An Algorithm for Unrestored Flow Optimization in Survivable Networks Based on p-Cycles

Adam Smutnicki

Chair of Systems and Computer Networks,
Wrocław University of Technology,
Wybrzeze Wyspianskiego 27, 50-370 Wrocław, Poland
adam.smutnicki@pwr.wroc.pl

Abstract. This paper provides compound algorithm for Unrestorable Flow Optimisation (UFO) problem formulated for computer networks protected by p-cycles, created on the base of mathematical model and solution approaches proposed in our complementary paper [1]. Components of the algorithm have been selected carefully and then experimentally tested in order to compose the best final algorithm. Results of the wide computer tests on common benchmarks have been also presented as well as some practical conclusions following from the research made.

Keywords: Computer network, survivability, optimization, p-cycles, UFO problem.

1 Introduction

In [1] we have shown that Unrestorable Flow Optimization (UFO) problem, as a complicated task, may be decomposed into several auxiliary subproblems with dedicated solution methods. The mentioned above paper outlines also solution methods without practical realizing comments. This paper refers chiefly to solution algorithms derived from theory, [1] and their numerical properties tested for the best practical performance. Paper [1] is complementary to this research.

2 Algorithm Components

Consider the problem of allocating demands into p-cycles while restoring the net flow in case of failure. The problem can be modeled by Multiple Knapsack Problem (MKP) which is strongly $\mathcal{NP}$-hard. One can find several approximate and exact algorithms for MKP in [2]. In our research we use a greedy algorithm for MKP, based on principle of best fit rule – element is packed into this knapsack in which most of free space is left. This algorithm has pseudo-polynomial complexity $O(nm)$ (n is a number of elements and m is a number of knapsacks) acceptable in practice, taking into account that overall solution is approximate. Up to now, research have been made using commonly Greedy algorithm for MKP; quite recently there have been proposed and analyzed profits of usage branch-and-bound algorithm for MKP in UFO problem in [3].

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 315–324, 2009.

Next subproblem, described in [1], consists in determining the maximum spare capacity of all p-cycles. For non-disjoint p-cycles there can be written set of linear equations or inequalities, where criteria function, as mentioned in [1] is to maximize sum of residual capacity of all p-cycles. This forms a set of linear equations and can be solved using algorithm dedicated Linear Programming task. Simplex method, as most popular and easiest to use, has been chosen.

At the end, for each demand there have to be prepared a set of alternative routing paths. Because demands are defined from source to target nodes, k-SPD algorithm based on Dijkstra's Shortest Path Algorithm, can be used [4].

2.1 Generation of p-Cycles

The optimal configuration of p-cycles protecting the network can be either constructed from scratch or selected among certain subset of p-cycles called cycle-candidates set. The number of all possible cycles in network with the full mesh topology (i.e. full undirected graph) is $\omega(2^V)$, i.e. is rising faster than 2^V, where V is the number of vertices in the net, see [5] for detail. Thus, assuming usage of the step-by-step search technique, even for quite small networks, the total number of cycle candidates is usually too big to store all of them in the computer memory and then to check any of them. Therefore, two alternative approaches one can propose next: (a) use pre-generated set of cycle candidates, of limited reasonable cardinality, generated in advance, in off-line mode, (b) generate on-line cycle candidates on demand, whereas the number of generated cycles is a priori unknown. The former approach is called *pre-generation technique*, whereas the latter – *enumeration technique*, [5,6]. In both cases a "good" p-cycles are expected from the generation algorithm. The problem of generation cycle components is nontrivial and has been studied among others in [5,6,7,8,9,10].

The presented paper provides research made for conception (a). Application of (b) remains an open research task. A decision process, which particular p-cycles are used for protection, will be done by tabu search (TS) algorithm, see Section 2.2 for detail. In order to evaluate the impact of generation scheme onto overall solution quality, we tested four different generation algorithms, to compare results obtainable for different sets of p-cycles. A lot of algorithms generate larger set of p-cycles by extending a small one with the help of some transformations. We refer here to the well-known algorithm Straddling Link, proposed first time in [9]. It is quite simple and never generates more cycles than number of spans in network. In [7,11] there have been described several different algorithms providing rules for transformations from simple p-cycles to complex ones. Among them, three have been chosen as the most promising: SP-Add (Span Add), Expand and Grow, described in [7].

