A Hybrid Metaheuristic for Routing on Multicast Networks

Carlos A.S. Oliveira and Panos M. Pardalos

Abstract Multicast routing systems have the objective of simultaneously transferring data to multiple destination nodes while using a single "push" operation. This leads to cost savings associated with reduced bandwidth utilization, which results from a decrease in data duplication across network links. An important problem on multicast networks, known as the delay constrained multicast routing problem (DCMRP), asks for the determination of an optimal route for packet transfers between members of a multicast group. Several heuristics have been proposed in the last few years to solve the DCMRP, which is of great interest for telecommunication engineers. In this paper we propose a novel, hybrid metaheuristic approach for the DCMRP, where a greedy randomized adaptive search procedure is used along with variable neighborhood search algorithm to find near optimal solutions. Computational experiments show that the proposed technique provides superior solution quality, while it is also efficient in terms of the use of computational resources.

1 Introduction

Multicast services have been used in modern network applications to allow direct communication between a source node and a set of receivers, referred to as multicast destinations [2, 13]. In recent years, the number of applications of multicasting has increased steadily, following the rapid advances in the availability and use of the Internet as well as intranets in the corporate world. Multicast networks are known

C.A.S. Oliveira (✉)
Quantitative Research Dept., F-Squared Inc., Princeton, NJ, USA
e-mail: oliveira@ufl.edu

P.M. Pardalos
Department of Industrial and Systems Engineering, Center for Applied Optimization, University of Florida, Gainesville, FL, 32608, USA

Laboratory of Algorithms and Technologies for Networks Analysis, National Research University Higher School of Economics, Nizhny Novgorod, 603155, Russia
e-mail: pardalos@ufl.edu

M.V. Batsyn et al. (eds.), *Models, Algorithms and Technologies for Network Analysis: From the Third International Conference on Network Analysis*, Springer Proceedings in Mathematics & Statistics 104, DOI 10.1007/978-3-319-09758-9_9

to provide robust and efficient data delivery for a wide spectrum of applications, including video-on-demand, groupware, and data streaming, among others [14, 22].

A number of algorithmic issues, however, remain as a major problem for the wide adoption of multicasting networks. For example, routing is an issue that has not been completely resolved on such systems due to the high computational cost of exact algorithms. While for traditional unicast systems the routing problem can be solved in polynomial time using well-known methods such as the Dijkstra's algorithm [3], multicast routing is better modeled by the Steiner tree problem, which is one of the basic NP-hard problems [7, 17].

Given the inherent complexity of exact approaches to the DCMRP, a large number of heuristics have been proposed to find good, if non-optimal, solutions that can be calculated in polynomial time [2, 6, 10, 11, 14]. However, many of these local search methods proposed in the telecommunications engineering literature suffer from a lack of optimality guarantees and may be easily trapped into local optima [19–21, 23]. As such, these methods are indicated only for application to small- to medium-scale instances. On the other hand, applications of the DCMRP become even more challenging for instances with a large number of multicast members, since the efficient use of resources turns into a critical factor for the success of such network implementations.

In this paper we propose a metaheuristic solution for the delay constrained multicast routing problem. In particular, we propose a new method for computing routing trees for multicast networks using a hybridization of greedy randomized adaptive search procedure (GRASP) and variable neighborhood search (VNS). The strategy here is to improve the performance of the algorithm by avoiding spending too much time exploring suboptimal solutions and their solution spaces.

At the same time, our contribution may be extended to the general application of a hybrid GRASP metaheuristic. By combining the general structure of GRASP with VNS, the result is a novel search algorithm that may be used to produce fast implementations for several related problems.

The paper is organized as follows. In the next section (Sect. 2), we provide a detailed definition for the DCMRP, using graph theoretical concepts as well as a mathematical programming model. Then, in Sect. 3 we develop a GRASP metaheuristic for the DCMRP, followed by a description of the VNS strategy employed. We present computational results for our approach in Sect. 4, and finally some concluding remarks are provided in Sect. 5.

