Genetic Algorithm for Solving Multilayer Survivable Optical Network Design Problem

Huynh Thi Thanh Binh, and Ha Dinh Ly

***Abstract*—Given an undirected graph $G_1 = (V_1, E_1)$, a complete undirected and weighted graph $G_2 = (V_2, E_2, c)$ and a set of customers' demands. The goal is to design connections based on customers' demands with the smallest network cost to protect the network against all failures. This problem is NP-hard. This paper proposes a genetic algorithm for solving the Survivable Network Design Problem (SNDP). We experiment our proposed algorithm on random [12] and real world instances [1]. The experimental results are reported to show the efficiency of proposed algorithm comparing to the Branch and Price algorithm [1].**

***Index Terms*—Survivable network design, genetic algorithm, branch-and-price.**

I. Introduction

The birth of network, especially the appearance of Internet, marked an important turning-point in the field of information technology and communications. Hence, network has brought more and more useful in all of life areas such as economic, culture, military, etc. So, sometimes, only a network failure also can cause serious consequences, particularly about economic. This sets a requirement for the network providers is how to ensure the reliability for their products. It is also the reason for defining Survivable Network Design Problem (SNDP).

Later, along with the birth and the ongoing development of optical network, and now is multilayer optical network, SNDP has switched to a new stage and improved tremendously. The advantages of optical network (such as heavy flow, less signal decline, etc) make it more and more popular than the traditional one. And so, SNDP for optical network is becoming essential and gaining much more attention. In this paper, we will propose new approach for solving SNDP for multiplayer optical network which is the latest model of this problem and called the Multilayer Survivable Optical Network Design Problem (MSONDP).

MSONDP problem is defined as the following: Given an undirected graph $G_1 = (V_1, E_1)$ and a complete undirected and weighted graph $G_2 = (V_2, E_2, c)$ such that $V_1 \subset V_2$, $E_1 \subset E_2$ and c is edge weight of G_2. G_1 represents the logical layer, G_2 represents the physical layer and each value in c is cost of a corresponding edge in G_2. In this problem, a set of demands T is required by customers. Each demand t_i in T consists of two node-disjoint paths (L_i^1, L_i^2) that connect the source node o_i to the destination node d_i in G_1, where L_i^1 is working path and L_i^2 is backup path. The goal of the problem is to design connections based on customers' demands with the smallest network cost to protect the network against all failures. In the other word, it is to find two node-disjoint paths mapping of L_i^1, L_i^2 in graph G_2 for each t_i such that total cost of all demands is minimal.

We can formulate the problem as following:

Find an appropriate connection for each demand in customer's set of demands so as to minimize:

$$totalCost = \sum_{i=1}^{|T|} c(t_i) \tag{1}$$

where $c(t_i)$ is cost of the i-th demand, |T| is the number of customers' demands.

To solve this problem, we have to build a set of paths satisfying all demands to minimize the total network cost.

This is a new model of SNDP and according to [1], it is a NP-hard problem even for one demand. Up to now, only research group of Sylvie Borne research this problem and publish their results in [1]. So, in this paper, we propose a genetic algorithm for solving MSONDP, called GAMSONDP. In our genetic algorithm, a chromosome represents a pair of paths (L^1, L^2) and so, an individual has |T| chromosomes. Taking advantages of the complete graph, we encode an individual by an ordered nodes list of its paths. And a new characteristic of GAMSONDP is that we use two crossover and three mutation operation types simultaneously to bear new individuals more diversely, which makes GA more effective. We experiment with the data which used in [1] and compare the result with Branch and Price in [1].

The rest of this paper is organized as following: Section II describes the related works. In section III, we present the proposed algorithms to solve MSONDP. Our experiments and computational and comparative results are given in section IV. The paper concludes with section V with some discussions on the future extension of this work.

II. Related Works

Since the survivability is an essential necessity in any network system, there have many research works about this topic. Many models of problem were researched and solved by specific algorithms. There are two approaches for solving SNDP [3]: exact and approximate algorithms.

