Improving Genetic Algorithms with Solution Space Partitioning and Evolution Refinements

Ay-Hwa Andy Liou
Department of Information Management, Tamkang University, Taiwan
liou@mail.tku.edu.tw

Tzong-Heng Chi
Department of Information Management, Aletheia University, Taiwan
chi@email.au.edu.tw

I-Jun Yu
Department of Information Management, Tamkang University, Taiwan
oodukoo@hotmail.com

Abstract

Irregular Sum Problem (ISP) is an issue resulted from mathematical problems and graph theories. It has the characteristic that when the problem size is getting bigger, the space of the solution is also become larger. Therefore, while searching for the feasible solution, the larger the question the harder the attempt to come up with an efficient search. We propose a new genetic algorithm, called the Incremental Improving Genetic Algorithm (IIGA), which is considered efficient and has the capability to incrementally improve itself from partial solutions. The initial solutions can be constructed by satisfying the constraints in stepwise fashion. The effectiveness of IIGA also comes from the allowing of suitable percentage of illegal solutions during the evolution for increasing the effectiveness of searching. The cut-point of the genetic coding for generating the descendants has carefully planned so that the algorithm can focus on the key factors for the contradiction and has the chances to fix it. After comparing the results of IIGA and usual genetic algorithm among different graphs, we found and shown that the performance of IIGA is truly better.

Keywords: *Genetic Algorithms, Graph Theory, Irregularity Sum, Evolution Refinement, Problem Decomposition.*

1. Introduction

The study of regular graphs can be dated back to Petersen and Faudree [19][11] in 1890s, but the irregular properties of graphs were not being investigated till Alavi et al. [2]. For a simple graph G, it is regular if its vertices have the same degree. A network is a simple graph to which each edge is assigned a positive integer value or weight. A network is locally irregular if it is connected and the vertices in the neighborhood of any vertex have mutually distinct degrees. Otherwise, a network is globally irregular or just irregular if all the vertices have distinct degrees [2][8][11]. Given an assignment which is a function $w:E(G) \rightarrow \{1, 2, 3, \ldots, s\}$, the weight of a vertex v is $w(v)= \Sigma v \in e w(e)$. The strength of a network is the maximum weight assigned to any edge. Also, an irregular assignment is valid or admissible if the weight of each vertex is different and thus the graph G is distinguishable. The irregularity strength s(G) of graph G is the minimum strength among irregular networks with underlying graph G. The study of irregular strength stems from problems related to highly irregular graphs and multigraphs first introduced by Chartrand et al. [6]. The irregularity sum of a graph G is the minimum sum of the induced weights of all vertices in an irregular assignment for G originally considered by [17]. We abbreviate the problem of finding the minimal irregularity sum as ISP, irregularity sum problems. The problem of studying s(G) was proved by Chartrand et al. to be rather hard, even for very simple graphs [6][10].

Besides, the study of irregular assignments have done well to prove the existence of lower or upper bound of s(G) [5][8][10]. However, there was less study in the construction of valid assignments under the consideration of irregularity sum. Though [4][18] has done many good results, they focused on trees only. For these reasons, in this paper, we shift our attention to the construction of solutions in finding the irregular assignments related to the irregular strength and irregular sum for graphs.

The original form of genetic algorithms introduced by Holland [16] is usually called simple genetic algorithms, abbreviated SGA, contrast to successive many variants. Related literatures can be found in [12][14][21]. We have constructed the SGA for ISP problems showed later in this paper. However, while problems are large-scale with complex solution landscape, the research results of [7] suggested that the standard GAs are not efficient in getting a suboptimal solution in limited computational power[21]. There are two ways to improve genetic algorithms for large-scale problems, which are problem decomposition [20] and searching-space reduction [21][7]. We work along with both lines.

The goal of this work is to develop an effective GA partition algorithm for ISP problems. The problem of ISP is a combinatorial optimization problem rather than a numerical optimization problem. Meanwhile, the mathematical definition of ISP is usually graph-based. As [9], we hence adopt ordering representation for the

Third International Conference on Natural Computation (ICNC 2007)
0-7695-2875-9/07 $25.00

genotype as the individuals to evolve. Though the structure of neighborhood determines the order of edges according to the practical graphs, however, there is not necessarily only one ordering mapping.

