A hybrid metaheuristic algorithm to solve the Capacitated *m*-Ring Star Problem

Antonio Mauttone, *Universidad de la República, Uruguay*
Sergio Nesmachnow, *Universidad de la República, Uruguay*
Alfredo Olivera, *Universidad de la República, Uruguay*
Franco Robledo, *Universidad de la República, Uruguay*
Email: {mauttone, sergion, aolivera, frobledo}@fing.edu.uy

Keywords: metaheuristics; network design; survivability.

1. Introduction

This work presents a metaheuristic approach to solve the Capacitated m-Ring Star Problem (CmRSP), a problem introduced by Baldacci et al. [1] which models the design of telecommunication networks with survivability properties. The CmRSP consists of finding a set of m cycles (rings), each of them including the central depot (the central telephone office), a subset of customers, and a set of optional nodes (secondary stations) used to diminish the costs of the network design. The rings must be node-disjoint (except for the central depot) in order to provide survivability to the network when node failure occurs. Customers that are not part of the rings must be directly connected to nodes in the rings. An additional constraint is that no ring (the cycle itself and pendants) can have more than Q customers, being Q a prefixed parameter (the ring capacity). The objective is to minimize the sum of routing and connection costs. The CmRSP is a NP-Hard problem, since it generalizes the *Traveling Salesman Problem*.

System survivability is the ability to provide service despite failures on some components [5]. Survivability is an important goal in the design of communication network backbones, to ensure that the topology is able to resist node malfunctions as well as connection lines failures. In this sense, telecommunication companies have particular interest in problems like the CmRSP, which may help to design large metropolitan optical fiber networks at minimum cost.

Baldacci et al. [1] introduced the CmRSP, proposing a branch-and-cut exact algorithm using inequalities from two integer programming formulations as cutting planes. The branch-and-cut algorithm was able to achieve highly accurate results on small and medium-size CmRSP instances. However, numerical results degrade when solving bigger scenarios, even employing up to two hours of execution time. Other similar problems have been previously proposed in the literature. A well-known particular case of the CmRSP is the *Ring Star Problem* (RSP) [7, 2], which consists of building a simple cycle to minimize the sum of routing and assignment costs. The *Steiner Ring Star Problem* (SRSP) is a variant of the RSP, where the goal is to find a minimum cost cycle considering only optional nodes (Steiner nodes) and the customers must be connected to exactly one node on the cycle. The *Median Cycle Problem* (MCP) [8, 9] consists of finding a simple cycle that minimizes the routing cost, while the assignment costs are bounded by a prefixed value. Other location models have also been used, where instead of a ring, a topology such as a path or a tree must be designed and the customers not integrated to this topology must be linked to it [6]. All these related problems have been widely solved using several algorithms, including metaheuristic approaches. However, due to its recent formulation, there have not been attempts to solve the CmRSP using metaheuristic algorithms. This article proposes the application of a combined GRASP and Tabu Search algorithm for solving the CmRSP, which was able to find accurate results in moderate execution times.

The rest of the paper is organized as follows. Next section presents the CmRSP formalization. The combined GRASP-Tabu Search algorithm to solve the CmRSP is described Section 3. Section 4 presents the discussion on the empirical analysis on applying the algorithm on a set of CmRSP benchmark instances, and summarizes the numerical results. Finally, the last section formulates the conclusions as well as main lines for future work.

2. Notation and CmRSP formalization

Consider a mixed graph $G = (V, E \cup A)$, where $V = \{0, n+1\} \cup V'$ is the node set, $E = \{\{i, j\} | i, j \in V, i \neq j\}$ is the edge set, and A is the arc set. Node set V' is composed of two subsets: U modelling the set of customers and W modelling the set of optional transit nodes (named *Steiner nodes*). Node 0 represents the depot and node $n+1$ is a copy of node 0, which is introduced to simplify the model. A path from node 0 to node $n+1$ is called a "ring" in the solution.