In 2006, S. Borne et.al studied survivable IP-over-optical network design problem [2]. They gave a 0-1 integer pro-

Manuscript received September 15, 2012; revised November 14, 2012. This work was partially supported by the project "Models for next generation of robust Internet", grand number 12/2012/HĐ-NĐT, funded by the Ministry of Science and Technology, Vietnam. The Vietnam Institute for Advanced Study in Mathematics provided part of the support funding for this work.

The authors are with Ha Noi University of Science and Technology, School of Information and Communication Technology (e-mail: binhht@soict.hut.edu.vn; greeny255@gmail.com).

gramming formulation for this problem, described some valid inequalities and discussed separation algorithms for these inequalities, simultaneously, introduced some reduction operations. Basing on these, they proposed Branch-and-cut algorithm to solve. They experiment with the random instances which are generated with 10 to 45 nodes and the number of edges is 10, 15, 20 and 30. The real instances they used to test were provided by the French telecommunications operator France Télécom. These instances have 18 to 60 nodes and 31 to 102 edges. However, their algorithm only can solve small test sets (less than 35 nodes and 30 edges in random instances, less than 60 nodes and 102 edges in real instances). With larger problem instances, the running time is quite long (about 5 hours) and some of them could not be solved.

After that, in 2009, S. Borne et. al. proposed Branch-and-Cut and Branch-and- Cut- and-Price algorithm based on the path formulation for solving the multilayer capacitated survivable network design problem [9]. However, they only solved with graphs having 10 nodes, 20 edges and the number of demands is 20.

Adrian Zymolka defined the cost-efficient design of survivable optical telecommunication networks problem [8]. Then, he proposed Branch-and-Price with four branching rules after modeling this problem in integer linear program [8]. His problem was separated into two individually hard sub-problems, one of which is to rout the connection with corresponding dimensioning of capacities and the and the other is to seek for s conflict-free assignment of available wavelengths to the light paths (a common characteristic of optical network) using a minimum number of involved wavelength converters. The first problem was solved by three steps algorithm as follows: 1. A preprocessing transforming the hardware model accordingly; 2. The application of DISCNET as main optimization routine; 3. A post-processing to adapt solutions for the original problem. With the second one, deriving linear program formulation, he proposed an exact Branch-and-Price method to solve. That is an integer linear programming approach for exact solution.

In 2011, MSONDP was proven to be NP-hard by S.Borne et.al [1]. They formulated this problem in terms of 0-1 linear program based on path variables. Then, they discussed the pricing problem and proved that it reduces to a shortest path problem. Using this, they proposed a Branch-and-Price algorithm. However, this is an exact approach for NP-hard problem, so they can only offer solutions for the maximum input is 17 nodes on G_1, 20 nodes on G_2 and 25 demands.

So, MSONDP was only defined by S.Borne in 2011 [1], which shows this is a new model of SNDP. And for now, there have not any effective algorithms to solve this model with the large test sets. Moreover, it is NP-hard problem, so, in this paper, we propose Genetic Algorithm for solving it and we hope our algorithm could solve large problem instances.

III. Proposed Algorithm

In this section, we propose genetic algorithm, GAMSONDP, to solve MSONDP. As mentioned in section 1, one of the goals of MSONDP is to find two node-disjoint paths mapping of L_i^1, L_i^2 in completed graph G_2. So, we describe find-path algorithm first. And then, GAMSONDP will be presented specifically.

A. Find-Path Algorithm

This is a way to build our initial solution for the problem. Then, using GA, we improve individuals to reach the best ones as possible in this population.

The main idea of the find-path algorithm is simple. Assume, we find a path from A to B. We will choose x nodes (x $\in N$) that are different from both A and B in the graph G_2 randomly and insert them between A, B. This operation is ensured to give us a solution because G_2 is the completed graph. However, if both the node set V and x is too great, the path from A to B will be over many nodes, which makes the algorithm not effective about run time. To improve that, we take x from {0, 1, 2}. Then, the find-path algorithm has the pseudo code as follow:

```
1 pathMulti <- A;
2 if (|V|-2) < 3 then
3       x= random(|V|-2)
4       if x = 0 then pathMulti <- B
5       else
6           for i = 0 to x do
7               C <- random(V\{A,B})
8               path <- C
9           end for
10          pathMulti <- B
11      end if
12 else
13      x = random({0,1,2})
14      if x = 0 then
15          pathMulti <- B
16      else
17          for i = 0 to x do
18              C <- random(V\{A,B})
19              path <- C
20              end for
21              pathMulti <- B
22      end if
23 end if
24 return pathMulti
```

If L_i has more two nodes, we will apply above algorithm for two consecutive nodes respectively, and then combine them with note that each node is only used once. And now, we will present GAMSONDP.