Since the construction phase is very time consuming for the SGA, we have made some efforts to improve it. On one hand, we attack it with problem decomposition by stepwise the procedure for generating the initial genetic coding. By the empirical results, it shows great improvements. On the other hand, further improvements are pursued by exploring the necessity of retaining partial illegal but usable solutions for improving the evolution procedure. It is also considered efficient and has the capability to incrementally improve itself from partial solutions. The cut-point of the genetic coding for generating the descendants is carefully planned so that the algorithm can focus on the key factors for the contradiction. This new algorithm is called the Incremental Improving Genetic Algorithm (IIGA). After comparing the results of IIGA and conventional genetic algorithm among experimental graphs, the results are satisfactory.

We refer the related results found in [1][3][4][6][11][13] and [15], within which the most important results are For any connected graph with at least three vertices, $s(G) \leq 2n - 3$ [6]. It was strengthened by [1] as $s(G) \leq n + 1$ for any graph with $s(G) < \infty$. Both developed SGAs and GPMGAs for ISP problems utilize this result to reduce the search space in force. Without this, the search space can be very huge.

Next section will cover the main ideas of IIGA along with some explanation. The experiment and the comparison detail of the results is reported in Section 4 and, finally, Section 5 concludes the research.

2. The Genetic Algorithm

2.1. The SGA for ISP

In order to employee the Simple GA for solving the ISP problem, following stages are necessary: (1) Constructing (2) Limiting (3) Encoding (4) Crossover (5) Mutation. The procedures of the first three stages are explained below using the example graph in Fig 1.

(1) Constructing: Construct possible solutions by giving arbitrary weights to the edges

(2) Limiting: Confirming the restriction that all the weights of the nodes are distinguishable.

(3) Encoding: After knowing that the assignment is valid, the next step is to transform the weight assignments of the edges into genetic coding. For example, the assignments on the diagram in Fig. 2. can be transformed into genetic coding of [2 3 1 1 3 4 4 5 1]. This coding will be used as the chromosome in the subsequent stages.

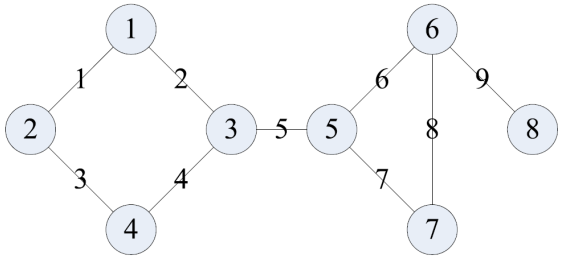

Fig. 1. An example graph showing all the labels on the vertices and the edges.

After encoding, stage (4) and (5) for crossover and mutation are carried out conventionally.

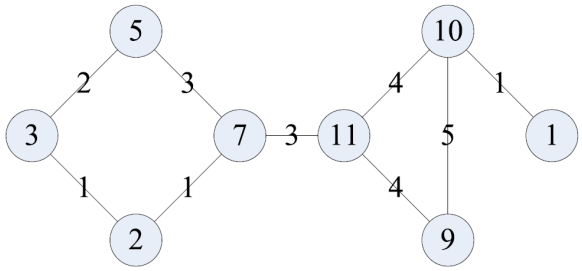

Fig. 2. The graph with arbitrarily constructed valid assignment. This assignment will then be used as a starting chromosome for crossover and mutation.

2.2. Stepwise Decomposition of the Problem

The effectiveness of generating the complete genetic coding of the chromosomes can be improved by constructing gradually in stepwise phases. Not finishing it in one shot has several advantages. First, the quality of the chromosome is better than original method. Second, the speed of generating the chromosome is faster. The stepwise construction phases are based on the partitioning characteristics of the network under consideration. The details of the procedure are described below.

Step 1) Construct the network and determine the boundaries of the partitions.

Step 2) Starting from one partition and create its corresponding chromosomes according to the limit function and determine the upper limit of the genetic coding.

Step 3) Continue on next partition and combining it with the previous partitions(s). The combined graph will be used for the overall consideration for the genetic coding when the limit function is applied again but with previous results fixed. Of course the combined graph now has a new

upper limit.

