A parallel GRASP heuristic for the 2-path network design problem

Celso C. Ribeiro[1] and Isabel Rosseti[1]

Department of Computer Science, Catholic University of Rio de Janeiro, Rua Marquês de São Vicente 225, Rio de Janeiro, RJ 22453-900, Brazil.
{celso,rosseti}@inf.puc-rio.br

Abstract. We propose a parallel GRASP heuristic with path-relinking for the 2-path network design problem. A parallel strategy for its implementation is described. Computational results illustrating the effectiveness of the new heuristic are reported. The parallel implementation obtains linear speedups on a cluster with 32 machines.

1 Introduction

Let $G = (V, E)$ be a connected undirected graph, where V is the set of nodes and E is the set of edges. A k-path between nodes $s, t \in V$ is a sequence of at most k edges connecting them. Given a non-negative weight function $w : E \rightarrow R_+$ associated with the edges of G and a set D of pairs of origin-destination nodes, the *2-path network design problem* (2PNDP) consists in finding a minimum weighted subset of edges $E' \subseteq E$ containing a 2-path between every origin-destination pair. Applications can be found in the design of communication networks, in which paths with few edges are sought to enforce high reliability and small delays.

Dahl and Johannessen [3] proved that the decision version of 2PNDP is NP-complete. They proposed a greedy heuristic for 2PNDP, based on the linear relaxation of its integer programming formulation. They also gave an exact cutting plane algorithm and presented computational results for randomly generated problems with up to 120 nodes, 7140 edges, and 60 origin-destination pairs.

In the next section, we propose a GRASP with path-relinking heuristic for 2PNDP. An independent-thread parallelization strategy is given in Section 3. Computational results illustrating the effectiveness of the new heuristic and the speedups achieved by its parallel implementation on a cluster with 32 machines are reported in Section 4. Final remarks are presented in the last section.

2 A GRASP heuristic

GRASP is a multistart process, in which each iteration consists of two phases: construction and local search [4]. The best solution found after `MaxIterations` iterations is returned. The reader is referred to Resende and Ribeiro [10] for implementation strategies, variants, and extensions. In the following, we customize a GRASP for the 2-path network design problem. We describe the construction and local search procedures, as well as a path-relinking intensification strategy.

2.1 Construction phase

The construction of a new solution begins by the initialization of modified edge weights with the original edge weights. Each iteration of the construction phase starts by the random selection of an origin-destination pair still in D. A shortest 2-path between the extremities of this pair is computed, using the modified edge weights. The weights of the edges in this 2-path are set to zero until the end of the construction procedure, the origin-destination pair is removed from D, and a new iteration resumes. The construction phase stops when 2-paths have been computed for all origin-destination pairs.

2.2 Local search phase

The local search phase seeks to improve each solution built in the construction phase. Each solution may be viewed as a set of 2-paths, one for each origin-destination pair in D. To introduce some diversity by driving different applications of the local search to different local optima, the origin-destination pairs are investigated at each GRASP iteration in a circular order defined by a different random permutation of their original indices.

Each 2-path in the current solution is tentatively eliminated. The weights of the edges used by other 2-paths are temporarily set to zero, while those which are not used by other 2-paths in the current solution are restored to their original values. A new shortest 2-path between the extremities of the origin-destination pair under investigation is computed, using the modified weights. If the new 2-path improves the current solution, then the latter is modified; otherwise the previous 2-path is restored. The search stops if the current solution was not improved after a sequence of $|D|$ iterations along which all 2-paths have been investigated. Otherwise, the next 2-path in the current solution is investigated for substitution and a new iteration resumes.

2.3 Path-relinking

Successful applications of path-relinking [5] used as an intensification strategy within a GRASP procedure are reported in [1, 7, 11]. Implementation strategies are investigated in detail by Resende and Ribeiro [10].

