A Variable Neighborhood Search Approach for the Capacitated m-Ring-Star Problem

Carlos Franco[1(✉)], Eduyn López-Santana[2], and Germán Mendez-Giraldo[2]

[1] Universidad Católica de Colombia, Bogotá, Colombia
cafranco@ucatolica.edu.co
[2] Universidad Distrital Francisco José de Caldas, Bogotá, Colombia
{erlopezs,gmendez}@udistrital.edu.co

Abstract. In this paper, we proposed an algorithm based on variable neighborhood search (VNS) for the capacitated m-Ring-Star problem. This problem has several real applications in communications networks, rapid transit system planning and optical fiber networks. The problem consists in design m rings or cycles that begins of a central depot and visits a set of customers and transition or steiner nodes. While the nodes don't belong to a ring these must be allocated or assign to a customer or steiner node that belongs to a ring. The number of customers allocated or visited in each ring must not exceed the maximum capacity. The goal is to minimize the visiting and allocation cost. For solving the problem, we propose a VNS approach based on random perturbation for escaping from the local optimal solutions. Our method reached the optimal solution in a reasonable amount of time in a set of instances from the literature.

Keywords: m-Ring-Star problem · Variable neighborhood search · Network design · Combinatorial optimization

1 Introduction

The capacitated m-Ring-Star problem (CmRSP) was introduced by Baldacci et al. [1]. This problem is a variant of the classical capacitated vehicle routing problem with one depot. The CmRSP consists in designing a set of rings (with size m) passing throw a central depot and visiting a subset of customers and a subset of steiner nodes. Rings may include transit nodes (Steiner nodes) so that star connections can be established between a customer and a ring through a transit node. Each ring and its star connections (ring-star) is limited by a maximum number of customers. A feasible solution is represented by a set of m rings. Each node is a visited node if is in a ring, if not, is an allocated node. If in a node is assigned an allocated node, it is call a connecting node. Figure 1 gives an example of a feasible CmRSP solution, where there are three rings, and nodes assigned with the dotted line are the allocated nodes. The goal is to minimize the total cost of visiting and assigning the customers to the routes that comes from the depot. This problem has many real-world applications like optical fiber networks, rapid transit system planning and telecommunication systems.

D.-S. Huang et al. (Eds.): ICIC 2016, Part I, LNCS 9771, pp. 3–11, 2016.
DOI: 10.1007/978-3-319-42291-6_1

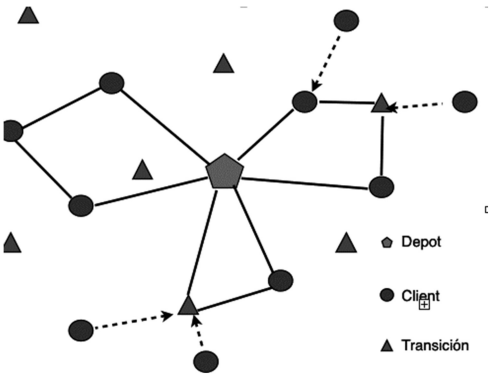

Fig. 1. An example of a feasible CmRSP solution

There are two variations of this problem, the ring star problem and the steiner ring star problem. In the ring star problem the steiner nodes are not available and the capacity constrain are not imposed. The problem consist in design a cycle through a subset of know customers looking for minimize the cycle length and the cost of assign a non-visited customer to the nearest customer visited. There are several algorithmic development for solving this problem, see [2–5].

On the other hand, the steiner ring star problem is a variation that has application in designing of digital data service networks. The problem consist in design a cycle over a set of steiner nodes and assign each customer to a node visited by a cycle. The objective is minimize the cycle and allocation costs. A branch and cut algorithm has been proposed to solve the problem [6].

The first work on this problem proposed two integer programming models inspired in classical routing formulations and introduced valid inequalities for improve the quality of the lower bounds, the problem was inspire in the design of a fiber optic communication network [1]. For solving the linear problem, authors develop a branch and cut approach and embebbed two heuristics procedure to speed up the convergence of the branch and cut algorithm.

