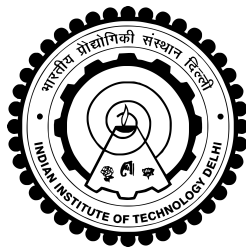


**THE EIGENANT ALGORITHM: EXTENSIONS,
APPLICATIONS AND HARDWARE
IMPLEMENTATION**

UDIT KUMAR



**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI
OCTOBER 2016**

©Indian Institute of Technology Delhi (IITD), New Delhi, 2016

**THE EIGENANT ALGORITHM: EXTENSIONS,
APPLICATIONS AND HARDWARE
IMPLEMENTATION**

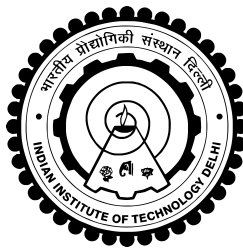
by

UDIT KUMAR

Department of Electrical Engineering

Submitted

in fulfillment of the requirements of the degree of Doctor of Philosophy
to the



Indian Institute of Technology Delhi

OCTOBER 2016

Certificate

This is to certify that the thesis entitled “**The EigenAnt Algorithm: Extensions, Applications and Hardware Implementation**”, being submitted by **Mr. Udit Kumar** for the award of the degree of **Doctor of Philosophy** to the Department of Electrical Engineering, Indian Institute of Technology Delhi, is a record of bonafide work done by him under our supervision and guidance. The matter embodied in this thesis has not been submitted to any other University or Institute for the award of any other degree or diploma.

Dr. Jayadeva

Professor
Department of Electrical Engineering,
Indian Institute of Technology Delhi,
Hauz Khas, New Delhi - 110016,
INDIA.

Dr. Kaushik Saha

Co-supervisor
Chief Technology Officer,
Samsung R&D Institute India - Delhi,
2A, Sector - 126, Noida,
Uttar Pradesh - 201303, INDIA.

Acknowledgments

I would like to express my greatest gratitude to my supervisors Prof. Jayadeva and Dr. Kaushik Saha for their guidance, interest, advice, support and help throughout the course of the present work. I am grateful for having the opportunity to work with them and hope that I shall continue to have their blessings in the future. I am also indebted to my research committee members, Prof. Santanu Chaudhury, Prof. Amit Kumar and Dr. Shouri Chatterjee for their valuable insights during my research work. Dr. Shouri Chatterjee's help in chip fabrication and PCB design requires special mention.

I am grateful to the faculty and staff at the Department of Electrical Engineering at the Indian Institute of Technology Delhi for providing me an excellent work environment during the past years. I wish to acknowledge the funding provided by Department of Science and Technology under the project "Efficient Algorithms for Global Optimization and for Learning and their Hardware Implementation" for chip manufacturing. I am grateful to Natronix Semiconductor Technology (formerly SPEL Semiconductor Limited) for helping me with their expertise on chip packaging. Rajesh Gupta and Madhur Kashyap's help in chip tape-out has been instrumental. I would also like to acknowledge the help which Mr. Rakesh Kumar, Mukesh Singh, Ritwick Pahari provided me during my research work.

I would like to thank the students who made my experience at IIT Delhi a memorable one. Words fail me to express my deepest thanks to Sumit Soman for being a great friend, his cooperation helped me to reach this milestone. I enjoyed working with Pawas, Munish, Aashish, Rajshree, Kinde, Roohie, Gajendranath, Manpreet, Navneet and others. I would also like to thank my former employer, STMicroelectronics, the management and my colleagues there for their support, particularly Olivier Florent and Bhanu Prakash. I am also grateful to my present employer Atrenta India Pvt. Ltd.

This thesis would not have been possible without the help and unconditional support offered by my family. I am truly grateful to my parents, sisters, wife and son, who have stood by me in all times.

Udit Kumar

Abstract

Biological phenomena prevalent in nature and among organisms have inspired development of algorithms for self-organization in dynamic scenarios. The behavior of ants has been found to particularly suit tasks such as optimization, and this has been modeled as the class of Ant Colony Optimization (ACO) algorithms. This has primarily been motivated by the fact that ants travelling from a source node (nest) to a destination node (food source) deposit pheromone on the path traversed, and over time the paths traversed by the ants converge to the shortest path, which represents the stabilized system at equilibrium. Eventually, all ants end up choosing the shortest path as it has the highest pheromone concentration.

