13

Greedy Algorithms in Survivable Optical Networks

Xiaofei Cheng
Institute for Infocomm Research (I2R)
Singapore

1. Introduction

Greedy algorithm is a simple and straightforward heuristic algorithm which is usually used to solve optimization problems. It has been widely applied in communication networks. In this chapter, we focus on greedy algorithm and its applications in survivable optical networks. After introducing basic concept and design method of greedy algorithm, we build up a mathematic model and design a greedy algorithm to solve backup resource reservation problems for protection against multiple-link failures in survivable optical networks. The design of the greedy algorithm is introduced in detail. Computational complexity and performance of the greedy algorithm are analyzed. Advances in greedy algorithms are discussed.

2. Optimization problem

An optimization problem is a problem in which the object is to find, not just a solution, but the best solution from all feasible solutions. More formally, optimization problem refers to study of problems and find a solution which has the minimum or maximum objective function value by systematically choosing the values of real or integer variables within a feasible region while satisfying all the constraints. Optimization problems are made up of three basic ingredients: (a) an objective function which the object is to minimize or maximize; (b) a set of variables which affect the value of the objective function; (c) a set of constraints that allow the variables to take on certain values but exclude others. Constraints are sometime not necessary. Some optimization problems are unconstrained. However, most of optimization problems, in practice, do have some constraints. In mathematics, the optimization problem can be represented as follows:

$$\min f(x) \quad or \quad \max f(x)$$

$$\text{Subject to: } g(x) \geq 0$$

$$x \in D$$

where *f(x)* is the objective function. g(*x*) represents constraint functions and often specified by a set of constraints, equalities or inequalities that the variable x has to satisfy. D is the

Source: Advances in Greedy Algorithms, Book edited by: Witold Bednorz,
ISBN 978-953-7619-27-5, pp. 586, November 2008, I-Tech, Vienna, Austria

feasible region or search space. Typically, D is some subset of the Euclidean space and is a set with limited elements. The elements of D are called candidate solution or feasible solution. Such a formulation is an optimization problem and the feasible solution that minimizes or maximizes the objective function is the optimal solution.

3. Advanced algorithms

We can get the optimal solution by enumerating all possible candidate solutions and comparing the value of objective function. However, it is time-consuming and the computational time is sometime unacceptable for complicated optimization problems. When the general methods are not available to find the optimal solution in an acceptable computational time, advanced algorithms are required to solve these problems.

Two fundamental goals are needed to be considered to design advanced algorithms: (a) running time; (b) optimal solution quality. Heuristic algorithm refers to an algorithm that is constructed by intuition or prior experiences and the heuristic algorithm abandons one or both of these goals. For example, some heuristic algorithms give up finding the optimal solution for an improvement in run time. Advanced algorithms belong to heuristic algorithm. Recently, advanced algorithms develop very quickly. These advanced algorithms include tabu search, simulated annealing, genetic algorithms, neural networks algorithms, greedy algorithm, etc. Among these advanced algorithms, greedy algorithms are used to solve optimization problems and sometimes work very well. Greedy algorithms are very simple and straightforward. They are widely used as heuristic algorithms and sometimes are designed to embed into other heuristic algorithms to solve the optimization problem due to its rapid calculation speed and acceptable solution quality.

4. Greedy algorithm

Greedy algorithm is a heuristic algorithm which can be used to solve optimization problems. The principle of greedy algorithm is to recursively constructe a set of objectives and makes the choice that is the best at each step. It is a step-by-step recursive method for solving optimization problems. The basic idea behind greedy algorithms is to build large solutions up from smaller ones while keeping only the best immediate or local solution they find as they go along. That is why we call "Greedy". Greedy algorithm works in phases. At each phase, a decision is made that appears to be the best at the step, without regard for future consequences**.** Generally, this means that some local optimum is chosen whereas this will not lead to globe optimum at the end of the algorithm. A greedy algorithm sometime can find the overall or globally optimal solution at terminate for some optimization problems. It is because a locally optimal choice leads to a globally optimal solution. However, it does not mean that is always yield optimal solutions. In most cases, it finds less-than-optimal solutions and produces a suboptimal solution. Even more, many optimization problems cannot be solved correctly by greedy algorithms. However, the optimal solution is not always required for some optimization problems in practices. In these cases, simple and fast greedy algorithms are always good algorithms which are used to generate approximate answers, rather than using the more complicated algorithms generally required generating an exact answer.

