Solve Constrained Minimum Spanning Tree
By cross-entropy (CE) method

Jing Li, Zaifu Guo
Institute of Public Safety, China Academy of Safety Science and Technology, Beijing, China
jacklongdon@126.com

Abstract—Given an undirected graph and nonnegative numbers as weights for each edge, we consider the problem of finding a spanning tree that has lowest total cost with respect to cost weight and has constrained budget with respect to constrains weight [1]. Different from the traditional Lagrangean relaxation method, we proposed a new randomized method for solving the constrained spanning tree problem in this paper. For this newly proposed method every solution found is in the original problem’s feasible region, there is no need to do the hard work to close the gap between the Lagrangean relaxation and the original problem. And the proposed algorithm is very easy to parallelize, can take full advantage of multi-core processors to improve problem solving efficiency. For most optimization algorithms, properly selecting the super parameters have big impact on the algorithm’s practical performance. Through computer numerical simulation, we can see that our algorithm is robust to super parameters.

1. INTRODUCTION
Given an undirected graph $G = (V, E)$ and nonnegative numbers $l_{ei} (0 \leq i \leq n)$ as weights for each edge $e \in E$, we consider the problem of finding a spanning tree that has lowest total cost with respect to weight $l_{ei}$ and has constrained total cost with respect to $l_{ei} (1 \leq i \leq n)$ [1]. For convenience, we will refer to $l_{e0}$ of an edge $e$ as its cost weight and $l_{ei} (1 \leq i \leq n)$ of an edge $e$ as its constrain weights respectively. Thus the problem we consider is that of finding spanning tree with smallest total cost weight and with all its total constrain weights beneath predefined constrained value.

In the next section, we review some basic algorithm we used in this paper. Then we present the constrained spanning tree problem’s formal description in Section 3. In Section 4 we developed a novel approximation algorithm for solving the constrained span tree problem. In Section 5, we do some experiment through computer simulation to prove that our algorithm can find the constrained span tree problem’s sub-optimal solution robustly. In the last section we conclude the paper and post the future work’s direction.

2. BACKGROUND KNOWLEDGE

2.1 Lagrangean relaxation
Lagrangean relaxation is a classical technique to get rid of a set of "hard" constraints in an optimization problem. This gives lower bounds (for minimization problems) on the optimum value, and their solutions, while usually infeasible for the original problem, can often be used as staring points for specialized heuristics. The literature on Lagrangean relaxation, its extensions and applications is enormous. We refer the reader to Nemhauser [2] for a discussion of the method, and M.Guignard [3] is...
a very good tutorial. In this paper, we consider the application of Lagrangean relaxation to the constrained spanning tree problem, although we does not take a traditional approach.

2.2 The cross-entropy (CE) method
The cross-entropy (CE) method [4] is a new generic approach to combinatorial and multi-extremal optimization and rare event simulation. The CE method was motivated by an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks. The CE method involves an iterative procedure where each iteration can be broken down into two phases:

- Generate some random data samples (trajectories, vectors, etc.) according to a specified mechanism.
- Update the parameters of the random mechanism based on the data to produce “better” sample in the next iteration.

The significance of the CE method is that it defines a precise mathematical framework for deriving fast, and in some sense optimal updating/learning rules, based on advanced simulation theory. Other well-known randomized methods for combinatorial optimization problems are simulated annealing, tabu search, genetic algorithms and the ant colony optimization meta-heuristic.

An increasing number of applications are being found for the CE method. Recent publications on applications of the CE method include: buffer allocation; static simulation models; queuing models of telecommunication systems; neural computation; control and navigation; DNA sequence alignment; scheduling; vehicle routing and more.

3. PROBLEM FORMAL DESCRIPTION
This is a multi-criteria problem. A natural way to formulate such problems is to specify budgets as inequalities and minimize the other objective under these constraints. The problem therefore becomes a capacitated problem. In this case, we can specify some budgets \( L_i \) on the total constrains weights of the spanning tree and require a tree of minimum cost weight under these budgets restriction. We call this problem the Constrained Minimum Spanning Tree problem. In this section, we give this problem’s formal description and transform it to a more algorithm friendly form.