Step 4) Extend the chromosomes according to the generated genetic codings and confirm that the legal status is fulfilling the limit function. Since the previous partitions have been used to generate a smaller values for the coding (because the previous partition was smaller before the combination), the newly generated coding will only expand the coding belong to the part that is just added.

Step 5) Repeat Step 3 until all the partitions are combined.

In the example of Fig. 1, the network is apparently partitioned into two clusters– one with node numbering {1, 2, 3, 4} and another with {5, 6, 7, 8}. They are connected with the edge 5. Let's follow the previous procedure for constructing the initial chromosome.

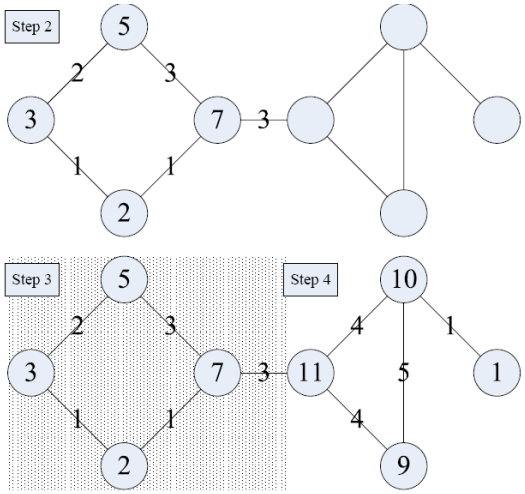

Fig. 3. The assignment of weights during different steps. Step 2 is only considering the left partition and Step 3 is combing with the assignment of Step 4 to become an overall assignment.

2.3. Refining the Evolution by Retaining Illegal Solutions

Because of the existence of complicated and enormous restrictions on the solutions of the problem, there are many illegal solutions generated during the evolution. These illegal solutions are not useless. However many of them can be re-used. Usually the approach taken punishment to the genetic algorithm was unable to distinguish the distance between the solutions to the legal solutions. That means the algorithm doesn't know which solutions are the better solutions. Unable to screen out bad quality solutions limits the potential of improving the search process.

Therefore this research studies the possibility of come out with the strategy of allowing illegal solutions such that the restriction of not generating illegal solutions can be released to some extent. However, deciding the characteristics of the solutions such that understanding its *potential* of being adjusted to a legal solution is not an easy task. This paper has working on the method for measuring the distance between the illegal solution and the safe area for this purpose.

2.4. Cut-points on Illegal Edges

The main reason for a solution to be an illegal solution and therefore violating the restrictions in ISP is resulted from the contradictions between the weights of the neighboring nodes which are constructed from the connecting edges. This research takes these properties for further exploration of the solutions by focusing on those edges that is causing the contradictions. The cut-points of the mutation or crossover will be chosen from these edges such that increasing the chances of fixing the solution.

For example, when there are two chromosomes [2 3 2 1 2 4 1 3 1] and [2 3 1 1 3 4 3 3 1], the weights of the nodes are (5 4 6 3 7 8 4 1) and (5 3 7 2 10 8 6 1), respectively. As shown in Figure 5., there are two nodes (both weighted 4) violating the restriction and the status of these two nodes will be recorded. This recording will cause the cut-point to be falling on the edges that are connecting to these nodes only.

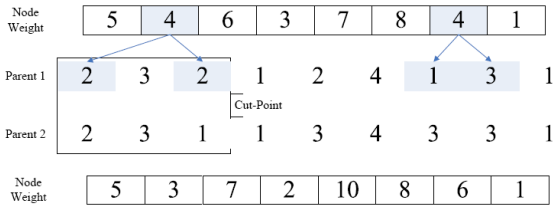

Fig. 4. The cut-point of the chromosome is decided by the contradiction nodes of the graph.

If crossover is applied onto our example by taking the cut-point on the third edge showing on the figure, the result will be two legal solutions.

3. The Experiment

The experiment mainly focusing on two issues: (1) differences of the initial solutions between ISP and IIGA, (2) the effect of allowing illegal solutions involved in the evolution.

Section 4.1 introduces the graphs employed for performing the experiment and Section 4.2 shows the deployment and the results of the experiments.

3.1. Graphs Used for Experiments

This experiment generates the graphs with proper clustering automatically for conducting the tests which are designed for realizing the behavior of the evolutions.

