Turnitin Originality Report

An Improved Approximation Algorithm for the Steiner Tree by Anil Kumar

From Project Report (MTP Thesis Check 2015)

Processed on 05-May-2015 21:31 IST
ID: 534211101
Word Count: 12510

Similarity Index	Similarity by Source	
16%	Internet Sources:	9%
	Publications:	13%
	Student Papers:	4%

sources:

1 2% match (student papers from 23-Apr-2014)
Submitted to Indian Institute of Technology, Madras on 2014-04-23

2 < 1% match (Internet from 05-Apr-2014)
http://www.akami.mx/wp-content/uploads/Posgrado/Algoritmos/Cormen%20Introduction%20to%20Algorithms.pdf

3 < 1% match (publications)
Byrka, Jaroslaw, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. "An improved LP-based approximation for steiner tree", Proceedings of the 42nd ACM symposium on Theory of computing - STOC 10 STOC 10, 2010.

4 < 1% match (publications)
S.J. Phillips. "Finding the hidden path: time bounds for all-pairs shortest paths", [1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science, 1991

5 < 1% match (student papers from 10-May-2014)
Submitted to Indian Institute of Technology, Madras on 2014-05-10

6 < 1% match (Internet from 20-Nov-2006)
http://hss.ulb.uni-bonn.de/diss_online/math_nat_fak/2004/hauptmann_mathias/0380.pdf

7 < 1% match (Internet from 22-Feb-2014)
http://zi.zavantag.com/tw_files2/urls_10/1066/d-1065768/7z-docs/1.pdf

8 < 1% match (publications)
N. Pissinou. "A parallel algorithm for the Steiner tree problem", Proceedings of ICCI 93 5th International Conference on Computing and Information ICCI-93, 1993

9 < 1% match (student papers from 07-Nov-2014)
Submitted to Higher Education Commission Pakistan on 2014-11-07

10 < 1% match (Internet from 08-Jul-2008)
http://elib.zib.de/steinlib/showset.php?D

11 < 1% match (publications)
Dirk Müller. "Faster min–max resource sharing in theory and practice", Mathematical Programming Computation, 03/01/2011

12 < 1% match (publications)
Du, . "k -Steiner Ratios and Better Approximation Algorithms", Steiner Tree Problems in Computer Communication Networks, 2008.

13 < 1% match (publications)
Bouillet. "Path Routing– Part 1: Complexity", Path Routing in Mesh Optical Networks, 10/10/2007

14 < 1% match (Internet from 19-Feb-2014)
http://www.idsia.ch/~grandoni/Pubblicazioni/GGLMSS08focs.pdf

15 < 1% match (publications)
Karger, David R., Daphne Koller, and Steven J. Phillips. "Finding the Hidden Path: Time Bounds for All-Pairs Shortest Paths", SIAM Journal on Computing, 1993.

16 < 1% match (publications)
Pradhan, Anu, and G. (Kumar) Mahinthakumar. "Finding All-Pairs Shortest Path for a Large-Scale Transportation Network Using Parallel Floyd-Warshall and Parallel Dijkstra Algorithms", Journal of Computing in Civil Engineering, 2013.

17 < 1% match (Internet from 07-Jun-2014)
http://www.sommer.jp/csom_phdthesis.pdf

18 < 1% match (publications)
Bruce Weide. "A Survey of Analysis Techniques for Discrete Algorithms", ACM Computing Surveys, 12/1/1977

19 < 1% match (student papers from 13-May-2014)
Submitted to Indian Institute of Technology, Madras on 2014-05-13

20 < 1% match (student papers from 30-Mar-2015)
Submitted to Universiti Teknologi Malaysia on 2015-03-30

21 < 1% match (Internet from 05-Feb-2015)
http://etd.lib.nsysu.edu.tw/ETD-db/ETD-search/getfile?URN=etd-0710100-164312&filename=paper_8734604.pdf

22 < 1% match (publications)
Goldman, . "Abstract Weighted Graph and Weighted Graph Algorithms", Chapman & Hall/CRC Applied Algorithms and Data Structures series, 2007.

23 < 1% match (publications)
Anh Dung Nguyen. "Pervasive Intelligent Routing in Content Centric Delay Tolerant Networks", 2011 IEEE Ninth International Conference on Dependable Autonomic and Secure Computing, 12/2011

24 < 1% match (publications)
LawrenceT. Kou. "On efficient implementation of an approximation algorithm for the Steiner tree problem", Acta Informatica, 03/1990

25 < 1% match (publications)
Tancredi, Urbano; Accardo, Domenico; Fasano, Giancarmine; Renga, Alfredo; Rufino, Giancarlo and Maresca, Giuseppe. "An Algorithm for Managing Aircraft Movement on an Airport Surface", Algorithms, 2013.

26 < 1% match (publications)
Hakim Badis. "A QoS-aware multicast routing protocol for multimedia applications in mobile ad hoc networks", Proceedings of the 11th international symposium on Modeling analysis and simulation of wireless and mobile systems - MSWiM 08 MSWiM 08, 2008

27 < 1% match (student papers from 10-Jun-2009)
Submitted to Lebanese American University on 2009-06-10

28 < 1% match (Internet from 09-Jan-2014)
http://etd.lsu.edu/docs/available/etd-11162005-111427/unrestricted/Yi_dis.pdf

29 < 1% match (publications)
David P. Williamson. "The primal-dual method for approximation algorithms", Mathematical Programming, 02/01/2002

30 < 1% match (publications)
Goyal, Uphar, Suneeta Agarwal, Vaibhav Singhal, Swami Nath Singh, and Rajesh Prasad. "Reoptimization for Steiner tree problem Adding and deleting an edge", 2012 1st International Conference on Recent Advances in Information Technology (RAIT), 2012.

31 < 1% match (publications)
Noor Hasnah Moin. "Genetic Algorithm and Variable Neighborhood Search for Point to Multipoint Routing Problem", Studies in Computational Intelligence, 2008

32 < 1% match (publications)
Hauptmann, Mathias. "On the approximability of Dense Steiner Problems", Journal of

Discrete Algorithms, 2013.

33 < 1% match (publications)
Didi Biha, M.. "Steiner trees and polyhedra", Discrete Applied Mathematics, 20010915

34 < 1% match (Internet from 26-Sep-2014)
http://www.designofapproxalgs.com/book.pdf

35 < 1% match (Internet from 03-Sep-2008)
http://www-lsviii.mathematik.uni-dortmund.de/lsv/lehre/ss2005/opt/LectureNotes.pdf

36 < 1% match (Internet from 28-May-2014)
http://isa.unomaha.edu/wp-content/uploads/2012/08/Advanced-Data-structures.pdf

37 < 1% match (publications)
Schulze, Anna. "Approximation Algorithms for Network Design Problems", Kölner UniversitätsPublikationsServer, 2011.

38 < 1% match (publications)
Baochun Li. "The Cost Advantage of Network Coding in Uniform Combinatorial Networks", 2008 5th IEEE Annual Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks Workshops, 06/2008

39 < 1% match (student papers from 04-May-2015)
Submitted to Indian Institute of Technology, Madras on 2015-05-04

40 < 1% match (publications)
L. R. Foulds. "STEINER PROBLEMS IN GRAPHS: ALGORITHMS AND APPLICATIONS", Engineering Optimization, 1983

41 < 1% match (publications)
Goel, A.. "An online throughput-competitive algorithm for multicast routing and admission control", Journal of Algorithms, 200504

42 < 1% match (publications)
Seufert, Stephan, Srikanta Bedathur, Julian Mestre, and Gerhard Weikum. "Bonsai: Growing Interesting Small Trees", 2010 IEEE International Conference on Data Mining, 2010.

43 < 1% match (publications)
Jean-François Macq. "Trade-offs on the location of the core node in a network", Networks, 10/2004

44 < 1% match (publications)
Aaron Archer. "A Faster, Better Approximation Algorithm for the Minimum Latency Problem", SIAM Journal on Computing, 2008

45 < 1% match (Internet from 02-Apr-2010)
http://pluto.huji.ac.il/~levinas/restricted_diameter.pdf

46 < 1% match (Internet from 11-Jul-2010)
http://www.cas.mcmaster.ca/sqrl/papers/sqrl23.pdf

47 < 1% match (Internet from 19-Aug-2014)
http://graphsjl-docs.readthedocs.org/en/latest/algorithms.html

48 < 1% match (publications)
S. Misra. "An Efficient Dynamic Algorithm for Maintaining All-Pairs Shortest Paths in Stochastic Networks", IEEE Transactions on Computers, 6/2006

49 < 1% match (publications)
Anna Hać. "A Multicast Routing Algorithm Reducing Congestion", International Journal of Network Management, 05/1997

50 < 1% match (student papers from 18-Aug-2014)
Submitted to Indian Institute of Technology, Madras on 2014-08-18

51 < 1% match (Internet from 17-Feb-2014)
http://web.mit.edu/~cjoseph/Public/Cormen,%20Leiserson,%20Rivest,%20Stein%20-%20Introduction%20to%20Algorithms%20-%203rd%20Edition.pdf

52 < 1% match (Internet from 08-Jul-2008)
http://elib.zib.de/steinlib/showset.php?B

53 < 1% match (publications)
Robert Endre Tarjan. "Complexity of Combinatorial Algorithms", SIAM Review, 1978

54 < 1% match (publications)
Calinescu, G.. "Approximate Min-Power Strong Connectivity", SIAM Journal on Discrete Mathematics, 2013.

55 < 1% match (publications)
Uri Zwick. "Exact and Approximate Distances in Graphs — A Survey", Lecture Notes in Computer Science, 2001

56 < 1% match (publications)
D. Banerji. "Shrubbery: a new algorithm for quickly growing high-quality Steiner trees", 17th International Conference on VLSI Design Proceedings ICVD-04, 2004

57 < 1% match (publications)
Rahul, Saladi. "Improved Bounds for Orthogonal Point Enclosure Query and Point Location in Orthogonal Subdivisions in ℝ3", Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 2014.

58 < 1% match (student papers from 11-Dec-2008)
Submitted to CSU, San Francisco State University on 2008-12-11

59 < 1% match (Internet from 29-Jun-2010)
http://home.earthlink.net/~morgenstern/proof/p12x12.htm

60 < 1% match (Internet from 08-Aug-2014)
http://www.laernu.com/unrealfiles/Maps/DMIntelInside.unr

61 < 1% match (publications)
Fatih Köksal. "Multicasting for all-optical multifiber networks", Journal of Optical Networking, 2007

62 < 1% match (publications)
Rong Qu. "Particle swarm optimization for the Steiner tree in graph and delay-constrained multicast routing problems", Journal of Heuristics, 03/02/2012

63 < 1% match (Internet from 07-Jul-2014)
http://www.coursehero.com/file/5684034/all-notes/

64 < 1% match (Internet from 08-Apr-2010)
http://granmapa.cs.uiuc.edu/~jeffe/teaching/373/notes/allnotes.pdf

65 < 1% match (Internet from 03-Sep-2011)
http://granmapa.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/20-apsp.pdf

66 < 1% match (Internet from 07-Feb-2014)
http://www.coursehero.com/file/132316/Introduction-To-Algorithms-2Nd-Edition-SolutionsInstructorsManual/

67 < 1% match (Internet from 06-Sep-2006)
http://ftp.dis.uniroma1.it/pub/demetres/papers/JCSS.pdf

68 < 1% match (publications)
van Bevern, René, Rolf Niedermeier, Manuel Sorge, and Mathias Weller. "Chapter 2: The Complexity of Arc Routing Problems", Arc Routing, 2013.

