A Composition Algorithm for Very Hard Graph 3-Colorability Instances*

Seiichi Nishihara, Kazunori Mizuno, and Kohsuke Nishihara

Institute of Information Sciences and Electronics, University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
nishihara@is.tsukuba.ac.jp
mizuno@algor.is.tsukuba.ac.jp
pml01582@mail1.accsnet.ne.jp

1 Introduction

Graph colorability (COL) is a constraint satisfaction problem, which has been studied in the context of computational complexity and combinatorial search algorithms. It is also interesting as subjects of heuristics [2]. Many research reports discuss the complexity of COL [2,3,4,8,9,10]. Examples of possible candidates of order parameters that explain the mechanism making COLs very hard include the 3-paths [10], the minimal unsolvable subproblems [8], and the frozen developments [4]. Instead of generate-and-test approaches, we propose a constructive approach producing 3-colorablity problems (3COLs) that are exceptionally hard for usual backtracking algorithms adopting Brélaz heuristics and for Smallk coloring program [1]. Instances generated by our procedure (1) are 4-critical, (2) include no near-4-cliques(n4c's; 4-cliques with 1 edge removed) as subgraphs, and (3) have the degree of every node limited to 3 or 4: quasi-regular.

2 Graph 3-Colorability and 4-Critical Graphs

Let $G = (V, E)$ be a graph to be colored, where V and E corresponds to the set of vertices and edges. Let $n =\mid V \mid$ and $m =\mid E \mid$. An edge $(i, j) \in E$ has the constraint that prohibits assigning the same color to vertices, i and j. Phenomena similar to physical phase transitions are generally observed in COLs, where search cost follows an easy-hard-easy pattern as a function of constraint density, or $\gamma(= 2m/n)$. The region where median search cost becomes the most time-consuming lies very close to the cross-over point, at which half the instances are solvable and half unsolvable (primary PT). An interesting region also exists at a slightly lower constraint density than that of primary PT, in which exceptionally hard instances (EHIs) [5] tend to occur, although most are solved easily (secondary PT).

* This research was supported in part by the Ministry of Education, Culture, Sports, Science and Technology of Japan, Grant-in-Aid for Scientific Research (B)(2), No. 14380134, 2002–2005.

F. Rossi (Ed.): CP 2003, LNCS 2833, pp. 914–919, 2003.
© Springer-Verlag Berlin Heidelberg 2003

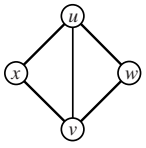
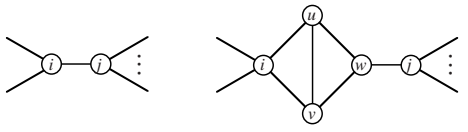


Fig. 1. An n4c.



(a) A 4-critical graph (part). (b) Embedding completed.

Fig. 2. Embedding operator embed_$K_4(i, j)$.

A hard non 3-colorable graph necessarily contains large 4-critical subgraphs [4,8], i.e., non 3-colorable but any proper subgraph is 3-colorable. K_4, 4-clique, is the smallest 4-critical graph because removing an arbitrary edge from it makes a 3-colorable graph, which we call an n4c (Fig. 1) [4]. The n4c contains an interesting constraint, constraint(x, w), that claims the colors for x and w must be the same. Let Fig. 2(a) be part of a 4-critical graph, where the degree of vertex i, deg(i), is 3. Introduce an operation, embed_ $K_4(i, j)$, where an n4c is added in place of edge (i, j) merging i and x and connecting j and w [1]. Starting with K_4 as the initial graph, arbitrarily large 4-critical instances are constructed by repeating embed_$K_4(i, j)$ recursively to meet the many known conditions EHIs may have to satisfy [10, 7, 4, 8].

3 Composition Algorithm for EHI without n4c's

Because embed_$K_4(i, j)$ always leaves an n4c in the graph, we can find at any stage of graph construction at least 1 n4c, which is the footprint where the latest embedding operation was executed. By repeating collapse, i.e., inverse operation of embed_$K_4(i, j)$, the given graph straightforwardly is reduced to a single K_4 that is unsolvable. To overcome this drawback, we introduce a set of original n4c-free 4-critical graphs independent of each other in that no graph is a subgraph of any other. We found 7 such graphs by trial and error (Fig. 3), in which each graph is termed MUG$_{nt}$, where MUG stands for "minimal unsolvable graph," n means the number of vertices included, and t is used to identify the type if necessary. Let us naturally extend the embedding operation to embed_MUG$_{nt}(i, j)$. These operations are the same as Hajós' join construction [6] except that both vertices to be merged should have the degree of 3.

