that, a set of instances were generated in order to evaluate and validate the methods.

Keywords: Wireless Sensor Networks; Data Mule Scheduling Problem; General Variable Neighborhood Descent

[1] Universidade Federal Fluminense – Instituto de Computação
Email: {pmunhoz,ueverton,satoru,lucia}@ic.uff.br
[2] Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
Email: pegonzalez@eic.cefet-rj.br
[3] Université d'Avignon et de Pays de Vaucluse
Email: philippe.michelon@univ-avignon.fr

1 Introduction

Data Mule Scheduling Problem (DMSP) was proposed by [1] and deals with the planning of the data mules attendance. This planning covers three phases: Path Selection, Speed Control and Job Scheduling. The Path Selection phase aims to define a route to the data mule to collect information from each of the sensors in the network. In the defined route there must be at least one contact between the data mule and each sensor. The focus in the Speed Control phase is the speed management during the route created by Path Selection. The data mule needs to change the speed during the route so that it stays in contact with each sensor enough time to be able to change all the information with each of them. The Job Scheduling phase is important when the mule is in contact with more than one sensor at the same time and, since the mule serves only one sensor at a time, a scheduling of services has to be done, taking into account both the path and the speed of the data mule.

Both sensors and the data mule have radio equipment to exchange information and different sensors may have different ranges of communication. The communication occurs only when the distance between the sensor and the mule is within the range of communication of their equipment. In this scenario we say that the sensor and the data mule are in contact, whenever the data mule is in the communication range of the sensor. During each contact, the mule can upload messages destined to this sensor and download the messages that this sensor desires to send. This process is called attendance, and the mule can execute only one attendance at a time.

In [3] the Message Ferrying is defined as the agent responsible for collecting and transmitting the information between the other sensors. In this paper, it is proved that the problem of routing the Message Ferrying (Data Mule) is $\mathcal{NP}$-hard through a generalization of the Traveling Salesman Problem and a mathematical formulation was proposed to solve it. This work was extended by [4] using multiple ferries and constructive heuristics were proposed to solve it. In [1] and [2], the Data Mule Scheduling Problem is presented. In this problem, two more perspectives are added to the basic problem of routing: speed control and the order of sensors serving. They developed a mathematical model and heuristics to solve each perspective independently.

The goal of this paper is solving the three steps of the Data Mule Scheduling Problem in a unified way with the objective of minimizing the time spent by the mule to serve all the sensor nodes. To the best of our knowledge, this is the first study which deals with these steps simultaneously. In this paper, we considered a constant speed along the complete mule route as a

first approach. Thus, a mathematical formulation and two heuristics were proposed to tackle the DMSP. The first heuristic, called GRVND is based on a Greedy Randomized Adaptive Search Procedure approach. The second one is a General Variable Neighborhood Search, named GVNS-RVND. We show that GVNS-RVND overperfomed GRVND in all tested instances.

2 Mathematical formulation

The purpose of the DMSP is to serve the demand of all sensors. In order to do this the mule must stay in the communication range of the sensors enough time to be able to exchange all the required data. Thus, the mule does not necessarily have to visit directly all sensors.

The intersections of communication between sensors are parts of the path where the mule can serve more than one sensor. The attendance can be divided in this intersection area, but it should respect the constraint that the mule cannot connect to more than one sensor at a time. In order to highlight the attendance areas, fictitious nodes were added to the problem representation at each time that the possible path of the mule intercept the border of the radius range of a sensor, as shown in Figure 1. Thus the edges can be divided in edges where the mule can serve a node, $\{(a, 4), (4, d), (e, 1), ...\}$, and edges where the mule cannot attend any node, $\{(BS, a), (d, e), ...\}$.

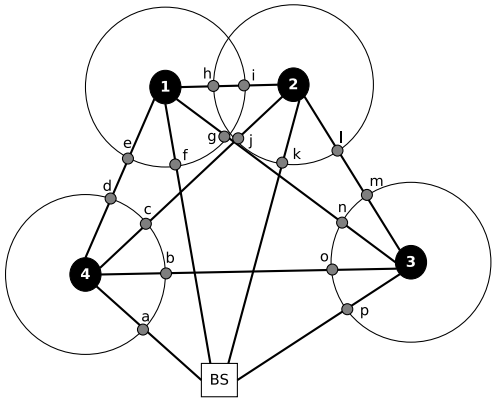

Fig. 1. Problem representation

The DMSP can be defined as follows. Let $G = (V, E)$ be a complete graph without adding fictitious nodes, where $V = \{0, 1, 2, ..., n\}$ is the set of $n + 1$ nodes of the network, and $E = \{(i, j) : i, j \in V, i \neq j\}$ the set of edges. Node 0 represents the Base Station (BS), from where the data mule departs. Each node $i \in V \setminus \{0\}$ has a demand s_i, a communication range r_i radius and a $rate_i$

