An incremental search heuristic for coloring vertices of a graph

Subhankar Ghosal and Sasthi C Ghosh

Advance Computing and Microelectronics Unit
Indian Statistical Institute
Kolkata, India

September 10, 2020

Overview

Graph Coloring Problem (GCP)

- Given a graph $G(V, E)$ we have to assign color $c(v)$ to each vertex v such that $c(u) \neq c(v)$ if $uv \in E$. We have to minimize total number of color $span(C)$ used in color vector $C = (c(v))$.
- It is known that graph coloring problem is NP-Complete and even providing $n^{1/\epsilon}$ approximation of it is NP-hard (Zuckerman et al).
- Greedy coloring is a natural choice for solving this problem where following an order S we visit the vertices and while visiting a vertex v we put the minimum color that is absent in all of its already allocated neighbor.
- Since there is an optimum order among the $n!$ orders, where $n = |V|$, performance of greedy coloring is entirely order dependent.
- There are some good orders like largest degree fast, Dsature, Kempe order etc.

Heuristics to solve GCP

- Since solving GCP is NP-Complete authors applied simulated annealing, genetic algorithms, hill climbing, memetic-algorithm and tabu-search heuristic.
- It is evident that all of the above mentioned heuristic algorithms may eventually reach to a poorer order than the current best order during the search process.
- In this paper we propose a selective search (SS) heuristic *which never search a worse order than the current best order.* This is the key difference of this algorithm from others.
- Using SS we propose a incremental search heuristic (ISH) in which we call SS with random initial orders.

Definition (Equivalent order)

Let S_1 and S_2 be two orders of the vertices of G. Let C_1 and C_2 be the two color vectors obtained by applying greedy coloring on G following S_1 and S_2 respectively. Then orders S_1 and S_2 are said to be equivalent to each other if $C_1 = C_2$.

Definition

Let S_1 and S_2 be two orders of the vertices of G. Let C_1 and C_2 be the two color vectors obtained by applying greedy coloring on G following S_1 and S_2 respectively. Then order $S_1 \triangleleft S_2$ if and only if $span(C_1) \leq span(C_2)$.

Definition (Pseudo-vertex)

Let S be an order of the vertices of G and C be its color vector obtained by applying greedy coloring on G following S. Let k be the span of C. A pseudo-vertex V_i is a subset of vertices of G all of which get color i in C, where $1 \leq i \leq k$.

Let S be an order and $V_1, V_2, \cdots, V_k$ be the k pseudo-vertices obtained by applying greedy coloring on G following S, where $n_1 = |V_1|$, $n_2 = |V_2|$, $\cdots$, $n_k = |V_k|$. Let $\Pi(S)$ be the set of all permutation of $V_1, V_2, \cdots, V_k$, and $\pi = (V_{l_1}, V_{l_2}, \cdots, V_{l_k}) \in \Pi(S)$. Let $L(\pi)$ be the set of all orders generated from π by permuting vertices within the same pseudo-vertex but keeping the order of pseudo-vertex intact. Then all orders in $L(\pi)$ are considered as represented by π.Let $N_k = |L(\pi)| = n_1!n_2! \cdots n_k!$.

Theorem

Let S be an order and $V_1, V_2, \cdots, V_k$ be the k pseudo-vertices obtained by applying greedy coloring on G following S. Let $\pi = (V_{l_1}, V_{l_2}, \cdots, V_{l_k}) \in \Pi(S)$. All orders in $L(\pi)$ are equivalent to each other.

Definition (Cardinal order)

An order $S = (v_{l_1}, v_{l_2}, \cdots, v_{l_n})$ of the vertices of G is said to be a cardinal order if we apply greedy coloring on G following S and get color vector C such that $c(v_{l_1}) \leq c(v_{l_2}) \leq \cdots \leq c(v_{l_n})$, where $c(v_{l_k})$ denotes the color of v_{l_k}.

Theorem

Let S be an order and $V_1, V_2, \cdots, V_k$ be the k pseudo-vertices obtained by applying greedy coloring on G following S. If $\pi \in \Pi(S)$, each $S' \in L(\pi)$ is $S' \lhd S$.

Theorem

Given any coloring C' of G we could generate an order S by sorting the vertices of G according to ascending order of their colors in C'. If we apply greedy coloring on G following S, we will get a color vector C such that $span(C) \leq span(C')$.