3.1 The constrained span tree problem
Given a graph \( G = (V, E) \), let \( S \) denote the set of incidence vectors of spanning trees of \( G \). The constrained minimum spanning tree problem can be formulated by the following optimization problem:

\[
W = \min \sum_{e \in E} l_e x_e
\]

subject to:

\[
\sum_{e \in E, 1 \leq i \leq n} l_{ei} x_e \leq L_i, \quad (3.1.1)
\]

3.2 Apply Lagrangean relaxation
The Lagrangean relaxation of the constrained spanning tree problem described by formula (3.1.1) can be expressed by the following formula:

\[
LW - \sum_{e \in E} \sum_{0 \leq i \leq n} \lambda_i l_{ei} x_e - \sum_{1 \leq i \leq n} \lambda_i L_i
\]

(3.2.1)
To simplify our proposed algorithm’s parameter searching procedure, we want to change the parameter’s range from \([1, \infty)\) to \((0, 1)\). More specifically, we change the Lagrangean relaxation to the following form:

\[
LW' = \sum_{e \in E} \sum_{1 \leq i \leq n} \lambda_i^* e_i x_e - \sum_{1 \leq i \leq n} \lambda_i^* L_i
\]  

subject to:

\[
x \in S
\]

\[
0 \leq \lambda_i^* \leq 1 (0 \leq i \leq n)
\]

\[
\sum_{i = 0}^{n} \lambda_i^* = 1
\]

\[
\lambda_i^* = \frac{1}{1 + \sum_{i=1}^{n} \lambda_i^*}
\]

\[
\sum_{e \in E, 1 \leq i \leq n} l_{ei} x_e \leq L_i
\]  

(3.2.4)

In traditional method, we get a good \(W''\)’s lower bound by solve the following problem:

\[
\min_{\lambda} \min_{x_e} LW''
\]  

(3.2.5)

Then branch and cut or other gap closing method [5] is used to solve similar problems. Because what we can get from \((3.2.5)\) is not a directly feasible solution for the original problem, the time consuming gap closing process can’t be avoid.

Different from the traditional Lagrangean relaxation method, we take another approach. We observed that:

\[
W \leq \min_{\lambda'} \sum_{e \in E} l_{0e} \arg \min_{x_e} LW'
\]  

(3.2.6)

Our basic idea is using right part of formula \((3.2.6)\) to approximate \(W\). By this method, every solution we found is in the original problem’s feasible region, although possibly only a sub-optimal solution. There is no need to do the hard work to close the gap between the Lagrangean relaxation and the original problem.

3.3 Some analysis

In right part of the formula \((3.2.6)\), there are two minimizing operations. To minimize over \(x_e\), we can simply use graph’s classical minimum spanning tree algorithm [7][8][9][10] for one weight, it’s a polygonal time complexity procedure. To minimize over \(\lambda'\), we use a cross-entropy (CE) like method to search the \(\lambda'\) parameter space.

We sample some \(\lambda'\) parameters uniformly from the parameter space defined by \((3.2.4)\), select the top \(k\) most optimal parameters, and then generate new parameters near these optimal parameters. In order to prevent us from getting stuck at local minima, we also sample \(\lambda'\) parameters uniformly from the parameter space defined by \((3.2.4)\) in the pilot process. In the next section, we will describe our proposed algorithm in detail.
4. PROPOSED METHOD

4.1 The pseudo code for the proposed method

Algorithm 1 Lagrangean relaxation by CEM

procedure LRCEM

Step 1:

Sample $m$ samples $\lambda'_i$ uniformly:

$\lambda'_i = \{\lambda'_{i0}, ..., \lambda'_{im}\}$ $i \in \{1, ..., m\}$

$0 \leq \lambda'_{ij} \leq 1$ $j \in \{0, ..., n\}$

$\sum_{j=0}^{n} \lambda'_{ij} = 1$

$\text{SET}_p \leftarrow \{\lambda'_{00}, ..., \lambda'_{mn}\}$

$opt_{old} \leftarrow \infty$

$\delta \leftarrow$ a predefined value.