69 < 1% match (publications)
Khandekar, R.. "Approximating fault-tolerant group-Steiner problems", Theoretical Computer Science, 20120127

70 < 1% match (publications)
Saamaja Vupputuri. "On providing event reliability and maximizing network lifetime using mobile Data-Collectors in Wireless Sensor Networks", 2009 International Conference on High Performance Computing (HiPC), 12/2009

71 < 1% match (publications)
Go Tanaka. "Minimum spanning tree-based random-valued impulse noise detection for a switching median filter", Optics Letters, 09/01/2008

72 < 1% match (Internet from 15-Nov-2006)
http://compgeom.cs.uiuc.edu/~jeffe/teaching/373/notes/allnotes.pdf

73 < 1% match (Internet from 24-Dec-2009)
http://www.mfo.de/programme/schedule/2004/24/OWR_2004_28_web.pdf

74 < 1% match (Internet from 25-Feb-2014)
http://kharkov.10ki.biz/catalog/p11763-0785-trojnik-latun-12-n-12v-12n-v-upakovke-10

75 < 1% match (Internet from 07-May-2014)
http://www.absoluteastronomy.com/topics/Edsger_Dijkstra

76 < 1% match (Internet from 02-Nov-2009)
http://www.sm.luth.se/~lenka/GraphAlg/articles/research/Finding%20the%20k%20Shortest.pdf

77 < 1% match (Internet from 11-Sep-2014)
http://bf-badcompany2.info/?cat=99&paged=5

78 < 1% match (publications)
Pandurangan, Gopal, and Maleq Khan. "Theory of Communication Networks", Chapman & Hall/CRC Applied Algorithms and Data Structures series, 2009.

79 < 1% match (publications)
Liam Roditty. "Replacement Paths and k Simple Shortest Paths in Unweighted Directed Graphs", Lecture Notes in Computer Science, 2005

80 < 1% match (publications)
Uri Zwick. "All pairs lightest shortest paths", Proceedings of the thirty-first annual ACM symposium on Theory of computing - STOC 99 STOC 99, 1999

81 < 1% match (publications)
Blum, A.. "A Constant-Factor Approximation Algorithm for thek-MST Problem", Journal of Computer and System Sciences, 199902

82 < 1% match (publications)
Sriram, . "Background Terminology And Notation", Signal Processing and Communications, 2009.

83 < 1% match (publications)
Goldman, . "Weighted Graph ADT", Chapman & Hall/CRC Applied Algorithms and Data Structures series, 2007.

84 < 1% match (publications)
Sashka Davis. "Models of Greedy Algorithms for Graph Problems", Algorithmica, 07/2009

85 < 1% match (publications)
Khuller, S.. "On strongly connected digraphs with bounded cycle length", Discrete Applied Mathematics, 19960827

86 < 1% match (publications)
Uri Zwick. "A Slightly Improved Sub-Cubic Algorithm for the All Pairs Shortest Paths Problem

with Real Edge Lengths", Algorithmica, 10/2006

87 < 1% match (publications)
Chen, Wen-Tsuen, Po-Yu Chen, Yu-Chee Tseng, and Chi-Fu Huang. "Models and Algorithms for Coverage Problems in Wireless Sensor Networks", Handbook on Theoretical and Algorithmic Aspects of Sensor Ad Hoc Wireless and Peer-to-Peer Networks, 2005.

88 < 1% match (publications)
Zimao Li and Wenying Xiao. "Nearly Optimal Solution for Restricted Euclidean Bottleneck Steiner Tree Problem", Journal of Networks, 2014.

89 < 1% match (publications)
Crina Grosan. "Problem Solving by Search", Intelligent Systems Reference Library, 2011

90 < 1% match (Internet from 27-Jan-2015)
http://www.docstoc.com/docs/75040523/Introduction_to_Algorithms__3rd_Edition__%C3%8B%C3%A3%C2%B7%C2%A8%C2%B5%C2%E

91 < 1% match (Internet from 10-Mar-2014)
http://www.docstoc.com/docs/11640827/Algorithms-Course-Materials

92 < 1% match (Internet from 22-May-2014)
http://www.lix.polytechnique.fr/ctw09/ctw09-proceedings.pdf

93 < 1% match (Internet from 27-Apr-2015)
http://ceur-ws.org/Vol-1231/proceedings.pdf

94 < 1% match (Internet from 09-Jan-2010)
http://student.ulb.ac.be/~sfiorini/papers/concave_soda_update.pdf

95 < 1% match (Internet from 16-May-2014)
http://cs.indstate.edu/arjun/project.pdf

96 < 1% match (Internet from 24-Apr-2013)
http://www.cispa-security.de/en/investigators/pi/rseidel

97 < 1% match (publications)
Sun-Yuan Hsieh. "On the partial terminal Steiner tree problem", The Journal of Supercomputing, 05/04/2007

98 < 1% match (publications)
Surender Baswana. "Faster Algorithms for All-pairs Approximate Shortest Paths in Undirected Graphs", SIAM Journal on Computing, 2010

99 < 1% match (publications)
Shin-ichi Nakano. "A New Approach to Graph Recognition and Applications to Distance-Hereditary Graphs", Journal of Computer Science and Technology, 05/2009

100 < 1% match (publications)
Chakrabarty, Deeparnab, Jochen Könemann, and David Pritchard. "Hypergraphic LP Relaxations for Steiner Trees", SIAM Journal on Discrete Mathematics, 2013.

101 < 1% match (publications)
Neal E. Young. "Faster parametric shortest path and minimum-balance algorithms", Networks, 03/1991

102 < 1% match (publications)
Anupam Gupta. "Approximation via cost sharing", Journal of the ACM, 6/1/2007

103 < 1% match (publications)
Karpagam, T.; Sudhakar, V.J.; Arunsankar, N. and Dhanasekaran, M.. "Flow Based Algorithm", American Journal of Applied Sciences, 2012.

104 < 1% match (publications)
Dutta, S., and T. J. Overbye. "Optimal Wind Farm Collector System Topology Design

Considering Total Trenching Length", IEEE Transactions on Sustainable Energy, 2012.

105 < 1% match (publications)
Uri Zwick. "Approximate distance oracles", Proceedings of the thirty-third annual ACM symposium on Theory of computing - STOC 01 STOC 01, 2001

106 < 1% match (publications)
Ma, Lin, Kunal Agrawal, and Roger D. Chamberlain. "A Memory Access Model for Highly-threaded Many-core Architectures", 2012 IEEE 18th International Conference on Parallel and Distributed Systems, 2012.

107 < 1% match (publications)
James Kadirire. "Minimising packet copies in multicast routing by exploiting geographic spread", ACM SIGCOMM Computer Communication Review, 7/1/1994

108 < 1% match (publications)
Giovanna Melideo. "Sharing the cost of multicast transmissions in wireless networks", Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures - SPAA 04 SPAA 04, 2004

109 < 1% match (publications)
D. T. Lotarev. "Conversion of the Steiner Problem on the Euclidean Plane to the Steiner Problem on Graph", Automation and Remote Control, 10/2005

110 < 1% match (publications)
Sanjiv Kapoor. "Efficiently Constructing the Visibility Graph of a Simple Polygon with Obstacles", SIAM Journal on Computing, 2000

111 < 1% match (publications)
Zahrani, M.S.. "Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing", Computers and Operations Research, 200806

112 < 1% match (publications)
Milind Dawande. "", IEEE Transactions on Communications, 4/2007

113 < 1% match (publications)
Weifa Liang. "Constrained Resource Optimization in Large-Scale Wireless Sensor Networks with Mobile Sinks", Journal of Communications, 2012.

114 < 1% match (publications)
Ying-Meng Li. "Efficient rectilinear Steiner tree construction with rectilinear blockages", 2005 International Conference on Computer Design ICCD-05, 2005

115 < 1% match (publications)
Lokshtanov, Daniel, and Jesper Nederlof. "Saving space by algebraization", Proceedings of the 42nd ACM symposium on Theory of computing - STOC 10 STOC 10, 2010.

116 < 1% match (publications)
I. Averbakh. "A Simple Heuristic for m-Machine Flow-Shop and its Applications in Routing-Scheduling Problems", Operations Research, 01/01/1999

117 < 1% match (publications)
Cohen, E.. "All-Pairs Small-Stretch Paths", Journal of Algorithms, 200102

118 < 1% match (publications)
Y. F. Wu. "A faster approximation algorithm for the Steiner problem in graphs", Acta Informatica, 05/1986

119 < 1% match (publications)
Noga Alon. "Fast Algorithms for Maximum Subset Matching and All-Pairs Shortest Paths in Graphs with a (Not So) Small Vertex Cover", Lecture Notes in Computer Science, 2007

120 < 1% match (publications)
D.J. Guan. "Ensemble motion planning in trees", 30th Annual Symposium on Foundations of Computer Science, 1989

< 1% match (publications)

121 U. Zwick, "All pairs shortest paths in weighted directed graphs-exact and almost exact algorithms", Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat No 98CB36280) SFCS-98, 1998

122 < 1% match (publications)
Williams, Virginia Vassilevska, "Faster replacement paths", Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, 2011.

123 < 1% match (publications)
Wegner, Peter, Deborah Johnson, Keith Miller, Edward Reingold, Roberto Tamassia, Bryan Cantrill, Eric Allender, Michael Loui, and Kenneth Regan, "Combinatorial Optimization", Computer Science Handbook Second Edition CD-ROM, 2004.

124 < 1% match (publications)
Nowozin, Sebastian, "Learning with Structured Data: Applications to Computer Vision", Technische Universität Berlin, 2009.

125 < 1% match (publications)
Sankar Biswas, Siddhartha; Alam, Bashir and Doja, M. N., "GENERALIZATION OF DIJKSTRA'S ALGORITHM FOR EXTRACTION OF SHORTEST PATHS IN DIRECTED MULTIGRAPHS", Journal of Computer Science, 2013.