2 Delay-Constrained Multicast Routing

Multicast networks have been designed with the explicit goal of allowing fast data transmission from a source node to a set of destinations, while using a single send operation. This is made possible by sending data only once over a network link whenever one or more destinations have requested the same content. A set of nodes interested in a particular piece of data is called a *multicast group*. The main task

faced in the operation of a multicast network consists of delivering the requested data to all members of a multicast group. To accomplish this goal, the system needs to determine a set of routes connecting sources to destinations.

Let $G = (V, E)$ be a graph where V is the set of nodes and E a set of links connecting adjacent nodes. The source is denoted by s with destinations $D = \{d_1, \ldots, d_k\}$, such that $D \subset V$. The cost function $c : E \to Z_+$ represents link costs and the delay function $\tau : E \to Z_+$ returns the time $\tau(e)$ elapsed when traversing edge $e \in E$. We also denote by $c(E')$ the cost associated with a set of edges $E' \subseteq E$, that is, $c(E') = \sum_{e \in E'} c(e)$. Similarly, we denote the time delay for path $\mathscr{P}$ in G as $\tau(\mathscr{P}) = \sum_{e \in \mathscr{P}} \tau(e)$.

The DCMRP asks for a set of edges $E' \subseteq E$ such that s is connected to every node $d \in D$ on $G' = (V, E')$, and the maximum acceptable delay Δ_d at destination d is a constant, i.e.,

$$\sum_{e \in \mathscr{P}_d} \tau(e) \leq \Delta_d, \quad \text{for every destination } d \in D,$$

where $\mathscr{P}_d$ is the path induced by E' in G, connecting s to d. Moreover, we require that the total cost $\sum_{e \in E'} c(e)$ of the subset E' be minimum.

Additionally, real-world instances of the DCMRC problem frequently have extra requirements on the kind of paths that can be used to connect sources and destinations [2]. For example, the model may require that a minimum capacity be available for each edge selected in the final solution. We will consider variations of this problem in the next sections. First, let us define a formal MIP model for the problem.

2.1 MIP Model for the DCMRP

A mixed integer programming (MIP) model for the DCMRP can be described as follows. Let $x_i \in \{0, 1\}$, for $i \in \{1 \ldots |E|\}$, be a decision variable that is 1 whenever an edge is part of the routing tree and 0 otherwise. Then, the objective function can be written as a minimization problem over the vector x, while considering the cost of each edge:

$$\min \sum_{e_i \in E} x_i c(e_i),$$

where $c : E \to \mathbb{R}$ is the cost function as described above. Then, we need a set of constraints that guarantee the connectedness of the solution set $\{(u, v) \in E : x_j = 1\}$:

$$\sum_{e_i = (v,w) \in U \times V} x_i \geq 1 \qquad \forall \text{ partitions } U, W \text{ s. t. } |(S \cup D) \cap U| \text{ is odd.}$$

That is, there is at least one link connecting each partition of V where the number of sources and destinations is different. Next, we have constraints that indicate the boundedness of the delay.

$$\sum_{e_i \in E} y_i^v \tau(e_i) \leq \Delta_v \qquad \text{for } v \in D$$

where y_i^v for $e_i \in E$ is an indicator variable with value 1 whenever the edge e_i is part of a path from source to destination $v \in D$. The variables y_i can be 1 only if link e_i is part of the solution, so we also have

$$y_i \leq x_i \qquad \text{for all } e_i \in E.$$

Finally, we need to apply the standard integrality constraints to our model variables:

$$y_i \in \{0, 1\} \qquad \text{for } e_i \in E$$

$$x_i \in \{0, 1\} \qquad \text{for } e_i \in E$$

2.2 DCMRP and the Steiner Problem

The minimum routing cost problem as described above has close connections to the minimum cost Steiner tree problem. In graph theory, a tree that connects a set of required nodes, while using other nodes only if necessary, is called a Steiner tree [8, 16]. Thus, we can restate the problem as that of finding a minimum cost Steiner tree such that maximum delay restrictions are also satisfied.