Step 2:

$opt_{new} \leftarrow \min_{p \in \text{SET}_p, p \text{ agree with } (3.2.2)} LW'(p)$

if $\|opt_{new} - opt_{old}\| \leq \delta$ then

return $\min \{opt_{new}, opt_{old}\}$

end if

$\{p_1, ..., p_k\} \leftarrow \arg\min_{p \in \text{SET}_p, p \text{ agree with } (3.2.2)} LW'(p)$

Step 3:

for all $p$ in $\text{SET}_p$ do

$\rho = \min_{p \in \text{SET}_p, q \neq p} \|p - q\|_{L^2}$

Sample $\{m * \rho/k\} - 1$ samples uniformly from a $n+1$ dimensional cube centered at $p$.

Add these samples to $\text{SET}_p$.

end for

Sample $m * (1 - \rho)$ samples $\lambda'_i$ uniformly

$\lambda'_i = \{\lambda'_{i0}, ..., \lambda'_{im}\}$ $i \in \{1, ..., m * \rho\}$

$0 \leq \lambda'_{ij} \leq 1$ $j \in \{0, ..., n\}$

$\sum_{j=0}^{n} \lambda'_{ij} = 1$

Add these samples to $\text{SET}_p$.

go to Step 2:

end procedure

Figure 1 Pseudo code for the proposed algorithm
4.2 The description for the proposed method

Aggarwal, Aneja and Nair [11] studied the constrained minimum spanning tree problem; they prove weak NP-hardness and describe computational experience with an approach for exact solution. For more than one constraints, the problem is even harder. The novel method we present here has natural parallelism, and can use multicore processor’s computing resource efficiently and easily.

The proposed method search optimal $x$ in parameter space defined by (3.2.4). The parameter searching processing includes three steps.

At the first step, we do some variable initialization and sample initial parameters search set. We sample $m$ parameters uniformly in the space defined by (3.2.4), and save these parameters to a parameter set that we call $SET_p$, $m$ is a predefined number. Initiate the current optimal total spanning tree edge cost sum to positive infinity. We also defined a small value $\delta$, if the parameter searching process can’t find a more optimal value improved by $\delta$, then the algorithm stop.

At the second step, for every parameter in $SET_p$, the algorithm use classical minimum spanning tree algorithm to solve the problem defined by (3.2.3). Then check that whether the returned spanning tree is in the feasible region defined by (3.2.4). For all the feasible solutions, select the $k$ smaller ones and replace $SET_p$ with these parameters. Then we check whether the newly obtained optimal value is improved more than $\delta$. If the answer is not, stop the algorithm and output the current found optimal value. If the enhancement is big enough, go to the third step.

At the third step, at the beginning, $SET_p$ only has $k$ parameters, we enlarge the $SET_p$ by two means. Some parameters are sampled near the $k$ optimal parameters found at step two, we call these parameters are exploitation parameters. Some parameters are sampled uniformly in the space defined by (3.2.4), we call these parameters are exploration parameters. After the third step, parameters in $SET_p$ is $m$, the ratio between exploitation parameters and exploration parameters is controlled by a predefined value $r$.

For more detailed algorithm procedure, readers can refer to the pseudo code in figure 1.

5. EXPERIMENT AND ANALYSIS

We create a 50$\times$100 randomly initiated grid graph, and assigned three weights to every edges of the graph, weight 1 for the cost optimization, weight 2 and weight 3 for the constrains. If we find the minimum spanning tree with respect to weight 1, the spanning tree’s total cost for the weight 1, 2, 3 is 65118.42, 167915.89, and 161741.22 respectively.