In this context, path-relinking is applied to pairs (z, y) of solutions, where z is an initial solution randomly chosen from a pool with a limited number of `MaxElite` previously found elite solutions and y is the guiding solution obtained by local search. The pool is originally empty. Each locally optimal solution is considered as a candidate to be inserted into the pool if it is different from every other solution currently in the pool. If the pool already has `MaxElite` solutions and the candidate is better than the worst of them, then the former replaces the latter. If the pool is not full, the candidate is simply inserted.

The algorithm starts by determining all origin-destination pairs whose associated 2-paths are different in the two solutions z and y. These computations amount to determining a set of moves $\Delta(z, y)$ which should be applied to the initial solution to reach the guiding one. Each move is characterized by a pair of 2-paths, one to be inserted and the other to be eliminated from the current solution. The best solution $\bar{y}$ along the path to be constructed from z to y is initialized with the initial solution z. At each path-relinking iteration the best yet unselected move is applied to the current solution, the incumbent $\bar{y}$ is updated, and the selected move is removed from $\Delta(z, y)$, until the guiding solution is reached. The incumbent $\bar{y}$ is returned as the best solution found by path-relinking and inserted into the pool if it satisfies the membership conditions. The pseudo-code with the complete description of procedure `GRASP+PR_2PNDP` is given in Figure 1, with $f(x)$ denoting the weight of a solution x.

procedure `GRASP+PR_2PNDP`;
Set $f^* \leftarrow \infty$ and `Pool` $\leftarrow \emptyset$;
for $k = 1, \ldots,$ `MaxIterations` **do**;
    Construct a randomized solution x (construction phase);
    Find y by applying local search to x (local search phase);
    if y satisfies the membership conditions **then** insert y into `Pool`;
    Randomly select a solution $z \in$ `Pool` $(z \neq y)$ with uniform probability;
    Compute the set $\Delta(z, y)$ of moves;
    Let $\bar{y}$ be the best solution found by applying path-relinking to (z, y);
    if $\bar{y}$ satisfies the membership conditions **then** insert $\bar{y}$ into `Pool`;
    if $f(\bar{y}) < f^*$ **then do**; $x^* \leftarrow \bar{y}$; $f^* \leftarrow f(\bar{y})$; **end**;
end;
return x^*;
end `GRASP+PR_2PNDP`;

Fig. 1. Pseudo-code of the sequential GRASP with path-relinking heuristic

3 Parallel implementation

Strategies for the parallel implementation of metaheuristics were reviewed by Cung et al. [2]. Parallel implementations of metaheuristics are much more robust than their sequential versions. Typical parallelizations of GRASP correspond to multiple-walk independent-thread strategies, based on the distribution of the iterations over the processors [8, 9]. The iterations may be evenly distributed over the processors or according with their demands, to improve load balancing.

The processors perform `MaxIterations`$/p$ iterations each, where p is the number of processors and `MaxIterations` the total number of iterations. Each processor has a copy of algorithm `GRASP+PR_2PNDP`, a copy of the problem data, and its own pool of elite solutions. One of the processors acts as the master, reading and distributing the problem data, generating the seeds which will be used by the pseudorandom number generators at each processor, distributing the iterations, and collecting the best solution found by each processor.

4 Computational results

The parallel GRASP heuristic was implemented in C, using version egcs-2.91.66 of the gcc compiler and the LAM 6.3.2 implementation of MPI. The computational experiments were performed on a cluster of 32 Pentium II-400 processors, each one with 32 Mbytes of RAM and running under version 2.2.14-5.0 of the Red Hat implementation (release 6.2) of the operating system Linux. Since neither the instances nor the codes used in [3] were available for a straightforward comparison with the parallel GRASP algorithm, we randomly generated 100 new test instances with 70 nodes and 35 origin-destination pairs, with the same parameters used in [3]. Each instance was run five times, with different seeds.

We compared the results of the parallel GRASP algorithm with those obtained by the greedy heuristic in [3], using two samples of solution values and the statistical test t for unpaired observations [6]. The main statistics are summarized in Table 1. These results made it possible to conclude with 40% of confidence that GRASP finds better solutions than the other heuristic. The average value of the solutions obtained by GRASP was 2.2% smaller than that of the solutions obtained by the other heuristic. The dominance of the former over the latter is even stronger when harder or larger instances are considered. The parallel GRASP was applied to problems with up to 400 nodes, 79800 edges, and 4000 origin-destination pairs, while the other heuristic solved problems with no more than 120 nodes, 7140 edges, and 60 origin-destination pairs.