In the Steiner problem, one is given a graph $G = (V, E)$ together with a cost function $c : E \to Z_+$, and a set $R \subset V$ of required nodes. The nodes in $V \setminus R$ are called Steiner nodes. The objective is to find a tree T linking the nodes in D, passing through Steiner nodes $(V \setminus R)$ if necessary, such that the cost $\sum_{e \in T} c(e)$ is minimized. The Steiner problem on graphs is well known to be NP-hard [7].

Consider the following transformation from instances of the Steiner problem to the DCMRP. Given an instance of the minimum cost Steiner tree problem, let us construct an instance of DCMRP using the same underlying graph. Select a node among the required nodes to become the source and let the remaining required nodes be destinations. Then, set $\Delta_d \leftarrow \infty$, for all $d \in D$. As can be easily confirmed, an optimal solution to the transformed problem will also be a solution to the original instance of the Steiner problem. Conversely, an optimal solution for the original instance will give an optimal solution for the transformed instance. This argument shows that the DCMRP is also NP-hard.

Given the computational complexity of the DCMRP, it is extremely difficult to solve general large-scale instances of the problem. However, our goal is to devise algorithms that can provide near optimal solutions for typical instances of

```
Read instance data
Initialize GRASP data structures
S* ← ∅
while termination criterion not satisfied do
    S ← new greedy randomized solution
    S ← LocalSearch(s)
    if cost(S) < cost(S*) then
        S* ← S
    end
end
return S*
```

Algorithm 1: Generic GRASP algorithm

the problem. Moreover, we want such algorithms to perform efficiently, returning feasible solutions quickly and avoiding getting stuck in local optima.

3 GRASP Approach for DCMRP

Most modern metaheuristic techniques are based on finding solutions close to the global optimum with the help of gradient methods combined with randomization rules, which are designed to avoid local optima. Metaheuristics main contribution is on the development of intelligent strategies for mixing existing non-optimal techniques and algorithms. Such metaheuristics use similar principles in slightly different ways. A conceivable goal for the algorithm designer is, therefore, to borrow techniques from different metaheuristics, in order to create algorithms that better reflect the characteristics of the problem at hand. In this paper we use some of the techniques proposed by GRASP metaheuristic as well as by the variable neighborhood search metaheuristic (VNS) to efficiently solve the DCMRP.

Greedy Randomized Adaptive Search Algorithm (GRASP), proposed by Feo and Resende [4], aims at finding near optimal solutions for combinatorial optimization problems. It is composed of a number of iterations, where a new solution is picked from the available feasible set, using a greedy construction algorithm. The initial solution is subsequently improved using some local search method. GRASP has been very successful in a number of applications such as QAP [15], Frequency Assignment [9], Satisfiability, and many others [5]. The steps of standard GRASP are summarized in Algorithm 1.

The GRASP algorithm is a multi-start method, where a new solution is constructed, and subsequently improved. In the construction phase, at each iteration the algorithm tries to add a new element using a randomized greedy strategy. The second phase is concerned with improving the current solution. Its goal is to achieve a local optimum state by performing a local search, which is usually based on a gradient decent strategy. In the next sections, we describe the approach used in the GRASP implementation for DCMRP.

$S \leftarrow \emptyset$
while *solution* s *is not feasible* **do**
  Sort $e_1, \ldots, e_k$ using greedy function
  Select a random α, such that $0 < \alpha \leq k$
  $R \leftarrow \{e_1, \ldots, e_\alpha\}$
  $e \leftarrow$ arbitrary selected element of R
  $S \leftarrow S \cup \{e\}$
end
return S^*

Algorithm 2: Generic GRASP constructor

3.1 GRASP Constructor

The construction phase of the GRASP algorithm is responsible for creating a new solution, using a greedy randomized strategy. The basic idea behind greedy randomization is to add elements to the solution according to a greed criterion. However, the element that is chosen at each step to compose the new solution is not necessarily the best available element. Instead, the selection is taken randomly from a subset of best available elements, which have been previously sorted using a greedy objective. The general algorithm is displayed in Fig. 2. In the algorithm, the subset of best elements is called a restricted candidate list (RCL). The RCL is created and used to track the best elements available to be added to the current solution.