In this experiment, we have fixed the number of clusters to 3. The number of the nodes of each cluster is fixed at 17 and the number of edges is 200 in total (summing all three clusters). These randomly generated clusters are all connected with single edge.

For understanding the behavior of the algorithm when working on the ISP, this research tries to gather the data of working on four different cases of the graphs which are having different densities between clusters. The densities of the clusters are increasing with different slops.

3.2. Effectiveness of Generating Initial Solutions

Our experiment has been conducted on a workstation with Pentium 3.0 GHz CPU with 512MB ROM. The Borland C++ Builder has been used to implement the algorithm.

For comparing the speed and quality of creating an initial solution for the ISP, an experiment has been conducted on four different cases of the graphs. For the first case of the graphs (No. 1 in Table 1), the clusters inside have densities of 10%, 50% and 90%, represented as 10-50-90. The distributions of densities for other types of graphs are showing in the table with No. 2, 3, and 4.

Table 1. Comparing initial solutions and final solutions between IIGA and SGA.

No.	Densities of clusters	Percentages of illegal solutions allowed in the evolution procedure			
		0%	5%	10%	50%
1)	10-50-90	778.41	760.02	722.7	744.72
2)	20-50-80	824.41	756.4	725.57	742.17
3)	30-50-70	923.85	881.26	810.49	849.55
4)	40-50-60	1115.61	1025.82	944.53	1007.71

The experiment is monitoring the time and quality of performing the generation of 100 solutions for 100 times. It is observed that the IIGA has an advantage of using less time (in average) to generate solutions that are better than SGA.

3.3. Resolution without Control

The test allow illegal solutions to exist continuously during the evolution without giving any punishment to them as long as the solution has a lower score which means closer to a better answer of irregular sum.

As expected, shown in Table 2, the popularity distinguished (die out) in a short period after 6 to 7 generations. Doesn't matter what is the percentage of mutation, the crossover and mutation failed so many times that is close to overall failure. Compare to the results in the next subsection, the effectiveness of the IIGA can be manifested.

Table 2. The different mutation rates all resulted in early elimination during the evolution.

No.	Mutation rate	Die out generations	Crossover failed (Number of time)	Mutation failed (Number of time)
1	0	7.18	785.84	0
2	0.05	6.8	728.59	18.6
3	0.1	6.75	704.25	35.68

3.4. Probing the Percentages of Illegal Solutions Allowed

In order to effectively narrow down to useful chromosomes to attend the evolution process so that better solutions can be generated, this research adopted two approaches: (1) evaluate the number of nodes that are violating the restrictions, and (2) control, but not eliminate, the percentage of illegal solutions during the evolution.

All the experiments have been set up with 90% crossover rate and 10% mutation rate. One hundred chromosomes are employed and the generation of the evolution is bounded at 2000.

Table 3. Comparing the percentages of illegal solutions allowed in the evolution procedure.

No.	Densities of clusters	Initial Solutions		Final Solutions	
		IIGA	**SGA**	**IIGA**	**SGA**
1	10-50-90	3780.64	4603.3	722.7	801.93
.2	20-50-80	3766.84	4606.03	725.57	838.79
3	30-50-70	3471.00	4631.08	810.49	953.15
4	40-50-60	3277.79	4707.39	944.53	1143.63

In this experiment, we are trying to comprehend how different percentages of illegal solutions allowed in the population can differently affect the performance. Four different percentages of allowing illegal solutions are used for the experiment, which are 0%, 5%, 10% and 50%.

The results are obvious. The 10% percentage has the best score as comparing to other percentages (Table 3). That means the percentage should not be too small and should not be too large either.

3.5. IIGA vs. SGA

With these encouraging results, we have the opportunity to compare the final version of the IIGA with SGA. The performance of the IIGA is obviously better than the SGA, as showing in Table 4. For different type of

cluster densities, the initial solutions as well as the final solutions are all reaching a better outcome.