126 < 1% match (publications)
Monika R. Henzinger, "Randomized fully dynamic graph algorithms with polylogarithmic time per operation", Journal of the ACM, 7/1/1999

127 < 1% match (publications)
Liam Roditty, "A faster and simpler fully dynamic transitive closure", ACM Transactions on Algorithms, 3/1/2008

paper text:

An Improved Approximation Algorithm for

1**Steiner Tree A Project Report submitted by** ANIL KUMAR **in partial fulfilment of the requirements for the award of the degree of MASTER OF TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.** DECEMBER **2014** THESIS **CERTIFICATE This is to certify that the** thesis titled An **Improved** Approximation Algorithm **for Steiner Tree, submitted by** Anil Kumar, **to the Indian Institute of Technology, Madras, for the award of the degree of Master of Technology , is a** bonafide **record of the research work** done **by him under my supervision. The contents of this** thesis, **in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.** Dr. **N. S.** Narayanswamy Research **Guide Associate Professor Dept. of Computer Science and Engineering IIT-Madras, 600 036 Place: Chennai Date:** December, 2014 **ACKNOWLEDGEMENTS I would like to**

my gratitude to some of my friends, Abil N George, Anurag Jain, Chirag Garg, Govind C.,Madhusudan Patidar and Alok Singh Chauhan. They helped me a lot and encour- aged me through out my project work. they have given their full support for resolving several of my technical and non-technical doubts. Anil Kumar i ABSTRACT KEYWORDS: Graphs, Minimum Spanning Tree, Terminal Steiner Tree, All-pairs Shortest-Path

88**Steiner Tree Problem Steiner Tree Problem** (STP) **Steiner tree problem**

(named after Jacob Steiner)

111**one of the NP- Complete Problems.** If **we** apply restriction **on**

the problem than we can solved some of the problems in polynomial time. This problem is some what similar to the

73minimum spanning tree problem. In mst problem, we are

required construct a

7tree that cover all the vertices of a graph with minimum

cost. while

34in the Steiner tree problem, we are given

some vertices, which we called as the terminals(required vertices) and some other points which are the optional points(Steiner Point), we are required to connect all the terminal vertices with minimum cost and in this connection, we may use some optional points(Steiner Point) to cover all the terminal vertices with minimum cost.

14Given a graph G = (V, E) with edge weights as c:E→ R≥0 i.e, and subset of

nodes S⊆V , S is call required vertices (terminals), formed a tree Ts that cover every required vertices(S) with optimal cost c(Ts) := e∈Ts c(e). This contructed tree Ts, we called as the steiner tree, it might contain some other v∑ertices which are optional(Steiner nodes).

97Terminal steiner tree is a tree that having all required vertices

as leaves i.e, all the vertices which having degree one are the required vertices. The

6cost of tree is defined as the sum of all the edges cover the

path in the contruction of tree, this cost minimum then other tree then, we called this tree as the minimum cost and there is no other tree, which have cost less then that, even for unit weighted graph this problem is NP-Hard.

19ii TABLE OF CONTENTS ACKNOWLEDGEMENTS ABSTRACT LIST OF FIGURES ABBREVIATIONS NOTATIONS 1 INTRODUCTION 1.1 Overview and Motivation . . . . . 1.1.1 Steiner Tree Problem . . . . . 1.1.2 Terminal Steiner Tree 1.2

Applications of Steiner Tree . . . . . 1.3 Existing Approximation Algorithms . . . . . 1.4 All-Pairs Shortest-Paths Algorithm . . . . . 1.5 Preliminaries . . . . . .

501.6 Contribution of the Thesis . . . . . 1.7 Organization of the Thesis 2

ALL ABOUT STEINER TREE 2.1 Steiner Tree . . . . . 2.1.1 2.1.2 2.1.3 Types of Steiner Trees problems . . . . . .

3Approximation Algorithm for Steiner Tree . . . . . Analysis of Approximation Ratio 3

A SURVEY FOR BEST APPROXIMATION ALGORITHMS i ii vi vii viii 1 1 1 4 5 6 7 8 9 10 11 11 11 16 20 23 iii 3.1 3.2 For Approximation Algorithm algo1 . . . . . All-Pairs Shortest-Paths . . . . . 3.2.1 Dijktra's Algorithm . . . . . 3.2.2 Description of Dijktra's Algorithm . . . . . 3.2.3 Run Time of Dijktra's Algorithm . . . . . 3.3 Conclusion . . . . . 4 PROPOSED HEURISTIC ALGORITHM 4.1 Overview ..... 4.1.1 Modification in The All-pairs Shortest-Path Algorithm . . . . . 4.1.2 Approximation Algorithm for Steiner Tree . . . . . 4.2 Input Dataset Description . . . . . 4.3 Computational Results . . . . . 4.4 Conclusion . . . . . .

. 5

21**CONCLUSION AND FUTURE WORK 5.1 Conclusion and Future Work**

. . . . . 23 24 27 27 29 30 31 31 32 34 36 39 41 42 42

1**LIST OF TABLES 1.1 Existing approximation** ratio **for** the **steiner tree**

. . . . . 4.1 P4E instance . . . . . 4.2 B, C, D are the instances . . . . . 4.3 Detail results of computation on test instance . . . . . 4.4 Detail results of computation on test instance . . . . . 7 38 39 40 41 v LIST OF FIGURES 1.1 A connected Graph G, with edges weights, having steiner vertices(hallow) and termi- nal vertices(dark) . . . . . 1.2 A Graph G, with edges weights and steiner vertices as hollow vertices, terminal ver- tices(dark) and steiner tree(dark edges) . . . . . . 1.3 Terminal Steiner Tree in which all the degree vertices are terminals, while in case of steiner tree some internal node can also be terminal nodes. . . . . 2.1 (a) three points are given in the euclidean plane. (b) these point form trianle suppose edges lenght is 1 for each. (c) minimum spanning tree of cost 2. (d) steiner tree form using one steiner point . . . . . 2.2 A triangle ABC, and Torricelli Point T (Black dot at the center of the triangle ABC) . . . . . 2.3 (a) Grig having four terminal point int rectilenear plane (b) minimum spanning tree of cost 6 using all four points (c) minimum steiner tree of cost 4 . . . . 2.4 Connected weighted Graph G, with steiner vertices(hollow), and 4 terminal vertices(dark) . . . . . 2.5 Finally Steiner Tree in which all the terminals covered by the tree with mini- mum weight. . . . . 2.6 2.7 Eulerian graph by doubling of edges of steiner tree. . . . . Hamiltonian Cycle that cover all the Terminals of Cteiner tree . . . . . 4.1 Connected weighted Graph G, with steiner vertices(hollow), and 4 terminal vertices(dark) . . . . . 4.2 All the steps of our algorithm are cover by above graph, final Steiner tree is shown figure 4.2(e). . . . . 4.3 Running Time Comparision for Both Algorithms algo1 and algo2 on P4E dataset . . . . . 4.4 Running Time Comparision for Both Algorithms algo1 and algo2 on B, C, D dataset . . . . . 4 4 5 13 13 15 18 19 22 22 36 37 38 39 vi ABBREVIATIONS STP Steiner Tree Problem BCR Bidirected Cut Relaxation APSP All-pairs Shortest-Path DFS Depth First Search vii NOTATIONS G Ts c(Ts) S A ∈ B dist(u,v) MST Graph Minimum Steiner Tree Minimum cost of tree Require vertices in G All elements of A are present in B Distance

126**between u and v** in G **Minimum Spanning**

Tree viii CHAPTER 1 INTRODUCTION In This thesis, we are presenting the study on two problems, SteinerT ree and All-pair shortest- path problem, and with the help of all-pairs shortest-path, we will reduce the running time of approximation algorithms for Steiner tree. Mainly

29**we will focus on Steiner Tree** problems. **In this** chapter, **we**

present a brief introduction to both the problems. Also, some preliminaries and basic definitions that are used throughout this thesis. And what are the present approximation

32**algorithms for the Steiner tree,** which are fastest **algorithms for** constructing a minimum **Steiner**

tree for a given graph G. 1.1 Overview and Motivation 1.1.1

32**Steiner Tree Problem Steiner Tree Problem** (STP) Problem **of Steiner tree**

is one of the combinatorial optimization problem, which is used in a number of applications, applications in an area like in which some items or points having some common property, we called these types of items or points as the required points in perspective of Steiner tree. So we are required to find the interconnections between these require points with minimum weight or cost of covering these points. While eliminating some unnecessary points, we can use these points, in the formation if these points, helping in construction of the minimum

62**Steiner tree.** STP **problem can be defined as follows: STEINER TREE PROBLEM(**

STP) INPUT

54**Given a** connected **undirected graph G = (V, E) and** weight of edges **c:E→ R≥** 0, **a**

33**subset of nodes S⊆V** , where S is **called** as **the**

required ver- tices(Terminals) given as the input.

31**OUTPUT A Steiner Tree** Ts, **for** Graph **G and S.** Finding **the**

shortest path between two places is an important problem in real life applica- tions and technology. The shortest path problem is also studying in the context of the commu- nication networks. In communication networks, it is important to send data from one hub to other through a shortest available path. In the twentieth century, the development of communi- cation networks forced researchers to design more efficient algorithms for better communica- tion. Here we are presenting some motivational application of Steiner tree problem. Suppose, India want to start some new communication technology between some cities, Let technol- ogy is very useful for country, but installation may be costly, but have to start this technology, suppose Indian goverment is thinking of starting this technology in four major city like Delhi, Mumbai, Kolkata and Chennai, but starting this communication direct to direct from one city to other city it may be costly in some situation like direct linking between cities, there may be a chance of reducing the cost of communication and installation by using some other cities as the intermediary for providing connection between these four cities. This type of problem comes under the categary of

69**Steiner tree problem,** as **in Steiner tree problem we are given**

the required nodes and some non-required nodes, we have to find the con- nection between the required nodes with minimum cost of interconnection, in this connection we can use some of the non-required nodes. Same case in Steiner tree problem, in that we have

84**to find a minimum cost tree** connecting between all **the required vertices,**

these required ver- tices called terminal vertices, these nodes are basically a subset of the nodes of the connected undirected, weighted graph G. Steiner tree

45**problem is NP-Hard. Unless P=NP, there is no polynomial time algorithm**

for finding optimal solution for steiner tree problems, so we cannot use well known currently available algorithms for finding an optimal solution of the problems. If we applied some constraints on these problems, then we may be a chance of solving some

18**of the problems in polynomial time.**

So we have to think for solution in different way, our focus is to find a better algorithm for solving the problems, but

18**there is no polynomial time algorithm for solving** NP **-Hard problems**

in polynomial time, so we have to change our approach for solv- ing these problems, so now we are focusing on the better approximation algorithms with good running time. Till now

43**best known approximation** bound **for Steiner tree** problems is **1.55**

given by Robins and Zelikovsky[2000]. There is no improvement for a long time in this Approximation, So we have to think problem differently to get the best approximation. For this our focus is on the LP-Based approximation like directing-component cut relaxation and Iterative randomized rounding. By LP-Based we are getting an approximation (ln 4 + ∈) factor for some graph. In this report our focus is to get the better

run time for approximation algorithms. We will test the algorithm by running a test instance provided

38**for the Steiner tree. STEINER TREE PROBLEMS** GOAL A MST **that**

cover all the required vertices with minimum cost. INPUT A connected undirected

3**graph G = (V, E) with edge costs c:E→** R≥0, **and a subset of nodes** S ⊆**V**

, S is called as the required nodes.