Greedy algorithms have many advantages and thus are very attractive. The greedy algorithms employ simple strategies that are simple to implement and require minimal

amount of resources. They are fast and generally linear to quadratic and require little extra memory. They are shortsighted in their approach in the sense that they take decisions on the basis of information at hand without worrying about the effect these decisions may have in the future. They are easy to invent, easy to be implemented and most of the time quite efficient. Even when greedy algorithms do not produce the optimal solution, they often provide fast heuristics (non-optimal solution), and are often used in finding good approximations. More over, the greedy algorithm can be designed to embed into other heuristic algorithms. In such cases the greedy method is frequently the basis of other heuristic approaches. Greedy algorithms have been widely applied to solve optimization problems in communications networks, e.g., Dijkstra's algorithm for finding the shortest path; Kruskal's algorithm for finding a minimum-cost spanning tree and Prim's algorithm for finding a minimum-cost spanning tree, etc. Dynamic programming is a powerful technique, but it often leads to algorithms with higher than desired running times.

5. Survivability in optical communication networks

Optical communication networks are high-capacity telecommunications networks which use fiber as transmission media to increase transmission distance and make use of wavelength-division multiplexing (WDM) technologies to increase the network capacity. Optical communication networks can provide routing, grooming, and restoration capabilities at the wavelength level. With the explosive growth of the Internet and the emergence of new services, optical communication networks are being rapidly deployed to satisfy the high-speed transport capacity demand economically. Optical communication networks have become the primary transport networks. Recently, optical communication network employing dense wavelength division multiplexing (DWDM) technology has currently harnessed several Terabits/s bandwidths into one fiber. As the size and capacity of transport network increase rapidly, any failure in the transport network will result in a significant loss of data and service. Therefore, survivability for high-speed communication networks becomes critical. In following section, we will discuss the greedy algorithm applications in survivable optical networks for protection against multiple-link failure [1].

6. Greedy algorithm in multiple-link failure protection in survivable optical networks

In a survivable optical networks, we model the network as a graph G (N, L), where N and L are the set of nodes and the set of links respectively. A link in the graph G represents bidirectional fiber between two nodes. For each link, set the cost of the link equals to the link distance. F is the number of simultaneous link failures. In order to provide 100% guarantee for dynamic traffic, when a new connection request arrives at an ingress node, the ingress node would choose a working path (WP) and F link-disjoint backup paths (BP) between itself and the intended egress node from a pre-computed routing table. Let k denote the arrival sequence of a connection and w_k denote the bandwidth requirement of the k^{th} connection. We make use of the following notations to describe the problem and the proposed solution [1, 2].

A'_e : Set of connections whose working paths traverse link e.

A_e : Total bandwidth of all connections in A'_e.

B'_e: Set of connections whose backup paths traverse link *e*.

B_e: Backup bandwidth reserved on link *e*. Since sharing of protection bandwidth is allowed, $B_e \le \sum_{k \in B'_e} w_k$.

R_e: Residue bandwidth of link *e*; i.e., $R_e = C - A_e - B_e$ where C is the total capacity of link *e*.

S'^{b}_{a}: Set of connections whose working paths traverse link *a* and backup path traverse link *b*, i.e. $S'^{b}_{a} = A'_a \bigcap B'_b$.

S^{b}_{a}: Total bandwidth of all the connections in S'^{b}_{a}. This is the total bandwidth of all connections whose AP traverse link *a* and BP traverse link *b*.

$U'^{e}_{a,b,c,\ldots n}$: Set of connections whose working paths transverse any of the links *a*, *b*, *c*… *n* and whose backup paths traverse link *e*. In case of all links *a, b, c … n* failure simultaneously, all connections in $U'^{e}_{a,b,c,\ldots n}$ will be protected by link *e*.