3.2 Speeding Up the GRASP Constructor

To be effective as the construction phase in a GRASP algorithm, it is desirable that the construction method be very fast. A construction algorithm that is not quick enough may become a bottleneck for the whole algorithm, since it needs to be executed every time a new solution is desired. While the traditional approach for GRASP construction works well, it requires the maintenance of an additional data structure, which needs time to build, and as a result it can waste computational resources. Instead, we use the following observation proved in [14].

Observation 1 *Let* $x_1, \ldots, x_n$ *be an unordered sequence, and* $y_1, \ldots, y_n$ *the corresponding ordered sequence. Then, to find a random element among the* $y_1, \ldots, y_\alpha$, *for* $0 < \alpha \leq n$ *is on average equivalent to selecting the best of* α *random elements of* $x_1, \ldots, x_n$.

The observation above gives a very efficient way of implementing the RCL test, which gives us, on average, the same results. Start with the full set C of candidate elements. Then, at each step generate a value of α, and pick at random $k = 1/\alpha$

```
Input: parameter α, instance size N
S ← ∅
while S is an incomplete solution do
    α′ ← uniform(0, α)
    k ← N/α;
    c ← ∞ (−∞ for maximization problems)
    for j ∈ {1, . . . , k} do
        C_j ← set of candidates at this iteration
        x ← arbitrary element from C_j
        if c(x) < c then
            c ← c(x)
            y ← x
        end
    end

    S ← S ∪ {y}
end

return S
```

Algorithm 3: Improved construction phase for GRASP

elements of C. From the picked elements, store only the one that is the best fit for the greedy function. This method is shown in Algorithm 3.

A clear advantage in terms of computational complexity is achieved by the proposed construction method for GRASP. The greatest advantage is that, while in the original technique the candidate elements must be sorted, this is not necessary in the proposed algorithm. Moreover, the complexity of traditional construction is dependent on the number of candidate elements. In our method, the complexity is constant for a fixed value of α. For example, if alpha is $n/2$, then we need just two iterations to find an element in the RCL, with high probability.

Theorem 1 ([14]). *The complexity of selecting elements from the RCL in the modified construction algorithm is* $n \log n$.

3.3 GRASP Construction Phase for DCMRC

We proceed to describe how to find a solution for the DCMRP that will be used in the construction phase of the GRASP algorithm. The construction algorithm is composed of several steps, in which we build a spanning tree that contains all the nodes in the required set of sources and destinations, along with other nodes required as intermediaries.

The first part of the construction phase is to compute paths connecting the source to each of the destinations $d \in D$. This is done using a randomized version of Dijkstra's algorithm, which finds shortest paths from a source to a single destination.

$S \leftarrow \emptyset$
while *there is* $d \in D$ *that is unconnected in* S **do**
  $\mathscr{P} \leftarrow$ randomized shortest path from s to d
  $S \leftarrow S \cup \mathscr{P}$
end
while *the is at least one cycle in* s **do**
  $e' \leftarrow \arg\max_{e \in s} c(e)$ s.t. e is contained in a cycle
  $e'' \leftarrow \arg\max_{e \in s \setminus e'} c(e)$ s.t. e is contained in a cycle
  $\hat{e} \leftarrow e'$ with prob. p, otherwise e''
  $S \leftarrow S \setminus \hat{e}$
end
return S

Algorithm 4: GRASP construction for DCMRP

The adaptation necessary here is such that the shortest path is updated only with high probability p. This guarantees that the solution found in any two executions of the algorithm will be close to the optimum, but still with random differences that make the result useful for our stochastic optimization methods. The randomized shortest path is then used to create an initial solution as shown in Algorithm 4.

