Competitive Decision Algorithm for the Rooted Delay-constrained Minimum Spanning Tree

Xiaohua Xiong
College of Computer and Information
Shanghai Second Polytechnic Univ.
Shanghai, China
xhxiong@sspu.cn

Xuemin Chen
Dept. of Engineering Technology
Texas Southern University
Houston, USA
chenxm@tsu.edu

Aibing Ning
School of Management
University of Shanghai for Science and Technology, Shanghai, China
nab@ usst.edu.cn

***Abstract*— In this paper, we investigate a rooted delay-constrained minimum spanning tree (RDCMST) problem. RDCMST seeks to find a minimum spanning tree in which no path from a specified root node to any other nodes may exceed a given delay bound. RDCMST is a NP-hard combinatorial optimization problem arising both in scientific research and practical engineering. Competitive decision algorithm (CDA) is a newly proposed meta-heuristic algorithm for solving complex combinatorial optimization problems. A new CDA algorithm for RDCMST problem is proposed in this paper. Restricted candidate list (RCL) and randomly choosing resource are introduced in CDA for the first time. We reduce the search space based on the mathematical properties of RDCMST. To evaluate the algorithm, numerical computational experiments are performed.**

Keywords-competitive decision algorithm; spanning tree; rooted delay constrained; competitiveness function; Decision function

I. INTRODUCTION

Rooted delay-constrained minimum spanning tree (RDCMST) problem is a combinatorial optimization problem. The task is to find a spanning tree for a given graph in which the edges have cost and delay [1]. No path from a specified root node to any other nodes may exceed a given delay bound, and the total costs shall be a minimum. More formally, we are given a graph $G=(V, E)$ with a set of n nodes V, a set of m edges E, a cost function $c: E \to R^+$, a delay function $d: E \to R^+$, a fixed root node $s \in V$ and a delay bound $B > 0$. An optimal solution to the RDCMST problem is a spanning tree $T = (V, E'), E' \subseteq E$, with minimum cost $c(T) = \sum_{e \in E'} c(e)$, subject to $\sum_{e \in P(s,v)} d(e) \le B$, where $P(s, v)$ denotes the unique path from the specified root node s to a node $v \in V$. The RDCMST problem is NP-hard because a special case called hop-constrained minimum spanning tree problem, where $d(e) = 1$, $\forall e \in E$, is NP-hard [2].

RDCMST problem arises in a number of application scenarios, such as in network design and transportation design. An example would be a shipment organization with a central storage depot providing its customers with goods within a given period of time, i.e. perishable products. Another example is real-time central broadcasting service that is required to transmit its information to all receivers with a certain delay boundary. Real-time traffic is usually bandwidth-intensive and requires quality of service (QoS) guarantees from the underlying network. The delay constraint is an important QoS requirement because most real time applications are delay sensitive.

In the recent papers, the researches about solving methods of RDCMST problem developed rapidly [3-9]. Exact approaches to RDCMST problem have been examined by Gouveia *et.al.* in [3], but it only can solve small graphs with less than 100 nodes. A constructive heuristic was presented in [4] based on Prim's algorithm to find a RDCMST. The performance will affected greatly by the cheap cost edge with large delay. So a more de-centralized constructive heuristic approach by applying the basic concept of Kruskal's algorithm was presented in [5]. And some meta-heuristic approaches based on greedy randomized adaptive search procedure (GRASP) and variable neighborhood were presented to solve RDCMST problem [6-9].

In this paper, we propose a competitive decision algorithm (CDA) to solve the RDCMST problem. CDA is a newly meta-heuristic algorithm for solving complex optimization problems. The principle and main common process of algorithm has been showed in [10]. CDA has already been proved to be an effective algorithm to solve some constrained spanning tree problems, for example, degree-constrained minimum spanning tree [11], multi-object minimum spanning tree [12] and minimum ratio spanning tree [13]. But till now, CDA hasn't been applied to solve the RDCMST problem.

The rest of the paper is organized as follows. In section 2, a general introduction of CDA is presented. In section 3, a competitive decision algorithm for RDCMST will be discussed in detail. Numerical illustrations are demonstrated in section 4 and conclusions follow in the final section.

II. THE COMPETITIVE DECISION ALGORITHM

A. Algorithm Introduction

Nature is the source of inspiration. Applying the characteristics and mechanism from nature into solving the practical problems has turned out to be a successful way.