2.2 Tabu Search

It has been shown, [1], that UFO problem is strongly $\mathcal{NP}$-hard. Thus the use of heuristic or metaheuristic algorithms is fully justified. We decided to apply Tabu Search (TS) technology, because of good results achieved for many other

applications in optimization. Referring the reader to foundations of TS, [12], we present here only implementations of its crucial elements in our case. Each *solution* of our problem will be represented by current configuration of p-cycles (a subset of components selected from the whole candidates set) plus current set of routing paths (only one path is selected for each flow demand). *Quality of the solution* is measured by the amount of unrestored flow, expressed as fraction of total flow. The whole evaluation of UFO value consists of calculation of capacities for chosen p-cycles (an LP task) and determining which demands can be restored (by solving the sequence of MKP instances), see [1] for detail.

Because our solution consists of two inhomogeneous structures (p-cycles and routing paths) the *move* (transformation) from the current solution can be either *p-cycle move* or *routing path move*. All moves, generated from the current solution, generate its *neighborhood*. Let us describe these transformations more precisely. First, because of small change philosophy, we assume that transformed solutions differ from original one by single element. For transformation performed on routing configuration, we had to choose solution which will not increase drastically the size of TS neighborhood. For example, if there are three alternative routing paths for some demand, two new neighboring solutions will be generated for this demand — each of them corresponds to one of two remaining routing path (except this already used). Thus, having k alternative paths and d demands, $(k-1)d$ new neighboring solutions will be generated. For p-cycles transformations we have defined three possible changes (three neighbors) for each component of p-cycle : add one p-cycle to current configuration from candidates set (not used yet), remove one p-cycle from current configuration, exchange one p-cycle from current configuration into one from candidates set (not used yet). Each new element in neighborhood is generated by performing only one of mentioned transformations. Whole neighborhood coming from p-cycles set transformation is done by performing for each p-cycle in current configuration following steps: (a) remove a component p-cycle; (b) add n times randomly chosen with weights (value of AE ([11]) metric as weight) p-cycle from candidates set; (c) exchange current cycle component into one from candidates set being not far than $|l|$ steps from actual. p-Cycles in candidates set are sorted decreasingly by value of AE metric. We see that parameters l and n values has significant influence on size and type of generated TS neighborhood. Next, we will call l a size of p-cycles neighborhood and n as number of randomly chosen p-cycles.

The starting solution of TS is created by using the following principles: (1) for each demand, the first (shortest) path on the candidate list is set, (2) current configuration of p-cycles contains single component, the first one from the list of candidates (with greatest value of metrics AE).

Tabu list (short term memory of the search history) is considered as the crucial control element of TS, since it is responsible for proper behavior of the whole algorithm (it prevents cycling). In our implementation we assume that tabu list stores the move made, both types of moves on the common list. The move to be performed is considered as tabu if its inverse move is stored on tabu list. The form of inverse move is easy to define for the moves introduced earlier.

Selection of TS *control parameters* is discussed in Section 3.4.

3 Computational Results

The quality of proposed (metaheuristic) algorithm will be evaluete in computer test on common benchmarks.

3.1 Common Benchmarks

For network optimization tasks there exists a library, called SNDlib [13], providing a set of network topologies, as well as all necessary information about flows and possible routes. SNDlib consists of several network topologies, which are in most cases real networks, or at least advanced projects, e.g.: COST-266, germany50, poland, france. Test described in this paper have been performed using COST-266 topology, because it is quite similar to COST-239 (COST-239 is the project older than COST-266). COST-239 was used in most of papers in literature dealing with p-cycles optimization.

3.2 Generation Technology Test

First performed test compares the behavior of algorithm depending on p-cycles generation algorithm. Four, mentioned in Section 2.1, algorithms have been tested. Results are presented on the Fig. 1, where one routing path for each demand is used. First and most obvious conclusion is that the Grow algorithm finds final result using smaller number of TS iterations. Grow also descends toward final result quicker that other algorithms. We see that best results are achieved using p-cycles set from Grow algorithm, but it is quite interesting that for this combination, the TS algorithm needed less time to find better result than using other p-cycles algorithm. So the configuration of parameters giving the best

