Solving Maximum Cut Problem with an Incremental Genetic Algorithm

Jinhyun Kim
School of Computer Science & Engineering
Seoul National University
1 Gwanak-ro, Gwanak-gu
Seoul, 151-744 Korea
jh@soar.snu.ac.kr

Yourim Yoon
Department of Computer Engineering
College of IT
Gachon University
Gyeonggi-do 461-701, Korea
yryoon@gachon.ac.kr

Byung-Ro Moon
School of Computer Science & Engineering
Seoul National University
1 Gwanak-ro, Gwanak-gu
Seoul, 151-744 Korea
moon@snu.ac.kr

ABSTRACT

In this paper, we propose an incremental genetic algorithm applied to solve the maximum cut problem. We test the implementation of the algorithm on benchmark graph instances. We propose several methods to build up the sequence of subproblems, and they are tested through experiments. The performance of a genetic algorithm makes an improvement when the incremental approach is applied with respect to an appropriate sequence of subproblems.

Keywords

Incremental genetic algorithm; maximum cut problem

1. INTRODUCTION

Graphs are representative discrete data structures, and there are lots of important combinatorial optimization problems on graphs. For the problems that are classified as NP-hard problems, we need to examine an extreme number of combinations to find a good solution. Genetic algorithms(GAs), and other stochastic approaches are widely used to deal with this difficulty.

In contrast to NP-hard graph problems, there exist polynomial time algorithms for some of the graph problems in P class. The algorithms for these problems are often a greedy method or a dynamic programming. Both of them rely on the optimal substructure of the problem, which is a property that optimal solutions of subproblems could be extended to an optimal solution of the original problem.

The idea of utilizing optimal substructure could be applied to solve NP-hard graph problems, by means of an incremental genetic algorithm(IGA). IGA solves a graph problem in multiple steps, and a subproblem is tackled at each step. The algorithm begins with solving a small subproblem and the problem is gradually expanded to the original problem. The subgraph isomorphism problem(SIP) was tackled in this manner, and the subproblem was defined by the subgraph structure [1]. When the subproblems are expanded in an appropriate way, the incremental approach has brought a significant performance improvement upon it's predecessor.

In this paper, we investigate the mechanism of an IGA for graph problems through empirical analysis. We propose an IGA applied to the maximum cut problem(MCP) [2]. We use the natural subproblem structure defined by the subset of the edges, and we seek methodologies to build up a fine sequence of subproblems. Results of the experiments to verify our analysis are provided with discussions. Using an incremental approach in an appropriate way has brought a performance improvement, and the incremental genetic algorithm was able to find a better graph cut.

GECCO '16 July 20-24, 2016, Denver, CO, USA

ACM ISBN 978-1-4503-4323-7/16/07.

DOI: http://dx.doi.org/10.1145/2908961.2908976

2. GENETIC FRAMEWORK

We use a typical GA with and without incremental approach in our experiment. The operators and parameters are as follows.

- **Population management**: The size of the population is 100. Twenty new offspring are generated in each generation. The best 100 out of 120 chromosomes survive. The population is randomly initialized in the first step of the IGA, and is reused in the rest of the steps.
- **Representation**: We use a binary representation. If a chromosome has a different number of 0s and 1s, we randomly repair it.
- **Selection**: We randomly select eight chromosomes and run a tournament. The better chromosome wins the match with probability 80%. The best two chromosomes are finally selected.
- **Crossover and Mutation**: We use a uniform crossover and a random mutation. Based on Hamming distance, we first normalize the chromosomes before crossover. Each gene is inherited from one of the two parents with equal probability, and is toggled afterward with 0.5% of chance.
- **Stopping criterion**: For a fair comparison, we use a fixed number of generations. We use 10^5 for a traditional GA, and it is evenly distributed to each step for an IGA.