The first part of Algorithm 4 guarantees that the solution created is connected, by finding a separate path from the source to each destination $d \in D$. At the end of this phase, we will have a solution where separate paths may result in cycles. Therefore, the second phase of the construction algorithm aims at removing such cycles. At each step, it finds the two highest cost edges contained in a cycle. Then, the algorithm removes one of them in an arbitrary way. The goal is to reduce the cost of the final solution, while at the same time allowing for randomized results. The resulting solution is returned at the end of Algorithm 4 to be later used by GRASP.

3.4 Variable Neighborhood Search

Once a feasible solution has been created by GRASP, the next step of our algorithm is to try to improve its quality using a local search procedure. Traditionally, local search has been performed using gradient descent techniques, which try to incrementally improve a solution until a local minimum is reached. The disadvantage of such methods, however, is that they can quickly become stuck in a local neighborhood, hindering the computational effort employed during the search phase. We try to avoid this behavior by using instead an alternative search technique based on variable neighborhood search (VNS).

VNS is a metaheuristic programming technique that has be successfully used to solve several combinatorial optimization problems [12]. Its main approach is to perform local search using successively larger neighborhoods, until no more improvements can be found by increasing the neighborhood size up to a given parameter. The advantage of VNS is that it will not stop once the first local optimum

Input: current solution S
$S^* \leftarrow s$ /* initialize best solution */
$N \leftarrow 1$ /* set distance to 1 */
while $N < K$ **do**
  $N \leftarrow N + 1$
  while *improvement found in last* δ *iterations* **do**
    $u, v \leftarrow$ random pair or nodes in S such that $d_S(u, v) = N$
    $\mathscr{P} \leftarrow$ randomized shortest path $u \rightarrow v$ in G
    $S' \leftarrow S$ with $\mathscr{P}$ in place of path $u \rightarrow v$
    if $c(S') < c(S^*)$ **then**
      $S^* \leftarrow S'$
    end
  end
end
return S^*

Algorithm 5: Variable neighborhood search

has been found. Instead, it will continue to build better solutions by reaching more distant neighborhoods in a organized fashion, until it cannot find any additional improvements. By reaching these more distant neighborhoods, VNS can more easily avoid being trapped in the same optimal solution, therefore yielding better results than standard local search.

The VNS algorithm for the DCMRP tries to replace existing sub-paths in the current solution by using alternate paths that have the potential for improving the objective function. This is done by arbitrarily selecting a pair of nodes occurring in the existing solution and replacing its induced sub-path by a new path, found using a shortest path algorithm. For this purpose, one can use a method such as Dijkstra's shortest-path procedure, which will hopefully provide a local improvement to the existing solution.

It is also possible to use randomized shortest-paths, similarly to how these paths are calculated in the GRASP constructor, where the best path is updated according to a probability distribution, so that the best path is not always selected. In that case, possible improvements will come from the exploration of the neighborhood of the existing solution, although the local-optimality of this improvement is not guaranteed in the same form as when using a completely greedy procedure.

The difference between VNS compared to other local search strategies resides in the ability to change the underlying neighborhood structure as a new local minimum is found. In our case, the neighborhood is changed by increasing a reach parameter, so that larger subpaths are substituted by the algorithm. This kind of change will happen until the maximum of N, defined by the parameter K, is reached. The general approach used in our VNS implementation is provided in Algorithm 5.

The algorithm starts by defining the parameter N, which is interpreted as the distance between nodes for which a new path will be searched. For the first iteration, this parameter is set to value 1, and in this case only nodes that are neighbors in the current solution are substituted by new paths. This process is repeated as long as

we are able to perform improvements in the existing solution. A limit of δ moves without improvement is allowed before the inner part of the algorithm is interrupted. Then, the parameter N is increased by one and the process restarts.