B. Individual Representation

In our algorithm, each individual, which is a solution of the problem, has |T| chromosomes and each chromosome has two genes called working gene and backup gene. These two genes encode for two node disjoint paths corresponding with L^1 and L^2 of a demand and are represented by a list of nodes in those two paths.

We initialize population by two ways as follows: The first, we create individuals randomly. The other way: first, we create an empty individual with |T| empty chromosomes. Then, we insert node lists of each demand into corresponding chromosome respectively. After that, we use above find-path algorithm to create individuals for initial population.

Fig. 1 depicts an individual representation. This individual has k chromosomes. Chromosome t_1 encodes demand t_1 that has $L_1^1 = \{4, 1, 3\}$ and $L_1^2 = \{4, 2, 6, 3\}$. Number 4 and 3 are bold to show that they are the source and destination nodes. The other numbers represent nodes required to go through. It is the same to the other chromosomes.

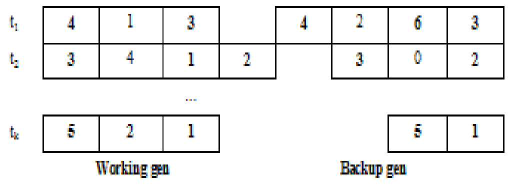

Fig. 1. An example of the individual representation

C. Selection

As mentioned above, an individual is a solution of the problem, i.e. it gives us for a way to design network with a specific cost. Obviously, the lower cost is, the better solution is, in other words, the more adaptable the individual is. Fitness function, therefore, can be calculated through cost function as (1):

$$totalCost = \sum_{i=1}^{|T|} c(t_i)$$

with:

$$c(t_i) = \sum_{e \in the\, i-th\, working\ path} c(e) + \sum_{e \in the\, i-th\, backup\ path} c(e)$$

where:

$$c(e) = \begin{cases} initial\ cost\ of\ e\ if\ e\ is\ not\ used \\ 0 \qquad\qquad if\ e\ was\ used \end{cases}$$

So, fitness function will be:

$$F = \frac{1}{totalCost} \tag{2}$$

When the stop condition of GAMSONDP is satisfied, an individual having the maximum F value is the best solution for the problem.

D. Crossover Operator

In this subsection, we implement crossover operator. It is an important operator affecting the efficiency of GA strongly. We propose two different crossover operators as follow:

The first one is called chromosome crossover operator. Its idea is to choose two different individuals in the population then random a "cut-point". The "cut-point" divides the number of chromosomes of each parent individual into two parts. Joining the first part of the parent 1 with the second part of the parent 2 together and inversely, we create two new children. If these children coincide with all of individuals in the current population, they will not be admitted. Fig. 2 shows chromosome crossover operator.

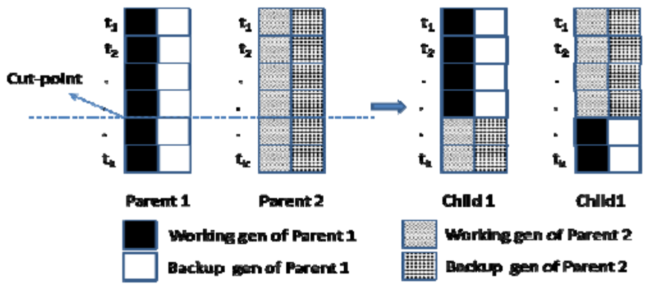

Fig. 2. Chromosome crossover operator

The other crossover type is path crossover operator. The idea of this crossover type is that a new individual is born by taking working gene of a parent and backup gene of the other parent. The number of each individual's chromosomes is divided into two equal parts. In the first half part, all of the parent 1's working genes are copied to the child, and backup genes of the child are copied from the parent 2, and do inversely with the other part.