One can analogously simulate this scenario for solving several optimization tasks, and expect a similar behavior to eventually lead to finding an optimal path (or solution). Though this has been the thrust of most of the ACO-based optimization approaches, a variant called the EigenAnt has been the first to have a mathematical proof of convergence. This thesis explores extensions, applications and hardware realization of the EigenAnt algorithm to illustrate its widespread practical applicability.

The initial chapters establish the EigenAnt as an algorithm of choice for discrete and continuous optimization tasks. There are a plethora of tasks that involve optimization. The use of EigenAnt based approaches for these tasks helps attain better solutions to the problems, often with lower computational cost. Applications include problems involving the optimization of discrete variables as well as others that employ continuous valued variables. The inferences lead to conclude that the biological phenomena of traversal of ants indeed benefits optimization routines, irrespective of the nature of the variables involved.

The true prospective benefits of such biologically inspired algorithms can be gauged only when they are implemented in hardware. This enables such approaches to be run with data generated from natural sources, which is crucial to evaluate its practical benefit. An additional aspect that stems from hardware implementation is the choice of optimal

design architecture in terms of design area and power consumption that would enable these algorithms to be used efficiently despite limited precision or accuracy. Hardware implementation imposes constraints on the algorithm that can potentially alter its behaviour and needs careful study. The sequel focuses on determining the effect of quantization on algorithm behavior. This is with a view to understanding the effect of precision during floating to fixed-point conversion for hardware implementation of EigenAnt. It is shown that for the case of the EigenAnt, a feasible architecture is possible, and the same has been implemented on a VLSI ASIC that was fabricated in a 180 nm CMOS process.

Contents

1	Introduction	1
2	Preliminaries	5
2.1	The EigenAnt Algorithm	5
2.2	EigenAnt for General Graphs	7
2.3	The $IACO_{\mathbb{R}}$ Algorithm	9
2.4	Local Search in $IACO_{\mathbb{R}}$ using Mtsls1	10
2.5	BFGS approach	12
2.6	Support Vector Regression	12
2.7	SOCO Benchmarks	14
3	Combinatorial Optimization using EigenAnt	15
3.1	Set Cover Problem	16
3.1.1	Pre-processing	18
3.1.2	Obtaining continuous solutions	18
3.1.3	Randomized Rounding	20
3.1.4	The EigenAnt for Combinatorial Optimization	21
3.1.5	Post-processing	22
3.2	Summary of our approach	22
3.3	Experimental Framework	25
3.4	Results	25
3.4.1	Analysis of problem size	25
3.4.2	Analysis of solution quality	26

3.4.3	Incremental improvement using solution processing steps	32
3.4.4	Results using an LP solver	35
3.4.5	Results with heuristic methods	39
3.5	Conclusion	39
4	EigenAnt for Continuous Optimization	41
4.1	A hybrid approach for local search in $IACO_{\mathbb{R}}$	42
4.1.1	Hybrid Local Search using Mtsls1 and BFGS	42
4.1.2	Performance of the Mtsls1-BFGS approach	46
4.2	Gradient Based $IACO_{\mathbb{R}}$ for Continuous Global Optimization	53
4.2.1	The Proposed Algorithm for Gradient based $IACO_{\mathbb{R}}$	54
4.2.2	The EigenAnt2 Algorithm	55
4.2.3	Gradient Based $IACO_{\mathbb{R}}$	56
4.2.4	Performance on representative functions	58
4.2.5	Results on SOCO Benchmarks	60
4.3	The Modified Mtsls1 and Parallel Mtsls1	63
4.3.1	Handling bound constraints	64
4.3.2	Modified Mtsls1	66
4.3.3	Parallel Approach for Mtsls1	67
4.3.4	Results using the MMtsls1 and PMtsls1	68
4.4	The EigenAnt algorithm using Modified Mtsls1 (Eigen-MM)	73
4.4.1	Performance of the Eigen-MM approach	77
4.4.2	Comparative Performance of Eigen-MM	80
4.4.3	Function evaluations using Eigen-MM	83
4.5	Conclusion	86
5	EigenAnt Quantization Noise Analysis	87
5.1	Floating-Point to Fixed-Point conversion	88
5.1.1	Fixed-point System: Simulation based fractional length deter- mination	89
5.1.2	Fixed-point System: Stability Analysis	89

