An Iterated Backtrack-based Removal Approach for Finding k-vertex-critical Subgraph

Wen Sun[1], Jin-Kao Hao[1,2]

[1] LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
[2] Institut Universitaire de France, Paris, France

Mots-clés : *k-VCS ; removal algorithm ; heuristics ; graph coloring.*

1 Introduction

Given an undirected graph $G = (V, E)$ and a positive integer k, a k-vertex-critical subgraph (k-VCS) of G is a subgraph H such that its chromatic number equals k (i.e., $\chi(H) = k$), and removing any vertex causes a decrease of $\chi(H)$. The k-VCS problem (k-VCSP) is to find the smallest k-vertex-critical subgraph H^* of G. Using increasing values of k, the associated k-VCS can be used to determine improved lower bounds on the chromatic number of graph G.

k-VCSP is a NP-hard problem [2, 3] and thus computationally challenging. In addition to its application in graph coloring, k-VCSP has a number of other applications like analysis in infeasible linear programs [1] and solving the set covering problem [4]. To handle this difficult problem, heuristics constitute a popular approach to find near-optimal solutions in an acceptable computing time.

2 A iterated backtrack-based removal approach for k-VCSP

This paper proposes an iterated backtrack-based removal (IBR) heuristic to find k-VCS as small as possible for a given graph G. IBR extends a classical vertex removal procedure [3] by a backtracking scheme and a perturbation procedure. The removal procedure removes uncritical vertices of the given G. The backtracking phase modifies the current subgraph by adding back some removed vertices, when the chromatic number of the current subgraph H is less than k. The perturbation strategy is used to jump out of local optimum traps by randomly changing some vertices of the current subgraph H.

3 Computational results

Computational results of the proposed approach on a set of 44 well-known DIMACS benchmarks are presented in Table 1. We compare our IBR algorithm with the state-of-the-art algorithm ($Ins + h$) proposed in [3]. From Table 1, we observe that for 9 out of the 44 instances, our algorithm obtains improved best-known solutions (column IBR vs. column $Ins + h$, k-VCS with the same k and smaller $|E|$, or k-VCS with a larger k). For the 35 remaining instances, our algorithm obtains the best results (known to be optimal). These results demonstrate thus the effectiveness of the proposed algorithm for the tested instances. We will verify the performance of IBR on additional and larger DIMACS instances.

Références

[1] J.W. Chinneck. Finding a useful subset of constraints for analysis in an infeasible linear program. *INFORMS Journal on Computing*, 9(2) : 164–174, 1997.

[2] Nilotpal Chakravarti. Some results concerning post-infeasibility analysis. *European Journal of Operational Research*, 73(1) :139–143, 1994.

[3] C. Desrosiers, P. Galinier, and A. Hertz. Efficient algorithms for finding critical subgraphs. *Discrete Applied Mathematics*, 156(2) : 244–266, 2008.

[4] P. Galinier and A. Hertz. Solution techniques for the large set covering problem. *Discrete Applied Mathematics*, 155(3) : 312–326, 2007.

TAB. 1: Computational results of IBR in comparison with the results of the best reference algorithm ($Ins + h$) [3] on 44 DIMACS benchmark instances. Improved results are indicated in bold compared to the previous best results.