Proposition 1 *When embed_MUG$_{nt}(i, j)$ is applied to a 4-critical graph, the result remains 4-critical.*

Proposition 2 *Quasi-regularity is maintained by embed_MUG$_{nt}$ operation where nt is 9, 10, 11a, 11b, or 12c.*

Proposition 3 *Let the graph to embed contain, m edges and $n = n_3 + n_4 + n_5$ vertices, where n_i is the number of vertices with degree i. The numbers of vertices*

[1] Note that 4-criticality is maintained because the constraint, constraint(i, w), remains after embedding while u and v are not adjacent to other vertices.

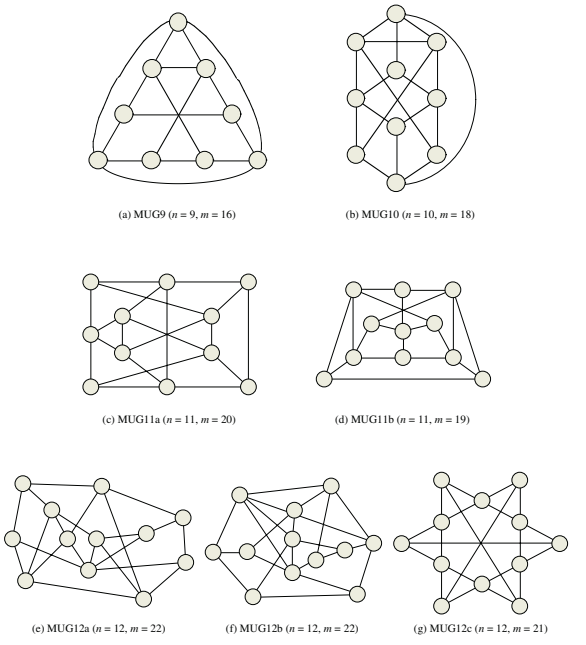


Fig. 3. 4-critical n4c-free graphs.

with degrees 3,4,5 increase by $n_3 - 2, n_4 + 1, n_5$. *The total number of vertices increases by* $n - 1$, *and edges by* $m - 1$.

Starting with a 4-critical graph, we construct arbitrarily large 4-critical graphs, i.e., including an arbitrary number of vertices, by repeating embedding. Fig. 4 gives the procedure "graph-generator(k)" which repeats embedding operations k times randomly. When we start with 1 of 7 graphs (Fig. 3), we produce graphs contain no n4c's. Further, if a quasi-regular graph is assigned initially to G_{init} at (1) in Fig. 4, and both MUG12a and MUG12b are excluded from candidates at (2), then the graph-generator produces quasi-regular graphs.

4 Experiments and Discussion

We test the difficulty of 3COL instances generated by "graph-generator(k)" where all graphs except for MUG12a and MUG12b are used to generate quasi-regular graphs. For 8 cases from $k = 5$ to $k = 12$, 100 instances are generated for each case, i.e., a total of 800 generated instances. These instances are applied to the backtracking algorithm with Brélaz heuristics and the Smallk coloring program. In the Brélaz algorithm, only 500 instances from $k = 5$ to $k = 9$ are used for testing. These algorithms are implemented in C on a PC with 1 GHz of Pentium III and 512 Mbytes of RAM. Fig. 5 gives results for search costs

```
procedure graph-generator(k)
begin
  input an initial graph G_init;                                    (1)
  G := G_init;
  for w := 1 to k do
    choose randomly an edge(i, j) ∈ E(G) where deg(i)≤ 3;
    choose randomly MUG_nt, (nt =9, 10, 11a, 11b, 12a, 12b, or 12c);  (2)
    embed_MUG_nt(i, j);
  end for;
end.
procedure embed_MUG_nt(i, j)
begin
  choose randomly an edge (x, y) ∈ E(MUG_nt) where deg(x)≤ 3;
  remove edges (i, j) and (x, y);
  add an edge (j, y);
  merge x with i;
end.
```

Fig. 4. 3COL instance generator.

and CPU time, where "average line" shows the variation in average search cost and CPU time for each k as a function of the average number of vertices for each k. Smallk is more sophisticated than the Brélaz algorithm, but both search cost and CPU time clearly exhibit exponential growth[2]. We also conduct experiments on randomly generated instances. For 33 cases from $\gamma = 3.0$ to $\gamma = 5.0$ at the intervals of 0.2 in $n = 100$, 200, and 300, 10,000 instances are randomly generated for each cases, i.e., a total of 3.3 million generated instances, each of which is solved using Smallk. In the Brélaz algorithm, only 1.1 million instances with $n = 100$ are used. It is obvious that the hardness of our instance set cannot be compared with that of the huge set of random instances (Fig. 6)[3].