5.1.3	Fixed-point System: Noise Analysis and Signal to Noise ratio	93
5.2	Simulation results	94
5.3	Conclusion	95
6	Hardware Implementation of EigenAnt	97
6.1	Deterministic pheromone update	99
6.2	Hardware Implementation of EigenAnt	100
6.2.1	Basic Units for Hardware Implementation	100
6.2.2	Design Architectures	108
6.2.3	Experimental results	111
6.2.4	VLSI implementation	118
6.2.5	Validation on a Field Programmable Gate Array (FPGA)	118
6.3	ASIC Implementation of EigenAnt for General Graphs	121
6.3.1	Design Implementation, on-chip testing and configuration methodology	124
6.3.2	Chip Packaging and Test setup	126
6.3.3	Simulation results	127
6.4	Comparison with software implementation	131
6.5	Conclusions	132
7	Conclusions and Future Work	133

List of Figures

2.1	EigenAnt Algorithm	6
2.2	Network for EigenAnt for general graphs.	7
2.3	Mtssl1 search along one dimension from the first dimension to the last dimension	11
2.4	Iterations for the global minimization of Rastrigin function. Starting points are depicted by squares, local minima by circles and the regressor obtained is shown by using dotted lines.	13
3.1	An example of the set cover problem	16
3.2	Node selection paths for converting a continuous solution to a binary one.	20
3.3	Flowchart illustrating EigenAnt for Combinatorial Optimization	21
3.4	Phases for solving combinatorial optimization problems.	23
3.5	Performance of EigenAnt based combinatorial optimization approach for the SCP. The use of randomized rounding on the average solution obtained from 10 runs results in a deviation of 2.64% from the optimal solution, while with the use of EigenAnt the deviation is 1.62%, which amounts to a decrease of 39.4% (without pre-processing). For the case of deviation of the best solution of 10 runs from the optimal solution, the values for randomized rounding and EigenAnt are 2.12% and 1.09% respectively, which corresponds to a decrease of 48.58%.	29

3.6	Run time comparison between different approaches. The lower portion of the bars indicates the runtime till obtaining the continuous solution, while the upper portion of the bars represent runtime till finding the optimal binary solution from the continuous solution. EigenAnt approach takes 44.50% (in case of A1), 43.87% (in case of A2) and 39.10% (in case of A3) less time than randomized rounding based approach.	32
3.7	Percentage deviation of the best solution from the 20 runs (10 using B1 and 10 using B2) from the optimal solution.	35
4.1	Hybrid algorithm.	44
4.2	Function $F1$: Comparison of error of fitness value in Mtsls1 and BFGS. This is the case where both algorithms find the global optimum.	49
4.3	Function $F3$: Comparison of error of fitness value using all three approaches. In this case, BFGS finds the global optimum, whereas Mtsls1 fails.	50
4.4	Function $F13$: Comparison of error of fitness value on all three approaches, indicating that the standalone use of the BFGS or Mtsls1 algorithm fails to find the global optimum, but our hybrid algorithm is able to reach it.	50
4.5	Function $F17$: Comparison of error of fitness values on all three approaches, indicating that the standalone use of the BFGS or Mtsls1 algorithm fails to find the global optimum, but our hybrid algorithm is able to attain the optimal solution.	51
4.6	Comparison of average number of function evaluations used in all three approaches. Our hybrid approach requires fewer iterations.	51
4.7	Box plot for average errors	52
4.8	Box plot for median errors	52
4.9	A summary of our approach	54
4.10	Sphere function - Performance of EigenAnt2 is similar to SDM	58
4.11	Sphere function for 50 dimensions - EigenAnt2 performs better than SDM	59