Table 4. Final results comparing IIGA with SGA

No.	Densities of clusters	IIGA		SGA	
		Average Score	Spend Time	Average Score	Spend Time
1	10-50-90	4222.54	267.55	5217.09	2383.48
2	20-50-80	4098.24	317.64	5229.35	2294.60
3	30-50-70	3849.05	357.23	5216.71	4989.10
4	40-50-60	3644.71	582.65	5331.42	27977.80

4. Conclusions

This research is working on the attempt of improving the effectiveness of generating the initial solution of ISP by decomposing the problem of ISP and the possibilities of searching for an ideal number of percentages of allowing illegal solutions to be continuously existed in the population of the evolution. The proposed generic algorithm, IIGA, behaves satisfactorily for the purpose of generating a solution that is comparatively better than the results of SGA.

5. References

[1] M. Aiger and E. Triesch, "Irregular assignments of trees and forests," *SIAM Journal of Discrete Mathematics*, vol. 3, pp. 439-449, 1990.

[2] Y. Alavi, G. Chartrand, F. R. K. Chung, P. Erdos, R. L. Graham and O. R. Oellermann, "Highly irregular graphs," *J. Graph Theory*, vol. 11, pp. 235-249, 1987.

[3] D. Amar, "Irregularity strength of regular graphs of large degree," *Discrete* Mathematics, vol. 114, pp. 9-17, 1993.

[4] T. Bohman and D. Kravitz, "On the irregularity strength of trees," *Journal of Graph Theory*, vol. 45, pp. 241-254, 2004.

[5] A. C. Burris, "The irregular coloring number of trees, *Discrete Mathematics*," vol. 141, pp. 279-283, 1995.

[6] G. Chartrand, M. S. Jacobsen, J. Lehel, O. R. Oellermann, S. Ruiz, and F. Saba, "Irregular Networks," *Congressus Numerantium*, vol. 64, pp. 187-192, 1988.

[7] Stephen Y. Chen and Stephen F. Smith, "Improving Genetic Algorithms by Search Space Reductions," Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), pp. 13-17, 1999.

[8] P. R. Christopher, "Highly irregular asymmetric graphs," *Congressus Numerantium*, vol. 91, pp. 93-97, 1992.

[9] L. Davis, "Order-based genetic algorithms and the graph coloring problem," In *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, pp. 72-90, 1991.

[10] G. Ebert, J. Hemmeter, F. Lazebnik, and A. Woldar, "On the number of irregular assignments on a graph," *Discrete Mathematics*, vol. 93 , pp. 131-142, 1991.

[11] R. Faudree, M. Jacobson, J. Lehel, and R. Schlp, "Irregular networks, regular graphs and integer matrices with distinct row and column sums," *Discrete Mathematics*, vol. 76, pp. 223-240, 1989.

[12] D. B. Fogel, Evolutionary computation: toward a new philosophy of machine intelligence. Piscataway, N J: Institute of Electrical and Electronics Engineers, 1995.

[13] A. Frieze, R. J. Gould, M. Karoński, and F. Pfender, "On graph irregularity strength," *Journal of Graph Theory*, vol. 41, pp. 120-137, 2002.

[14] D. E. Goldberg, Genetic algorithm in search, optimization and machine learning. Reading, MA: Addison-Wesley Publishing Co; 1989.

[15] A. Gyarfas, "The irregularity strength of $K_{m,m}$ is 4 for m odd," *Discrete Mathematics*, vol. 71, pp. 273-274, 1988.

[16] J. Holland, Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI, 1975.

[17] M. S. Jacobson, E. Kubicka, and G. Kubicki, "Irregularity Sum for graphs," *Vishwa International Journal of Graph Theory*, vol. 1, no. 2, pp. 159-175, 1992.

[18] D. Kravitz, "Investigation of the Irregularity Strength of Trees," Undergraduate thesis, University of Delaware, 2000.

[19] J. Petersen, "Die Theorie der regulären Graphen, " *Acta Math.*, vol. 15, pp. 193-220, 1891.

[20] N. Sannomiya, H. Iima, K. Ashizawa, Y. Kobayashi, "Application of genetic algorithm to a large-scale scheduling problem for a metal mold assembly process, " Proceedings of the 38th IEEE Conference on Decision and Control, pp. 2288 - 2293, 1999.

[21] L. Darrell Whitley, "An overview of evolutionary algorithms: practical issues and common pitfalls," *Information & Software Technology*, Vol. 43, Issue 14, pp. 817-831, 2001.