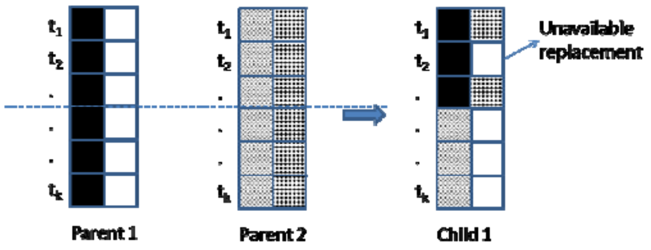

Fig. 3. Path crossover operator without conflict

If conflict occurs in any chromosomes, these chromosomes keep their initial backup genes. This is unavailable replacement.

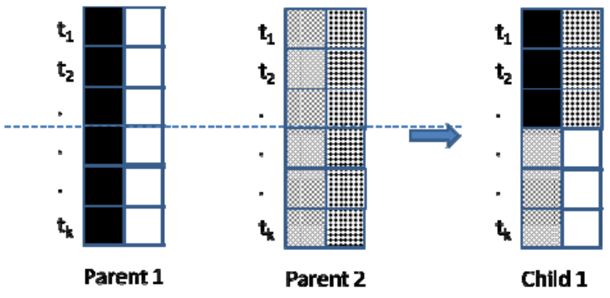

Fig. 4. Path crossover operator with conflict.

E. Mutation Operator

Obviously, by using above crossover operators, we just bear new individuals, but not change nodes of genes, i.e no new paths are created. This can make us not to find global solution for the problem. So, we use mutation operator to improve that. Below, we implement four mutation types: gene mutation, chromosome replacement mutation, individual renew mutation 1 and individual renew mutation 2.

Gene mutation: we implement operations: replace a node, add a node and delete a node. Note that in replace and delete operations, replaced or deleted node must be different from all nodes in L_i^1 and L_i^2 and added node must be unused by the other genes of the mutated chromosome.

Chromosome replacement mutation: we choose an individual and its chromosome randomly then replace the chosen chromosome by another.

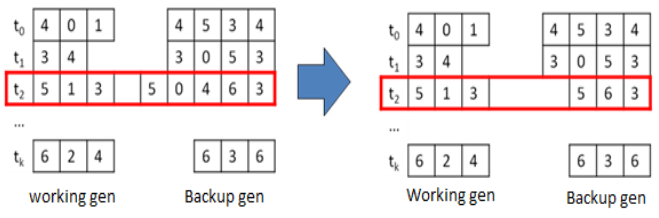

Fig. 5. Mutation operator

In individual renew mutation 1: after choosing a random individual, the first one-third of the number of its chromosomes is copied to its child then cost of edges that the copied chromosomes equal 0. The remaining is created by heuristic that used edges will be chosen first.

Individual renew mutation 2 is similar to Individual renew mutation 1, only replace the first one-third by the final one-third.

Note that, it always tests a new individual to guarantee that there have two identical ones. If a new individual existed in the current population, we will implement again.

IV. Experimental Results

A. Problem Instances

In our experiments, we used both random and real world instances. Random instances can be found at TSP library [12] and real world ones are based on S.Borne's Gravitory model [1]. Denote problem instances as namen2_n1_k, in which name is 'a' if it is random instance and is 'g' or 'Germany' if it is real world instance; n2, n1 is respectively the number of nodes in G_2 and G_1; and k is the number of demands. We tested our algorithms with 45 random and 44 real world instances. The smallest data set we experimented is a6_4_2 and the biggest is a60_55_150 for both random and real world instances.

B. Experiment Setup

We experiment proposed algorithms GAMSONDP and compare its performance with Branch-and-Price algorithm in [1].

C. System Setting

In our algorithm, the population size is 600. 300 individuals are initialized randomly and 300 individuals are initialized by find-path algorithm. The maximum number of generations is 800, crossover rate in each generation is 40% (each of crossover types rate is 20%) and mutation rate is 30% (where gene mutation rate equals chromosome replace mutation rate is 5%, the others, each of them is 10%).
Our system is run 20 times for each problem instances. The programs were run on a machine with Intel Pentium G840 2.80 Ghz, RAM 4G DDR3, HDD 160G Seagate and were installed by Java language.