Naji-Azimi et al. [7] proposed another heuristic based on linear programming and column generation scheme. The linear programming model selects the best point when a customer can be insert, and with the column generation scheme a reinsertion problem is solved. Some other methods are proposed for solving the problem, see [8–11].

The remainder of this paper is organized as follows: Sect. 2 present the description of the proposed algorithm based on VNS. Section 3 shows some numerical results from a set of instances of literature. Finally, Sect. 4 concludes this work and provides possible research directions.

2 Description of the Proposed Algorithm

Our algorithm is based on variable neighborhood search (VNS), for implementing the algorithm, we use the following premises:

(1) For representing the solution we use a vector that indicates the nodes on the ring and the assignation (2) We don't allow infeasible solutions in any iteration of the algorithm (3) In every moment of the algorithm, new solutions that improve the actual solution are accepted. For escaping to local optimal solutions we use a perturbation for begin in a new neighborhood (4) The stop criterion is when the algorithm can not find new best solutions.

The outline of the proposed algorithm is described in Table 1. In the following subsections, we give the details of each step.

Table 1. Proposed VNS algorithm for the CmRSP

```
current solution = initialization procedure ()
Best cost = cost (current solution)
Best solution = current solution
Stop criterion = 0
while Stop criterion=0
     current solution = perturbation within rings ()
     current solution = improve solution ()
     current solution = perturbation between rings ()
     current solution = improve solution ()
     current solution = addition of transition nodes ()
       current solution = resizing operator ()
       current solution = improve solution ()
       New cost = cost (current solution)
       New solution = current solution
       If New cost < Best cost then
             Best cost= New cost
             Best solution = New solution
             else
               current solution = random perturbation()
               New cost = cost (current solution)
               New solution = current solution
                      If New cost < Best cost then
                          Best cost= New cost
                          Best solution = New solution
                             Else
                             Stop criterion=1
                       end if
       end if
end while
```

2.1 Initialization Procedure

In this procedure we look for create a feasible solution using only the customers (not the transition points or steiner nodes) that generate the structure of the ring. To do so, we select the m customer as close as possible one from each other and then we define m rings by connecting each (selected) customer to the depot. After, each one of the remaining customer is added to the best feasible position, this is the closest customer that belongs to a ring.

2.2 Perturbation Operator Within Rings

Each one of the rings is consider separately and transform each ring as the traveling salesman problem (TSP). The algorithm consists in applied variable neighborhood search (VNS) to the nodes that belongs to a ring, in this case a TSP for finding a better solution. If a better solution is find, the change is accepted. In Fig. 2 an illustrative example is presented. The node 0 represents the depot, the swap is between the customer one and three, if the cost of the new ring is lower than the actual ring then the swap is accepted. Table 2 shows the pseudocode of the perturbation operator.

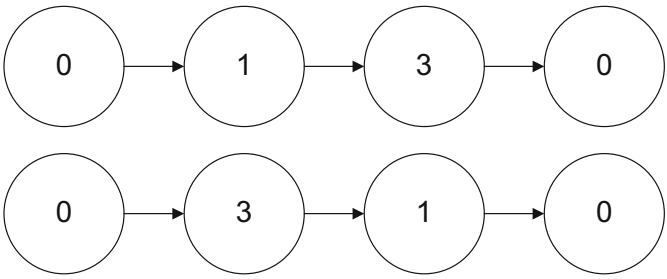

Fig. 2. An illustrative example: perturbation of a ring

2.3 Perturbation Operator Between Rings

Once the process of find better independent solutions for each ring is finished, we generate a perturbation or exchange between two rings. The exchange is made between two closest rings, if the new solution improves the oldest one, the new solution is updated. An illustrative example is presented in Fig. 3. Nodes 1 and 3 belongs to the first ring while nodes 4 and 5 belongs to the second ring. A pair of nodes is exchanged between the two rings generating a new solution that contains the nodes 4 and 3 in the first ring and nodes 1 and 5 are in the second ring. Table 3 shows the pseudocode of the perturbation operator.