31**OUTPUT A Steiner Tree** Ts, **for** Graph **G and S.** In figure **1.**

1 dark vertices are representing the required vertices, while hollow vertices rep- resenting the steiner vertices, so we are required to cover every required vertices of the graph G with the optimal cost, cost is the total sum of the edge weight that are participating in the steiner tree formation. In figure 1.2 dark edges represent the Steiner tree that cover every ter- 10 g 2 f 1 e a 1 1 2 1 10 2 h k d 4 i 2 2 1 1 1 l j 1 b 8 1 c 9 Figure 1.1: A connected Graph G, with edges weights, having steiner vertices(hallow) and terminal vertices(dark) 10 f 1 e a 1 1 1 2 10 2 h k d g 2 Figure 1.2: A Graph G, with edges weights and steiner vertices as hollow vertices, terminal ver- tices(dark) and steiner tree(dark edges) minal(require) vertices by using some of the steiner vertices, the cost of covering all the require vertices is minimum, as there is no other tree that cover all the require vertices minimum cost which is less than that, and cover all the require vertices. 1.1.2 Terminal Steiner Tree For a given graph G, Steiner tree whose all terminal vertices are of degree one, i.e., all the required vertices which are covered by the tree to form Steiner tree of minimum cost are leaves. 2 10 10 f 10 10 f

28**2 a** 2 **a 2 2** 2 **2 b c b c**

2 10 2 10 (a) Steiner tree of a graph with four terminals (b) Terminal steiner tree of a Graph G e 10 d e 10 d 2 2 2 Figure 1.3: Terminal Steiner Tree in which all the degree vertices are terminals, while in case of steiner tree some internal node can also be terminal nodes. All the require nodes of the Steiner tree must be leaves, but there is no restriction on the require nodes in the Steiner tree. In Steiner tree terminal node can also be some internal node. For example, in figure 1.3(a) a

109**Steiner tree is** formed **from the** a given **graph G, in**

that steiner tree some of the internal nodes are also terminal node. While in figure 1.3(b) a Steiner tree all the required vertices are leaves only, so this

30**Steiner tree is** called **as** the terminal **Steiner tree.** TERMINAL **STEINER TREE**

INPUT A connected weighted

3**graph G = (V, E) with edge costs c:E→** R≥0, **and a subset of nodes** S ⊆**V**

, S is call required vertices (Terminals) as a input. Output A steiner tree Ts of G, such that all the terminal node must be of degree one. 1.2 Applications of Steiner Tree The Steiner tree problems have so many applications like in circuit layout, network design, and telephone communications. Everywhere where the shortest path required between the interconnecting, some of the require points with some specific properties and we are required to connect these points with the help of some optional points, there we can use Steiner tree. Here are the some applications of steiner tree. In physical VLSI designs Steiner tree is used, whenever the placement of component on a chip, there may be a chance that, a set of pin are to be connected with in the free space. A set pin which sharing the networks in the designs, these pins are called as the required vertices, and some other pins which are not useful in the network designs, but with the help of these pins we may connect the require vertices in designing of the network. with the help of these optional pins, we may connect the required pins and fulfill the purpose of the require pins in the designs of networks. On-line and Dynamic Steiner tree On-line application of the Steiner tree comes, whenever a new change in the network be- cause of some new additional points for service, then there is a requirement of new connection in the telephone network. If a new points for the service is added then, a new line of connect for the new point in the network has to build. In this network custruction is like a way the every time, whenever location is change a new network is formed based on the previous network. this brings us a issue of dynamic Steiner tree, in that whenever new network is built, we have to calculate an

approximation for the performance of the network. 1.3 Existing Approximation Algorithms There are series of research in the improvement the approximation ratio of the Steiner tree problems. It starts from 2-approximation and after so many of the improvement it reached to 1.55, i.e., the best known approximation by Robins and Zelikovsky[2000] [9]. Here is the list of Existing approximation ratio for the steiner tree [9]. Approximation Bound References 2.0-approximation minimum spanning tree heuristic 1.83-approximation Zelikovsky 1993 1.78-approximation

12**Berman and Ramaiyer** 1994 **1.** 73-approximation **Borchers and Du** 1997 **1.**

67-approximation

12**Prömel and Steger** 1997 **1.** 65-approximation **Karpinski and Zelikovsky** 1997 **1.** 59-approximation **Hougardy and** Proömel 1999 **1.**

12**55-** approximation **Robins and Zelikovsky** 2000 **Table** 1 **.1:** Existing approximation ratio **for** the **steiner tree**

A series of improvement, maximize with Robins and Zelikovsky's [6] famous 1+ ln2(3) + ε < 1.55 (here ε

3**>0 is an arbitrary small constant)**

algorithm improved this ratio by iteratively improving upon the minimum weight terminal Steiner tree. The above results

3**are based on the** Steiner tree as **a full** components. A **component**

of the

30**Steiner tree is** the **tree** whose **all the** required **(terminal) vertices** are the leaves **of the**

tree. But considering the full component at a time will not give a good approximation ratio. If we consider the Steiner tree with special structure, So called r-restricted Steiner, in this tree every componets have no more than r- terminals. To get a better approximation, It is better to pay attention to r-restricted Steiner tree. By using this approach we may get a better approximation ratio [6]. Steiner Ratio:-

1**The Ratio between the total length of the minimum** Steiner **tree and the total length of** a **minimum** Spanning **tree,**

Steiner ratio represented as ρ. 1.4 All-Pairs Shortest-Paths Algorithm All-Pair

67**Shortest-path(APSP)** problem **is one of the fundamental problems** in **the**

graph theory. A classical ways for solving APSP in general weighted graphs is given by Floyd and Warshall [1]. It outputs an V × V matrix, in that every pairs of vertices at the minimum distance, it takes Θ(V 3) time. In

4**all-pairs Shortest-paths problem,** we are required **to find** the **shortest path between each** and every pairs **of vertices** of the graph **G**

= (V, E). i.e., we are given a Vi and Vj where (Vi, Vj) ∈ V , this algorithm

48**used to find the shortest paths**

between vertices Vi,Vj.

2**All-Pairs Shortest-Paths problem** can be solved **by** using the **a single-source shortest-** path **algorithm** by running **|V | times**

to find the

83**shortest path for each** and every **vertex of the graph** as **a**

source. With the help of All-Pairs Shortest-paths algorithms, we are coming on a better running time approximation algorithm for steiner tree. Run time of algorithms totally depends on algorithmic and datastructure that we are using in the implementation. 1.5 Preliminaries Weighted Graph:- A graph has two elements, vertices and edges. The edges connecting the two end points or vertices of the graph, have some cost of connecting two vertices, this is called as the weight for that edge. If weights are associated with every edge, this type of graph called as the weighted graph. Weight sum of all the edges is called as the weight of graph. Path in a Graph:- A path is a sequence of edges which connects the vertices of the graph. The Sum of all associated weights with edges so called as the weight of the path. Subgraph:- A subgraph P is the subgraph of the graph G, whose edges and vertices are the subset of the G. Tree:-

64**A tree is a acyclic graph. A tree is a**

graph is which every vertex is connected with exactly one path, one degree vertices are called as the leaves of the tree. Tree became disconnected after removal of any edges. Minimum

2**Spanning Tree:- A spanning tree of** a **graph is**

the tree which connects every vertces of the graph, A minimum spanning tree is the tree which have optimal cost of covering every vertices. i.e., there is no other tree having weight less than a minimum spanning tree. The

81**sum of** all **the** weight **of** all **the edges of tree** is called as **the**

weight

49**of the minimum spanning tree.** Algorithms for finding **the**

47**minimum spanning tree** in **a graph** are **Prim's algorithm, kruskal's algorithm.** Dijkstra's **Algorithm:-** Dijkstra's **algorithm**

one of the shortest path algorithm, it use single vertex as a source for finding the

58**shortest path. Run time complexity of this algorithm is O(**| E| + **|V** |log **|V**

|), if we use fibonacci heap as in implementation,

123**where |E|** is **the number of edges.**

1.6 Contribution of the Thesis In this thesis, we restrict our attention on fastest

102**approximation algorithm for steiner tree.** Not **only on the**

theoretical interest about the

108**approximation algorithm** of **the Steiner tree,** But also **on the** practical impact **of**

the algorithms are present in this thesis. All-Pairs Shortest-Paths algorithms have been shown to find some better solution in practice for approximation algo- rithm, But because of the high run time on a big

graph and different type of problems limits their applicability. Underlying graph structure can play an important role in setting up efficient approximation algorithms. All-pairs shortest-paths algorithms, tries to improve an existing so- lution by modifying the current exiting algorithm,

> 4**for finding the** best **path between** the **pair of** terminals. **In**

this All-pairs shortest paths basically focusing on the find the best minimum path between the terminals, by that it for providing some better solution for our approximation algorithm, if we apply some change in algorithm. By this way we may get better run time for some limited problems, This is also confirming that nontrivial data structures and algorithmic are also very important for the solution of some real-world problems for optimization. 1.7

> 39**Organization of the Thesis This thesis is organized in** 5 **chapters.** Current **chapter is** an **introduction to the**

thesis. It contains an introduction and motivation for the Steiner tree problem. We presented some graph theoretic preliminaries which are used throughout this thesis. In Chapter 2 we study various types of Steiner tree and existing algorithms, and we explained some of the approximation algorithms. Also, we prove the various important properties of Steiner tree. Chapter 3 presents a nice survey of the work done on all-pairs shortest-path algorithms and how, it will use in the steiner tree algorithm for improving the running time for heuristic algorithm. In chapter 4 we present an algorithm to construct a shortest distance between all-pairs of terminals using APSP. How we are using this algorithm to solve the approximation algorithm for steiner tree. Finally, Chapter 5 concludes the thesis. This chapter also lists some interesting direction for research and open problems for further study, and some case that can be consider for further improvement in the approximation algorithms. CHAPTER 2 ALL ABOUT

> 27**STEINER TREE In this** chapter, **we will present** all about **the**

> 27**Steiner tree** problems. **In this we will present** what are **the** properties **of** the **steiner tree**

and types of the steiner tree, what are the algorithms for the different type of Steiner tree and their approximation algorithms. 2

> 1**.1 Steiner Tree** The **Steiner Tree** in **a** given **graph G = (V, E) is a tree that** cover all **the** terminal vertices **of**

the graph with minimum cost, terminals are the subset of the vertices which are required for the connection. In this section we define and prove many type of steiner tree problems. 2.1.1 Types of Steiner Trees problems Based on the applications of the Steiner tree is of different types. For example, if a Steiner tree is applied in the VLSI network, then this type of Steiner tree is called a terminal

> 6**Steiner tree. In this section, we will** discuss **a** few types **of**

Steiner tree Problems. 1. Euclidean Steiner Tree The Euclidean Steiner Problem(ESP) was well researched by Zacharis, the origin of ESP is often traced

> 6**back to Fermat(1601-1665) who** proposed **the following problem:** We are **given** some terminal **points**

in the 2-dimensional space and some optional points. Here we considering the cost of connecting the points is same as the distance between the points.