$U^{e}_{a,b,c,\ldots n}$:Total backup bandwidth needed on link *e* for all the connections in set $U'^{e}_{a,b,c,\ldots n}$.

$C^{e}_{a,b,c,\ldots n}$: Additional backup bandwidth needed on link *e* in order to use it as part of a BP for a new connection to protect against simultaneous failures of links *a*, *b*, *c*… *n*.

C_e : Additional bandwidth needed on link *e* when a new connection is established.

$W^{k,j}$: A variable that takes on the value of 1 if the j^{th} alternative route between the node pair of the k^{th} connection is selected as the AP of the connection; 0 otherwise.

$P^{k,j}$: A variable that takes on the value of 1 if the j^{th} alternative route between the node pair of the k^{th} connection is selected as the BP of the connection; 0 otherwise.

R(k): The number of alternative routers between the node pair of the k^{th} connection.

V (.): A function that returns the total bandwidth for all the connections in a set *S*, such that $V(S) = \sum_{k \in S} w_k$, where *S* is a set of connections.

Based on the above definition, we can get the following equations:

$$A_e = \sum_{k \in A'_e} w_k \tag{1}$$

$$S'^{b}_{a} = A'_a \bigcap B'_b \tag{2}$$

$$S^{b}_{a} = \sum_{k \in s^{b}_{a}} w_k \tag{3}$$

$$U'^{e}_{a,b,c,\ldots n} = S'^{e}_{a} \bigcup S'^{e}_{b} \bigcup \ldots \bigcup S'^{e}_{n} \tag{4}$$

$$U^{e}_{a,b,c,\ldots n} = \sum_{k \in U'^{e}_{a,b,c,\ldots n}} w_k \tag{5}$$

The mathematic model of the optimization problem is given as follows:

Objective: Minimize the total network bandwidth consumption:

$$Minimize \sum_{e \in L} [A_e + B_e] \tag{6}$$

where:

$$B_e = \max_{L_f \in L} (U^e_{L_1, L_2, \cdots L_n}) \tag{7}$$

Subject to:
Link capacity constraints:

$$A_e + B_e \le C \qquad \forall e \tag{8}$$

Traffic demand constraints:

$$\sum_{j=1}^{R(k)} W^{k,j} = 1; \sum_{j=1}^{R(k)} P^{k,j} = F \quad \forall k \tag{9}$$

Link-disjoint constraints:

$$W^{k,j} + P^{k,j} \le 1 \qquad \forall k, j \tag{10}$$

Integer flow constraints:

$$W^{k,j}; P^{k,j} \in \{0,1\} \qquad \forall k, j \tag{11}$$

The mathematical model described in this section is suitable for static traffic pattern. To expedite a fast on-line algorithm for dynamic traffic, we present a fast on-line Greedy heuristic algorithm. The design of the greedy algorithm is analyzed as follows.
In order to provide 100% guarantee for network connections, total backup bandwidth reserved on a link *e* should protect against random *F*-link simultaneous failure. Assume that *F*-link set is represented by $L_f = \{l_1, l_2, ..l_i..l_F \mid l_i \in L\}$. To protect against this *F*-link failure, the total backup bandwidth reserved on the link *e* is $(U^e_{l_1, l_2, ..l_i .., l_F})$. For random *F*-link failure case, the total backup bandwidth on link *e* must be enough to cater for the worst case and is therefore given by:

$$\begin{aligned} B_e &= \max_{\substack{l_i \in L \\ i=1,2,...F}} (U^e_{l_1, l_2, \cdots l_F}) \\ &= \max_{\substack{l_i \in L \\ i=1,2,...F}} \sum_{k \in S^e_{l_1} \cup S^e_{l_2} \cup ... \cup S^e_{l_F})} w_k \end{aligned} \tag{12}$$