Instance				$Ins + h$ [3]				IBR			
name	$\|V\|$	$\|E\|$	k_UB	k	$\|V\|$	$\|E\|$	$time(s)$	k	$\|V\|$	$\|E\|$	$time(s)$
myciel5	47	236	6*	6	47	236	0.1	6	47	236	0.00
myciel7	191	2360	8*	8	191	2360	30.65	8	191	2360	11.46
mug88_1	88	146	4*	4	88	146	0.05	4	88	146	0.03
mug100_25	100	166	4*	4	100	166	0.05	4	100	166	0.03
1_Insertion_4	67	232	5*	5	67	232	0.05	5	67	232	0.3
4_Insertion_4	475	1795	5	5	475	1795	0.75	5	475	1795	0.95
le450_5a	450	5714	5*	5	5	10	5.35	5	5	10	2.24
le450_25d	450	17425	25*	25	25	300	8.95	25	25	300	6.65
school1	385	19,095	14*	14	14	91	6.25	14	14	91	1.94
miles250	128	387	8*	8	8	28	0.1	8	8	28	0.04
miles1500	128	5198	73*	73	73	2628	1.6	73	73	2628	0.64
anna	138	493	11*	11	11	55	0.15	11	11	55	0.07
david	87	406	11*	11	11	55	0.15	11	11	55	0.04
homer	561	1629	13*	13	13	78	0.4	13	13	78	0.23
huck	74	301	11*	11	11	55	0.1	11	11	55	0.04
jean	80	254	10*	10	10	45	7.1	10	10	45	0.09
games120	120	638	9*	9	9	36	0.2	9	9	36	0.07
fpsol2.i.1	496	11654	65*	65	65	2080	104.95	65	65	2080	21.93
fpsol2.i.3	451	8691	30*	30	30	435	26.25	30	30	435	4.34
mulsol.i.1	197	3925	49*	49	49	1176	2.2	49	49	1176	0.47
mulsol.i.5	186	3973	31*	31	31	465	3.45	31	31	465	1.86
zeroin.i.1	211	4100	49*	49	49	1176	2.2	49	49	1176	0.45
zeroin.i.3	206	3540	30*	30	30	435	2.8	30	30	435	1.68
inithx.i.1	864	18707	54*	54	54	1431	448.85	54	54	1431	38.13
inithx.i.3	621	13969	31*	31	31	465	7.1	31	31	465	1.9
DSJC125.1	125	736	5*	5	10	26	0.4	5	10	26	2.29
DSJC125.5	125	3891	17*	14	70	1341	46.35	14	**66**	**1266**	11.69
								15	**82**	**1862**	35.4
DSJC250.1	250	3218	8	6	64	362	27.65	6	**51**	**290**	21.45
								7	**120**	**983**	134
DSJC250.5	250	15,668	28	14	74	1505	59.6	14	**50**	**836**	32.54
								15	**63**	**1265**	31.08
								16	**75**	**1742**	43.92
DSJC500.1	500	12,458	12*	6	65	369	73.15	6	**32**	**159**	48.23
								7	**79**	**617**	353.06
DSJR500.1	500	3555	12*	12	12	66	1.9	12	12	66	41.23
DSJR500.1C	500	121,275	85*	80	84	3477	710.75	80	**80**	**3160**	153.96
								82	**82**	**3321**	634.89
								83	**83**	**3403**	1280.19
DSJR500.5	500	58,862	122	90	90	4005	373.6	90	90	4005	15.78
								119	**119**	**7,021**	29.08
queen8_8	64	728	9*	9	54	538	12.6	9	**53**	**519**	2.6
queen9_9	81	2112	10*	10	74	897	13.8	10	**72**	**869**	920.45
ash331GPIA	662	4185	4	4	9	16	1.6	4	**7**	**12**	287.08
1-FullIns_3	30	100	4	4	7	12	0.1	4	7	12	0.09
1-FullIns_5	282	3247	6	6	31	144	7.3	6	31	144	2.10
2-FullIns_3	52	201	5	5	9	22	0.1	5	9	22	0.09
2-FullIns_5	852	12,201	7	7	39	244	13.3	7	39	244	22.4
3-FullIns_3	80	346	6	6	11	35	0.15	6	11	35	0.15
3-FullIns_5	2030	33,751	8	8	47	371	78.6	8	47	371	396.26
4-FullIns_3	114	541	7	7	13	51	0.2	7	13	51	0.25
5-FullIns_3	154	792	8	8	15	70	1.2	8	15	70	0.47