Algorithm 1: *Selective Search (SS) Algorithm*

Input: G,C, ρ
Output: C

Find S by sorting the vertices according to ascending order of their colors in C;
Apply greedy coloring on G following S and generate color vector C';
Set $C = C'$;
Set $k = span(C)$;
Find pseudo-vertices $V_1, V_2, \cdots, V_k$ of C;
for $i = 1, 2, \cdots \rho$ **do**
    Let $\pi \in \Pi(S)$;
    Let $S' \in L(\pi)$;
    Apply greedy coloring on G following S' and find color vector C';
    if $span(C') < span(C)$ **then**
        $C = C'$;
        Set $k = span(C)$;
        Find pseudo-vertices $V_1, V_2, \cdots, V_k$ of C;
        Reset $i = 1$;
return C

Algorithm 2: *Incremental Search Heuristic (ISH)*

Input: G, ρ_1, ρ_2
Output: C_{min}

Set $C_{min} = (1, 2, \cdots, |V(G)|)$;
for $i = 1, 2, \cdots, \rho_1$ **do**
    Generate a random order S of the vertices;
    Apply greedy coloring on G following O and find color vector C;
    $C = SS(G, C, \rho_2)$;
    if $span(C) < span(C_{min})$ **then**
        $C_{min} = C$;
        Reset $i = 1$;
return C_{min};

Theorem

If ISH produces span less than or equals to k in each step, then expected number of steps to find optimum is $\mathbb{E}[\rho_1\rho_2] = O(\frac{1}{k!}(\frac{ek^2}{n})^n)$.

Corollary

If $k \leq \sqrt{\frac{n}{e}}$ *then* $\mathbb{E}[\rho_1\rho_2] = O(1)$.

Corollary

If $\Delta(G) + 1 \leq \sqrt{\frac{n}{e}}$, *then* $\mathbb{E}[\rho_1\rho_2] = O(1)$.

It is a well-known fact that solving the graph coloring problem even for sparse graph is hard. It is known that even coloring a planer graph with maximum degree just 4 is NP-Complete. Using ISH we can solve these sparse graph instances in expected polynomial time. Thus *any NP-Complete problem could be solved in expected polynomial time.*

Theorem

In Erdos-Reyni random graph $G(n, p)$*, if average degree*

$$\Delta_a = np \leq_n \sqrt{\frac{n}{e}}$$

then $\mathbb{E}[\rho_1 \rho_2] = O(1)$.