> 6**Find a point in the plane** that sum **the** connection **of**

minimum distance, and point is

> 17**in such a way that the** sum of whose **distances** from three given **points**

should be minimal. This

116**problem will be referred to as** the Fermat **problem.** In figure **2.**

1 three points are shown in the Euclidean plane, if we connect these point with help of a steiner point(shown as hollow in figure) then we will get a tree that covers all these points with minimum distance 3. √ A solution for this problem is given by torricelli in 1640, he gave solution by using some geometric triangle for this problem. He propose that, if we draw three circles using circum- scribing of equilateral triangles sides for construction of the circle, these circle meat at point that point called as called the Torricelli point. In fig 2.2 a triangle, a Torricelli point is formed after the intersection of three circles. The circle are drowned at the edges of the equilateral tri- angle every circle pass through the two points of side of the triangle and the intersection point of these three circles is so called Torricelli point. The distance from the Torricelli point and the give point is called Euclidean distance and the sum of all the distance is minimum. Here in this figure 2.1 shown that the solution of the 3 points problem with the Torricelli method, if we found a Torricelli point, we can draw edges having an equal distance from this point to the other point. In this fig 2.1 point form a triangle, if we form a minimum weight tree from this triangle the total cost of that mst will be 2 which is not the minimum cost of covering all points in the plane, and if we form a steiner minimum tree by taking the steiner point as the center of these point, then we are getting a tree that covers all points with minimum distance this is so called Steiner tree. characteristics of this Torricelli point that it always lies inside the triangle and the distance sum from all other point triangle corner to the torricelli point is minimum. If triangle is equi- lateral triangle than ∠ ATC = ∠ BTC = ∠ CTA = 120. As Shown in figure 2.2. Circle are draw by taking center as the outside intersection point of arch drown by takening the corner of the triangle edge as a center, by taking that intersection point as center and draw a circle

59**1 1 1 (a) (b)** √ **1 1** 3 √ **1** 3 **1**√1 3 **(c) (d)**

Figure 2.1: (a) three points are given in the euclidean plane. (b) these point form trianle sup- pose edges lenght is 1 for each. (c) minimum spanning tree of cost 2. (d) steiner tree form using one steiner point Figure 2.2: A triangle ABC, and Torricelli Point T (Black dot at the center of the triangle ABC) , intersection of all the circle. which passes through the end points of edges, this process is done for all the three circle, the intersection point of these three circle is so called steiner point or torricelli point. 2. Rectilinear Steiner Problem A Steiner tree, whose

96**spanning tree** overs **a set of points in the**

collection by using vertical and horizontal lines which covers the points in the given space is called rectilinear tree. These sets of lines are called Steiner

78**trees. The** Length **of a tree is the** weight **sum of**

its lines, and a rectilinear Steiner minimal tree(RSMT) is a tree of minimum length. According the statement by Hanan's we can find the number of Steiner tree exist in a grid, that having the points. This can be drown by horizontal and vertical line than can be form by connecting the set of points that are presents in the plane [5]. Rectilinear minimum

37**Steiner tree problem is a** one **of the problem in the**

geometric plane

112**for the Steiner tree problem, for** that **we** have **to**

replaced euclidean distance by rectilinear distance that form a grid like structure. Given the points there is a interconnections between the points in fashion of

37**horizontal and vertical lines.** This arises the **application of**

steiner tree vlsi design, because in vlsi design there is a linear relationship between the wirelength and wiring area. In figure 2.3(a) grid structure is given to the four points lies in the rectilinear plane, suppose that the distance from one grid to another grid is 1, from both way horizontal or vertical. In figure 2.3(b) a

minimum spanning tree is formed by using these points, the cost of forming minimum spanning tree will be 6, this cost is not minimal cost of covering all the points in rectilinear plane. In figure 2.3(c) we are using one steiner point(show as hollow) to connect all the required vertices, than the cost of covering all the points will be 4 which is minimum cost of covering all the points. 3. Steiner Tree Problems in Network

> 113**This problem is the** combinatorial version **of the** Euclidean problem for **steiner tree.**

If G is not connected and terminals appear in at least two components, then there is no solution for the 14 Figure 2.3: (a) Grig having four terminal point int rectilenear plane (b) minimum spanning tree of cost 6 using all four points (c) minimum steiner tree of cost 4 Steiner problem. If G is not connected but all terminals appears in one component, then the remaining components can be disregarded and we can get the solution for the by using only the terminals that are present in the component. For this problem in the netwrok by using steiner tree, it is assumed that the G is connected graph. The Problem of steiner tree in networks was originally formulated by Hakimiand independently by levinin 1971. After that this problem received considerable attention for the research in the network.

> 11**STEINER TREE PROBLEM IN** NETWORK INPUT **A undirected graph G** = (V, E), **edge costs c:E→ R≥** 0, **and a** nodes S ⊆ **V**

, S is the subset if vertices, called required nodes. OUTPUT A Steiner Tree Ts, Such that there is a path between every paire of terminals. 4.

> 42**Prize-Collecting Steiner Tree(PCST)** The **Prize collecting steiner tree(PCST)** in a **given undirected graph G,**

with edges cost and vertex profits. In PCST we form a steiner tree by using ternimal vertices such that it minimizing total

> 44**sum of the edge'** s weight **in the** steiner **tree plus the** profit **of** all **the** vertices which are **not** in **the** steiner **tree.**

PRIZE-COLLECTING

> 11**STEINER TREE(** PCST) **INPUT A** undirected **connected graph G** = (V, E) **with edge costs c : E → R≥** 0, **and a** |S| nodes S ⊆ **V**

, |S| is so called terminal nodes and p is vertex profit. OUTPUT A Steiner Tree Ts = (Vs, Es) of G, Es⊆E, that minimize the objective function c(s) = v∈/Vs p(V) + E∈Es c(E) . ∑ ∑ 2.1

> 29.**2 Approximation Algorithm for Steiner Tree** Here **we** are presenting **the** approximation **algorithm**

from a graph G for finding the Steiner tree. A graph G is given with edges weight and a vertex set |S| which, we called as the terminal nodes. Sequence of vertices (v1, v2, v3 . . . vk) is known as the

> 80**path in the graph. The** Weighted **sum of** all **the edges**

which are present in the path is known as the weight of the path. A

> 71**spanning tree is** the **tree** that cover **all the vertices of the** graph, if **the**

weight of a path is minimum than this tree is so called as the minimum spanning tree. If we applied some conditions on this like, we have to cover some particulear nodes, and have to find the minimum distance between these nodes, then these type of cases comes under the category

> 38**of the Steiner tree problems,** in that we **have to**

cover all the terminals, if all the ternimals cover by a tree with minimum weight, i.e., there is no other optimal weighted tree which have less weight than this, this type of tree so called as the minimum Steiner tree.

> 8**Finding a minimum steiner tree** in a graph **G and S is** the **NP-complete**

problem. So have to think about some approximation solution for these problems. An approximation algorithm best known for Steiner tree problems. We start with a heuristic algorithm that can be use for finding the Steiner tree from a given graph [8]. We will show the steiner tree algorithm and their explanation by taking some graph as an example. Heuristic Algorithm :- Heuristic

> 118**Algorithm for** construct **a Steiner tree** from **a graph**

[8].

> 124**Given a connected undirected graph G,**

and edges weight or cost is the minimum distance d between two vertices vi and vj if there is a edge. We are given some require vertices S, S is the subset of vertices S ⊆ V .

> 40**Heuristic algorithm for the steiner tree**

> 92**following. ALGORITHM** 1:- **INPUT A** undirected **graph G = (V,E),**

distance d between the vertices vi and vj, i.e., edge weight d

> 17**and a subset of nodes** S ⊆ **V** , S **is** called **as**

required vertex. OUTPUT A Steiner Tree Ts = (Vs, Es) of G and S. Step 1. First take the given graph G and the required vertices(S), and

> 114**construct a complete** distance **graph G1=(V1, E1).** Step 2. **From**

that graph G1 find a minimum spanning tree, T1. there may be chance that more than one minimun spanning tree are possible, than choose arbitrary one of the mst. Step 3. Construct the subgraph Gs, by using the graph G by replacing each edges by it's shortest path in G, according to the edges present in the mst from step 2.(If more than one shortest paths are their choose all shortest path.) Step 4. Find mst form that subgraph Gs.(If more that one mst are present, then any mst from that one.) Step 5. Now delete all the unnecessary edges which are not connecting the required ver- tices(S), final tree is so called as the Steiner tree. a 3 1 k 1 2 h i 2 10 2 g 2 1 2 1 j b 1 1 f 1 e 5 4 8 2 2 c 9 d Figure 2.4: Connected weighted Graph G, with steiner vertices(hollow), and 4 terminal ver- tices(dark) In this algorithm run time taken by each Step is as, Step 1 run time

> 5**O(|S||V |2), Step 2** can **be done in O(|S|2) time, Step 3** can **be done in** the order of **O(|V |) time, step 4 could be done in O(|V |2) time and**

time taken by the Step 5 will of order O(|V |) time. Here in these steps only first step is taking more runnig time then the other steps. So can say the overall running time complexity for this algorithm will be O(|S||V |2) [8]. With the help of an example, we are explanating each and every steps of this heuristic algorithm. Example: In this figure 2.4, a graph G with edge weights with four terminals are given. After applying this algorithm step by step we are getting the final Steiner tree that shown in figure 2.5(e). Here is the step by step explaination for the algorithm by using the graph for explaination, if we apply this algorithm step by step on this graph, after each step on this algorithm we are coming with a new graph. After applying first step of this algorithm, we are getting a complete graph between all the ternimal nodes with edge connection as shortest path between the terminals as shown in figure 2.5(b), only in this complete graph we are using the terminals nodes. On

> 60**a 4 d** a **d 4 4 4** 4 4 4 **b 4** c **b 4**

c (a) A complete graph of G using terminals nodes, (b) Minimum spanning tree from complete graph with minimum edges weight between terminals a 1 a

46**1 g 1** 2 h **1 2** b **g 1** 2 **h 1 2** b 1 **1** b f **2** e **2**

b f 2 e 2 11

28**1 1 1 c d c d (c)**

A subgraph of graph G (d) Minimum spanning tree from subgraph a

13**1 b 1** b **f 2 e 2** 1 1 **c**

d (e) Terminal steiner tree of a Graph G Figure 2.5: Finally Steiner Tree in which all the terminals covered by the tree with minimum weight. this complete graph if we apply second step of this algorithm we come up with a output as the

36**minimum spanning tree of** that complete **graph, that** cover all **the** vertice **as**

shown in figure 2.5(c), by using this

120**minimum spanning tree,** we **construct a** subgraph of **graph G**

by using only those edges which are participating in

91**the minimum spanning tree of** complete **graph,** in **this** subgraph **there**

is no leaves which are steiner nodes, only terminal nodes can be leaves of this subgraph because, when we form a complete graph in step 1, this is done only by using the terminals nodes of the graph G, so whenever we form a minimum spanning tree from that graph than only leaves have to be terminals, so in the subgraph formation in the step 3 of the algorithm have leaves as a terminals. On this subgraph form after applying step 3, we applied step 4 of algorithm in that some