which is the amount of data that this node can transmit per unit of time. After adding fictitious nodes V_f, a new graph $G' = (V', E')$, where $V' = \{V \cup V_f\}$ and $E' = \{(p, q) : p, q \in V', p \neq q\}$, is created and used as the input of the problem. The set E' can be partitioned into two subsets W and U, representing the edges from which the data mule can and cannot connect to some nodes, respectively. To each edge $(p, q) \in E'$ is associated a non-negative cost c_{pq}, which represents the distance between these nodes, and a pre-calculation of the time which the mule can stay in a edge $(p, q) \in E'$ is made, $T_{pq} = \frac{c_{pq}}{K}$. In addition, two sets are defined: $A_i = \{(p, q) \in E' : (p, q)$ can attend the node $i \in V\}$ and $Fict_{pq} = \{i \mid$ node $i \in V$ can be attended in $(p, q)\}$. The decision variables are: $x_{pq}, \forall (p, q) \in E'$, equal to one if the edge (p, q) is used in the solution, zero otherwise; $z_{pq}, \forall (p, q) \in E'$ is a flow variable related of each edge; $y_p, \forall p \in V'$, equal to one if the node p is used in the solution, zero otherwise; and H^i_{pq} is the time spend by the mule in edge $(p, q) \in E'$ serving the sensor $i \in V$. The proposed mathematical formulation is given by:

$$min \sum_{(p,q)\in W} \sum_{i\in V} H^i_{pq} + \sum_{(p,q)\in U} \sum_{i\in V} H^i_{pq} \tag{1}$$

$$s.t.$$

$$\sum_{p\in V'} x_{pl} + \sum_{q\in V'} x_{lq} = 2y_l \qquad \forall l \in V' \tag{2}$$

$$\sum_{q\in V'} z_{0q} = 1 \tag{3}$$

$$\sum_{q\in V'} z_{lq} = \sum_{p\in V'} z_{pl} + y_l \qquad \forall l \in V' \setminus \{0\} \tag{4}$$

$$\sum_{p\in V'} z_{p0} = \sum_{p\in V'} y_p + 1 \tag{5}$$

$$x_{pq} \leq \frac{z_{pq}}{2} \qquad \forall (p, q) \in E' \mid p \neq 0 \tag{6}$$

$$x_{pq} \geq \frac{z_{pq}}{|V'| + 1} \qquad \forall (p, q) \in E' \tag{7}$$

$$x_{0q} = z_{0q} \qquad \forall q \in E' \tag{8}$$

$$\sum_{i\in Fict_{pq}} H^i_{pq} = T_{pq} x_{pq} \qquad \forall (p, q) \in E' \tag{9}$$

$$\sum_{i\in V} H^i_{pq} = T_{pq} x_{pq} \qquad \forall (p, q) \in U \tag{10}$$

$$\sum_{(p,q)\in A_i} rate_i H^i_{pq} \geq s_i \qquad \forall i \in V \tag{11}$$

$$x_{pq} \in \{0, 1\} \qquad \forall (pq) \in E' \tag{12}$$

$$y_p \in \{0, 1\} \qquad \forall p \in N' \tag{13}$$

$$z_{pq} \geq 0 \qquad \forall (p, q) \in E', z_{pq} \in \mathbb{Z}^+ \tag{14}$$

$$H^i_{pq} \geq 0 \qquad \forall (p, q) \in E', i \in V, H^i_{pq} \in \mathbb{R}^+ \tag{15}$$

The objective function (1) aims to minimize the time spent by the data

mule to serve all sensors. Constraints (2) ensures that if a node is used in the solution, only one edge must enter and another edge must leave this node. The set of constraints (3-5) represent the flow restrictions that define a path and avoid sub-cycles formation. Constraints (6-8) make the connection between the flow variables z_{pq} and the solution variables x_{pq}. Constraints (9) define the time spent by the mule if the mule use this edge $(p, q) \in E'$ in its path. The demand attendance is guaranteed through constraints (11) and constraints (12-15) ensure integrality and non-negativity of the variables.

3 Proposed heuristics

In this section we present the two proposed heuristics, GRVND and GVNS-RVND, for the Data Mule Scheduling Problem.