Given a customer node $i \in U$, $C_i \subseteq V'$ denotes the subset of nodes to which i can be connected. Let assume that $i \in C_i$, $\forall i \in U$ and that a customer will be connected to itself when it belongs to a ring. The set E is the set of feasible ring edges, whereas the set A models the feasible direct connections between customers and nodes, that is to say, $A = \{(i, j) | i \in U, j \in C_i\}$. Two types of cost structures are defined: the *routing costs* and *connection costs*. Each edge $\{i, j\} \in E$ has associated a non-negative routing cost c_{ij} and each arc $(i, j) \in A$ has associated a non-negative connection cost d_{ij}. Given a subset $E' \subset E$, $V(E')$ denotes the set of nodes which are end of some edge in E'.

A pair $R = (E', A')$ is called a ring if $E' \subset E$ is a simple path (without loops) from 0 to $n+1$ and $A' \subseteq A$ are arcs connecting customer nodes of $U \setminus V(E')$ and nodes of $V(E')$. Furthermore, a customer i is *assigned* to a ring R if it is either visited by the simple path (i.e. $i \in V(E')$) or there exists a node j on the path such that $(i, j) \in A'$ (i.e. i is directly connected to a ring node). A ring will be *feasible* if the number of customers assigned to it does not surpass the capacity Q.

The *cost* of a ring $R = (E', A')$ is the sum of the routing costs of the edges in E' plus the sum of the connection costs of the arcs in A'. An input parameter m indicates the number of rings in the network solution ($mQ \geq |U|$). The goal of the CmRSP is to build m rings so that each customer is assigned to exactly one ring, and the sum of the ring costs is minimum. Optional (Steiner) nodes of W can be used to diminish the solution cost.

3. GRASP and Tabu Search algorithm

GRASP and Tabu Search are well known metaheuristics that have been successfully used to solve many hard combinatorial optimization problems. GRASP [10] is an iterative process which operates in two phases. In the *Construction Phase* an initial feasible solution is built whose neighborhood is then explored in the *Local Search Phase*. Tabu Search [4] is a strategy to prevent local search algorithms getting trapped in locally optimal solutions. It operates accepting non improving moves and uses a penalization mechanism called *tabu list* to avoid returning to previously visited solutions. For a complete description of these methods the reader is referred to the works of Glover and Laguna [4] and Resende and Ribeiro [10].

Algorithm 1 shows the hybrid metaheuristic algorithm designed for tackling the CmRSP. The algorithm follows the generic two-phase global skeleton of a GRASP algorithm, incorporating the Tabu Search strategy in order to enhance the local search. Additionally, the local search includes a reinitialization procedure (named "Shaking") designed to provide diversity to the exploration method. The specific CmRSP algorithm is described in detail in the following subsections.

Algorithm 1 Hybrid metaheuristic pseudo-code.

```
1: Scenario_Initialization (G, C) // Read data from scenario files
2: Parameter_Initialization ()
3: for i = 1 to GRASP_iterations do
4:    G_sol = CmRSP_Construction_Phase(G, C, k);
5:    CmRSP_Local_Search(G, C, G_sol, Local_Search_iterations, Shaking_iterations);
6: end for
7: return G_sol;
```

Construction Phase

The construction phase of the proposed algorithm is depicted in Algorithm 2. The algorithm starts constructing iteratively m rings with exactly one customer each (lines 3–9). Step by step, it randomly chooses as candidates customers from the k farthest to the set X of nodes already in the solution (using as the distance from a customer to a set of nodes the Euclidean distance from the customer to the barycenter of the set). Each constructed ring is included in the initial solution. After that, (lines 10–12), the remaining $|U| - m$ customer nodes are iteratively added to that partial solution. Each customer is either inserted in a cycle or assigned to a node in a cycle, depending on which decision implies the minor increase in the solution cost. The order in which the remaining customers are added to the solution is randomized, which contributes to diversify the solutions built in this phase. The constructed solution $\mathcal{G}_{sol}$ is feasible by construction, and does not contain Steiner nodes (these will be suitably analyzed as potential improvers of the solution in the local search phase).

Algorithm 2 Construction Phase pseudo-code.