104**minimum spanning tree algorithm is applied on the**

subgraph, we come up with

22**a minimum spanning tree of that**

subgraph. Finally, after applied step 5 of algorithm, and we come up with shortest cover of all the terminal nodes i.e., Steiner tree of graph G. 2.1.3 Analysis of Approximation Ratio Theorem 1. 2-Approximation Ratio for Algorithm algo1. Lets the Steiner tree's optimal cost is OPT. By appling doubling of it's edges of this tree. then, we are getting a Eulerian graph that connecting all the terminals of steiner tree. After applying DFS(depth first search) on this Eulerian graph, we will come up with an Euler tour of this graph. Cost of this Euler tour will be 2.OP T , the cost of Euler tour caot will be double of the cost of the Eulerian graph because the Eulerian graph is viewed as a directed graph that contain two directed edges for each of the edge present in the tree, as shown in the figure 2.6, of steiner tree in which all dark nodes are represening the

100**terminals and** hollow **(white) nodes are** representing **steiner nodes.** we are getting **a**

Eulerian graph connecting all the required vertices and Steiner veritces by doubling edges for every edge present in the tree, and after applying DFS(depth first search) on that Eulerian graph, we will come up with a Euler tour in which, we have to cover each and every edge twice so the cost will be 2.OP T . We got a hamiltonian cycle by short-cutting the steiner vertices and the previously visited vertices os the terminals. Hamiltonian cycle by short-cutting is shown in the figure 2.7, that cover all the required nodes of the graph. because of the triangle inequality short-cutting

94does not increase the cost of the path, i.

e., the cost of hamiltonian cycle will be same as the cost of the cost of Euler tour. So, we obtain a tree the cover all the terminals by deleting the one of the edge of the hamiltonian cycle, this tree is called as the spanning tree or hamiltonian path. This tree covering all the steiner vertices, so we can say that this is also

43a minimum steiner tree on the terminals. The

cost of hamiltonian cycle Hc will be cost(Hc) ≤ 2.OP T In hamiltonian cycle total number of edges are equal to number of terminals, so total number of edges will be |S|, out of these edges maximum weight of the edge is atleast cos|tS(H|c). We are getting the hamiltonian path Hp by deleting the one of the maximum weight edge from the hamiltonian cycle so the cost of the hamiltonian path will be cost(Hp) ≤ 2.OPT - cos|tS(H|c) cost(Hp) ≤ 2.OPT - 2.O|SP|T cost(Hp) ≤ 2.OPT(1 - |S1|) So cost of steiner tree, that will formed after deleting one of the edge of hamiltonian cycle. cost(Ts) ≤ 2.OPT(1 - |S1|) Figure 2.6: Eulerian graph by doubling of edges of steiner tree. Figure 2.7: Hamiltonian Cycle that cover all the Terminals of Cteiner tree CHAPTER 3 A SURVEY FOR BEST APPROXIMATION ALGORITHMS 3.1 For Approximation Algorithm algo1 Our previous approximation algorithm algo1 is taking

95a running time of O(| S ||V |2) for

com- puting the

8minimum Steiner tree for a given graph G and |S|,

most of the run time is taken by the first step of the algorithm, total running

89time taken by the algorithm is in the contruction of the

complete graph by using all the terminals of the graph G. After that we are constructing the minimun spanning tree from that complete graph, that taking total time of O(|S|2). And step 3 of this algorithm taking a time of O(|V |), so total run

101time taken by the algorithm is by the

step first only. By using these steps[1,2,3], we are coming at a position, there we can say that every terminals of graph G are connecting by using the sortest path between the terminals only, i.e., we are only connecting

107the shortest path from one terminal node to the other terminal node.

so we can say that this problem is some what reduced to the

9all-pairs shortest-path problem, in that we are required to find the shortest path between each pairs of

vertice. So, Now we have to think problem in a different ways, either reduce the running time of step 1,

18so that the running time of the our algorithm

algo1 may reduce or we can find a way that directly

16find the shortest path between the all-pairs of

terminals by using one of the best known all-pairs shortest-path algorithm. So now our focus is change, now we have to think about this so that we can get the shortest path between terminals. A all-pairs

75**shortest-path problem is** graph theory **problem** for **finding** the shortest **path between** pairs of **vertices**

of graph G. One of the most interesting ways to approach this problem is to preprocess a graph to setup data structure which are required for storing the edges weights. So, that it can efficiently answer queries of

110**distance between any pair of vertices,** this data **structure** like heap. We **can**

say one thing that, we are finding the distance between the terminals of the graph, that distance is minimum as we are using the shortest

7**path between every pair of vertices,** we are retrieving **the** minimum distance **of**

edgs only from the data structure. So now we have to come up with a

25**best known** algorithm **for finding the shortest path** between **the** terminals, with **the**

help of this algorithm we can skip the step [1,2] of our algorithm algo1, and come up directly to the step 3 of heuristic algorithm algo1. For this we are using dijktra's algorithm

2**to find the shortest path from one** terminal vertices **to the**

other terminals. 3.2 All-Pairs Shortest-Paths A classical algorithm for solving APSP in general weighted graphs is given by Floyd and Warshall [1]. It outputs an V × V matrix that contains optimal weight of edges for every vertices if edge is present between these two vertices, this problem takes the running by order Θ(V 3) time. Here, we are considering only the problem of weighted graph. For APSP problem [11], we have to come up with a best run time algorithm that reduced

115**the running time of** our **algorithm** algo1, **and** find **the** optimal path between **the**

terminals, we are thinking about this problem, be- cause in the middle of this algorithm algo1, we are at the stage, where, a subgraph constructed using a

55**shortest paths between all-pairs of** the terminals present **in the graph, this** construction **is** done by **the**

steps from 1 to 3 of the algorithm algo1, while other steps only taking a linear or logarithmic time. So we have to think to solve steps [1,2,3] of algorithm algo1 more efficiently to get a optimal solution, and some better running time bound. For that we changed our think- ing about

48**finding the shortest** path **between all pairs of**

terminal vertex, for that we come up with a solution to use the

105**all-pairs shortest-paths** algorithm, **as in** these steps **we are**

costructing the subgraph from a given graph G, this subgraph contains only the minimum distance path between terminals, while using some of the steiner nodes. this shortest path is contructed by using all the terminals as a source,

125**i.e.,** a **shortest path from** one terminal as **a**

source, and

72**find the shortest path from the** terminal **to other** terminals by using **the**

76**single source shortest path** algorithm. we **run the** single source **shortest path algorithm**

for every terminal vertices by taking as a source vertice, and

90**a shortest path between every pair of** terminal **vertices**

is found, as we choosing the optimal edges every time in the cover path from one vertices to the other, so weight of the path is increase whenever a new edge is chosen in the path. as only non-negative edge weights are present in the graph. A edge is optimal in the path from one vertice Vi to other Vj in G, if

24**there is no** other **shortest path between vertices** Vi **and**

Vj [2], which, is less than that path. A classical

122**algorithm for** solving **APSP in** general weighted **graphs is** given **by**

Floyd and Warshall [1]. It outputs an V × V matrix that contains shortest

93**distance between every pair of vertices, in**Θ **(V**

3) time. So we are using (APSP) algorithm for establishing the minimum path connection between all the pair of terminals of given graph G. In this algorithm we are

7**running a single source shortest path algorithm dijkstra's on**

each and every pairs of terminals whenever, other than terminal node is encounter, we just skip that node, we are considering only terminal nodes as a source vertex of our algorithm [1]. for our algorithm we need some of the preliminary, that we are going to be used in our algo- rithm. A path is a sequence of vertices, (v1, v2, v3 . . . , vn), a path from a vertice v1 to vn going through the vertices (v2, v3 . . . , vn−1) [2]. A symbol (u ❀ v) is used for path representation

86**from u to v (if there** exits **a** edge between **(u,v)).**

A path concatenation symbol is denoted like (u ❀ w ❀

99**v) is the** path concatenation **of** two **paths (u**❀ w) **and** (w ❀ **v)**

[2], and the symbol use to concatenation of edges in path is like (u,v ❀ w) this is what we called edge adding

15**(u,v) to the path** (u ❀ **w).** Lenght of **the path**

(v1, v2, v3 . . . , vn) is [11]. |(v1, v2, v3 . . . , vn)| = n − 1 Edge (v,u) cost is denoted as ||(v,u)||. So total weight of the path (v1, v2, v3 . . . , vn) is the weighted's sum for it's constituent edges [2]. ||(v1, v2, v3 . . . , vn)|| = ni=−11 ||(vi, vi+1)|| ∑ Lemma 3.1.

15**An edge is** called **optimal if it is** present in the **optimal path.**

Lemma 3.2. A path (v ❀ u) is called optimal if no other path (v ❀'u) is optimal, in between (v, u) is called optimal path. ||v ❀′ u|| ≤ ||v ❀ u|| Lemma 3.3. As we are covering only the optimal edges in the path, so subpath of the optimal path also optimal. Lemma 3.4.

85**If there is a** optimal **path from (v** ❀ **u)** then there **is**

also optimal path from (u ❀ v) in undirected graph [2]. here we can say that,

> 40**if there** is **a** optimal **path from** vertices **vi to vj,**

then all the edges which are path of the optimal path form a minimum path between the two end points, because it is present in the optimal path. 3.2.1 Dijktra's Algorithm 3.2.2 Description of Dijktra's Algorithm Dijktra's Algorithm is one of the single source shortest path algorithm, that take one vertice (source) as a input and find a path of minimum distance till destination is found, means we can say that it find a minimum weight path

> 25**from a** given **vertex** as **a** source **to all other vertices** of **the graph**

G. We are using this algorithm in our implementation, beacause the good implementation of this algorithm is lower then run time of Bellman-Ford algorithm [1] [3]. DIJKTRA'S ALGORITHM INPUT

> 14**Given a** connected **graph G = (V,E), with** edges weight **c:E→ R≥0, and** one vertex as **a**

source is given as v ∈ V . OUTPUT The minimum lenght

> 22**path from a source vertex to all other** remaining **vertices.**

DATA STRUCTURE FOR DIJKTRA'S ALGORITHM DijktraAlgorithm(G, w ,s) - Distance of all the source as zero, dist[s]<-0. - For all the vertices other then source store distance as dist[v-s]<-infinity • S, set for holding all the visited vertices initialize as empty • Q, initially all vertices are present in the queue. • prev[s] <- undefined DIJKTRA'S ALGORITHM

> 20**while (Q**≠ empty) **do u <- minimumDistance(Q,dist) S = S** ∪ **{u}** for all vertex **v**∈ neighbour **[u].** U pdate(u, **v,** w) UPDATE **(U,** V **,W)** if dist **[u]** + w[u **,v] < dist[v]** set **dist[** v] = dist **[u] + w[u,v]**

set prev[v] = u The following

> 87**algorithm is** suffices **to find the minimum weight path** between vertices pair, **the**

> 63**running time of this algorithm is O(|E| + |V** |logV) as **we**

are using the Fibonacci heap in the implementation for the priority queue. Theorem 1 The dijktra's algorithm finds the optimal path between the vertices of

> 15**the graph.** Futhermore, **it discovers** path **in** the increasing **order of weight** [3] [1]. **Proof.**