Many heuristic algorithms have shown to be perfect examples, such as ant-colony algorithm, particle swarm algorithm and genetic algorithm.

Competitive decision algorithm is a newly proposed meta-heuristic algorithm for solving combinatorial optimization problems [10-19]. It analogizes the natural selection process in the real world by recognizing that an entity with more resource will have a high chance to survive. Specially, the algorithm considers multiple competitors and a common resource. Each competitor has a competitiveness function (CF) where a large CF implies a high chance of gaining more resource. There is a decision function (DF) that assigns resources to the competitors. In the algorithm, resources will be assigned and reallocated among the competitors through multiple iterations where a strong competitor may deprive another weak competitor of some resource. Any status of resource assignment represents a solution of the problem, and the equilibrium status without any further resource transfer will be the final solution.

CDA has been shown effective in a broad range of optimization problems, especially those NP-hard problems, for example, large-scale traveling salesman problem [15], 0-1 knapsack problem and its variants [16-18], and vehicle routing problem and its variants [14, 19].

B. Basic Concepts of CDA

Let one or multiple competitor(s) to enter for the competition of resources. According to the decision making principle, some competitor(s) obtain resources and increase their strength, some become weak, or even die because lose their resource. The main factors influencing the result of CDA are:

(1) Initial layout

It is the status of resource assignment among all the competitors before each round of competition.

(2)**Competitiveness function (CF)**

It stands for the competitiveness of the competitor on resource. Generally speaking, a larger value CF means more liable to possess resource.

(3) **Decision function (DF)**

It acts like a referee. The role of it is to assign resource.

(4)**Resource exchange rule**

When competition enters in equilibrium status, resource exchange rule will deprive some resource from one or some competitor(s) and assign the resource to other competitor(s). The process will make the competition entering a new non-stable status.

III. CDA FOR RDCMST

A. Property of RDCMST

Property 1: All the edges $e \in E$ with a delay $d(e)$ higher than the delay bound B can not be part of a feasible solution.

It is easy to prove that any edge with a higher delay than delay bound will not be included in any feasible solution of RDCMST. Otherwise it will violate the delay constraint. Using Property 1 can discard these edges safely to reduce the search space.

Property 2: Edges $e = (i, j) \in E$ which satisfy these conditions:

$$d_{\min}(s,i) + d(e) > B \wedge d_{\min}(s, j) + d(e) > B,$$

would exceed the bound in all possible trees and can not be included in any feasible solution where $d_{\min}(s,v) = \min_{p(s,v)} \sum_{e \in P(s,v)} d(e), \forall v \in V$.

It is easy to prove that edges satisfying these condition can not included in any feasible solution because there isn't any path can arrive at node i or node j with less than or equal to the delay bound. Using Property 2 these edges can be discarded safely to reduce the search space further.

Property 3: For a node $v \in V$, if $d_{\min}(s,v) > B$ then there is no feasible solution for RDCMST.

For a given node v, if there isn't any path can arrive at it with less than or equal to delay constraint, then node v can not be included in any spanning tree. So the entire problem has no feasible solution.

B. Notations

The following notations will be used to formulate the CDA for RDCMST problem.

$parent(s,v)$: parent node of node v upstream to root node s.

$d(s,v)$: sum of all the edges' delay in unique path from the specified root node s to node v.

$subtreedelay(v)$: delay caused by sub-tree rooted at node v.

$d_{\min}(s,v)$: the shortest delay path from the specified root node s to node v.

$arrive(i, j)$: bool value of the adjacent relationship between node i and node j. If edge (i, j) is included then $arrive(i, j)$ =true, otherwise $arrive(i, j)$ = false.

$t\_arrive(i, j)$: the transitive, symmetric closure [1] of *arrive*. $t\_arrive(i, j)$ indicates whether there has a path from node i to node j in current solution.

$flag(i)$: a bool value identifying node i in or not in the tree.

$L(i)$: all the edges incident to node i and can be formulated as:

$$L(i) = \{(i, j) \mid (i, j) \in E \wedge t\_arrive(i, j) = false \wedge flag(j) = true\}$$

$dw(i)$: the edge in $L(i)$ with the minimal sum value of cost and delay.

$dw\_j(i)$: the node incident to edge $dw(i)$.

$power(i)$: the CF value on the edges incident to node i.