In both algorithms, the construction of the initial solution is based on the Nearest Insertion proposed for the Traveling Salesman Problem. The method changes the criterion of choice from the next node with the shortest distance to a ratio given by the $\frac{attendance}{distance}$ of an edge. The method chosen as local search is the randomized version of the Variable Neighborhood Descent heuristic (RVND). Three neighborhood structures were developed: N^{Swap}: swapping two sensors on the solution, including sensors that are not in the route; N^{Shift}: reallocating a sensor to another point of the route; and $N^{Swap(2,1)}$: swapping two consecutive sensors and another sensor in the solution. All neighborhoods structures consider movements using nodes that are not in the route.

We developed two heuristics, GRVND and GVNS-RVND, based on Greedy Randomized Adaptive Search Procedure (GRASP) and General Variable Neighborhood Search (GVNS), respectively. The GRASP is an iterative method that has two phases: construction and local search. The GRVND construction phase builds a solution using the Nearest Insertion, and the neighborhood is explored by RVND. The best solution over all iterations is returned.

The VNS is a method whose basic idea is the systematic change of neighborhood structures within the local search algorithm. This metaheuristic uses a shake procedure that applies random movements in order to escape from local optimum. The GVNS is an extended version of the basic VNS which uses more than one neighborhood in a local search, like the RVND. In order to compare this two heuristics, the stop criteria for both algorithms is was the same: 50 iterations without improvement. The procedures GRVND and GVNS-RVND are presented in Algorithm 1 and Algorithm 2, respectively.

In the *Shake* phase of GVNS-RVND, the algorithm applies $2+p$ times a random movement of one of the three neighbourhood structures, N^{Swap}, N^{Shift} or

Algorithm 1 GRVND Heuristic

Inputs:
$IterMaxWithoutImp$, $f()$
$IterWithoutImp \leftarrow 0$;
$s^* \leftarrow \emptyset$;
while $IterWithoutImp < IterMaxWithoutImp$ **do**
  $s \leftarrow NearestInsertion()$;
  $s' \leftarrow RVND(s)$;
  if $f(s') < f(s^*)$ **then**
    $s^* \leftarrow s'$;
    $IterWithoutImp \leftarrow 0$;
  else
    $IterWithoutImp \leftarrow IterWithoutImp + 1$;
  end if
end while
return s^*

Algorithm 2 GVNS-RVND Heuristic

Inputs:
$IterMaxWithoutImp$, $f()$, $shakePerc$
$s_0 \leftarrow NearestInsertion()$;
$s^* \leftarrow RVND(s_0)$;
while $IterWithoutImp < IterMaxWithoutImp$ **do**
  $s' \leftarrow s^*$;
  $s' \leftarrow Shake(s', shakePerc)$;
  $s'' \leftarrow RVND(s')$;
  if $f(s'') < f(s^*)$ **then**
    $s^* \leftarrow s''$;
    $IterWithoutImp \leftarrow 0$;
  else
    $IterWithoutImp \leftarrow IterWithoutImp + 1$;
  end if
end while
return s^*;

$N^{Swap(2,1)}$, randomly selected too. The value of p is initially defined as 0, and it is increased by two units at each $shakePerc\%$ of total $IterMaxWithoutImp$. The idea is that the more the iterations without improvement increases the more the *Shake* phase will try do diversify the current solution.

4 Computational Experiments

For the validation of the methods, a new set of instances was created, having between 6 and 21 sensors. The coordinates of the sensors were randomly defined within a *grid* (0, 300)×(0, 300). The Base Station was positioned in three configurations for each instance: `Central`, in the center of the grid; `Eccentric`, in the corner of *grid*; and `Random`, at a random point on the *grid*. The other sensors were positioned randomly. The radius range r_i, transmission rate $rate_i$ and service demands s_i were randomly selected in the intervals (1, 50), (1, 10) and (1, 20), respectively. Thus, for each set node × Base Station

position, 50 instances were generated, totaling 1200 used for the test.

The tests were performed on a computer with Intel Core i7-4770 3.40GHz processor with 16GB of RAM and Linux Ubuntu 14.04 operating system. For the execution of the exact method, the mathematical solver CPLEX 12.5.1 was used in its default configuration.

The GRVND and GVNS-RVND were tested in the proposed set of instances and GAP metric was used for the comparison between them. Its formula is given by $GAP = \frac{(f_{Heuristic} - f_{Exact})}{f_{Exact}}$, where $f_{Heuristic}$ represents the value of the objective function of the solution found by the proposed heuristics and f_{Exact} represents the solution of the mathematical model. The results of the two proposed heuristics compared to the mathematical formulation executed with a time limit of 1 hour (3600 seconds) are presented in Table 1. If the time exceeded the given limit, the CPLEX is interrupted and the best solution found is reported. Each heuristic was executed 10 times for each instance.

