Solving a Variant of the Hub Location-Routing Problem

Mauro Cardoso Lopes, Thiago Alves de Queiroz, Carlos Eduardo de Andrade and FlávioKeidi Miyazawa

Abstract We investigate a variant of the many-to-many (hub) location-routing problem, which consists in partitioning the set of vertices of a graph into cycles containing exactly one hub each, and determining an extra cycle interconnecting all hubs. A local search heuristic that considers add/remove and swap operations is developed.Also, a branch-and-cut approach that solves an integer formulation is investigated.Computational experiments on several instances adapted from literature show that our algorithms are good to deal with small to medium-sized instances.

Keywords Location-routing problem · Branch-and-cut · Local search

1 Introduction

In this work, we investigate a version of the *many-to-many (hub) location-routing problem* (MLRP) with the following assumptions: any node is candidate to be a hub in the network; the number of required hub nodes is initially given; each non-hub is served by exactly one hub and must be in only one local route;for each hub there exists only one local route starting and ending on it that serves two or more non-hub nodes; hubs are unlimited in capacity, but any local route is limited in size by a given maximum number of nodes;all hubs must be in an inter-hubs route forming a ring, the edges of which have a discount factor α, $0 \leq \alpha \leq 1$, applied to their costs. Such a discount factor is considered when using the inter-hubs route due to bulk

T. A. de Queiroz (✉)
Department of Mathematics, UFG-Campus Catalão, 75704-020 Catalão, GO, Brazil
e-mail: taq@ufg.br

M. C. Lopes · C. E. de Andrade · F. Miyazawa
Institute of Computing, IC/UNICAMP, Campinas, SP 13084-971, Brazil
e-mail: maurolopes@gmail.com

C. E. de Andrade
e-mail: andrade@ic.unicamp.br

F. Miyazawa
e-mail: fkm@ic.unicamp.br

Z. Zhang et al. (eds.), *LISS 2014,* DOI 10.1007/978-3-662-43871-8_58

transportation, see (O'Kelly 1987). The objective is to minimize the total cost given by the inter-hub transportation route (with the discount factor) and the local routes connecting non-hubs to hub nodes.

The MLRP was raised in (Nagy and Salhi 1998) and has applications in freight industry as transportation of parcels including passengers, postal applications, and beverage industry (Lin et al. 2012) and, telecommunication systems (Wang et al. 2006). We denominate the problem here addressed as *Many-to-Many k-Location-Hamiltonian Problem* (MLHP), due to the covering of all nodes by a set of exactly *k* cycles.It is an NP-hard problem.

In the following sections we present algorithms and computational experiments to solve the MLHP. Section 2 describes a local search based heuristic, while in Sect. 3 we present an integer formulation.Computational experiments are reported in Sect. 4 and conclusionsare given in Sect. 5.

2 A Local Search Based Heuristic

In the remainder of the paper, we consider an instance *I* of the MLHP given by: the number *k* of required hub nodes; a complete undirected graph $G = (V, E)$, in which V is the set of nodes (all of which are hub candidates), and E is the set of edges $\{i, j\}$ connecting nodes i and j in V, with i not equal to j, for which there is an associated cost $c_{\{i,j\}}$ for a vehicle to traverse it; a discount factor α, for those edges connecting hubs; and a maximum length C allowed for any local route.The term *solution cost*stands for the overall cost given by the cost of traversing of all routes: inter-hubs (multiplied by α) and local ones.

The heuristic is based on generating an initial solution at random, which is improved by a Local Search routine exploring its neighborhood by three operators. When the solution value does not change anymore, the Chained Lin-Kernighan heuristic (Applegate et al. 2013) is then used. Next, it goes back to local search, repeating the previous steps while those operators can improve the solution value.

Operator OP_1 works by removing a node from a local route r_s and inserting it into another local route r_t. In OP_2 we have swap operations. For each pair of routes r_s and r_t and each nodes i in r_s and j in r_t, i is moved to r_t and j to r_s. The last operator, OP_3, works over the hub nodes. In each local route, it searches (and replaces, if any) for nodes that, if converted to hub nodes, decrease the current solution cost. The Local Search phase cycles through these operators as long as the solution is improved. If they can no longer improve the solution, the Chained Lin-Kernighan heuristic is used in each route of the current solution.

3 Integer Formulation

We propose an integer model for the MLHP that uses binary variables. Variable $x_e = 1$ indicates that edge *e*links two hubs, while variable $z_e = 1$ considers *e* as present in the solution but involving at least one non-hub. Also, variable $y_i = 1$ assumes

a hub located at node *i*. Moreover, for $S \subseteq V$, let $\delta(S)$ be the set of edges with one extremity in *S* and the other in $S \backslash V$; consider *B[S]* as the set of edges with both extremities in *S*. We denote by $\mathfrak{D}_{ij}$ the set of all paths connecting node *i* to node *j*. For P in $\mathfrak{D}_{ij}$, *V[P]* denotes the set of nodes in *P*.