RDCMST is a connected sub-graph of original graph with n nodes and n-1 edges and it contains no cycle. All the edges in original graph can be treated as resources. There is only one competitor, namely C, which is the graph with all the nodes and no edges on it. So when the algorithm begins, the virtual competitor N occupies all the resources. Competitor C will try to get n-1 edges from N according to competitive force function and decision function. During the process of adding edge to competitor C, there should be no cycle in C. And the initial layout competitor C hasn't any edge. At the end of one competition, there will be n-1 edges on C without violating the delay bound.

C. Competitiveness Function

In an attempt to estimate how promising an edge is, it is more likely that an edge with comparatively low cost and low delay is part of an optimal solution than an edge with very low cost but high delay. So the CF is defined as:

$$power(i) = \begin{cases} -dw(i) & d(s,i) + d(i,j) <= B,\ j = dw(i) \\ -\infty & otherwise \end{cases}$$

D. Decision Function

There are two kinds of decision function.

In previous works about CDA, CDA always is a deterministic algorithm. Decision function always chooses the node with the biggest *power*(*i*). This is a greedy policy and the result of CDA is deterministic no matter how much times running the algorithm.

In order to extending the search space, a restricted candidate list (RCL) and randomly choosing resources are first introduced in CDA. First, all the nodes are sorted by *power*(*i*). Then construct the RCL from the first k elements whose *power*(*i*) is greater than $-\infty$. k is set no more than 5. Finally randomly choose one node from the RCL. The effect of deterministic DF and random DF will be evaluated.

E. CDA for RDCMST

From the above analysis, the competitive decision algorithm for RDCMST problem may be sketched as:

Algorithm: CDA-RDCMST

Step 1: Pre-process procedure

Construct the shortest delay path of each node.

Discard those edges satisfying conditions in Property 1 and 2.

If the shortest delay path from root to node v is greater than delay bound, no feasible solution can be found and exit the algorithm.

step 2: Competition and decision making

p_count=1; // number of competitiveness function

d_count=2; // number of decision function

la_count=1; //number of initial layout

for *p*=1 to p_count

for *d*=1 to d_count

for *la* =1 to la_count

{ Initialize *arrive* and *t_arrive*;

Compute the *dw*(*i*,k), *dw_j*(*i*,k) of each node *i* ($1 \le i \le n, 1 \le k \le 3$);

Compute the *power*(*i*) according to the *p*th competitiveness function;

Construct the RCL according the *power*(*i*);

Random choose one node *max_power_id* from RCL;

flag(*s*)=true; *partent*(*s*)=nil; *d*(*s*,*s*)=0;

line_count=0; // the number of edges in current solution

Step 2.1: Phase of resource allocation

repeat

*dot*_1=*max_power_id*;

*dot*_2= *dw _j*(*dot*_1, 1);

line_count= *line_count*+1;

arrive(*dot*_1, *dot*_2)= true; *arrive*(*dot*_2, *dot*_1)= true;

for *i*=1 to *n*

if (*t_arrive* (*i*, *dot*_1)=true) or(*i*=*dot*_1) then

for *j*=1 to *n*

if (*t_arrive*(*j*, *dot*_2)=true) or(*j*=*dot*_2) then

if (*i*<>*j*) then

{ *t_arrive*(*i*, *j*)=true; *t_arrive*(*j*, *i*)=true;}

Update the information of node *max_power_id*, including the parent, delay value from root node to *max_power_id*.

Re-compute the competitiveness function for according the *p*th competitiveness force function;

Get the *max_power_id* according to decision function.

until (*line_count*=*n*-1) or no more node can be found;

Compute the sub-tree delay of each node.

Step 2.2: modify the solution to meet delay bound

If *line_count* < *n*-1 then call relaxDelay procedure.

Step 2.3: Phase of resource exchange

Call SwapEdge procedure.

Step 3. Output the optimal solution of RDCMST**.**

When the phase of resource allocation finish, if not all nodes are included in the spanning tree it is because these nodes can not be added without violating the delay constraint. So the algorithm will resort to relaxDelay procedure to relax delay.

RelaxDelay procedure can be outlined as follows:

Procedure RelaxDelay:

For a node v not in solution add the shortest delay path of it to the tree. While adding each edge must make sure no cycle in the tree. For an edge also in the tree, it will be no problem. For an edge (*i*, *j*) not in the tree while *j* is already

in solution, deleting edge (*parent*(*j*),*j*) from tree then there will be no cycle.

