Finding shortest path in static networks: using a modified algorithm

Sahar Abbasi, Fahimeh Moosavi

Department of Industrial Engineering, Najafabad Branch, Islamic Azad University, Esfahan, Iran

Abstract

This paper considers the problem of finding the shortest path in a static network, where the costs are constant. The CE Algorithm based strategy that is presented by Rubinstein to solving rare event and combinatorial optimization problem is modified to finding shortest path in this research. To analyze the efficiency of the used algorithm three sets of small, medium and large sized problems that generated randomly are solved. The results on the set of problems show that the modified algorithm produces good solutions and time saving in computation of large-scale network.

Keywords: shortest path, CE Algorithm, rare event, combinatorial

© 2012 Published by SSBF.

1. Introduction

Shortest path problem is one of the most important network flows problem that seeks for the paths with minimum cost from source node to sink node in network. Considering many applications for this problem, it also can be used as starting point of solving more complicated problem in network flow area. When we want to send some material between two specified points in a network as quickly or as cheaply, applications of the shortest path problem are arises. This paper modifies Cross-Entropy algorithm to find the shortest path from the source node to the sink node in acyclic and static networks.

The cross-entropy (CE) method attributed to Reuven Rubinstein is a versatile adaptive Monte Carlo algorithm originally developed for rare-event. The method originated from the field of rare event simulation, where very small probabilities need to be accurately estimated [1]. It has been applied to a diverse range of difficult simulation problems, such as marginal likelihood computation in Bayesian statistics, estimation of large portfolio loss probabilities in credit risk models, efficient simulation of buffer overflow probabilities in queuing networks and network reliability estimation in telecommunications[2-4]. A recent review of the CE method and its applications can be found in Kroese [5]; a book-length treatment is given in Rubinstein and Kroese[6].

This paper is organized as follows: After review of the shortest path problem in Section 2, we define notation of the shortest path problem and CE algorithm Section 3, then we use CE algorithm to solving this problem and summarize our conclusions the related problems in Sect. 4, 5 respectively.
2. Literature Review

Some simple classic algorithms to solve shortest path problem used the label setting and label correcting algorithms. The simplest case of the shortest path problem, where weights of the network are static, can be solved in $O(n^2)$ time using Dijkstra’s algorithm [7]. There are several methods to find the shortest path from the source node to the sink node based on dynamic programming, zero-one programming and also network flows theory that some of these algorithms can be found in [8]. Okada and Gen discussed the problem of finding the shortest paths from a fixed origin to a specified node in a network with arcs represented as intervals on real line [9]. Gent et al. investigated the possibility of using genetic algorithms to solve shortest path problems [10]. Gupta and Pal presented an algorithm for the shortest path problem when the connected arcs in a transportation network are represented as interval numbers [11].

In this paper some basic definitions and Preliminaries of networks and CE algorithm are presented then the modified algorithm is proposed. To show the advantages of the used algorithm, numerical examples are solved and obtained results are discussed.

![Weighted network](image)

Fig1. Weighted network [12]

3. Basic Definitions and Preliminaries

We are given a directed network $G = (N, A)$ with node set $N = \{1, 2, \ldots, n\}$ and arc set $A \subseteq N \times N$. Let $m$ denote the number of arcs in network $G$, i.e., $m = |A|$. To simplify notation, we assume (without loss of generality) that every pair of nodes is connected by at most one arc. Each arc $(i, j) \in A$ has a traversal cost $c_{i,j}$. Without ambiguity, throughout the rest of this paper, we assume that the length of a path is equal to its cost and use interchangeably the terminologies cost and length. Let us assume that the graph is complete. A path $P$ is said to be a shortest path from node $i$ to node $j$ if $\text{Cost}[P] \leq \text{Cost}[\hat{P}]$ for all paths $P$ from node $i$ to node $j$, a sample of a weighted network is given in Fig.1.

We now present a generic algorithm based on cross-Entropy Method for solving this problem. Let $\mathcal{X}$ be the set of all possible paths and let $S(x)$ be the sum length of paths $x \in \mathcal{X}$ and $f(x)$ be the set of capacities of paths $x$. We can represent each path via a permutation of $(1,2,\ldots,N)$. For example for $N=4$, the permutation $(1,2,3,4)$ represents the path $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. In fact, we may as well represent a path via a permutation $x=(x_1,x_2,\ldots,x_n)$ with $x_1=1$. We may formulate the problem as follows:

$$\min_{x \in \mathcal{X}} S(x) = \min_{x \in \mathcal{X}} \left\{ \sum_{i=1}^{N} c_{x_i,x_{i+1}} \right\} \quad (1)$$
In order to apply the CE algorithm we need to specify (a) how to generate the random paths, and (b) how to update the parameters at each iteration. The easiest way to explain how the paths are generated and how the parameters are updated is to an equivalent minimization problem. Let

$$\tilde{\chi} = \{(x_1, ..., x_n): x_1 = 1, x_i \in \{1, ..., n\}, i = 2, ..., n\}$$  \hspace{1cm} (2)