D. Computational Results

Fig. 6 and 7 show that:

- On the random instances, the best results found by GAMSONDP are equivalent to ones of Branch-and-Price [1] in 17/35 test sets.
- On the real world instances, the best results found by GAMSONDP are equivalent to ones of Branch-and-Price [1] in 16/35 real test sets.
- The equivalent results found by both GAMSONDP and Branch-and-Price gather in small data sets (having 8, 10 nodes) and in data sets with small number of demands.

Table I and II show that the best and the average costs found by GAMSONDP on 9 random test sets and 10 real world test sets. These problem instances could be solved by GAMSONDP but could not be solved by Branch-and-Price [1].

Fig. 6, 7 and Table I, II shows that deviation of the results found by GAMSONDP over 20 running times are small. That is prove that the stability of GAMSONDP.

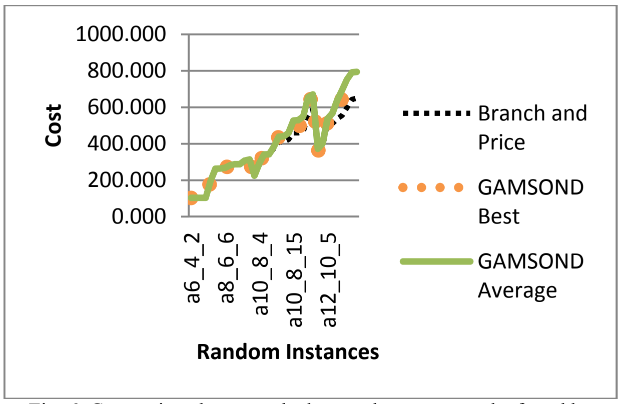

Fig. 6. Comparison between the best and average results found by GAMSONDP and Branch-and-Price [1] over 20 running times on the random instances.

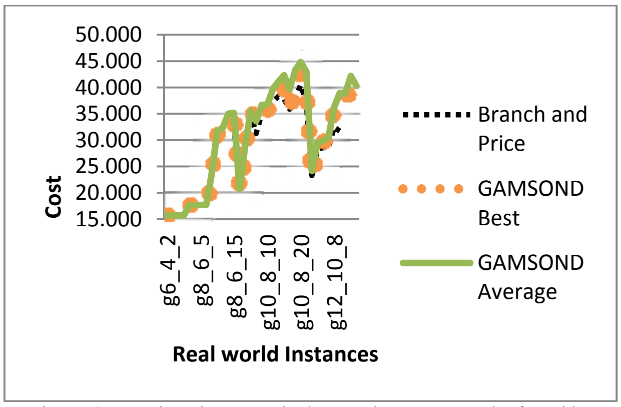

Fig. 7. Comparison between the best and average results found by GAMSONDP and Branch-and-Price [1] over 20 running times on the real world instances.

TABLE I: The Best and the Average Results Found by GAMSONDP Over 20 Running Times on the Random Instances which are Could Not Be Solved by Branch-and-Price [1].

Instance	GAMSONDP Best	GAMSONDP Average
a20_18_2	302.277	308.892
a20_18_40	2925.073	2940.400
a20_18_80	3697.477	3840.442
a40_35_2	790.975	797.357
a40_35_40	6926.569	7143.377
a40_35_80	11588.495	11879.296
a60_55_2	1051.291	1079.554
a60_55_40	9549.607	9696.495
a60_55_80	16842.176	17026.222
a60_55_150	27864.841	28493.118

TABLE II: The Best and the Average Results Found by GAMSONDP Over 20 Running Times on the Real World Instances which are Could Not Be Solved by Branch-and-Price [1].

Instance	GAMSONDP Best	GAMSONDP Average
Germany20_18_2	14.274	14.274
Germany20_18_40	79.484	82.878
Germany20_18_80	115.789	119.987
Germany40_35_2	12.918	12.918
Germany40_35_40	92.062	99.138
Germany40_35_80	168.715	174.438
Germany60_55_2	12.067	12.417
Germany60_55_40	114.176	115.518
Germany60_55_80	178.285	189.292

V. Conclusion

In this paper, we proposed a new genetic algorithm for solving MSONDP called GAMSONDP. We experimented on 45 random and 44 real world instances. With each data set, we run 20 times to take the best and average solutions. The results show that our algorithm is effective. It can solve big data sets that Branch-and-Price [1] does not. Besides, on small instance sets, our algorithm can give equivalent costs to Branch-and-Price with more than 50% test sets.