Table 2. Pseudocode of swap method

```
Set m as the number of rings
Set n_m as the number of nodes in each ring
For i→ 1 to m
  For j→ 1 to n_m
    c→cost of the ring
    Swap two nodes without including the depot
    cnew→cost of the new ring
        if cnew<c then
          accept the new solution and save the new ring
        end
  end
end
```

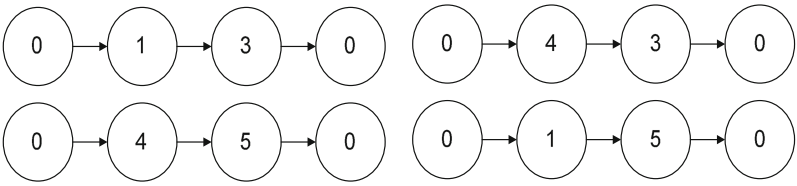

Fig. 3. An illustrative example: perturbation between rings

Table 3. Pseudocode of swap method between rings

```
Set m as the number of rings
For i→ 1 to m
   For j→ 1 to m
      c1→cost of the ring i
      c2→cost of the ring j
      Swap two nodes between the rings i and j
      cnew→cost of the new rings
      if cnew<c1+c2 then
         accept the new solution and save the new rings
      end
   end
end
```

2.4 Addition Operator of Transition Nodes

In each step of the iteration is evaluated if is convenient to replace a node of the ring for a transition node including the assigning of a customer to this transition node.

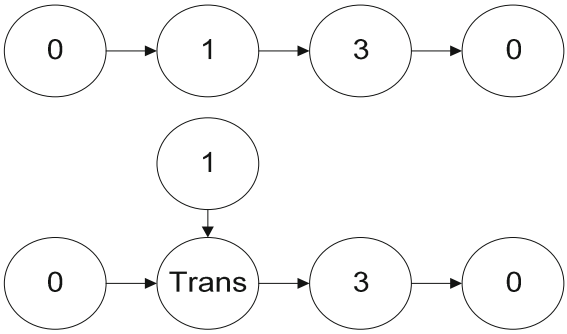

Fig. 4. An illustrative example: addition of transition nodes

Table 4. Pseudocode of adition of transition nodes

```
Set m as the number of rings
For i→ 1 to m
   For j→ 1 to m
      c1→cost of the ring i
      c2→cost of the ring j
      Swap two nodes between the rings i and j
      cnew→cost of the new rings
      if cnew<c1+c2 then
         accept the new solution and save the new rings
      end
   end
end
```

Figure 4 presents an illustrative example. Nodes 1 and 3 belongs to the ring and it is evaluated if a transition node can replace the node 1 and after assign the node 1 to the transition node. Table 4 shows the pseudocode of the perturbation operator.

2.5 Resizing Operator

The aim of this operator is to find a node in a ring that has the major cost of assignation, then extract the node of the actual ring, and reinsert in another ring, which present a less total cost. An illustrative example is presented in Fig. 5. Nodes 1, 3 and 4 belongs to ring 1 while node 5 belong to ring 2. After analyzing ring 1 the node 3 is the one with major cost of assigning, then is prove if this node can be assign in any position of ring 2. After evaluating the total cost this change can be done and the new ring 1 contains nodes 1 and 4 while ring 2 contains nodes 5 and 3. Table 5 shows the pseudocode of the resizing operator.