Be the set of arcs that that correspond to paths that start in 1 and end in N. Again we will define the function $\tilde{S}$

On $\tilde{\chi}$ by $\tilde{S}(x) = S(x)$, if $x \epsilon \chi$ and $\tilde{S}(x) = \infty$, otherwise. Then, obviously Eq.1 is equivalent to the minimization problem

$$\text{minimize } \tilde{S}(x) \text{ over } x \epsilon \tilde{\chi}$$  \hspace{1cm} (3)

A simple method to generate a random path $X= (x_1, x_2, ..., x_n)$ is to use a Markov chain of the graph $G$, starting at node one (source node) and stopping after $n$ steps. Let $P^n(p_{ij})$ denote the one-step transition matrix of this Markov chain. We assume that the diagonal elements of $P$ are zero, and that all other elements of $P$ are strictly positive, but otherwise $P$ is a general $n \times n$ stochastic matrix.

The pdf $f(\cdot; P)$ of $X$ is thus parameterized by the matrix $P$ and its logarithm is given by

$$\ln f(x; P) = \sum_{r=1}^{n} \sum_{i,j} f[\chi_{ij}(r)] \ln p_{ij}$$  \hspace{1cm} (4)

Where $\chi_{ij}(r)$ is the set of all paths in $\tilde{\chi}$ for which the $r$th transition is from node $i$ to $j$. The updating rules for this modified optimization problem derive of $\{S(X_i) \leq \gamma\}$, under the condition that the rows of $P$ sum up to 1. Using Lagrange multipliers $u_1, ..., u_n$ we obtain the maximization problem

$$\max_{P} \min_{u} \left[ \mathbb{E}_{P} \left[ S(X) \gamma \right] + \sum_{i=1}^{n} u_i \left( \sum_{j=1}^{n} p_{ij} \right) \right]$$  \hspace{1cm} (5)

Differentiating the expression within square brackets above with respect to $p_{ij}$, yields, for all $j = 1, 2, ..., n$,

$$\frac{\mathbb{E}_{P} \left[ S(X) \gamma \right] \sum_{k=1}^{\gamma} f[\chi_{ij}(r)]}{p_{ij}} + u_{ij} = 0$$  \hspace{1cm} (6)

Summing over $j = 1, 2, ..., n$ gives

$$\mathbb{E}_{P} \left[ S(X) \gamma \right] \sum_{j=1}^{n} f[\chi_{ij}(r)] = -u_{i},$$

where $\chi_{i}(r)$ is the set of paths for which the $r$-th transition starts from node $i$. It follows that the optimal $p_{ij}$ is given by

$$p_{ij} = \frac{\mathbb{E}_{P} \left[ S(X) \gamma \right] \sum_{j=1}^{n} f[\chi_{ij}(r)]}{\mathbb{E}_{P} \left[ S(X) \gamma \right] \sum_{j=1}^{n} f[\chi_{ij}(r)]}$$  \hspace{1cm} (7)

The corresponding estimator is

$$p_{ij} = \frac{\sum_{k=1}^{\gamma} f[x_k \chi_{ij}(r)]}{\sum_{k=1}^{\gamma} f[x_k \chi_{ij}(r)]}$$  \hspace{1cm} (8)

To update $p_{ij}$ we take the fraction of times that the transitions from $i$ to $j$ occurred, taking only those paths into account that have a total length less than or equal to $\gamma$.

Since we now only generate paths, the update value for $p_{ij}$ can be estimated as
\[ \hat{P}_{ij} = \frac{\sum_{x_{ij} \in X_{ij}} (e(x_{ij}))}{\sum_{x_{ij} \in X_{ij}}} \]  

(9)

Where \( X_{ij} \) is the set of paths in which the transitions from \( i \) to \( j \) is made. To complete the algorithm, we need to specify the stopping criterion. For the initial matrix \( \hat{P}_0 \) we could simply take all off-diagonal elements equal to

\[
\frac{1}{\text{non-zero elements in each row}}
\]  

(10)

And for the stopping criterion use

\[ \hat{y}_t = \hat{y}_{t-1} = \cdots = \hat{y}_{t-d} \]  

(11)

Where \( t \) denote the final iteration.