Table 1: The performance of the tested GA and IGAs for the MCP

G	GA	E-IGA			V-IGA			M-IGA		
		Random	Degree	BFS	Random	Degree	BFS	Random	Degree	BFS
1	11411.00	11373.20	11290.10	11283.90	11404.40	[**11411.90**]	11408.90	11366.40	11400.60	11384.70
2	11416.90	11379.00	11249.30	11290.20	11409.60	[**11417.10**]	11415.10	11371.50	11392.20	11389.20
3	11412.10	11375.60	11325.10	11286.50	11404.90	11410.10	11409.50	11365.70	[**11435.30**]	11385.30
4	11425.70	11386.00	11271.50	11299.00	11423.20	[**11442.40**]	**11425.80**	11380.10	11413.90	11398.50
5	11417.00	11380.40	11258.10	11291.00	11414.40	[**11450.10**]	**11418.70**	11374.50	11379.60	11391.20
14	2983.84	2964.97	2964.66	2962.03	**2989.07**	**2991.89**	[**2998.08**]	2971.87	2978.91	2982.76
15	2964.75	2946.12	2944.23	2944.41	**2970.86**	**2973.65**	[**2979.85**]	2953.22	2954.62	2961.28
16	2969.65	2951.15	2946.89	2944.18	**2974.30**	**2981.32**	[**2983.94**]	2957.30	2961.99	2965.21
17	2965.15	2946.83	2941.05	2943.74	**2971.31**	**2973.53**	[**2980.29**]	2954.07	2958.31	2962.56
43	6466.47	6410.77	6379.89	6375.71	6453.73	[**6474.56**]	6465.18	6423.30	6435.21	6443.29
44	6463.19	6408.30	6379.05	6370.58	6450.29	[**6470.49**]	6461.40	6418.64	6457.53	6439.86
45	6462.90	6406.78	6376.12	6370.45	6448.72	[**6469.92**]	6461.25	6419.64	6435.69	6440.85
46	6465.33	6410.04	6379.09	6372.30	6451.85	[**6475.64**]	6462.22	6422.36	6441.94	6443.28
47	6470.45	6414.29	6376.46	6376.91	6457.16	[**6475.07**]	6467.70	6426.59	6430.64	6448.11
51	3742.54	3710.64	3713.08	3698.30	**3743.31**	**3750.27**	[**3757.44**]	3720.95	3717.56	3727.72
52	3746.41	3715.03	3715.55	3706.69	**3747.10**	**3749.51**	[**3761.92**]	3725.74	3730.79	3737.34
53	3745.21	3712.13	3716.28	3704.69	**3745.72**	**3748.51**	[**3758.96**]	3723.85	3729.02	3733.04
54	3745.05	3712.70	3710.18	3707.08	3744.94	**3745.10**	[**3758.62**]	3724.93	3736.54	3735.34

3. SEQUENCE OF SUBPROBLEMS

The IGA solves the problem through solving a sequence of subproblems [1]. We first set the virtual 0^{th} subproblem to be a graph having no edges. The rest of the subproblems, from the first one to the last one, are obtained by adding some edges to the previous graph. We tried three different graph expansion methods described below.

- **Edge-wise expansion**(E-IGA): An edge is added to the previous graph at each step. The number of the steps equals the number of the edges.
- **Vertex-wise expansion**(V-IGA): A vertex is considered at each step. The incident edges connected to a vertex that has already considered in the previous step are added to the graph. This method is conceptually identical to adding the considered vertex at each step. The number of the steps equals the number of the vertices.
- **Mixed expansion**(M-IGA): A vertex is considered at each step and all of the incident edges are added to the previous graph. The philosophies of the two above methods are mixed in this one. The number of the steps equals the number of the vertices.

Determining the order of the edges or the vertices being considered at each step is another design issue. We tried three different ordering schemes; random ordering, degree based ordering(decreasing order), and BFS based ordering.

4. RESULTS AND DISCUSSIONS

The proposed algorithm was tested on benchmark graphs called G-set [2]. Table 1 shows the average cut size found by the algorithms. The average cut size is marked in bold if it is greater than the result of the traditional GA. And for each of the graph instances, we parenthesized the value by a square bracket, if it is the best result.

E-IGA and M-IGA were not effective for almost all of the cases regardless of the ordering scheme used. It was reported that falling in a bad local optimum is critical for the MCP, and most of the near state-of-the-art algorithms have a routine to avoid such situation [2]. Expanding the graph in the edge-wise way and in the mixed way as proposed seem to reinforce this situation.

There was no universally notable ordering scheme, and the characteristic of the graph decides which scheme works best on that graph. Basically, the vertex degree was a key factor for random graphs(1–5, 43–47), as degree based ordering showed the best result. For the random planar graphs(14–17, 51–54) having geometric property, the BFS ordering scheme seems to capture this property. When used with an appropriate ordering scheme, the incremental approach was effective.

5. CONCLUSION

In this paper, we analyzed incremental genetic algorithms for graph problems. A brief description of the process of an incremental genetic algorithm for a graph problem was given, and it was described in terms of a subproblem sequence. Graph expansion methods, reordering schemes, and their combinations were proposed and tested. Through experiments, we found an evidence that incremental approach is useful.

6. ACKNOWLEDGMENTS

The ICT at Seoul National University provided research facilities for this study.

7. REFERENCES

[1] H. Choi, J. Kim, and B.-R. Moon. A hybrid incremental genetic algorithm for subgraph isomorphism problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 445–452, New York, NY, USA, 2014. ACM.

[2] Q. Wu, Y. Wang, and Z. Lü. A tabu search based hybrid evolutionary algorithm for the max-cut problem. *Appl. Soft Comput.*, 34(C):827–837, Sept. 2015.