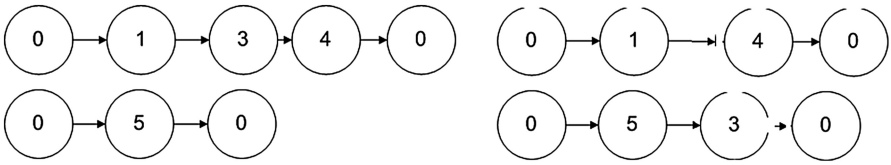

Fig. 5. An illustrative example: resizing operator

Table 5. Pseudocode of resizing operator

```
Set m as the number of rings
Set n_m as the number of nodes in each ring
For i→ 1 to m
  For j→ 1 to n_m
    Find the node with the highest cost
    Find a ring where the node has the lowest cost
    cnew→cost of the new rings
    if cnew<original cost then
       accept the new solution and save the new rings
    end
  end
end
```

2.6 Random Perturbation Operator

In order to avoid a local optimal solution, two random strategies are used for create a new solution. The first one generates a random number and select a random ring, with the random number is selected a random node and it is extract and reinsert in another random ring in a random position. The second one selects a random node of those which can be re assign in a transition node, then another random node is selected and is re assign the first node generating new rings. An illustrative example is presented in Fig. 6. Nodes 1, 4 and 5 belongs to ring 1 while nodes 3 and 2 belong to ring 2. After generate a random number on ring 1, node 4 is selected and reinserted in ring 2 between nodes 3 and 2 that is obtained by another random number.

3 Computational Experiments and Results

In order to test our algorithm we have used a well-known instances used in [1]. The results are presented in Table 6. The table is organized as follows; the first column presents the instance's name. The second column shows the best-known solution (BNS) reported in the literature. The Best solution (BSF) found by our algorithm is presented in column third, as our algorithm use a random perturbations, we run it twenty independent times and obtain the average solution on these runs (column four).

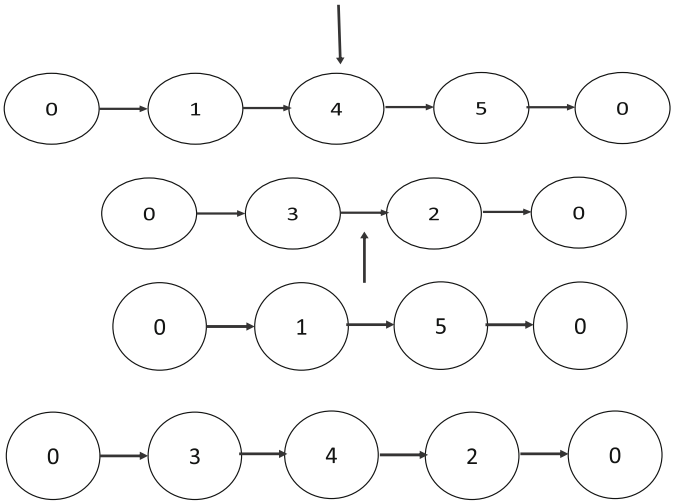

Fig. 6. An illustrative example: random perturbation operator

The standard deviation is presented in column five. The amount of computational time spend by our algorithm is presented in column six while the gap for the best solution obtained and the average gap of the twenty runs are on column six and seven respectively.

As we can see in Table 6, our algorithm is able to find the best solution know in the literature in all the instances except in the last one proving that our algorithm is efficient. For the instance that our algorithm cannot find the optimal solution, the gap is 9.75 %. The average GAP obtained in the twenty runs shows that is less than 3 %. Finally, the computational time of our algorithm is 2.96 s on average proving that it doesn't spend high computational times.