Experiments confirmed that our method stably produces EHIs whose computational cost is of an exponential order of n. Researchers adopting generate-and-test approaches found that conditions under which EHIs tend to occur are as follows: (1) Their constraint density is near the secondary PT region [7], (2) the smallest minimal unsolvable subproblem is very large compared to the instance size [8], and (3) their structure is homogeneous, i.e., quasi-regular [10]. It seems reasonable that instances produced by our method meet all these conditions. Because our instances contain no n4c's, most frozen pairs [4] are hidden from the surface, which makes our instances hard to solve even for sophisticated algorithms such as Smallk. We still do not know theoretically why our instances become EHIs. The ultimate question may be whether our instances are inher-

[2] As long as we see results of Culberson and Gent in [4], our instances seem to be much harder than their threshold graphs, although the complexity of their graphs also exhibit exponential growth.

[3] In Smallk, it is only 2.6 *sec.* and 103 *sec.* to determine the colorability of each hardest random instance with *200* and *300* vertices at $\gamma = 4.8$, whereas it requires more than 500 *sec.* on average in solving our instances with even *100* vertices or so in Fig. 5.

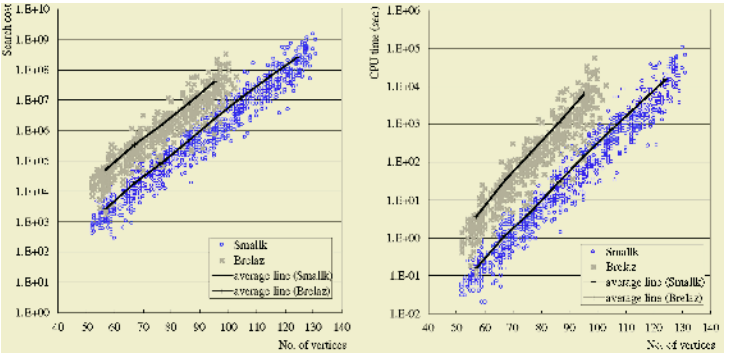


Fig. 5. Experimental results on 3COL instances generated by our procedure.

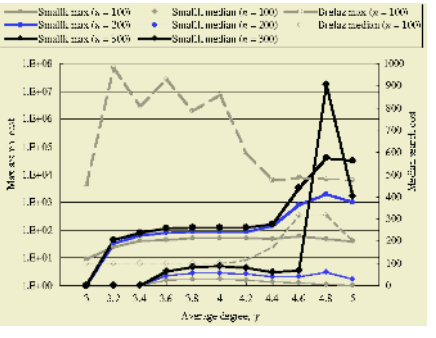


Fig. 6. Experimental results on randomly generated instances.

ently hard for any search algorithms. Let us move on to an issue probably related to heuristics. Fig. 3 introduces only 7 n4c-free MUGs independent of each other. Although we surmise that the number of such graphs is infinite, we still do not know how to generate them systematically. The method for producing such graphs may be necessary for hiding the structural weakness of an instance so that no clever heuristics can find and exploit it.

5 Conclusions

We have proposed a constructive algorithm to generate EHIs of 3COL, which recursively repeat self-embedding operations of MUGs. The EHIs generated are 4-critical and contain no n4c's, to hide a structural weakness that heuristics would be able to exploit. Using Brélaz heuristics and Smallk, we showed that the complexity of 3COL instances generated by our algorithm is an exponential order of the number of vertices. We plan to develop a systematic method to arbitrarily produce many MUGs independent of each other to construct hard instances, and to clarify whether heuristics exist that cope with these instances.

References

1. *Overview of the Smallk Graph Coloring Program*, 2000. http://www.cs.ualberta.ca/ joe/Coloring/Colorsrc/smallk.html.
2. D. Brélaz. New Methods to Color the Vertices of a Graph. *Comm. ACM*, 22(4), 1979.
3. P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where Really Hard Problems Are. In *Proc. 12th IJCAI*, 1991.
4. J. Culberson and I. Gent. Frozen development in graph coloring. *Theor. Computer Sci.*, 265, 2001.
5. S. A. Grant and B. M. Smith. Modelling Exceptionally Hard Constraint Satisfaction Problems. In *Proc. CP97*, 1997.
6. D. Hanson, G. C. Robinson, and B.Toft. Remarks on the Graph Colour Theorem of Hajós. *Congressus Numerantium*, 55, 1986.
7. T. Hogg and C. P. Williams. The hardest constraint problems: a double phase transition. *Artif. Inell.*, 69, 1994.
8. D. L. Mammen and T. Hogg. A New Look at Easy-Hard-Easy Pattern of Combinatorial Search Difficulty. *Jour. Artif. Intell. Research*, 7, 1997.
9. K. Mizuno and S. Nishihara. Toward Ordered Generation of Exceptionally Hard Instances for Graph 3-Colorability. In *Computational Symposium on Graph Coloring and its Generalizations(COLOR02), Ithaca, N.Y., Sept.*, 2002.
10. D. R. Vlasie. Systematic Generation of Very Hard Cases for Graph 3-Colorability. In *Proc. 7th IEEE ICTAI*, 1995.