When a new connection k_{new} (bandwidth requirement: $w_{k_{new}}$) arrives, the ingress node will choose a working path (WP) and *F* link-disjoint backup paths (BP) from a list of pre-computed routes between the source and the destination (We assume the network is *F*+1-

connected) to protect against *F* link failures. The backup bandwidth will be reserved on each link of the BPs of the new connection. The objective is to find a working path and *F* backup paths from pre-computed routing table with the minimal total additional bandwidth to carry the new connection:

$$\text{Minimize} \sum_{e \in WP \cup BP} C_e \tag{13}$$

For each link *e* of a candidate working path, an additional bandwidth $w_{k_{new}}$ is required to carry the new requested connection. For each link *e* of candidate backup paths, admission of connection k_{new} changes the sets A'_e, B'_e, $S'^{e}_{l_i}$ and $U'^{e}_{l_1,l_2,\cdots l_F}$. Since the total backup bandwidth on link *e* before the admission is B_e, to protect against the simultaneous failures of links $L_1, L_2, \ldots L_n$, the additional backup bandwidth needed on link *e* for setting up a new connection k_{new} is given by:

$$C^{e}_{L_1,L_2,\ldots L_n} = \begin{cases} 0 & \text{if } (U^{e}_{L_1,L_2,\ldots L_n} + w_{k_{new}}) < B_e \\ (U^{e}_{L_1,L_2,\ldots L_n}) + w_{k_{new}} - B_e & \text{if } B_e < (U^{e}_{L_1,L_2,\ldots L_n} + w_{k_{new}}) < R_e \\ \propto & \text{if } U^{e}_{L_1,L_2,\ldots L_n} + w_{k_{new}} - B_e > R_e \text{ or } e \in L_f \text{ or } R_e < w_{k_{new}} \end{cases} \tag{14}$$

Note that $C^{e}_{L_1,L_2,\ldots L_n} = \propto$ means that it is not feasible to set up the new connection with BP traverses link *e*. B_e is calculated according to the equation (12). The additional bandwidth on link *e* of a candidate backup path for protection against random *F* link-failure, $C_e (0 \le C_e \le w_{k_{new}})$ is given by:

$$C_e = \max_{\forall L_f \in L} C^{e}_{L_1,L_2,\ldots L_n} \tag{15}$$

Equation (12) ~ (15) gives a protection scheme for protect against random *F* (*F*>2) multiple-link failure scenarios. However, the computation time for algorithm in equation (12) is $O(L^F)$. The computation time increases exponentially with the number of simultaneous link faults number *F*. It is not suitable for on-line resource reservation for dynamic traffic. To solve this problem, we present a greedy algorithm described below to calculate B_e instead of equation (12). The detailed steps of greedy algorithm are described as follows:

- **STEP 1**: Set *j*=1. Find the relative connection set of link *e*, *S*'(*e*), as given by

$$S'(e) = \left\{ S'^{e}_{l_i} \mid l_i \in L, S'^{e}_{l_i} \neq \Phi \right\} \tag{16}$$

- **STEP 2**: Find in *S*′ (*e*) the $S'^{e}_{l_i}$ with maximum $V(S'^{e}_{l_i})$ and denote it as $S'_{\max}$.

  Calculate M (j) as given by:

$$M(j) = V(S'_{\max}) = \max_{S' \in S'(e)} V(S') \tag{17}$$

- **STEP 3**: Update S' (e) and every $S' \in S'(e)$ as follows:

$$S'(e) = S'(e) - \{S'_{\max}\} \tag{18}$$

$$S' = S' - S'_{\max}, \ \forall S' \in S'(e) \tag{19}$$

Where $A - B = A \bigcap \bar{B}$.

- **STEP 4**: Increment j. Repeat Step 2, 3 until M (1), M (2),… M (F) are found. We have then:

$$B_e = \sum_{j=1}^{F} M(j) \tag{20}$$

In equations (16) – (20), we iterate F times to get the F sequential maximal bandwidth required on link e for protect against random single link failure scenarios. After each iteration time, a connection set update operation is implemented to avoid iterate calculating a connection' bandwidth. The time complexity of the greedy algorithm (equations (16) – (20)) is O (LF).