Procedure CmRSP_Construction_Phase(G, C, k**)**;

1: $\mathcal{G}_{sol} \leftarrow \{0\}$;
2: $X \leftarrow \{0\}$;
3: **for** $i = 1$ to m **do**
4: Let Y_k be the k customer nodes of $U \setminus X$ farthest to X on G;
5: $v_i \leftarrow \mathrm{SelectRandom}(Y_k)$;
6: $R_i \leftarrow$ the ring conformed by $\{0, v_i, n+1\}$;
7: $\mathcal{G}_{sol} \leftarrow \mathcal{G}_{sol} \cup R_i$;
8: $X \leftarrow X \cup \{v_i\}$;
9: **end for**
10: **for all** $u \in U \setminus X$ **do**
11: $\mathcal{G}_{sol} \leftarrow \mathrm{BestAssignment}(\mathcal{G}_{sol}, u)$;
12: **end for**
13: **return** $\mathcal{G}_{sol}$;

Local Search Phase

For the local search phase, the neighborhood of a feasible solution $\mathcal{G}_{sol}$ is defined as the set of feasible solutions that can be reached applying one of the following exploration moves:

- **insert**: given a node $v \in \mathcal{G}_{sol}$ and a ring $R \subseteq \mathcal{G}_{sol}$, remove v from its current position and insert it in the cycle of the ring R.
- **remove**: given a node $v \in \mathcal{G}_{sol}$ in the cycle of a ring $R \subseteq \mathcal{G}_{sol}$, remove v from R.
- **swap**: given two nodes u and v in two different cycles $C^1 \subset \mathcal{G}_{sol}$ and $C^2 \subset \mathcal{G}_{sol}$ respectively, replace u with v in C^1 and replace v with u in C^2.

After each application of the exploration moves, all the pendant customers (if any) are reassigned using a greedy heuristic. Making use of the search pattern defined by the presented exploration moves, only feasible solutions are analyzed during the search. This decision simplifies the algorithm, avoiding exploring (and thus penalizing or repairing) non-feasible CmRSP solutions.

A tabu list is used to store tabu moves, avoiding to explore those solutions already visited. When a move is applied, the involved nodes are not considered for further moves for the next θ iterations, where θ is randomly drawn each time in the interval $[\theta^{\min}, \theta^{\max}]$ (this interval is a parameter of the algorithm).

Algorithm 3 presents a pseudo-code of the local search phase. The best solution found is initialized with the solution delivered by the construction phase and the tabu list is initialized empty. At each local search iteration (lines 4–6), the best non tabu move is computed, applied to $\mathcal{G}_{sol}$ and after that it is used to update the tabu list TL. If the new solution improves the best solution found in this phase, a post-optimization procedure is applied (line 8) just before updating the best solution found $\mathcal{G}_{best}$. The post-optimization procedure is based on the Unstringing and Stringing (US) procedure used for the TSP [3] and consists of iteratively removing and inserting customers in the solution until no improvement can be attained by applying this operation.

In order to reduce the size of the exploration neighborhood (and thus improving the computational efficiency of the metaheuristic algorithm), the following criterion is applied to discards non-promising moves. An exploration move is *not* evaluated if all the arcs that it adds to the solution have cost greater than $\gamma\bar{c}$, where $\bar{c}$ is the average routing cost in $\mathcal{G}_{sol}$ and γ is a parameter. This criterion was introduced in the Granular Tabu Search [13] and it was shown that if γ is appropriately chosen, it may dramatically reduce the number of moves to evaluate (and thus, the execution time).

The local search mechanism showed itself as a powerful instrument for finding local optima in solution neighborhoods. However, it failed to provide enough diversity in the search, due to its deterministic behavior. This deficit works against finding the optimal values when solving CmRSP instances with complex topologies. Trying to avoid the lack of diversity, a new operator was incorporated into the local search procedure. The "Shaking" operator performs a three-stage "deconstruction and reconstruction" move, following the procedure which is described next. It starts by "deconstructing" (wiping it out) the ring with higher cost value already in the solution, leaving as "orphans" all nodes assigned to the ring, except one. After that, a measure evaluating the "attraction" of each surviving ring is calculated for each orphan node. This attraction measure is iteratively considered for rebuilding the solution, assigning each orphan node to the ring with higher attraction value. The shaking operator is applied when a stagnation situation is detected (i.e., when no improvement is found after a prefixed value of local search iterations). This cataclysmic operator allows changing the exploration neighborhood in a more drastic way, even preserving some characteristics of the original solution.