Table 1
Computational experiments of proposed methods

# Sensors	Exact	GRVND				GVNS-RVND			
	T(s)	Sol_{min}	$T_{min}(s)$	Sol_{avg}	$T_{avg}(s)$	Sol_{min}	$T_{min}(s)$	Sol_{avg}	$T_{avg}(s)$
6	0.349	0.12%	0.082	0.17%	0.090	**0.00%**	0.036	**0.00%**	0.038
7	1.036	**0.00%**	0.157	0.19%	0.176	**0.00%**	0.070	**0.00%**	0.075
8	3.719	0.01%	0.276	0.32%	0.318	**0.00%**	0.125	**0.00%**	0.135
9	9.297	0.08%	0.413	0.31%	0.485	**0.00%**	0.194	**0.00%**	0.210
10	13.940	0.02%	0.706	0.17%	0.812	**0.00%**	0.217	**0.00%**	0.236
11	20.312	0.06%	1.049	0.29%	1.275	**0.00%**	0.273	**0.00%**	0.297
16	458.419	0.04%	6.110	0.15%	7.067	**0.00%**	0.706	0.01%	0.815
21	2521.781	-0.74%	14.418	-0.63%	16.926	-0.80%	2.098	-0.73%	2.692
Average	378.607	-0.05%	2.901	0.12%	3.394	-0.10%	0.465	-0.09%	0.562

In this table, the first and second columns represent the number of sensors and the average time spent by the exact method in each set of instances, respectively. The columns $\boldsymbol{Sol_{min}}$ and $\boldsymbol{Sol_{avg}}$ represent the average GAP for the best and average solutions found by the heuristics. The column **T (s)** shows the average execution time. The results are grouped by number of nodes, thus each line of the table presents the average of 150 instances.

GVNS-RVND outperforms GRVND in terms of both $\boldsymbol{Sol_{min}}$ and $\boldsymbol{Sol_{avg}}$, being able to find the optimal solutions in all instances until 16 sensors, except in one instance in the average measure. Both methods are capable of improving the solutions from 21 nodes, but the GVNS-RVND has a smaller gain when compared to GRVND. The negative gaps are obtained when the mathematical formulation can not prove the optimal solution in the limited executed time of 3600s, allowing the heuristics to improve the solutions. The time results clearly show that the heuristics are capable of obtaining good quality

solutions in lower computational times when compared with the mathematical formulation. In order to validate the statistical significance of all results, we performed a pairwise Wilcoxon test comparing the GAP values between GRVND and GVNS-RVND. This test led to a value $p < 2.2 \times 10^{-16}$, which indicates a significant difference of performance. As the methods are using the same constructive and local search modules, it is evident the importance of the shake phase of the GVNS-RVND, allowing to significantly improve the results in relation of the GRVND.

5 Conclusions

In this work we propose a mathematical formulation and two heuristics, GRVND and GVNS-RVND, to solve the three steps of the Data Mule Scheduling Problem (DMSP) in a unified way. The comparative study show that the GVNS-RVND provides results of great quality in comparison with the GRASP based heuristic. Besides that, the GVNS-RVND obtains the optimal solution for all instances until 16 nodes, being capable of overcoming the results of the mathematical formulation, with time limit, in instances with 21 nodes. The Wilcoxon test confirm that GVNS-RVNS is significantly better than the GRVND, so it can be considered an excellent choice for solving the DMSP, as it can achieve very good solutions in a small computation time. In future work, the mathematical formulation and the heuristics may be extended to problems where the mule can change its speed during the route.

References

[1] Sugihara, R. and R. K. Gupta, *Speed control and scheduling of data mules in sensor networks*, ACM Transactions on Sensor Networks **7** (2010), pp. 4:1–4:29.

[2] Sugihara, R. and R. K. Gupta, *Path planning of data mules in sensor networks*, ACM Transactions on Sensor Networks **8** (2011), pp. 1:1–1:27.

[3] Zhao, W. and M. Ammar, *Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks*, The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (2003), pp. 308–314.

[4] Zhao, W., M. H. Ammar and E. W. Zegura, *Controlling the mobility of multiple data transport ferries in a delay-tolerant network*, in: *24th Annual Joint Conference of the IEEE Computer and Communications Societies* (2005), pp. 1407–1418.