The search is completed only when N reaches the previously defined upper limit K. At this point, the algorithm has systemically investigated all paths that might improve the current solution by replacement of paths occurring between pairs of nodes. When that happens, we return to the upper-level GRASP process the best solution found during all iterations, which will be used as the new optimum solution.

3.5 Path Relinking

GRASP has the advantage of being easy to develop, since it is composed of relatively independent procedures (the constructor and local search phases). It is well suited for applications with existing heuristic algorithms, that can be combined with GRASP to find a better solution.

However, one of the weaknesses of GRASP is its incapacity to integrate good solutions found previously into the current search iteration. Since each iteration will create a completely different solution, there is no information added to the system when a good solution is found.

A method that has been used lately to overcome this problem is called *path relinking* (PR) [1,18]. In PR, a subset of the best solutions found is kept in a separate memory, called the *elite set*. At each iteration, one of the solutions s will be selected, and a process of comparing the current solution with s will start. Each component of the solution will be changed to the corresponding value on s, and after this a local search will be initiated to check for local optimality.

The disadvantage of PR is the large time it takes to run, in comparison with the rest of the GRASP algorithm. This, in practice, has been a restricting factor in the use of PR on practical applications. Although PR brings a relative boost in solution quality on each iteration, the effect can be negative since the number of iterations may be reduced due to the added complexity. Therefore, we need to use experimental data to determine the best trade-off between computational time and quality of results produced by path relinking.

The path relinking implementation used to solve the DCMRP is described in Algorithm 6. The first step of path relinking is to create a set of elite elements, denoted by $\mathscr{E}$. The maximum size of this set is given by ϵ, therefore for the first ϵ iterations we simply run the existing GRASP algorithm and add the resulting to solution to the elite set.

When the elite set is complete, we start using it to perform improvements to the solution generated by GRASP. This is depicted in the second `while` loop in Algorithm 6. The termination criterion is usually based on number of iterations, time, or the distance to a known lower bound. The first step of the loop is to create a new solution using GRASP. Then, an element of $\mathscr{E}$ is arbitrarily selected and subsequently used during the path relinking process. For each path $\mathscr{P} \in s$ going

```
ℰ ← ∅
while |ℰ| < ϵ do
    S ← GRASP
    ℰ ← ℰ ∪ {S}
end
while termination criteria not satisfied do
    S ← GRASP
    S′ ← arbitrary element of ℰ
    for each path 𝒫 ∈ S′ do
        S ← S ∪ 𝒫
        while s has a cycle do
            e ← arg max_e c(e) such that S \ e is connected
            S ← S \ e
        end
    end
    /* Update Elite Set ℰ */
    γ* ← arg max_{γ∈ℰ} c(γ)
    if c(s) < c(γ*) then
        ℰ ← (ℰ ∪ {s}) \ γ*
    end
end
return γ*
```

Algorithm 6: Path relinking improvement phase

from source to destination, the algorithm will try to replace the edges of the existing solutions with the edges of the same path in the elite solution. This is performed in the following way: first, we calculate the union of the two edge sets. Then, we proceed to remove redundant edges in a greedy fashion. That is, for each high cost edge, we check if we can remove it while maintaining a connected solution. This process continues until there are no cycles in the resulting solution.

The last step of the algorithm loop is to update the elite set. For this purpose, we retrieve the worst solution γ in the elite pool. If the current solution s improves on the cost of γ, then we replace γ with s in the elite set. These steps are then performed until a predefined termination criterion is satisfied.

4 Computational Results

To test the quality of the proposed metaheuristic, we designed a set of DCMRP instances. The instances range in size from 40 to 100 nodes, which is representative of medium-size problems occurring in large companies or in clusters of medium-sized organizations. Edges have been added to these test networks with costs that are uniformly distributed between 1 and 10. The distances are assumed to be Euclidian, but can be easily adapted to other metrics such as Manhattan distances (Fig. 1).