Table 6. Results of proposed method

Instance	BNS	BSF	Average	SD	Time (seconds)	GAP	Average GAP
eil26.tsp.3.12.5.A.BDS.cmrsp	242	242	242	0.00	1.11	0.00 %	0.00 %
eil26.tsp.3.25.10.B.BDS.cmrsp	2251	2251	2323.6	22.44	8.10	0.00 %	2.23 %
eil26.tsp.4.12.4.A.BDS.cmrsp	261	261	261	0.00	1.01	0.00 %	0.00 %
eil26.tsp.4.12.4.B.BDS.cmrsp	1827	1827	1827	0.00	0.79	0.00 %	0.00 %
eil26.tsp.4.18.5.A.BDS.cmrsp	339	339	343.4	3.16	1.20	0.00 %	1.30 %
eil26.tsp.4.18.5.B.BDS.cmrsp	2370	2370	2422	22.98	1.10	0.00 %	1.19 %
eil26.tsp.5.12.3.A.BDS.cmrsp	292	292	307.8	12.16	4.12	0.00 %	5.41 %
eil26.tsp.5.25.6.B.BDS.cmrsp	2674	2696	2745	31.44	5.20	0.82 %	2.66 %
eil51.tsp.3.12.5.A.BDS.cmrsp	242	242	242.4	0.69	1.92	0.00 %	0.17 %
eil51.tsp.5.50.12.B.BDS.cmrsp	3404	3736	3801.6	53.43	5.01	9.75 %	11.68 %
			Average	14.63	2.96	1.06 %	2.46 %

4 Conclusions

We have proposed a variable neighborhood search adaptation for solving the capacitated m-Ring-Star Problem. In order to avoid local optimal solutions, we have used random perturbations for avoid these local solutions and find new solutions close to another optimal local solutions or the global optimal solution.

We have used instances from the literature for testing our algorithm. The result shows the effectiveness of our algorithm in finding the optimal solution or closes one in a reasonable amount of computational time.

Future work should focus in to improve the decision-aid tool to allow speeding up the method. In addition, to decrease the computational time is possible to combine other techniques in order to decrease the computation time. Another real world constraints and characteristics can be explored as stochastic travel times, other objective functions, among others.

References

1. Baldacci, R., Dell'Amicoy, M., González, J.S.: The capacitated m-Ring-Star problem. Oper. Res. **55**, 1147–1162 (2007)
2. Calvetea, H.I., Galéy, C., Iranzo, J.A.: MEALS: a multiobjective evolutionary algorithm with local search for solving the bi-objective ring star problem. Eur. J. Oper. Res. **250**, 377–388 (2016)
3. Calvetea, H.I., Galéy, C., Iranzo, J.A.: An efficient evolutionary algorithm for the ring star problem. Eur. J. Oper. Res. **231**, 22–33 (2013)
4. Liefooghe, A., Jourdany, L., Talbi, E.G.: Metaheuristics and cooperative approaches for the bi-objective ring star problem. Comput. Oper. Res. **37**, 1033–1044 (2010)
5. Labbé, M., Laporte, G., Martíny, I., González, J.S.: The ring star problem polyhedral analysis and exact algorithm. Networks **43**, 177–189 (2004)
6. Lee, Y., Chiu, S.Y., Sanchez, J.: A branch and cut algorithm for the steiner ring star problem. Int. J. Manag. Sci. **4**, 21–34 (1998)
7. Naji-Azimi, Z., Salariy, M., Toth, P.: An integer linear programming based heuristic for the capacitated m-Ring-Star problem. Eur. J. Oper. Res. **217**, 17–25 (2012)
8. Azimi, Z.N., Salariy, M., Toth, P.: A heuristic procedure for the capacitated m-Ring-Star problem. Eur. J. Oper. Res. **207**, 1227–1234 (2010)
9. Hoshinoay, E.A., de Souza, C.C.: A branch-and-cut-and-price approach for the capacitated m-Ring-Star problem. Discrete Appl. Math. **160**, 2728–2741 (2012)
10. Hoshinoay, E.A., de Souza, C.C.: A branch-and-cut-and-price approach for the capacitated m-Ring-Star problem. Electron. Notes Discrete Math. **35**, 103–108 (2009)
11. Berinskyy, H., Zabala, P.: An integer linear programming formulation and branch-and-cut algorithm for the capacitated m-Ring-Star problem. Electron. Notes Discrete Math. **37**, 273–278 (2011)