This theorem can be proofed by inductive hypothesis, let OPT be the optimal path found by the algorithm. Inductive hypothesis is that at the beginning of the each step we are taking previous distance and comparing that distance with new calculated distance, if it is optimal add up this distance in the optimal path. At the beginning of the iteration we are taking the optimal path, and in each iteration we are keeping the minimum weight edge in the path of the heap, we are taking that path only so we can say that, we are getting

> 16**path between a pair of vertices,** that **path is**

optimal. By this theorem, we can say that each and every time we are choosing minimum weight edge, and appending this chosen edge to path previously covered by one of the shortest path. Lemma 2 Triangle inequality,

> 65**dist[u,v] ≤ dist[u,** w] + **dist[** w **,v].** Proof. Let **there is a**

minimum weight path between source

7**u and v.** Then, **there is no** other **path between u and v**

having weight less then that. Theorem 3 The dijktra's

36**algorithm finds the** optimal **path between** the **vertices of** the **graph.**

Futhermore it discovers path in the increasing order of weight [3]. Proof. As we are considering only the weighted graph with non-negative edge weight, when- ever we are finding the

106**path from** one **vertex to** the **other vertex,** then **we are**

adding some weight in our path. i.e., so we can say that, we are increasing the weight of the path as we dis- covered new edge, because we are considering only the nonnegative weight edges in the graph, whichever edges we discover we addup this edge to the previous optimal path, so we can say that dijktra's algorithm discovering the path in the increasing order of weight. Lemma 4 A subpath of a path is optimal. Proof. For a path sequence (v1, v2, v3 . . . vk), subpath will be (v2, v3 . . . vk−1), which have weight less then the path, because it only considering only subset of edges cover by the path. 3.2.3 Run Time of Dijktra's Algorithm

51**Running time of** dijktra's **algorithm** totaly **depends on, how we** implemented **the min-** priorty **queue.** suppose **we**

are implementing the

2**min-priority queue by** simply by storing **the vertices** from **1 to |V**

|. Removing the minimum element from this will take time of O(|V |), because we have to cover the entire array for finding the minimum element. So total running time for dijktra's algorithm will be

2**O(|V |2). We can achieve a** better **running time of** this algorithm **O(|V** |logV + **|E|) by**

implement this by using fibonacci heap [1]. Graph Terminology 1:-

35**Number of edges E** in **a** complete undirected **graph** with **V** ver- tices. **E**

= V∗(V − 1)/2 E = O(V 2) Graph Terminology 2:-

35**Number of edges E** in **a** undirected sparse **graph** with **V** vertices. |V | ≤ **E**

< |V |*logV 3.3 Conclusion In this chapter we presented, how to reduce the running time of some of the steps of the algo1, whichever steps are taking more running time, our focus was only those steps which are taking more running time. For that we come up with a solution, which provied better running time for our approximation algorithm algo1. As we are using the complete graph formation algorithm from a given graph G which will taking

82**a running time of** order **O(|** S||V |2), **where** S **is the** number **of**

terminals, which are the subset of vertices |V |. We are reducing some running time complexity of heuristic algorithm algo1, which was taking

13**running time of O(|** S **||V |2),**

and our new heuristic algorithm is taking

13**running time** complexity **of O(|** S **||V |logV**

) for the sparse graph and O(|S||V |logV + |E||S|) for the undirected simple graph, this is achieved by running dijktra's algorithm from every terminal as source. We are now able to skipping some of the steps of our heuristic algorithm algo1, and come up with a better running time algorithm, as step[1,2,3] are skipped from heuristic algorithm. So in this with help of dijktra's algorithm a new running bound is achieved that we will study a new algorithm algo2 in next chapter that reduce the running time of previous algorithm algo1. CHAPTER 4 PROPOSED HEURISTIC ALGORITHM 4.1 Overview

26**In this** chapter, **we propose a new heuristic Steiner tree** algorithm **for**

approximation,

127**with a** new **running time** complexity **of** order **O(|**

S||V |logV) for the sparse graph and O(|S||V |logV + |E||S|) for the undirected simple graph, where |S|

41**is the number of** terminal nodes **and** |V | **is the number of** vertex **nodes in the graph** G. **This**

running time, we are getting with the help of a shortest path finding algorithm between the vertices, so for that we are using

4**all- pairs shortest-paths** algorithm. **In** this algorithm **a graph G = (V, E),**

is given as a input. The

53**all pairs shortest path problem** on G **is to** compute **the minimum distance between** the vertices **of**

graph G. In this proposal, we are presenting, how we are reducing the

119**running time** complexity **of** our **algorithm** algo1 **as** given **in the**

previous chapter with the help of some other algorithm, for that we are using the algorithm that find shortest path between pairs of vertex, with better running time complexity. So for that we are using

9**all-pairs shortest-path** algorithm for **finding the shortest path between** the **pairs of** vertex. **In**

this part we are presenting how, we will used the gooness of all-pairs shortest path

8**algorithm, for finding** the **steiner tree in a given** graph **G.**

For our heuristic algorithm algo1 presented in the previous chepter, the running time complexity for that algorithm is O(|S||V |2), which is approximate to |V |3 if all vertices are terminals. Here we are presenting the improved version of that algorithm algo1, with the help of the dijktra's algorithm. As we know that the time bound for the dijktra's is O(|V |logV), if we implement min-priority queue with the help of fibonacci heap. In algorithm algo1 most of the running time taken by the step 1, so we have to think about this step how to reduce the running time for this step, and rest of steps are taking some what linear or O(|V |2) running time, so overall running time for that algorithm was O(|S||V 2|). If

61**we use the all-pairs shortest-path algorithm for finding the**

minimum distance between vertices, they we come up with a new running time complexity of O(|S||V |logV) for the sparse graph and O(|S||V |logV + |E||S|) for the undirected simple graph, if we find the shortest-

path between the vertices by dijktra's algorithm by running the dijktra's algorithm |S| times. 4.1.1 Modification in The

22**All-pairs Shortest-Path Algorithm** Here modification **for**

23**finding the shortest path between** the **pairs of** vertex, **in** this algorithm **we**

run the dijktra's algorithm for every vertex present in the graph for

103**to find the** minimum distance **between source and destination.** But **in**

the modified algorithm instead of running di- jktra's algorithm for each and every pairs vertex for finding the shortest path between then, but for our heuristic algorithm we are require to find the shortest distance between the terminals only, so if we run the dijktra's algorithm from each and every terminal taking as a source of the

79**algorithm. For finding the shortest path between** every **pair of**

terminals, we have to run the dijktra's algorithm from every terminals, total number of terminals are |S|, so our dijk- tra's algorithm will run O(|S||V |logV + |E||S|) times for

23**finding the shortest path between** all **-pairs of vertices in** G, if **we**

9**find the shortest path between** the **all pairs of**

terminals by running the dijktra's algorithm from every terminal as a source and rest of the vertices as the destination for the dijktra's algorithm. Then we have to run dijktra's algorithm almost O(|S|) times

17**to find the shortest path between** every **pairs of**

terminals, and total runnning time will be O(|S||V |logV) for the sparse graph and O(|S||V |logV + |E||S|) for the undirected sim- ple graph, where O(|V |logV) is the running time for the dijktra's algorithm [3], and we are running the dijktra's algorithms |S| times, as total number of terminals in a given graph G are |S|. So worst case running time

4**for finding the shortest path between each** and every pairs **of**

terminals will be O(|S||V |logV) for the sparse graph and O(|S||V |logV + |E||S|) for the undirected simple graph, by running the dijktra's algorithm from every terminals. We are just running the dijktra's algorithm from every terminals of the graph G, and

4**finding the shortest path between** the every **pair of the**

terminals, so we can say that the running time complexity for this algorithm

57**will be O(|S||V** |logV) and O(|S **||V**

|logV + |E||S|), which we can say that, this is better running time than the previous algorithm's running time for algorithm algo1 which was O(|S||V |2) and we are skipping some of the steps of algorithm algo1, but now we have a algorithm algo2 with running time O(|S||V |logV) in sparse graph for finding the minimum distance between the terminals, with the help this we form a subgaph, which we was construct in the step 3 of hueristic algorithm algo1, which is better then previous. Lemma 5

2**In a graph G = (V, E)** such **that** S ⊆ **V** , where S **is**

the terminals of the graph G, then we can stated that the

34**running time of the algorithm** will be **O(|** S **||V** |logV) ≤ **O(**

V 2logV). Modified

16**algorithm, is** use **to find the** minimum distance **between pairs of vertices** in **a**

graph. If

98**we run dijkstra's algorithm from** every pair **of the**

terminal vertices and find the shortest-path between other terminals. The dijktra's algorithm discover the path by using the optimal edges in the shortest path, while prunning the other unnecessary edges that are not usefull in the shortest path. Here we are presenting the description of the modified approximation algorithm. 4.1.2 Approximation Algorithm for Steiner Tree Here is our steiner tree approximation algorithm, with better running bound. We start with a heuristic algorithm and their explaination by using graph. Heuristic Algorithm :- Heuristic

8**Algorithm for** construct **a Steiner tree** from **a** graph. **Given** a connected graph **G, and**

edges weight or cost is the minimum distance d between two vertices vi and vj if there is a edge. We are given some require vertices S, S is the subset of vertices S⊆V. ALGORITHM algo2:-

68**INPUT A** undirected **connected graph G = (V,E) with edge** distance or **cost**

d between the OUTPUT vertices vi and vj,

33**and a subset of nodes S ⊆ V** , S is **called** required vertex. A **Steiner Tree**

Ts = (Vs, Es) of G and S. Step 1. Given a graph G and required vertices(S) as input, we have to construct a subgraph Gs, for this graph G. This subgraph having the shortest path between each and every pairs of terminals. This can be done by running dijktra's algorithm for each terminals as a source. Step 2. From that subgraph Gs find a

66**minimum spanning tree, by** applying **one of the minimum spanning tree**

algorithm.(If more that minimum spanning trees are present in the subgraph, pick arbitrary one, as all are minimum spanning tree so there weight will be same.) Step 3. From the minimum spanning tree, delete all unnecessary edges which are not connecting the required vertices(S). Do deletion till there is no leaves as nonterminal. Final tree after this, so called as the Steiner tree(No steiner nodes as leaves). In this algorithm running time taken by each Step, Step 1 running time O(|S||V |logV) for the sparse graph and for other graph it will be depend upon the order of O(E) in the graph, as we are applying the dijktra's algorithm on every terminals of the graph, and total number of ter- minals are |S|, so we have to run dijktra's algorithm |S| times to get the shortest distance from each and every terminals of the graph, we know running time for dijktra's algorithm is |V |logV , which we are applying |S| times, so total running time of our step 1 will be O(|S||V |logV) for the sparse graph and for other graph it depend on the order of O(E). Step 2 could be done in a 3 1 k 10 1 2 h i 2 2 g 2 1 2 1 j b 1 1 f 1 e 5 4 8 2 2 c 9 d Figure 4.1: Connected weighted Graph G, with steiner vertices(hollow), and 4 terminal ver- tices(dark)

5**O(|V |2) time and Step** 3 can **be done in O(|V |) time,**

as our previous algorithm algo1 is tak- ing. So running time complexity for our algorithm algo2 will be O(|S||V |logV) for the sparse graph and O(|S||V |logV + |E||S|) for the undirected simple graph. Example: 4.2 Input Dataset

1**Description The input instances are taken from** DIMACS website (www : **//steinlib.zib/testset.php), which contain several** dataset **for the steiner tree problem.**

instance dataset that, we used in our implementation. 1. P4E Dataset :- P4E dataset instances are randon generated complete graphs with euclid- ian weights. In these instance DC column classifies the difficulty of the instance [10]. 1 b f 2 e b 1 f 1 e g 11 1 1 2 1 g

74**2 h 1 2 b 1 2 h b**

c d 2 2 2 c d a 1 1 a 1 (a) A complete graph of G using terminals nodes, with minimum edges weight between terminals (b) Minimum spanning tree from complete graph a

131 **b 1** b **f 2 e 2** 1 1 **c**

d (c) Steiner tree for Graph G Figure 4.2: All the steps of our algorithm are cover by above graph, final Steiner tree is shown figure 4.2(e). Figure 4.3: Running Time Comparision for Both Algorithms algo1 and algo2 on P4E dataset When we run both the algorithm algo1 and algo2 on the dataset description in the table 4.1, algo1 is given heuristic and algo2 is proposed algorithm. In figur 4.3, as we observe

121**that the running time** for **our algorithm** algo2 **is**

almost better than the algorithm algo1 in all the cases. As we observe in all cases more is the Opt more is the difference in the running time.