In a centralized network, a control node will maintain the connection set, $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$ information. When a dynamic connection arrives, the control node will assign an optimal working and F backup path and reserve backup bandwidth according to equations (13)-(20). The connection set, $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$ will be updated. In a distributed network, each node will maintain the connection set, $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$. When a dynamic connection arrives, the ingress node will assign an optimal working and F backup paths and reserve backup bandwidth according to equations (13)-(20). Then, the connection set $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$ is updated and the updated information is broadcasted to all distributed nodes.

We simulate different routing and bandwidth allocation schemes on three network topologies: (a) a 8-node all connection network; (b) NJLATA network and (c) NSFNET network. The average node degrees $\bar{d}$ of the network topologies are 5, 4 and 3, respectively. We studied the following 4 algorithms: (1) greedy algorithm with alternate routing scheme (2) greedy algorithm with Fixed (Shortest path) routing scheme (3) ESPI algorithm [3, 4] and (4) NS: Shortest path with No backup bandwidth Sharing algorithm. We simulate these four algorithms with Matlab 6.2 version and 2.0GHZ CPU. In our simulation, connection request follows a Poisson process and has an exponential holding time. Connection request arrival is uniformly distributed over all node pairs and a total of 10^6 connection requests are generated. The bandwidth requirement of each demand varied randomly from 1 to 4 units. Simulation shows that our Greedy algorithm with alternate routing scheme has the least bandwidth consumption. Alternate routing is more superior in saving bandwidth than fixed routing. Greedy algorithms save more total bandwidth consumption (F=2, 12%; F=3, 17%; F=4, 22%) than the ESPI algorithm and save 32% of the total bandwidth consumption using NS algorithm. Our greedy algorithms have less blocking probability than other algorithms. Our greedy algorithms achieve superior performance both in total bandwidth consumption and blocking probability than other algorithms for protection against multiple-link failures.

7. Advances in greedy algorithms

Greedy algorithms are fast, simple straightforward and generally linear to quadratic. They employ simple strategies that are simple to implement and require minimal amount of resources. They are easy to invent, easy to implement and most of the time quite efficient. So, simple and fast greedy algorithms are always good algorithms to solve the optimization problems when the optimal solution is not required.

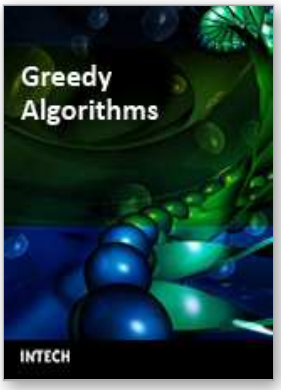
8. Reference

[1] Xiaofei Cheng, Xu Shao, Yixin Wang, "Multiple Link Failure Recovery in Survivable Optical Networks", Photonic Network Communications, pp. 159-164, 14 (2007), July 2007.

[2] Xiaofei Cheng, Teck Yoong Chai, Xu Shao, Yixin Wang, "Complementary Protection Under Double Link Failure for Survivable Optical Networks", IEEE GLOBECOM, OPN07-2, California, USA, 27 Nov.-1 Dec. 2006.

[3] Chunming Qiao, Dahai Xu,"Distributed Partial Information Management (DPIM) Scheme for Survivable Networks-Part I", INFOCOM 2002. Vol 1, pp. 302 –311, 2002.

[4] Murali Kodianalm,T.V.Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," IEEE, INFOCOM 2000, pp.902~910,2000.



Greedy Algorithms
Edited by Witold Bednorz

ISBN 978-953-7619-27-5
Hard cover, 586 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

Each chapter comprises a separate study on some optimization problem giving both an introductory look into the theory the problem comes from and some new developments invented by author(s). Usually some elementary knowledge is assumed, yet all the required facts are quoted mostly in examples, remarks or theorems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Xiaofei Cheng (2008). Greedy Algorithms in Survivable Optical Networks, Greedy Algorithms, Witold Bednorz (Ed.), ISBN: 978-953-7619-27-5, InTech, Available from: http://www.intechopen.com/books/greedy_algorithms/greedy_algorithms_in_survivable_optical_networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821