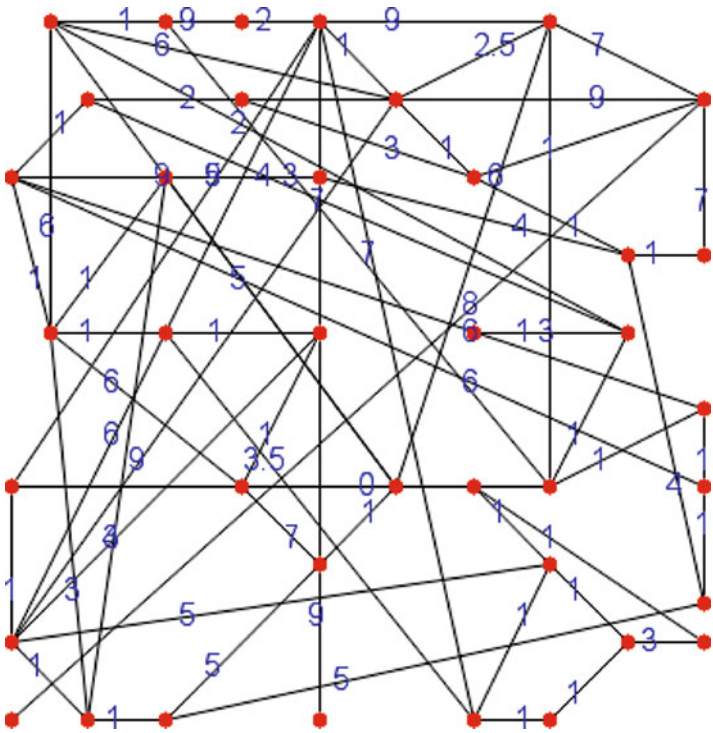

Fig. 1 Drawing of an instance with 40 nodes depicting node positions and costs between nodes

We run the proposed algorithm 10 times for each of the tested instances. We report on the average of these executions, to account for random fluctuations between different runs. The results are illustrated in Table 1. In this table, the first two columns give the number of nodes and edges in the network. The next three columns display the average best solution found by the GRASP without enhancements, the GRASP plus VNS strategy, and finally the GRASP with VNS enhanced with the PR method.

An area that we tested in the GRASP implementation just shown was the relative contribution of having two improvement methods (traditional local search and VNS) as components of the metaheuristic. As can be seen from the results, VNS is able to improve the quality of results in most of the cases, which shows that the use of multiple neighborhoods can provide a boost in efficiency for the algorithm.

Table 1 Experimental results of the GRASP and VNS metaheuristic implementation for the DCMRC

n	m	Best GRASP	Best GRASP+VNS	Best GRASP+VNS+PR	Time (s)
40	68	83	82	82	4.2
45	79	161	161	161	4.9
50	97	238	235	235	5.9
55	126	262	257	255	5.7
60	153	385	376	373	7.2
65	189	823	814	814	7.5
70	213	647	644	641	8.9
75	252	743	741	740	9.3
80	346	722	709	709	10.4
85	402	827	815	812	10.7
90	522	823	808	804	12.5
95	693	1028	1016	1009	13.3
100	816	1317	1311	1279	18.2

5 Concluding Remarks

In this paper, we presented a metaheuristic approach to solve the DCMRP, a problem arising on multicast routing systems, where the goal is to provide quick and accurate routing services to a set of source and destination points. Due to its occurrence in the design of telecommunication networks, the DCMRP has become the focus of intense research in the last few years. Our main contribution is the use of a fast construction algorithm along with an improvement method based on variable neighborhood search. The VNS has been used to enhance GRASP results, making it possible to explore existing solutions even faster.

The results of our experiments show that this method provides high quality solutions for realistic instances of the problem. The elegance of the method used also means that it can be easily incorporated to other algorithms for the DCMRP and related problems. In future research, it would be interesting to investigate the use of other intensification strategies for the proposed algorithm, such as improving the basic path relinking scheme used in this paper.