Now, we set the spanning tree’s total cost constrains for weight 2 and weight 3 at 120000, and solve the optimal spanning tree with low cost for weight 1.

We test our algorithm for eight rounds. At every round, our algorithm runs 20 batches. In every batch we generate 100 parameters to search the parameter space. In every batch, the 100 parameters are generated from two sources. In source one, we generate parameters near the found good parameters, we call this exploitation. In source two, we generate parameters just randomly, we call this exploration. At different round, parameters are generated from different exploration to exploitation ratio.

![Figure2 Best found optimal by Exploration to Exploitation ratio](image-url)

Figure2 Best found optimal by Exploration to Exploitation ratio
Figure 2 displays the optimal value obtained by eight computer simulation rounds. We can see that all the eight rounds return relatively optimal value. Compared to the most optimal value obtained, most of the final optimal values returned are within one percent interval. Only one round returns a value worse than the most optimal value by two percent also.

Figure 3 displays the convergence curve of the eight computer simulation rounds. We can see that when parameters are generated from different exploration to exploitation ratio, all of the curves are converging to a sub-optimal value robustly. In other words, the proposed method doesn’t need much work to adjust the super parameters to obtain a good sub-optimal object value.

6. CONCLUSION AND FUTURE WORK
We have proposed a novel algorithm to solve the constrained spanning tree problem. Different from the traditional Lagrangean relaxation method, every solution the proposed method found is in the original problem’s feasible region, although we possibly only found a sub-optimal solution, there is no need to do the hard work to close the gap between the Lagrangean relaxation and the original problem. The proposed algorithm is very easy to parallelize, and can take full advantage of multi-core processors to improve problem solving efficiency. Through computer numerical simulation, we can see that our algorithm is robust to super parameters. In the future we will integrate direction search tools into our algorithms and improve our algorithm’s efficiency further.

Acknowledgments
This work is sponsored by National Key R&D Program of China Project “Research and Development of Personnel Safety Protection Technology and Equipment for Typical Serious Industrial Accident” (2017YFC0804900), And this Project’s Subject 5 “Directional Emergency Alarm Distribution Technology and Equipment” (2017YFC0804905).

REFERENCES
[1] R. Ravi, M. X. Goemans, “The constrained minimum spanning tree problem,” SWAT 1996. Lecture Notes in Computer Science, vol.1097, Springer, Berlin, Heidelberg.
[2] G. L. Nemhauser, L. A. Wolsey, “Integer and Combinatorial optimization,” John Wiley & Sons, New York (1988).
[3] M. Guignard, “Lagrangean relaxation,” Top. vol.11, number 2, pp151-228, 2003.
[4] Pieter-Tjerk, “A Tutorial on the Cross-Entropy Method,” unpublished.
[5] Gang Feng, “Finding Multiple Shortest Paths Subject to Multiple Constraints: A Direction Tuning Algorithm,” unpublished.
[6] Refael Hassin, Asaf Levin. “An efficient polynomial time approximation scheme for the constrained minimum spanning tree problem using matroid intersection,” SIAM J. Comput.vol.33, number 2, pp261--268, 2004.
[7] R. C. Prim, “Shortest connection networks And some generalizations”, Bell System Technical Journal, vol.36, number 6, pp1389–1401, 1957.

[8] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” Proceedings of the American Mathematical Society, vol.7, pp48-50, 1956.

[9] Chazelle, Bernard, “A minimum spanning tree algorithm with inverse-Ackermann type complexity,” Journal of the Association for Computing Machinery, vol.47, number 6, pp1028–1047, 2000.

[10] David Karger, Philip Klein, and Robert Tarjan, “A randomized linear-time algorithm to find minimum spanning trees,” Journal of the ACM. vol.42, number 2, pp321-328, 1995.

[11] V. Aggarwal, Y. Aneja and K. Nair, “Minimal spanning tree subject to a side constraint,” Comput. Operations Res. vol.9, pp287-296, 1982.