Formulation in Eqs. (1)-(11) describes the integer model whose objective function (1) aims to minimize the overall cost of the inter-hubs and local routes. Constraints (2) ensure the number of hubs opened must be equal to *k*, while constraints (3) are the degree constraints for each node. Similarly to the latter, constraints (4) are the degree constraints for hub nodes, that is, each open hub must serve only one route. Constraints (5) impose that the edges cannot be simultaneously in the inter-hubs and any other local route. In order to avoid subtours, constraints (6) ensure that for any subset *S* of nodes, the number of edges (regardless of being in the inter-hubs or in a local route) in $\delta(S)$must be at least two. The same applies to constraints (7), but now only observing hubs that are open and the edges in the inter-hubs route. We have path elimination constraints in (8) that prevent routes starting at a hub and finishing at another one. Capacity constraints are given in (9) and impose that the number of incident edges in any subset of nodes must be at least double the number of routes required to serve such nodes demand, not considering those nodes as hubs. Constraints (10) and (11) impose that all variables are binary.

$$\min \alpha \sum_{e \in E} c_e x_e + \sum_{e \in E} c_e z_e. \tag{1}$$

Subject to:

$$\sum_{e \in V} y_e = k. \tag{2}$$

$$\sum_{e \in \delta(i)} z_e = 2, \text{for all } i \in V. \tag{3}$$

$$x_e + z_e \leq 1, \text{for all } e \in E. \tag{4}$$

$$x_e + z_e \leq 1, \text{for all } e \in E. \tag{5}$$

$$\sum_{e \in \delta(S)} (x_e + z_e) \geq 2, \text{for all } S \subset V : |S| \geq 1. \tag{6}$$

$$\sum_{e \in \delta(S)} x_e \geq 2(y_i + y_j - 1), \text{for all } i, j \in V, \text{for all } S \subset V : |S \cap \{i, j\}| = 1. \tag{7}$$

$$\sum_{e \in B(P)} z_e + \sum_{l \in V(P)} y_l \leq |V[P]|, \text{for all } i, j \in V : i \neq j, \text{for all } P \in \mathfrak{D}_{ij}. \tag{8}$$

$$\sum_{e \in \delta(S)} z_e \geq 2\left(\frac{\sum_{i \in S} (1 - y_i)}{c - 1} - \sum_{i \in S} y_i \right), \text{for all } S \subset V : |S| \geq 2. \tag{9}$$

$$x_e, z_e \in \{0,1\}, \text{for all } e \in E. \tag{10}$$

$$y_i \in \{0,1\}, \text{for all } i \in V. \tag{11}$$

The number of constraints (6)–(9) may be very large in practice. Therefore, separation routines are used to search by such inequalities that are violated, and they are added into the formulation in a cutting plane fashion. For constraints (6) and (7), we used the Gomory-Hu tree (Gomory and Hu 1961), while in the path constraints (8), the routine performs a breadth-first search in order to find hub j, so there is a path connecting i with j. Capacity constraints (9) use the heuristic procedure in (Lysgaard et al. 2004).

4 Computational Study

All algorithms are implemented in the C++ programming language and the experiments were carried out in a computer with 3.4 GHz Intel Core i7-2600 processor, 8 GB of RAM memory and GNU/Linux operating system. The maximum number of iterations used for the heuristic is 16,000. Formulation (1)–(11) is solved with a branch-and-cut algorithm provided by the Gurobi Optimizer version 5.6.2. Besides, each instance has a time limit of 3,600 s, and the heuristic solution is used as initial upper bound.We consider 28 instances adapted from the TSPLIB repository, for which we used $k = \lceil 0.2n \rceil$ and $C = \left\lceil \frac{n}{k} \right\rceil$ observing the total number of nodes n. The value of α is set to 0.4 in accordance with the literature (de Camargo et al. 2013).

Table 1 presents the results for all the instances. The last column contains the ratio of the solution value of the heuristic to the one of the branch-and-cut. Moreover, when a solution obtained with the branch-and-cut algorithm is proven optimal, an asterisk appears with the respective value. For the heuristic, we marked it with "+". Only seven instances are solved to the optimality with the branch-and-cut algorithm, while the other ones reached the time limit imposed. The total running time of the branch-and-cut approach is 2,722.67 s, on average. For all the instances, the total number of user cuts and nodes explored are 1,060,245 and 637,422, respectively.

The heuristic obtained equal solutions compared with the branch-and-cut algorithm for almost all the instances in Table 1. Note that it executes until it reaches the maximum number of iterations and its total running time is the shortest for almost all the instances, that is, for 27 out of 28 instances. Moreover, it computes the optimal solution for seven instances (observe the values marked with "+"). Note that both algorithms compute the same solution for 24 instances, according to the last column, and the worst difference in the solution value is for the instance *brazil58* with 1.22 %.