References

1. Aiex, R.M., Binato, S., Resende, M.G.C.: Parallel GRASP with path-relinking for job shop scheduling. Parallel Comput. **29**, 393–430 (2003)
2. Ballardie, A., Francis, P., Crowcroft, J.: Core-based trees (CBT) – an architecture for scalable inter-domain multicast routing. Comput. Comm. Rev. **23**(4), 85–95 (1993)

3. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1), 269–271 (1959)
4. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Global Optim. **6**, 109–133 (1995)
5. Festa, P., Resende, M.G.C.: An annotated bibliography of grasp, part ii: Applications. Int. Trans. Oper. Res. **16**, 131–172 (2009)
6. Ganjam, A., Zhang, H.: Internet multicast video delivery. Proc. IEEE **93**(1), 159–170 (2005)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company, San Francisco (1979)
8. Gilbert, E.N., Pollak, H.O.: Steiner minimal trees. SIAM J. Appl. Math. **16**, 1–29 (1968)
9. Gomes, F.C., Pardalos, P.M., Oliveira, C.A.S., Resende, M.G.C.: Reactive GRASP with path relinking for channel assignment in mobile phone networks. In: Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 60–67. ACM Press, New York (2001)
10. Hong, S., Lee, H., Park, B.H.: An efficient multicast routing algorithm for delay-sensitive applications with dynamic membership. In: Proceedings of IEEE INFOCOM'98, pp. 1433–1440 (1998)
11. Kompella, V.P., Pasquale, J.C., Polyzos, G.C.: Optimal multicast routing with quality of service constraints. J. Network Syst. Manag. **4**(2), 107–131 (1996)
12. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
13. Oliveira, C.A.S., Pardalos, P.M.: A survey of combinatorial optimization problems in multicast routing. Comput. Oper. Res. **32**(8), 1953–1981 (2005)
14. Oliveira, C.A.S., Pardalos, P.M.: Mathematical Aspects of Network Routing Optimization. Springer, New York (2010)
15. Oliveira, C.A.S., Pardalos, P.M., Resende, M.G.C.: GRASP with path-relinking for the QAP. In: 5th Metaheuristics International Conference, pp. 57.1–57.6, Kyoto, Japan, August (2003)
16. Pardalos, P.M., Du, D.-Z., Lu, B., Ngo, H.: Steiner tree problems. In: Floudas, C.A., Pardalos, P.M. (eds.), Encyclopedia of Optimization, vol. 5, pp. 277–290. Kluwer Academic, Dordrecht (2001)
17. Pardalos, P.M., Khoury, B.: A heuristic for the steiner problem on graphs. Comput. Optim. Appl. **6**, 5–14 (1996)
18. Resende, M.G.C., Ribeiro, C.C.: A GRASP with path-relinking for private virtual circuit routing. Networks **41**, 104–114 (2003)
19. Salama, H., Reeves, D., Viniotis, Y.: A distributed algorithm for delay-constrained unicast routing. In: Proc. IEEE INFOCOM'97, Kobe, Japan (1997)
20. Sriram, R., Manimaran, G., Siva Ram Murthy, C.: A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees. IEEE/ACM Trans. Network. **7**(4), 514–529 (1999)
21. Yang, D.-N., Liao, W.J.: On bandwidth-efficient overlay multicast. IEEE Trans. Parallel Distr. Syst. **18**(11), 1503–1515 (2007)
22. Yang, Y., Wang, J., Yang, M.: A service-centric multicast architecture and routing protocol. IEEE Trans. Parallel Distr. Syst. **19**(1), 35–51 (2008)
23. Zhu, Q., Parsa, M., Garcia-Luna-Aceves, J.J.: A source-based algorithm for delay-constrained minimum-cost multicasting. In: Proc. IEEE INFOCOM95, pp. 377–385 (1995)