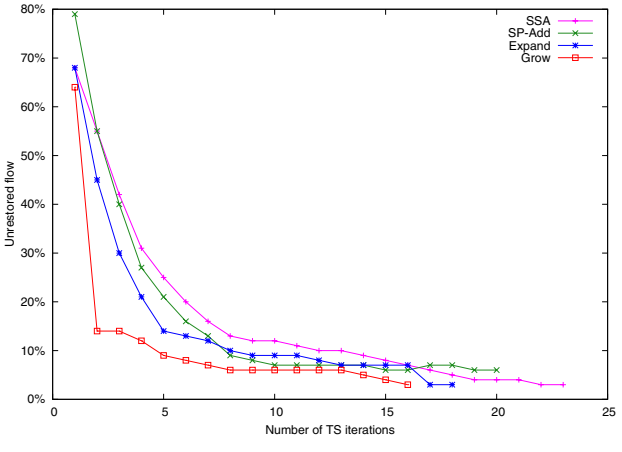

Fig. 1. Comparison of p-cycles generation algorithms for one routing path for each demand

Table 1. Comparison of parameters of p-cycles generated by four mentioned algorithms, for COST-266 topology

	SLA	SP-Add	Expand	Grow
number of p-cycles	45	96	89	1214
av. value of AE metrics	1.27	1.49	1.56	1.74
av. p-cycle length	7.20	11.04	13.01	21.38
av. number of straddling-spans	1.17	2.88	3.96	8.48
average generation time in seconds	0.0019	0.0076	0.0061	2.65

results produce them in shorter time than other, worse configurations. Only one disadvantage of this solution is that the time needed to generate p-cycles candidates set using Grow algorithm is significantly longer than for other algorithms. Average time needed for generation sets of p-cycles and parameters of those sets for test COST-266 topology, for tested four algorithms arepresented in Tab. 1.

3.3 Alternative Routing Paths Test

In Section 3.2 there have been analyzed influence of chosen p-cycles generation algorithm on final result. But as mentioned in [1], this is not only one control parameter. Second one, no less important is the number of alternative routing paths for each demand. Moreover during whole optimization in TS, the most useful path is chosen for each demand, so giving more optional paths, algorithm receives wider range of possible solutions. On Fig. 2 there have been presented comparison of p-cycles generation algorithm but for three alternative routing

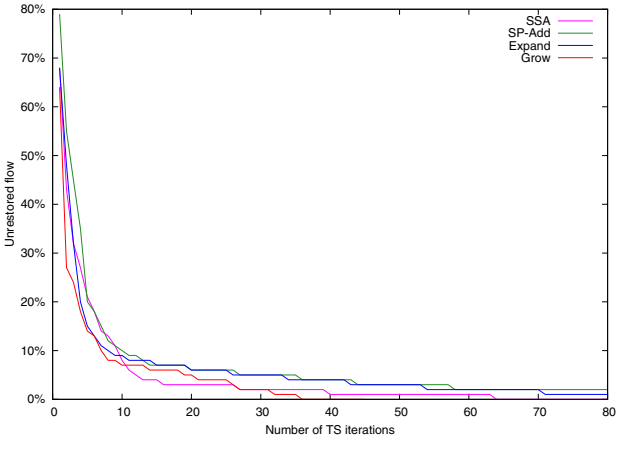

Fig. 2. Comparison of p-cycles generation algorithms for three routing path for each demand

Table 2. Comparison of example results depending on used p-cycles generation algorithm and number of alternative routing paths; presented values stands for % of unrestorable flow

	SLA	SP-Add	Expand	Grow
1 kSPD	3.84	6.70	6.87	3.68
3 kSPD	0.67	1.25	1.28	0.35

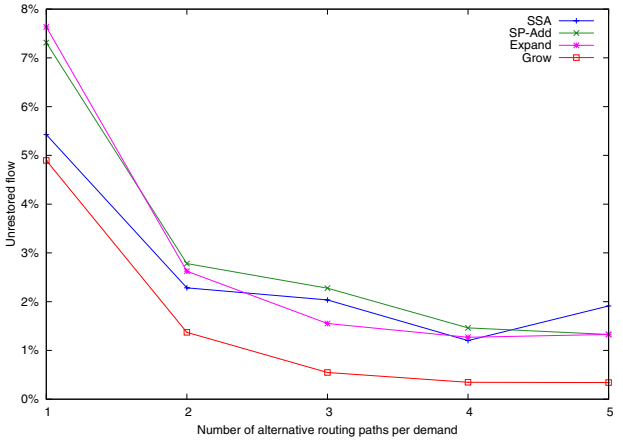

Fig. 3. Comparison of results for each p-cycles generation algorithm depending on number of alternative routing paths for each demand

paths for each demand. Comparing Fig. 1 and Fig. 2 one can notice, that giving more alternative paths improve the way that overall algorithm works. First improvement is that for each p-cycles generation algorithm TS descending toward final result is quicker. Second conclusion is that, TS is working longer (bigger number of TS iterations till stop condition). It is because using more alternative paths, allow TS algorithm to tune more the final result. Also, as previously, the best p-cycles generation algorithm is Grow.