Benchmark Instances	n	e	DBG [?]		MCOACOL [?]		Best of [?]		DR [?]		EBDA [?]		ISH	
			χ	T	χ	T	χ	T	χ	T	χ	T	χ	T
1-FullIns-3	30	100	–	–	4	0.4*s*	–	–	–	–	–	–	4	0.000077*s*
1-Insertion-4	67	232	–	–	5	0.5*s*	–	–	–	–	5	2.63*s*	5	0.000085*s*
2-FullIns-3	52	201	–	–	5	0.4*s*	–	–	–	–	–	–	5	0.000061*s*
2-Insertions-3	37	72	4	2.23*s*	4	0.4*s*	–	–	–	–	4	2.08*s*	4	0.000093*s*
3-Insertions-3	56	110	4	3.37*s*	4	0.5*s*	–	–	–	–	4	3.49*s*	4	0.00006*s*
anna	138	493	11	11.63*s*	11	0.8*s*	–	–	11	1.04*s*	11	2.53*s*	11	0.000395*s*
ash331GPIA	662	4181	–	–	5	42.2*s*	4	**3.29s**	–	–	–	–	4	21.3364*s*
david	87	406	11	10.89*s*	11	0.5*s*	–	–	11	0.26*s*	11	2.61*s*	11	0.000095*s*
fpsol2.i.1	496	11654	–	–	65	3.6*s*	65	82.03*s*	65	463.94*s*	–	–	65	0.002455*s*
fpsol2.i.2	451	8691	30	69.79*s*	30	7.4*s*	–	–	30	495.94*s*	–	–	30	0.002805*s*
fpsol2.i.3	425	8688	30	67.14*s*	30	6.4*s*	–	–	30	480.27*s*	–	–	30	0.002604*s*
games120	120	638	9	10.77*s*	9	0.8*s*	–	–	9	0.24*s*	9	3.24*s*	9	0.000176*s*
homer	561	1628	–	–	13	4.3*s*	–	–	13	59.01*s*	13	6.67*s*	13	0.00333*s*
huck	74	301	11	11.10*s*	11	0.5*s*	–	–	11	0.11*s*	11	4.78*s*	11	0.000075*s*
inithx.i.1	864	18707	–	–	54	19.4*s*	–	–	54	2443.43*s*	54	93.14*s*	54	0.0076*s*
inithx.i.2	645	13979	–	–	31	9*s*	–	–	31	1500.45*s*	31	39.16*s*	31	0.00429*s*
inithx.i.3	621	13969	–	–	31	6*s*	–	–	31	1432.43*s*	31	42.71*s*	31	0.00321*s*
jean	80	254	10	9.92*s*	10	0.5*s*	–	–	10	0.13*s*	10	4.42*s*	10	0.000082*s*
le450-25a	450	8260	–	–	25	4.4*s*	–	–	25	68.93*s*	25	3.65*s*	25	0.0028*s*
le450-25b	450	8263	–	–	25	7*s*	–	–	25	65.82*s*	25	5.58*s*	25	0.0025*s*
le450-5a	450	5714	–	–	–	–	5	**21.17s**	5	82.47*s*	–	–	5	32.7469*s*
le450-5b	450	5734	–	–	–	–	5	140.16*s*	5	238.33*s*	–	–	5	54.597*s*
le450-5c	450	9803	–	–	6	279*s*	–	–	5	68.68*s*	–	–	5	5.04865*s*
le450-5d	450	9757	–	–	7	80.8*s*	–	–	5	44.49*s*	–	–	5	7.0381*s*
miles1000	128	3216	–	–	42	1*s*	–	–	42	2.43*s*	42	5.73*s*	42	0.007641*s*
miles1500	128	5198	–	–	73	1.2*s*	–	–	73	24.65*s*	73	9.34*s*	73	0.0006*s*
miles250	128	387	8	4.39*s*	8	1.1*s*	–	–	8	0.4*s*	8	4.52*s*	8	0.0007*s*
miles500	128	1170	20	14.48*s*	20	1.2*s*	–	–	20	1.07*s*	20	5.11*s*	20	0.0008*s*
miles750	128	2113	–	–	31	1.5*s*	–	–	31	2.54*s*	31	5.64*s*	31	0.013746*s*
mug88-1	88	146	4	4.05*s*	4	1.1*s*	–	–	4	0.05*s*	4	2.46*s*	4	0.0002*s*
mug88-25	88	146	4	3.67*s*	4	1.3*s*	–	–	4	0.05*s*	4	1.65*s*	4	0.00038*s*
mulsol.i.1	197	3925	49	25.75*s*	49	1.2*s*	–	–	49	18.79*s*	49	9.68*s*	49	0.0015*s*
mulsol.i.2	188	3885	31	22.07*s*	31	1.1*s*	–	–	31	63.18*s*	31	7.43*s*	31	0.00028*s*
mulsol.i.3	184	3916	31	23.49*s*	31	1.3*s*	–	–	31	55.88*s*	31	6.49*s*	31	0.0014*s*
mulsol.i.4	185	3946	31	25.22*s*	31	1.7*s*	–	–	31	60.71*s*	31	6.75*s*	31	0.00022*s*
queen8-12	96	1368	–	–	13	1.1*s*	–	–	–	–	–	–	12	0.699735*s*
queen8-8	64	728	9	9.87*s*	10	**1.4s**	–	–	–	–	10	10.24*s*	9	4.66952*s*
queen9-9	81	1056	10	13.96*s*	11	94*s*	–	–	–	–	–	–	10	9.80493*s*
r125.1	125	209	–	–	5	1.2*s*	–	–	–	–	–	–	5	0.000265*s*
r125.1c	125	7501	–	–	46	1.8*s*	–	–	–	–	–	–	46	0.000565*s*
r125.5	125	3838	–	–	37	6.7*s*	–	–	–	–	–	–	36	0.328828*s*
r250.1	250	867	–	–	8	2.4*s*	–	–	–	–	–	–	8	0.001849*s*
r250.1c	250	30227	–	–	64	5*s*	–	–	–	–	–	–	64	0.012621*s*
			χ	T	χ	T	χ	T	χ	T	χ	T		
Greater			0	19	7	40	0	1	0	29	1	26		
Equal			19	0	34	0	3	0	29	0	25	0		
Smaller			0	0	0	1	0	2	0	0	0	0		