Algorithm 3 Local Search pseudo-code.

Procedure CmRSP_Local_Search($G, C, \mathcal{G}_{sol}, MaxIter, Shaking\_iterations$**)**;

1: $\mathcal{G}_{best} \leftarrow \mathcal{G}_{sol}$;
2: $TL \leftarrow \emptyset$;
3: **for** $iter = 1$ to $MaxIter$ **do**
4: $move \leftarrow$ Compute on $\mathcal{G}_{sol}$ the best move not in the tabu list TL;
5: $\mathcal{G}_{sol} \leftarrow$ Apply $move$ to $\mathcal{G}_{sol}$;
6: $TL \leftarrow$ Update the Tabu List TL;
7: **if** $\text{cost}(\mathcal{G}_{sol}) < \text{cost}(\mathcal{G}_{best})$ **then**
8: $\mathcal{G}_{sol} \leftarrow$ Apply Unstringing and Stringing to $\mathcal{G}_{sol}$;
9: $\mathcal{G}_{best} \leftarrow \mathcal{G}_{sol}$;
10: **end if**
11: **if** stagnation detected **then**
12: // no improvement found in the last $Shaking\_iterations$ iterations
13: Apply Shaking to $\mathcal{G}_{sol}$
14: **end if**
15: **end for**
16: **return** $\mathcal{G}_{best}$;

4. Computational results and discussion

The metaheuristic algorithm was evaluated using the set of 90 problem instances proposed by Baldacci et al. [1]. The instances are divided in two categories, having instances involving 25, 50, 75 and 100 nodes (categories A and B, regarding two different cost structures used over similar topologies). Preliminary experiments were made to tune the algorithm parameters. The number of GRASP iterations was fixed in 50 and each Local Search involved 250 iterations. The rest of the parameters were configured as follows: $k = n/10$, $\gamma = 1.25$, $[\theta^{\min}, \theta^{\max}] = [5, 10]$, and $Shaking\_iterations$ = 50. The algorithm was coded in C++ and was executed on a PC with a 2 GHz. processor and 1 GB of RAM, using the OpenSuse 10 Linux OS.

For each benchmark instance, 15 independent executions of the metaheuristic algorithm were performed. The solution values were compared to those reported in Baldacci et al. [1] using the well known GAP measure, defined by Equation 1, where z^{best} is the cost of the best solution found and LB a lower bound on the optimal value (obtained by Baldacci et al. [1] using an improved linear programming relaxation for the problem).

$$GAP = \frac{\|z^{best} - LB\|}{LB} \tag{1}$$

To compare the execution times, a *Time Ratio* was calculated dividing the execution time reported in the work used as a reference baseline [1] by the execution time of our algorithm. The results are summarized in Table 1, where the instances are grouped by category and number of nodes. The best, average and worst GAP values as well as the minimum and maximum Time Ratios are reported for each group of instances.

The GAP values obtained were below 8% for every problem instance solved. Moreover, average GAP values never exceeded 5%. These can be considered competitive results, considering that the values reported by Baldacci et al. [1] correspond to an exact algorithm which in most cases found optimal solutions. Regarding the execution times, significant improvements were obtained, especially for the bigger instances. However, it must be noticed that a fair execution time comparison is very difficult to attain. Finally, new best solutions were found for two instances, improving the best cost results achieved by Baldacci et al. [1]: for the instance **A26-n76-m4 (**$GAP = -0.43\%$**)** and for the instance **A29-n76-m4 (**$GAP = -0.19\%$**)**.

The *Time Ratio* values reported in Table 1 show that the heuristic algorithm is able to find accurate results in significantly lower execution times than those required by the method that produces the lower bound for the problem [1], specially when solving the most complex instances ($n = 76$ and $n = 101$).

Table 1 shows that the algorithm yields competitive results in terms of objective function values and execution times. Both the construction and the local search phase involve simple operations, which may be easily adapted to consider additional constraints. The metaheuristic algorithm was able to achieve promising results, especially when solving the most complex CmRSP instances. However, significant GAP values were achieved for some specific medium-size instances (i.e. B13-n51-m3, $GAP = 5.06\%$ and B21-n51-m5, $GAP = 5.35\%$), suggesting that the search get stuck in suboptimal solutions. This fact shows that there is still room to improve the quality of the obtained results, designing some mechanisms to increase the search diversity and allowing the algorithm to escape from local optima solutions.