Presented results on Fig. 1 show only general behavior of two described options. Exact results are presented in Tab. 2. Analyzing those results one can see that using three alternative paths improve the quality of final result several times. So without any additional resources, only changing the routing, adjusting it for chosen set of p-cycles, we can improve the level of final protection.

Comparing results of mentioned test with tests for only one routing path, we see that only changing flow in network, we can highly improve the protection performed by p-cycles set. So it is highly recommended that p-cycles optimization should be accompanied with dedicated routing.

One can ask, why not use more routing paths, if it gives such improvement. The answer is presented on Fig. 3. We can see that the best improvement of

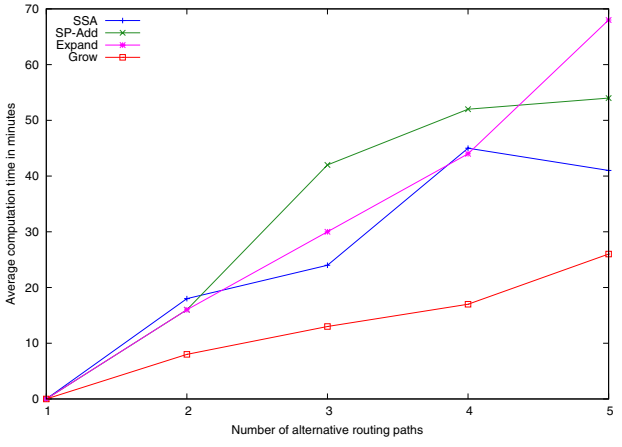

Fig. 4. Comparison of time needed to find best solution depending on used p-cycles generation algorithm and number of alternative routing paths

quality of final result is when changing from one to three alternative routing paths for each demand. Adding more paths till five, practically do not change the quality of received final result. What is important, adding more alternative paths increase the time of computation – Fig. 4.

For more that three routing paths, possible improvement is not worth because of amount of time needed to finish computations. What more, the increase of time needed for all algorithms except Grow is higher than for Grow.

Analyzing presented results for time of computation and time needed to generate a set of p-cycles (presented in Section 3.2) we can see, that using Grow is highly recommended because of best preformed results and shortest computation time, even taking into account time needed to generate p-cycles set. This additional time is several hundred times smaller that time saved in TS.

3.4 Finding the Optimal Configuration

In whole process of finding final solution several elements are important. Two main were analyzed and discussed in Section 3.2 and 3.3. Other are: size of tabu list, number of TS iterations till algorithm stops, size of p-cycles neighbourhood, number of random p-cycles configuration change. All mentioned parameters have been tested within defined ranges, to find which of them and how much influence the quality of final result.

During tests we have notices that TS algorithm was falling into cycle nearby found local minimum. In this case, we decided to try to block this cycles, as this prevents algorithm for further search. Those cycles came from situation, when reaching local minimum, TS while generating neighborhood received most of solutions with the same value of criteria function. Also there have to be mentioned, that in basic structure tabu list does not protect before cycles. We

decided to filter TS neighborhood – solutions having the same value of criteria function were not analyzed. Results presented in this paper compare version of TS with and without filtering.

Because of limited size of this paper, we are unable to present all results and comparisons, so we decided to present only conclusions drawn from performed tests. The conclusions are:

1. we have not notices influence of size of p-cycles neighborhood on quality of final result;
2. there exists dependency of quality of final result on number of randomly chosen p-cycles configuration – till the value of 5 it increases the quality, bigger values practically do not change anything;
3. for algorithm without filtering the neighborhood one can notice link between final solution quality and length of tabu list – longer list improves the quality, but influence is quite small;
4. for algorithm with filtering the neighborhood we have not noticed link between the quality of final solution and tabu list size – probably tabu list blocks only one step back and filtering process prevents algorithm from backtracking;
5. we have notices direct influence of number of TS iterations till stop on quality of final result – there is an improvement till value of 20, after this value there is no increase in quality;
6. analyzing all parameters, we have notices that algorithm with filtering neighborhood, generally gave better results that without filtering;
7. in all cases for option with filtering solutions and without, best results were received using Grow p-cycles generation algorithm;
8. quite surprising was that second best results were received using SSA p-cycles generation algorithm – the one that generates simplest and most basic p-cycles;
9. computations using Grow algorithm gave better results, because p-cycles set generated by this algorithm consists of p-cycles form SSA algorithm (which was second best) and some additional p-cycles with different feature, which improved the quality of final solution, over single set from SSA.