Repeat the procedure until all the nodes are in the tree.

Phase of resource exchange is used to improve the feasible solution. The main idea of SwapEdge is changing the tree-edge with non-tree edge without violating the delay constraint to reduce the cost.

SwapEdge procedure can be outlined as follows:

Procedure SwapEdge:

For each edge (*p*, *q*) in tree where *p* is the parent of *q*

Find a non-tree edge (*w*, *q*) where $d(s,w) + subtreedelay(q) + d(w,q) <= B \wedge w \notin subtree(q)$

and *subtree*(*q*) is a sub-tree rooted at node q.

If $c(p,q) > c(w,q)$ then this is a feasible change. Among all the feasible changes choose the one with the biggest reduction on cost.

Repeat the change until no more reduce on cost can get from it.

IV. COMPUTATIONAL RESULTS

To evaluate the performance of the proposed CDA for RDCMST, we implemented it in Delphi 7 and compared it with constructive heuristic algorithm proposed in [5] and meta-heuristic algorithm proposed in [8]. The instance sets R100, R200, R300 and R1000 were introduced and contain 30 complete instances with 100, 200, 500 and 1000 nodes respectively, and random integer edge costs and delays uniformly distributed between 1 and 99. All the instances of the benchmark have been tested by CDA. For reasons of space and clarity we will only give 5 instances with 100 and 500 respectively and compare the results with other approaches.

TABLE I. COMPARISON OF KBH, GVNS AND CDA ON RANDOM INSTANCE SETS WITH 100 NODES (B: DELAY BOUND)

	B	KBH[5]	GVNS[8]	CDA
1	20	2348	2171	2103
	25	1795	1681	1607
	30	1460	1355	1348
2	20	2761	2753	2016
	25	2032	1773	1746
	30	1430	1396	1358
3	20	2403	2053	2045
	25	1543	1539	1538
	30	1310	1261	1260
4	20	2651	2078	2066
	25	1952	1689	1668
	30	1701	1529	1498
5	20	2416	2053	1859
	25	1840	1538	1516
	30	1421	1291	1258

Due to the random choose node in RCL, the CDA for RDCMST is non-deterministic. So 30 runs are performed for every instance and best results are used for comparison with KBH presented in [5] and GVNS presented in [8]. The results as shown in Table I CDA is superior to the KBH and GVNS approaches.

RCL do not solely choose the node with the greatest value of *power*(*i*), it randomly chooses from several most promising node. RCL increases the search space and prevents the solution from immersing to local optimum. DETE is the result of CDA using deterministic DF. R_best and R_avg are the best and average result of CDA using the random DF. From the results shown in Table II, we know both the best value and average value of the random DF are better than the greedy deterministic approach.

TABLE II. COMPARISON OF DETE, R_BEST AND R_AVG RESULTS OF CDA ON RANDOM INSTANCE SETS WITH 500 NODES (B: DELAY BOUND)

	B	DETE	R_best	R_avg
1	20	4173	3590	3840.6
	50	1361	1181	1194.5
	100	766	738	746.67
2	20	3438	3153	3185.4
	50	1364	1167	1211.23
	100	825	725	736.67
3	20	3644	3607	3618.9
	50	1281	1173	1181.23
	100	757	736	747.07
4	20	2886	2826	2886
	50	1315	1091	1110.03
	100	774	704	709.5
5	20	3623	2676	2831.3
	50	1195	1068	1119.5
	100	761	695	701.7

V. CONCLUSIONS

In this paper, we studied the rooted delay-constrained minimum spanning tree problem. RDCMST problem is NP-hard problem and can be applied into many practical engineering applications. We developed a new meta-heuristic algorithm based on CDA to solve the problem. In designing the CF the edge with comparatively low costs and low delays were considered more promising than edges with low cost but high delay. Both deterministic and random strategies were used to test the performance of algorithm for designing the decision function. Deterministic one could get fairly good result in short time while random one could get better result when the running times was 30. Computational results also indicated that CDA is better than other heuristic algorithms.

ACKNOWLEDGMENT

The research is supported by Shanghai Leading Academic Discipline Project (No. XTKX2012).

REFERENCES

[1] R. Ahuja, T. Magnanti, and J. Orlin, Network Flows, Prentice Hall, Englewood Cliffs NJ, 1993.