	Parallel GRASP (sample A)	Greedy [3] (sample B)
Size	$n_A = 100$	$n_B = 30$
Mean	$\mu_A = 443.73$	$\mu_B = 453.67$
Standard deviation	$S_A = 40.64$	$S_B = 61.56$

Table 1. Statistics for GRASP (sample A) and the greedy heuristic (sample B)

We give in Table 2 the average elapsed times in seconds over four runs of the parallel GRASP heuristic for a fixed number of 3200 iterations on the same instance of 2PNDP with 400 nodes, using 1, 2, 4, 8, 16, and 32 processors. Almost-linear speedups are observed in all cases. We also report in this table the solution values obtained in the first run for each number of processors. As expected from an independent-thread parallelization strategy, solution quality slightly deteriorates (approximately 4%) when the number of processors increases from 1 to 32. Preliminary results obtained with a cooperative strategy show that the latter is able to overcome this drawback and preserves the linear speedups.

5 Concluding remarks

We proposed in this work a parallel GRASP heuristic with path-relinking for the 2-path network design problem. Statistical tests on instances with 70 nodes

Processors	Elapsed time (s)	Speedup	Solution value
1	27599.71	1.00	2968
2	13898.29	1.99	2976
4	6987.33	3.95	2977
8	3495.03	7.90	3012
16	1760.68	15.68	3049
32	885.72	31.16	3093

Table 2. Elapsed times and solution values for the parallel GRASP heuristic

showed that the GRASP heuristic compares favourably with another heuristic reported in the literature. This dominance is even stronger when harder or larger instances are considered. Linear speedups for up to 32 processors are observed for the parallel implementation on a cluster. A cooperative parallelization strategy using a centralized pool and detailed results obtained on a broader set of larger instances will be reported elsewhere.

References

1. S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro, "Local search with perturbations for the prize-collecting Steiner tree problem in graphs", *Networks* 38 (2001), 50–58.
2. V.D. Cung, S.L. Martins, C.C. Ribeiro, and C. Roucairol, "Strategies for the parallel implementation of metaheuristics", in *Essays and Surveys in Metaheuristics* (C.C. Ribeiro and P. Hansen, eds.), pages 263–308, Kluwer, 2001.
3. G. Dahl and B. Johannessen, "The 2-path network design problem", submitted for publication, 2000.
4. T.A. Feo and M.C.G. Resende, "Greedy randomized adaptive search procedures", *Journal of Global Optimization* 6 (1995), 109–133.
5. F. Glover, "Tabu search and adaptive memory programing – Advances, applications and challenges", in *Interfaces in Computer Science and Operations Research* (R.S. Barr, R.V. Helgason, and J.L. Kennington, eds.), pages 1–75, Kluwer, 1996.
6. R. Jain, *The art of Computer Systems Performance Analysis: Techniques for Experimental Desgin, Measurement, Simulation, and Modeling*, Wiley, 1991.
7. M. Laguna and R. Martí, "GRASP and path relinking for 2-layer straight line crossing minimization", *INFORMS Journal on Computing* 11 (1999), 44–52.
8. S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P. Pardalos, "A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy", *Journal of Global Optimization* 17 (2000), 267–283.
9. S.L. Martins, C.C. Ribeiro, and M.C. Souza, "A parallel GRASP for the Steiner problem in graphs", *Lecture Notes in Computer Science* 1457 (1998), 285–297.
10. M.G.C. Resende and C.C. Ribeiro, "GRASP", to appear in *State-of-the-Art Handbook of Metaheuristics* (F. Glover and G. Kochenberger, eds.), Kluwer.
11. C.C. Ribeiro, E. Uchoa, and R.F. Werneck, "A hybrid GRASP with perturbations for the Steiner problem in graphs", to appear in *INFORMS Journal on Computing*.