5 Concluding Remarks

We present heuristic and exact algorithms to solve a variant of the many-to-many location-routing problem. The heuristic allows the computation of optimal solution for several instances and also provides a good cutoff when solving the integer linear

Table 1 Results for all the 28 instances

				Heuristic		Branch-and-cut					
Instance	n	k	C	Time	Value	Cuts	Nodes	Time	Value	Gap (%)	Ratio
att48	48	10	5	36.46	13227.2	48036	31280	3601.31	13227.2	11.62	1
bayg29	29	6	5	5.42	1916	159082	165057	3600.07	1916	2.29	1
bays29	29	6	5	5.34	2429.8	51472	83794	3600.20	2429.8	5.10	1
berlin52	52	11	5	83.12	9172.6	46077	25029	3600.70	9172.6	6.66	1
brazil58	58	12	5	52.61	29334.4	56521	22248	3601.24	28975.2	9.04	1.0124
burma14	14	3	5	1.94	3891.4+	2374	2997	5.65	3891.4*	0.00	1
dantzig42	42	9	5	36.47	873.4	53248	31259	3600.83	873	9.32	1.00046
eil51	51	11	5	33.08	525.2	39739	25443	3601.51	524	8.93	1.00229
eil76	76	16	5	69.77	678.8	32421	9325	3604.91	678.8	12.64	1
fri26	26	6	5	6.11	1086.8+	8354	23046	209.62	1086.8*	0.00	1
gr17	17	4	5	3.32	2139.8+	323	277	0.40	2139.8*	0.00	1
gr21	21	5	5	4.59	3396.2+	6536	6337	19.32	3396.2*	0.00	1
gr24	24	5	5	4.85	1542.2+	15168	28885	229.79	1542.2*	0.00	1
gr48	48	10	5	44.24	6064.4	62504	31770	3602.33	6064.4	10.16	1
gr96	96	20	5	196.99	73155	23392	3531	3604.23	73155	19.20	1
hk48	48	10	5	32.86	14402.8	33357	30431	3602.15	14402.8	8.30	1
kroA100	100	20	5	88.54	30086.2	34623	3729	3602.52	30086.2	22.79	1
kroB100	100	20	5	138.26	31760.6	60819	5042	3601.11	31760.6	25.18	1
kroC100	100	20	5	124.09	30722	45359	4834	3600.82	30722	26.21	1
kroD100	100	20	5	89.78	30365.4	40237	4146	3600.61	30365.4	24.53	1
kroE100	100	20	5	239.47	31629.4	38196	3472	3600.51	31629.4	23.73	1
pr76	76	16	5	81.01	136435	40005	12647	3602.16	136435	14.57	1
rat99	99	20	5	105.50	1652.8	26690	3853	3600.85	1652.8	17.87	1
rd100	100	20	5	104.72	10734	29919	3728	3600.89	10734	19.90	1
st70	70	14	5	49.39	882.8	49736	16865	3602.00	881.2	17.75	1.00182
swiss42	42	9	5	14.52	1570.4	42474	34200	3600.96	1570.4	8.29	1
ulysses16	16	4	4	2.99	8073.4+	5332	7124	19.88	8073.4*	0.00	1
ulysses22	22	5	5	3.72	7544.2+	8251	17073	118.20	7544.2*	0.00	1

* means the optimal value.

formulation by the branch-and-cut algorithm. The worst solution computed with the heuristic differs from the optimal one by only 1.22 %.

The branch-and-cut algorithm reported optimal solution for 7 out of 28 instances, requiring a large computation time when compared with the heuristic. This behavior may be influenced by the upper bound computed with the heuristic and the sequence in which the separation algorithms are applied. Future work will consider the development of valid inequalities and hybrid heuristics.

Acknowledgements The authors would like to thank CNPq, FAPEG and FAPESP for their financial support.

References

Applegate D, Cook W, Rohe A (2003) Chained lin-kernighan for large traveling salesman problems. INFORMS J Comput 15(1):82–92

de Camargo RS, de Miranda G, Løkketangen A (2013) A new formulation and an exact approach for the many-to-many hub location-routing problem. Appl Math Model 37(12–13):7465–7480

Gomory RE, Hu TC (1961) Multi-terminal network flows. J Soc Ind Appl Math 9(4):551–570

Lin CC, Lin JY, Chen YC (2012) The capacitated p-hub median problem with integral constraints:an application to a chinese air cargo network. Appl Math Model 36:2777–2787

Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math Program 100(2):423–445 (Series A and B)

Nagy G, Salhi S (1998) The many-to-many location-routing problem. Top 6:261–275

O'Kelly ME (1987) A quadratic integer program for the location of interacting hub facilities. Eur J Oper Res 32:393–404

Wang Z, Lin C, Chan CK (2006) Demonstration of a single-fiber self-healing CWDM metro accessring network with unidirectional OADM. Photonics Technol Lett 18:163–165