In the future, we are planning to improve the algorithm to reduce running time and to achieve optimal results on small data sets more. We also hope that we can solve this problem with larger test sets by these approaches as well as different ones.

Acknowledgment

We would like to thank Prof. Raoutia Taktak for providing us the materials related to their works on MSONDP problem.

References

[1] Sylvie Borne, Virginie Gabrel, R idha Mahjoub, and Raoutia Taktak, "Multilayer Survivable Optical Network Design," presented at International Network Optimization Conference (INOC), Hamburg, June 13-16, 2011.

[2] S. Borne , E. Gourdin, B. Liau, and A.R Mahjoub, "Design of survivable IP-over-optical network," *Springer Science and Bussiness Media*, pp. 41-73, 2006.

[3] H. Kerivin and A.R Mahjoub , Design of survivable networks: A survey, Networks 46(1) , 2005.

[4] Thomas Bucsics, "Metaheuristic Approaches for Designing Survivable Fiber-Optic Networks," Master's thesis, Institute for Computer Graphics and Algorithms, the Vienna University of Technology, 2007.

[5] Colin R. Reeves and Jonathan E. Rowe, *Genetic algorithms-principles and perspectives*. Kluwer Academic Publishers, 2003.

[6] Muhammad S. Javed, Krishnaiyan Thulasiraman, and Guolian (Larry) Xue, "Logical Topology Design for IP-over-WDM networks: A Hybrid Approach for Minimum Protection Capacity," presented at the IEEE 17th ICCCN, St.Thomas U.S Virgin Islands, August 3-7, 2008.

[7] TSPLIB [Online]. Available: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

[8] Adrian Zymolka, "Design of Survivable Optical Networks by Mathematical Optimization," Ph.D. dissertation, Mathematik und Naturwissenschaften, Technischen Universit¨, Berlin, 2007.

[9] S. Borne, E. Gourdin, O. Klopfenstein and A. R. Mahjoub, "The Multilayer Capacitated Survivable IP Network Design Problem: Valid Inequalities and Branch – and – Cut" presented at International Network Optimization Conference (INOC), Pisa, Italy, April 26-29, 2009.

[10] Kulathumani Vinodkrishnan, Arjan Durresi, Nikhil Chandhuk, Raj Jain, Ramesh Jagannathan, and Srinivasan Seetharaman, "Survivability in IP over WDM networks," in *Journal of High Speed Networks*, 2nd ed. vol. 10, IOS Press, 2011, pp. 79-90.

[11] D. Wagner, U. Pferschy, P. Mutzel, G. R. Raidl, and P. Bachhiesl, " A directed cut model or the design of the last mile in real-world fiber optic networks" presented at International Network Optimization Conference (INOC), Spa, Belgium, April 22-25, 2007.

Huynh Thi Thanh Binh was born in 26 Sept 1975 at Ha Noi, Viet Nam. She has received Master and PhD degree in Computer Science field at Ha Noi University of Science and Technology on 1999 and 2011. Her research is genetic algorithm.

Dr. Huynh Thi Thanh Binh is now lecturer at Ha Noi University of Science and Technology, School of Information and Communication Technology. She is also researcher at Vietnam Institute for Advanced Study in Mathematic form July 2012 to Dec 2012.

Dr. Binh is member of IEEE (from 2006 to now). Now, she is treasure of IEEE Viet Nam Chapter. Dr. Binh also is member of ACM form 2006 to 2011. Dr. Binh has served as the organizing chair of the SoICT2011, SoICT2012, a program commitee of the international conference ICCCI2012, ICCASA2012, KSE2012, reviewer of International Journal Intelligent Information and Database Systems.

Dinh Thi Ha Ly was born in 25 May 1992 at Bac Ninh, Viet Nam. She is now student at Ha Noi University of Science and Technology.