Regarding these difficulties, the main lines of future work should be addressed to solve these important issues. Some enhancements can be made to the proposed algorithm in order to improve the accuracy and attain a high reduction of the execution times for the more complex instances. Allowing the local search to visit solutions that violate capacity constraints may simplify the constraint handling and lead to better results. Incorporating a long term memory as it is done in the Adaptive Memory Procedure [12], can also improve the quality of the results by providing a natural way of sharing information between the different GRASP iterations. Additionally, a parallel GRASP implementation should be able to take advantage of the complex structure of the CmRSP search space, by performing several concurrent explorations and exchanging information using the well-known Path Relinking technique [11], improving both the numerical results and the computational efficiency of the algorithm. Our group is working on these topics right now.

Group		GAP(%)			Time Ratio	
Class	n	best	average	worst	min	max
A	26	0.00	0.34	2.09	< 1	2
A	51	0.00	0.89	2.55	< 1	66
A	76	-0.43	1.51	3.47	3	208
A	101	0.28	2.04	4.71	< 1	194
B	26	0.55	1.24	3.29	< 1	< 1
B	51	0.88	3.43	5.35	< 1	49
B	76	1.42	3.19	5.99	4	121
B	101	3.20	4.24	6.18	1	70

Table 1: Summarized numerical results.

References

[1] R. Baldacci, M. Dell'Amico, and J.J. Salazar González, "The Capacitated m-Ring Star Problem", Technical Report DISMI 42, Università degli Studi di Modena e Reggio Emilia, 2003.

[2] T. C. S. Dias, G. F. de Sousa Filho, E. M. Macambira, Lucidio dos Anjos F. Cabral, and M. H. C. Fampa, "An Efficient Heuristic for the Ring Star Problem", In Proceedings of WEA 2006 (5th International Workshop on Experimental Algorithms), 2006.

[3] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem", Operations Research, Vol. 40, No. 6, pages 1086-1094, 1992.

[4] F. Glover and M. Laguna, "Tabu Search", Kluwer Academic Publishers, 1997.

[5] H. Kerivin, and A. Ridha Mahjoub, "Design of Survivable Networks: A survey", Networks, Vol. 46, No. 1, pages 1-21, 2005.

[6] M. Labbé, G. Laporte, and I. Rodriguez Martín, "Path, tree and cycle location", In T. G. Crainic and G. Laporte, editors, Fleet Management and Logistics, pages 187-204, Kluwer, 2002.

[7] M. Labbé, G. Laporte, I. Rodriguez Martín, and J. J. Salazar González, "The Ring Star Problem: Polyhedral Analysis and Exact Algorithm", Networks Vol. 43. No. 3, pages 177-189, 2004.

[8] M. Labbé, G. Laporte, I. Rodriguez Martín, and J. J. Salazar González, "Locating Median Cycles in Networks", European Journal of Operations Research, Vol. 160, pages 457-470, 2005.

[9] J.M. Pérez, M. Moreno-Vega, and I. Rodriguez Martín, "Variable Neighbourhood Tabu Search and its applications to the Median Cycle Problem", European Journal of Operations Research, Vol. 151, pages 365-378, 2003.

[10] M. G. C. Resende and C. C. Ribeiro, "Greedy Randomized Adaptive Search Procedures", In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, pages 219-249. Kluwer Academic Publishers, 2003.

[11] M.G.C. Resende and C.C. Ribeiro, "Parallel Greedy Randomized Adaptive Search Procedures", In E. Alba, editor, "Parallel Metaheuristics: A new class of algorithms," John Wiley and Sons, pages 315-346, 2005 .

[12] Y. Rochat and E. Taillard, "Probalistic diversification and intensification in local search for vehicle routing", Journal of Heuristics Vol. 1, pages 147-167, 1995.

[13] P. Toth and D. Vigo, "The Granular Tabu Search and Its Application to the Vehicle-Routing Problem", INFORMS Journal of Computing, Vol. 15, No. 4, pages 333-346, 2003.