[2] G. Dahl, L. Gouveia,etc. On Formulations and Methods for the Hop Constrained Minimum Spanning Tree Problem. In: handbook of optimization in telecommunications. Springer science,2006,pp.493-515.

[3] L. Gouveia, A. Paias, D. Sharma,: Modeling and Solving the Rooted Distance-Constrained Minimum Spanning Tree Problem. Computers and Operations Research vol 35 ,2008, pp.600–613.

[4] H. F. Salama,D.S.Reeves,Y. Viniotis,An Efficient Delay-Constrained Minimum Spanning Tree Heuristic. In: Proceedings of the 5th International Conference on Computer Communications and Networks, 1996.

[5] M. Ruthmair, G. R. Raidl. A Kruskal-based Heuristic for the Rooted Delay-constrained Minimum Spanning Tree Problem. Computer Aided Systems Theory - EUROCAST 2009 Lecture Notes in Computer Science Volume 5717, 2009,pp.713-720.

[6] N. ghaboosi, A.T. Haghighat. A Path Relinking Approach for Delay-constrained Least-cost Multicast Routing Problem. 19th IEEE International conference on tools with artifical intelligence,2007,pp.383-390.

[7] N. Skorin-kapov, M. Kos. A GRASP heuristic for the delay-constrained multicast routing problem. Telecommunication System , vol. 32,2006,pp.55-69.

[8] M. Ruthmair, G.R. Raidl,:Variable Neighborhood Search and Ant Colony Optimization for the Rooted Delay-Constrained Minimum Spanning Tree Problem.Schaefer, R., et al. (eds.) PPSN XI, Part II. LNCS, Springer ,vol. 6239, 2010, pp. 391–400.

[9] M. Berlakovich, M. Ruthmair, G.R. Raidl. A Multilevel Heuristic for the Rooted Delay-constrained Minimum Spanning Tree Problem. Computer Aided Systems Theory – EUROCAST 2011 Lecture Notes in Computer Science,vol. 6927, 2012,pp. 256-263.

[10] A. B. Ning,B. Wang,X. H. Xiong,L. Ma, Principles and Applications of Competitive Decision Algorithm, Journal of University of Shanghai for Science and Technology(in Chinese),vol. 30, 2008, pp. 369-373.

[11] A. B. Ning, L. Ma, Competitive Decision Algorithm for the degree-constained minimum spanning tree, Journal of System Engineering(in Chinese), vol. 20, 2005 , pp. 630-634.

[12] X. H. Xiong, L. Ma, A. B. Ning, Competitive Decision Algorithm for Multiple-objective Minimum Spanning Tree, System Engineering(in Chinese), vol.28, 2010, pp. 89-93.

[13] X. H. Xiong, A. B. Ning, Competitive decision algorithm for minimum ratio spanning tree,Computer Engineering and Application(in Chinese), vol.48, 2012, pp. 47-51.

[14] A. B. Ning, L. Ma, Competitive Decision Algorithm and its application to vehicle routing problem, Journal of Management Sciences in China(in Chinese), vol. 8, 2005 , pp. 10-18.

[15] A. B. Ning, L. Ma, Competitive Decision Algorithm for large-scale TSP, Computer Engineering(in Chinese), vol. 31, 2005, pp. 23-26.

[16] X. H. Xiong, A. B. Ning, L. Ma, Competitive Decision Algorithm for Multidimensional Knapsack Problem based on Multi-exchange Neighborhood Search, System engineering theory and practice(in Chinese), vol.30, 2010, pp. 1448-1456.

[17] X. H. Xiong, L. Ma, A. B. Ning, Competitive Decision Algorithm for Multiple-choice Knapsack Problem based on Reduction, International Conference on Computer Modeling and Simulation, IEEE, Sanya, China, 2010, pp. 344-348.

[18] X. H. Xiong, A. B. Wang, A. B. Ning, Competitive Decision Algorithm for 0-1 Multiple Knapsack Problem, International Workshop on Education Technology and Computer Science, IEEE, Wuhan, China, 2010, pp. 253-255.

[19] K. F. Wang, C. M. Ye, The Competitive Decision Algorithm for the Vehicle Routing Problem with Simultaneous Delivery and Pickup, Journal of Computational Information System, vol. 8, 2013, pp.3189-3198.