3.5 Final Solution Quality Test

In Section 3 there have been presented received results. The best parameters configuration have been identified and discussed. In this section we will analyze the quality of received results. Because for now one have performed such research like described in this paper, there is no point which we can refer to. Test have been performed for TS algorithm with filtering and without. Both of them have started from same basic solutions, which "unrestorability" was 64.15%. In Tab. 3 there have been presented average received results for 100 repeats. Also there are some statistic parameters of results sample. Time of computations for TS with filtering was ca. 30% longer that without. We can see that both versions have reached 0.0% of unrestorability and both have done it four times. Median

Table 3. Comparison of results quality for TS algorithm with filtering and without, for optimal algorithm configuration, for COST-266 topology; values are value of unrestorability in %

	without filtering	with filtering
av. result value	0.62	0.43
best value	0.00	0.00
best value found no. of times	4	4
worst value	5.11	3.82
median	0.35	0.35
standard deviation	0.87	0.42
av. deviation	0.49	0.17

is the same, version with filtering have much lower deviation from average value. Additionally there have been done T-Student statistic tests to confirm that both samples differ each other. Tests were positive.

4 Conclusions

We have presented a set of algorithms building overall solution for UFO problem formulated in our paper [1]. Presented algorithms are first ever developed for UFO problem. Performed tests analyze wide range of parameters influencing the quality of final result. Notice, we have proposed a joint optimization of routing and selecting p-cycles, which most of authors in literature have avoided, called them too complex. We have found configuration of the net ensuring unrestorable flow below 1% of total flow in network, and in many situations achieving 100% of restorability. Taking into account that this optimization is done in advance, we have enough time to find best solution. We have also confirmed that the combination of p-cycles with different size (generated by Grow algorithm) provides best restorability.

References

1. Smutnicki, A., Smutnicki, C.: Flow Optimization in Survivable Networks Based on p-Cycles. Submitted for International Conference on Computational Science (ICCS 2009) (2009)
2. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
3. Smutnicki, A.: Branch-and-bound Algorithm for Multiple Knapsack Problem in the Unrestorable Flow Optimisation Problem. In: Proceedings of 23rd IAR Workshop on Advanced Control and Diagnosis, Coventry, UK (November 2008)
4. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Elsevier Inc., San Francisco (2004)

5. Schupke, D.: An ILP for Optimal p-Cycle Selection Without Cycle Enumeration. In: Proceedings of Eighth IFIP Working Conference on Optical Network Design and Modelling (ONDM 2004), Ghent, Belgium (February 2004)
6. Wu, B., Yeung, K., Xu, S.: ILP Formulation for p-Cycle Construction Based on Flow Conservation. In: Proceedings of Global Telecommunications Conference, 2007. GLOBECOM 2007, Washington, DC, USA, pp. 2310–2314 (November 2007)
7. Doucette, J., He, D., Grover, W.D., Yang, O.: Algorithmic Approaches for Efficient Enumeration of Candidate p-Cycles and Capacitated p-Cycle Network Design. In: Proceedings of Fourth International Workshop on Design of Reliable Communication Networks, pp. 212–220 (October 2003)
8. Chang, L., Lu, R.: Finding Good Candidate Cycles for Efficient p-Cycle Network Design. In: Proceedings of 13th International Conference on Computer Communications and Networks, pp. 321–326 (October 2004)
9. Zhang, H., Yang, O.: Finding Protection Cycles in DWDM Networks. In: Proceedings of IEEE International Conference on Communications, pp. 2756–2760 (2002)
10. Lo, K., Habibi, D., Rassan, A., Phung, Q.V., Nguyen, H.N., Kang, B.: A Hybrid p-Cycle Search Algorithm for Protection in WDM Mesh Networks. In: Proceedings of ICON 2006. 14th IEEE International Conference on Networks, pp. 1–6 (September 2006)
11. Grover, W.D.: Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking. Prentice Hall PTR, New Jersey (2003)
12. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Pusblishers, Dordrecht (1997)
13. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0 — Survivable Network Design Library. In: Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium (April 2007), `http://sndlib.zib.de`