4.12	Plots comapring EigenAnt2 and gradient descent objective values and norm for RosenBrock function.	59
4.13	The box-plots show the distribution of average errors obtaints on 19 SOCO benchmark funtion of 50 dimensions	62
4.14	The box-plots show the distribution of the median errors obtaints on 19 SOCO benchmark funtion of 50 dimensions	62
4.15	Bound constraint handling: the penalty is proportional to the number of function evaluations done so far.	65
4.16	Bound constraint handling in the proposed approach. Here, the weight of the bound function $B(\mathbf{x})$ is updated only at the start of the local search.	65
4.17	Comparison of the impact of penalty function weight on the cost function for the two approaches	66
4.18	Mtssl1 search along one dimension from the first dimension to the last dimension	67
4.19	Parallel Mtssl1 searches independently on the problem dimensions.	68
4.20	Box plot for average and median Errors	70
4.21	Comparison of average number of function evaluations used in all the three approaches, $IACO_{\mathbb{R}}$ -Mtssl1, $IACO_{\mathbb{R}}$ -MMtssl1 and $IACO_{\mathbb{R}}$ -PMtssl1 indicated by red, green and blue bars respectively.	72
4.22	Plots for function $F1$ and $F4$ indicating reduction in function evaluations using $IACO_{\mathbb{R}}$ -PMtssl1 and $IACO_{\mathbb{R}}$ -MMtssl1 approaches, with lower or comparable error rates.	73
4.23	Illustrations on using our approach on Functions $F12$ and $F16$	73
4.24	The Eigen-MM Approach	74
4.25	Convergence plots for functions $F1$ and $F5$	75
4.26	Box plot for average error	80
4.27	Box plot for average error for 1000D functions.	81
4.28	Box plot for median error.	81
4.29	Box plot for median error for 1000D.	82
4.30	Plots for function evaluation comparison for 50D, 100D, 200D, 500D	85

4.31	Plots for function evaluation comparison for 1000D	85
5.1	Floating to fixed point conversion	89
5.2	The maximum error calculated using (5.15)	95
5.3	Pheromone concentration for different values of integer and fractional width	96
6.1	Summary of the architectures proposed for EigenAnt implementation.	98
6.2	Floating to fixed point conversion	99
6.3	Block diagram for EigenAnt implementation	101
6.4	AS pheromone sum computation	101
6.5	EigenAnt pheromone sum computation	102
6.6	Area requirement for pheromone sum in AS and EigenAnt approach.	102
6.7	Linear feedback shift register	103
6.8	Dynamic nature of LFSR for random number generation. In practical scenario, the pheromone sum will usually be a large number, so muxes for LSB can be avoided.	104
6.9	Area requirement for next path selection in AS and EigenAnt approach. In the estimation for AS, we assume that heuristic information is not considered ($\beta = 0$); if heuristic information is used, area requirements for AS will further increase.	106
6.10	Pheromone update logic	107
6.11	Pheromone update unit area for AS and EigenAnt	108
6.12	Representative architectures for <i>Type-1</i> , <i>Type-2</i> and <i>Type-3</i>	109
6.13	<i>Type-1</i> Simulation results for pheromone concentration.	112
6.14	Paths selected during ant trips.	112
6.15	<i>Type-3</i> , Pheromone concentration with the use of a deterministic next-path selection approach.	113
6.16	Scalability analysis for varying number of neighbors and pheromone width.	114
6.17	<i>Serial pheromone update (U1)</i> : Effect of the number of neighbors on design area, the pheromone width register has been kept fixed at 16 bits.	116

6.18	<i>Newton Raphson update (U2):</i> Effect of the number of neighbors on design area, the pheromone width register has been kept fixed at 16 bits.	117
6.21	Test chip design with ten paths and word length of 8 bits.	118
6.22	Block diagram for design validation using FPGAs.	118
6.19	<i>Serial pheromone update (U1):</i> Effect of varying the pheromone register width on design area, keeping number of neighbors fixed as 24.	119
6.20	<i>Newton Raphson update (U2):</i> Effect of varying the pheromone register width on design area, keeping number of neighbors fixed as 24.	120
6.23	Graph topology implemented on ASIC	121
6.24	Block diagram	121
6.25	Pheromone update logic	123
6.26	Pheromone weight calculation logic	123
6.27	Design of Input/Output block diagram and setup for chip configuration	124
6.28	Test Chip design: Routed design view	126
6.29	Chip diagram with bond pads	127
6.30	Chip Photomicrograph.	127
6.31	Chip Package.	128
6.32	PCB for testing the chip	128
6.33	Pheromone concentration corresponding to the graph architecture on each node after 200 cycles.	129
6.34	Pheromone concentration on the shortest path nodes from node 1 to the destination node.	130
6.35	Chip test setup	131