The algorithm to generation of paths is given below:

1. Define \( P^{(1)} = P \) and \( X_1 = 1 \). Let \( k = 1 \).
2. Obtain \( P^{(k+1)} \) from \( P^{(k)} \) by first setting the \( X_k \)-th column of \( P^{(k)} \) to 0 and then normalizing the rows to sum up to 1. Generate \( X_{k+1} \) from the distribution formed by the \( X_k \)-th row of \( P^k \).
3. If \( k = n \) then stop; otherwise \( k = k+1 \) and reiterate from step 2.

4. Numerical Examples

The algorithms were coded in MATLAB and run on a Microsoft Windows SEVEN Professional notebook with 2.3 gigabyte ram and 3 gigabyte swap, running Digital Intel Core i5 Duo CPU. to check for the efficiency and validity of the algorithm some experiments are generated randomly using the suggested algorithm for three sets of networks in small, medium and large sizes. Specifications of the problems is given in Table 1.

| #problem | # node | #arc | #problem | # node | #arc |
|----------|--------|------|----------|--------|------|
| 1        | 20     | 300  | 8        | 500    | 7500 |
| 2        | 40     | 600  | 9        | 800    | 12000|
| 3        | 60     | 900  | 10       | 1000   | 15000|
| 4        | 80     | 1200 | 11       | 1500   | 22500|
| 5        | 100    | 1500 | 12       | 2000   | 30000|
| 6        | 200    | 3000 | 13       | 2500   | 37500|
| 7        | 300    | 4500 |          |        |      |

A set of four small sized, five medium sized and four large sized acyclic network problems ranging from 20 to 2500 nodes are solved. Table 2 presents the performance of this algorithm for the mentioned problem. In all numerical problems we consider the value of each for each edge between (20,250) unit randomly and use \( \rho = 10^{-2}, \alpha = 0.5 \) and \( d=5 \).

In Table 2, \( n \) denotes the number of nodes in network, \( T \) denotes the total number of iterations needed before stopping, \( \gamma_1 \) and \( \gamma_T \) denote the initial and final estimates of the optimal solutions, \( \gamma^* \) denotes the best known solution and finally CPU denotes the CPU time in seconds.
Fig. 2 presents the running time CE Method for each of the 13 test problems. Fig. 2 shows running time of CE Method doesn't have significant increasing.

5. Conclusion
There are lots of classic algorithms for solving network flow problems like shortest path but in large size problems, the traditional algorithms have not capability of finding solutions in reasonable CPU time. The CE algorithm that is modified and applied in this paper can find high quality solutions and fairly quick for shortest path problem. Fig. 2 presents the running time of the CE algorithm for each of the 13 test problems in small, medium and large sized network. The figure shows running time is linear. It could be seen that the CE algorithm reduces CPU time. This algorithm could easily be modified to solve a problem with two objectives.

References
[1] P-T. De Boer, Kroese, D.P, Mannor, S. and R.Y. Rubinstein, “A tutorial on the cross-entropy method”, Annals of Operations Research, 134 (1), 19-67, 2005.
[2] J. C. C. Chan and E. Eisenstat. Marginal likelihood estimation with the cross-entropy method. 2011. Submitted.
[3] J. C. C. Chan and D. P. Kroese. Efficient estimation of large portfolio loss probabilities in t-copula models. European Journal of Operational Research, 205:361–367, 2010.
[4] K-P. Hui, N. Bean, M. Kraetzl, and D. P. Kroese. The cross-entropy method for network reliability estimation. Annals of Operations Research, 134:101–118, 2005.

[5] D. P. Kroese and R. Y. Rubinstein. The transform likelihood ratio method for rare event simulation with heavy tails. Queueing Systems, 46:317–351, 2004.

[6] D. P. Kroese. The cross-entropy method. In Wiley Encyclopedia of Operations Research and Management Science. Wiley, 2011.

[7] E.W. Dijkstra, A note on two papers in connection with graphs, Numerische Mathematics 1 -269–271. (1959).

[8] 1 M. Bazaraa, J. Jarvis, H. Sherali, Linear Programming and Network Flows, second ed., Wiley, New York, 1990.

[9] S. Okada and M. Gen, ‘Fuzzy shortest path problem,’ Comput. Ind. Eng.,27, 465-468, 1994.

[10] Gent, M., Cheng, R. and Wang, D. Genetic algorithms for solving shortest path problems,Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 401-406. (1997).

[11] Gupta, A. S. and Pal, T. K. Solving the shortest path problem with interval arcs, Fuzzy Optimization and Decision Making, Vol. 5, No. 1, pp. 71-89. (2006).

[12] C. Davies, P. Lingras, Genetic algorithms for rerouting shortest paths in dynamic and stochastic networks, European Journal of Operational Research 144:27–38. (2003).