10**Name |V | |E | |T | DC Opt** P455 100 4950 **5 Ps**

1138 P456 100 4950 5 Ps 1228 P457 100 4950 5 Ps 1609 P458 100 4950 5 Ps 1868 P459 100 4950 5 Ps 2345 P460 100 4950 5 Ps 2959 P461 100 4950 5 Ps 4474 P463 200 19900 20 Ps 1510 P465 200 19900 40 Ps 3853 P466 200 19900 100 Ps 6234 Table 4.1: P4E instance 1. BCD Dataset :- B, C, D are the instances of random generated graph with edges weights between 1 and 10. Out of these instances, we run both the algorithms on some of the instances of these dataset, as tabulated [10].

52**Name |V | |E | |T | DC Opt b01 50 63 9 Ls 82**

c01 500 625 10 Ps 144 c10 500 1000 250 Ps 1093 c15 500 2500 250 Ls 556 c18 500 12500 250 Ps 113 c19 500 12500 250 Ps 146 c20 500 12500 250 Ls 267

10**d17 1000 25000 10 Pm 23 d18 1000 25000 167 Ps 223 d19 1000 25000 250 Ps 310 d20 1000 25000 500 Ls 537**

Table 4.2: B, C, D are the instances Figure 4.4: Running Time Comparision for Both Algorithms algo1 and algo2 on B, C, D dataset As we can see in all the cases, for all the instances B, C, D our algorithm algo2 taking less running time than the algorithm algo2. 4.3

56**Computational Results Computational experiments were performed on test** data set **with quite different characteristics.**

All instance are available from the repositiry steinlib [10]. We run both the algorithms algo1 and algo2 on these instance. The running times for both the algorithms are present in the table 4.1, and table 4.2. First heuristic algorithm algo1 given by

26**L. Kou, G. Markowsky, and L. Berman** [8]. Second **algorithm** algo2 propose **in**

this thesis. Both algorithms were run on the same dataset, and we calculate their running time, how much time both algorihm are taking, in almost all cases our propose algorithm is taking less time as compare to the previous heuristic algorithm. Representations in tables are as follows, |V |

117**number of vertices** present **in the graph, |E|**

24**number of edges** present **in the graph,** |S| **number of** terminals present **in the graph.**

Running time is in seconds present for both the algorithms. Instance Heuristic Algo1 Heuristic Algo2 |V | |E | |S | Time in Sec |V | |E | |S | Time in Sec b01.stp 50 63 9 0.00369 50 63 9 0.000967 c01.stp 500 625 5 0.016047 500 625 5 0.034286 c20.stp 500 12500 250 1.13426 500 12500 250 0.928982 c19.stp 500 12500 150 0.573434 500 12500 150 0.456435 c18.stp 500 12500 83 0.369872 500 12500 83 0.30325 c16.stp 500 12500 5 0.015228 500 12500 5 0.010057 c15.stp 500 12500 250 1.11433 500 12500 250 0.879228 att48fst.stp 139 202 48 0.123454 139 202 48 0.156677 pr136fst.stp 196 250 136 0.408418 196 250 136 0.452413 d19.stp 1000 25000 250 2.34159 1000 25000 250 1.8951 d20.stp 1000 25000 500 4.76639 1000 25000 500 3.74008 d18.stp 1000 25000 167 1.61377 1000 25000 167 1.26696 d17.stp 1000 25000 10 0.10314 1000 25000 10 0.105337 berlin52fst.stp 89 104 52 0.318259 89 104 52 0.185827 bier127fst.stp 258 357 27 0.765978 258 357 27 0.443231 rat195fst.stp 560 870 195 1.43076 560 870 195 0.691303 rd400fst.stp 1001 1419 400 1.43076 1001 1419 400 1.50568 u574fst.stp 990 1258 574 5.17052 990 1258 574 2.18517 p455.stp 100 4950 5 0.0573699 100 4950 5 0.0309451 p456.stp 100 4950 5 0.040837 100 4950 5 0.031431 p457.stp 100 4950 10 0.0669429 100 4950 10 0.0486231 p458.stp 100 4950 10 0.067569 100 4950 10 0.0483148 p459.stp 100 4950 10 0.121697 100 4950 10 0.0835261 p460.stp 100 4950 20 0.144033 100 4950 20 0.0809169 p461.stp 100 4950 50 0.31225 100 4950 50 0.18301 p465.stp 100 4950 40 0.237409 100 4950 40 0.154525 p466.stp 200 19900 100 0.606883 200 19900 100 0.360251 Table 4.3: Detail results of computation on test instance Instance Heuristic Algo1 Heuristic Algo2 |V | |E | |S | Time in Sec |V | |E | |S | Time in Sec es10fst15.stp 16 18 10 0.71318 16 18 10 0.04878 es10fst14.stp 24 32 10 0.721951 24 32 10 0.04708 es10fst13.stp 18 21 10 0.72192 18 21 10 0.046977 es10fst13.stp 18 21 10 0.72192 18 21 10 0.046977 es10fst13.stp 18 21 10 0.72192 18 21 10 0.046977 es10fst13.stp 18 21 10 0.72192 18 21 10 0.046977 es10fst13.stp 18 21 10 0.72192 18 21 10 0.046977 es20fst101.stp 29 28 20 0.123531 29 28 20 0.08129 es20fst102.stp 29 28 20 0.125359 29 28 20 0.0812831 es20fst103.stp 27 26 20 0.132646 27 26 20 0.078793 es20fst104.stp 57 83 20 0.134467 57 83 20 0.0795579 es20fst105.stp 54 77 20 0.132566 54 77 20 0.0807161 es20fst106.stp 29 28 20 0.133814 29 28 20 0.0790079 es20fst111.stp 33 36 20 0.134738 33 36 20 0.0798302 es20fst113.stp 35 40 20 0.133753 35 40 20 0.0802221 es250fst01.stp 623 876 250 11.5419 623 876 250 0.893757 es250fst02.stp 542 719 250 11.5419 542 719 250 0.885769 es250fst03.stp 543 727 250 13.9157 543 727 250 0.890725 es250fst04.stp 604 842 250 11.2278 604 842 250 0.893235 es250fst05.stp 596 832 250 11.5419 596 832 250 0.891753 es250fst10.stp 662 951 250 10.8352 662 951 250 0.895662 es250fst11.stp 661 952 250 13.2844 661 952 250 0.900668 es250fst15.stp 713 1053 250 11.6048 713 1053 250 0.90136 Table 4.4: Detail results of computation on test instance 4.4 Conclusion Result we obtain for both the programs by running program using G++ compiler on a machine with configuration as

77**intel(R) core(TM) i5-2320 CPU @ 3.** 00Hz **3.**

00 GHz processor, with RAM 4.00GM and system type 32-bit operating system. CHAPTER 5

21**CONCLUSION AND FUTURE WORK 5.1 Conclusion and Future Work In this** work, **we**

have designed a heuristic approximation algorithm for steiner tree by using all-pairs shortest-path algorithm, with the help of this algorithm, we have came to a some better algorithm whose running time is better then previous heuristic algorithm presented by

49**L. Kou, G. Markowsky, and L, A**

F ast Alg orithm f or S teiner T ree [8], running time of this algo- rithm was O(|S

1**||V |2). where** |S | **is the number of terminals and |V | is the number of vertices in** the graph **G.**

In this algorithm running time mainly taken by the first step, rest of the step are taking a running time of some what smaller then that, so our thinking were how to reduce the running time for this step 1 only, and get a some better running time complexity. For that think we used all-pairs shortest=path problem, and we come up with a

70**better running time complexity i.e., O(|** S **||V** |log V) **for** the **sparse**

graph and

57**O(|S ||V |log V** + |E ||S |) for **the**

undirected simple graph. Computational results presented by our all-pairs shortest-path algorithm are competitive with the other heuristic algorithms for the steiner tree. For some instaces our algorithm fail to praduce better running time. In the implementation of algorithms, we used C++ as language for implementation, this imlementation is done on the reduced running time algorithms based on properties of special graphs. One can participate in the DIMACS implementation challenge with good implementations. REFERENCES [1] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. Introduction to algorithms, volume 2. MIT press Cambridge, 2001. [2] Steven J. Phillips David R. Karger, Daphne Koller. Finding the hidden path: Time Bound For All-Pairs Shortest Paths. SIAM Journal on Computing, 22(6):1199–1217, 1991. [3] Dijktra's. A Note on two problems in connection with graphs. Numerische mathematic, (1):269– 271, 1959. [4] P.Winter F.K. Hwang, D.S Richards. The Steiner Tree Problem. Elsevier, 1992. [5] V. Gopalkrishan C. K. Wong Ho, jan-ming. A New Approach to the Rectilinear Steiner problems. 26th ACM/IEEE, pages 161–166, 1989. [6] R. Thomas J. Byrka, F. grandoni and Laura Sanitá. An improved lp - based approximation for steiner tree. ACM 978-1-4503-0050-6, June 5-8 2010. [7] R. Thomas J. Byrka, F. grandoni and Laura Sanitá. Steiner tree approximation via Iterative randomized rounding. ACM, June 5-8 2010. [8] G. Markowsky L. Kou and L. Berman. A Fast Algorithm for Steiner Tree. Acta Informatica, 15:141–145, 1981. [9] H. Proömel S. Hougardy. A 1.598 approximation algorithm for the steiner problem in graphs, 1999. [10] A. Martin S. Voss and T. Koch. SteinLib Testdata Library Library. Online document at http://steinlib.zib.de/download.php, 2001. [11] Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. Journal of Computer and System Sciences, 51(3):400–403, 1995. 2 3 4 5 6 7 8 9 10 12 13 15 16 17 18 19 20 21 22 24 25 26 27 28 29 30 32 33 34 35 36 37 38 39 40 41 43