List of Tables

2.1	SOCO Benchmark functions and their description.	14
3.1	Description of SCP benchmarks	25
3.2	SCP benchmarks problem size after different steps and runtime	27
3.3	Percentage deviation from the optimal solution.	28
3.4	Wilcoxon signed rank test for the approaches.	28
3.5	Average and best solution from different approaches. Values in bold and italics indicate improved solution costs over randomized rounding, while values in italics indicate cases where the same solution costs have been attained by both EigenAnt and randomized rounding.	30
3.6	Run time for different approaches. Randomized rounding approaches takes 39.1-44.5% more runtime than the EigenAnt based approach.	31
3.7	Percentage deviation of the best solution of 20 runs (10 using B1 and B2 individually) from the optimal solution.	33
3.8	Best solution from different approaches: best of original and best solution from 20 runs of the benchmarks (10 from B1 and 10 from B2), Values in bold indicate the cases where solution quality has improved as a result of using this version.	34
3.9	Average and best solution with the use of LP solver.	36
3.10	Run time for different approaches with use of LP solver.	37
3.11	Average solution percentage deviation from the optimal solution.	38
3.12	Best solution percentage deviation from the optimal solution.	38
3.13	Comparison of average solution with various method	39

4.1	$IACO_{\mathbb{R}}$ -Hybrid Average error on SOCO benchmark functions (50D) . .	47
4.2	$IACO_{\mathbb{R}}$ -Hybrid Median error on SOCO benchmark functions (50D) . .	48
4.3	Wilcoxon signed-ranks tests with respect to $IACO_{\mathbb{R}}$ -Hybrid approach . .	49
4.4	Solution norms using EigenAnt2 and SDM	60
4.5	Average error on SOCO benchmark function (50D)	61
4.6	$GACO_{\mathbb{R}}$ -Mtsls1 Median error on SOCO benchmark function (50D) . .	61
4.7	Wilcoxon Signed-Ranks Tests for average and median error with respect to $GACO$ -Mtsls1 approach	63
4.8	Average errors on SOCO benchmark function (50D) with $IACO_{\mathbb{R}}$ -MMtsls1 & $IACO_{\mathbb{R}}$ -PMtsls1	69
4.9	Median errors on SOCO benchmark function (50D) with $IACO_{\mathbb{R}}$ -MMtsls1 & $IACO_{\mathbb{R}}$ -PMtsls1	70
4.10	Wilcoxon Signed-Ranks Tests with respect to $IACO_{\mathbb{R}}$ -MMtsls1 approach for average error	71
4.11	Wilcoxon Signed-Ranks Tests with respect to $IACO_{\mathbb{R}}$ -PMtsls1 approach for average error	71
4.12	Number of functions on which global optimum is achieved, in terms of average and median error	77
4.13	Mtsls1-EigenAnt results for 50D, 100D, 200D	78
4.14	Eigen-MM results for 50D, 100D, 200D	78
4.15	Mtsls1-EigenAnt results for 500D, 1000D	79
4.16	Eigen-MM results for 500D, 1000D	79
4.17	Wilcoxon signed rank test for average and median error w.r.t $IACO_{\mathbb{R}}$ - Mtsls1	82
4.18	Wilcoxon signed rank test for average and median error w.r.t. SaDE- MMTS	82
4.19	Method used to find out function for comparison between different ap- proaches.	83
4.20	Number of function evaluations w.r.t. Mtsls1 (%)	84
4.21	Average running time for all benchmark functions for Eigen-MM	84

5.1	Maximum quantization error based upon different test scenarios.	94
6.1	FPGA resource requirements for components of the Ant System and the EigenAnt algorithm, for different numbers of neighbours. Here, T_w denotes the minimum period, T_{in} denotes the minimum input arrival time before the clock, T_{out} denotes the maximum output required time after the clock, and T_{comb} denotes the maximum combinatorial path delay. FPGA settings are xc3s5000-4-fg900 target device, speed grade -4 and ISE version 10.1i.	103
6.2	FPGA resource requirement for Ant system and the EigenAnt algorithm.	114
6.3	Summary of ASIC synthesis	126