Benchmark Instances	n	e	DBG [?]		MCOACOL [?]		Best of [?]		DR [?]		EBDA [?]		ISH	
			χ	T	χ	T	χ	T	χ	T	χ	T	χ	T
mug100-1	100	166	4	8.34*s*	4	0.8*s*	4	0.13*s*	4	0.07*s*	4	2.98*s*	4	0.00007*s*
mug100-25	100	166	4	8.21*s*	4	0.5*s*	4	0.31*s*	4	0.06*s*	4	2.57*s*	4	0.000391*s*
school1	385	19095	–	–	14	1934.2*s*	–	–	–	–	–	–	14	0.644538*s*
qg.order40	1600	62400	–	–	–	–	40	534.83*s*	–	–	–	–	40	0.582656*s*
wap05a	905	43081	–	–	50	12.4*s*	50	125.45*s*	–	–	–	–	50	0.022395*s*
zeroin.i.3	206	3540	30	27.06*s*	30	1.6*s*	–	–	30	34.51*s*	30	9.87*s*	30	0.001792*s*
school1-nsh	352	14612	–	–	14	1352.6*s*	14	12.76*s*	–	–	–	–	14	0.656823*s*
DSJR500.5	500	58862	124	**728s**	–	–	**122**	**572.01s**	–	–	**122**	**587s**	123	4372.94*s*
DSJR500.1c	500	121275	85	581*s*	–	–	85	0.33*s*	–	–	–	–	85	0.05701*s*
DSJC125.5	125	3891	**17**	19.71*s*	–	–	20	*h*	–	–	–	–	18	5.58092*s*
DSJC125.9	125	6961	44	35.87*s*	–	–	44	*h*	–	–	–	–	44	3.34038*s*
DSJC250.9	250	27897	**72**	**75.81s**	–	–	–	–	–	–	**72**	63.28*s*	73	184.649*s*
queen10-10	100	2940	–	–	12	**9.6s**	12	*h*	–	–	–	–	11	223.45*s*
queen11-11	121	3960	–	–	14	**2.4s**	13	**h**	–	–	–	–	12	5748*s*
queen12-12	144	5192	–	–	15	**2.8s**	–	–	–	–	–	–	14	48.5491*s*
queen13-13	169	6656	–	–	16	**9.3s**	–	–	–	–	–	–	14	19011*s*
queen14-14	196	4186	–	–	17	**22.2s**	–	–	–	–	–	–	16	3933*s*
queen15-15	225	5180	–	–	18	**17.9s**	–	–	–	–	–	–	17	177438*s*
le450-15b	450	8169	–	–	–	–	**15**	700.50*s*	–	–	**15**	4.23*s*	16	0.54667*s*
le450-15c	450	16680	15	**93s**	–	–	25	**h**	–	–	–	–	15	12758.4*s*
le450-15d	450	16750	15	**228s**	–	–	26	**h**	–	–	–	–	15	23125.2*s*
le450-25c	450	17343	**25**	**740s**	–	–	30	**h**	–	∞	–	–	27	9213.9*s*
le450-25d	450	17425	**25**	**382s**	–	–	30	**h**	–	∞	–	–	27	52432*s*
1-FullIns-5	282	3247	–	–	6	1.9*s*	6	1.54*s*	–	–	–	–	6	0.002752*s*
2-FullIns-4	212	1621	–	–	6	1.2*s*	6	0.01*s*	–	–	–	–	6	0.000599*s*
2-FullIns-5	852	12201	–	–	7	10.7*s*	7	5.02*s*	–	–	–	–	7	0.0043*s*
3-FullIns-4	405	3524	–	–	7	1.6*s*	7	0.02*s*	–	–	–	–	7	0.001365*s*
4-FullIns-4	690	6650	–	–	8	7.7*s*	8	0.02*s*	–	–	–	–	8	0.004098*s*
5-FullIns-4	1085	11395	–	–	9	28*s*	9	0.04*s*	–	–	–	–	9	0.010039*s*
1-Insertions-5	202	1227	–	–	6	1.2*s*	–	–	–	–	–	–	6	0.0004*s*
1-Insertions-6	607	6337	–	–	7	8.1*s*	–	–	–	–	–	–	7	0.003357*s*
2-Insertions-4	149	541	–	–	5	1.1*s*	–	–	–	–	5	5.71*s*	5	0.00021*s*
2-Insertions-5	597	3936	–	–	6	6.5*s*	–	–	–	–	–	–	6	0.003145*s*
3-Insertions-4	281	1046	–	–	5	2.1*s*	–	–	–	–	–	–	5	0.000739*s*
3-Insertions-5	1406	9695	–	–	6	45*s*	–	–	–	–	–	–	6	0.018598*s*
4-Insertions-3	79	156	–	–	4	0.6*s*	–	–	–	–	4	3.72*s*	4	0.000069*s*
4-Insertions-4	475	1795	–	–	5	3.7*s*	–	–	–	–	–	–	5	0.005*s*
qg.order60	3600	212400	–	–	–	–	62	**h**	–	–	–	–	60	257186*s*
DSJC250.1	250	3218	**8**	26.07*s*	**8**	**3s**	–	–	–	–	**8**	**5.77s**	9	19.19942*s*
DSJC250.5	250	15668	**28**	**89s**	33	**600s**	–	–	–	–	30	**80.12s**	29	12511*s*
will199GPIA	701	6772	–	–	8	21.8*s*	7	6.68*s*	–	–	–	–	7	1.38738*s*
zeroin.i.1	211	4100	49	28.24*s*	49	1.7*s*	–	–	49	27.1*s*	49	7.74*s*	49	0.002171*s*
zeroin.i.2	211	3541	30	83.39*s*	30	2.1*s*	–	–	30	39.08*s*	30	15.11*s*	30	0.001804*s*
			χ	T	χ	T	χ	T	χ	T	χ	T		
Greater			1	9	8	23	8	17	2	7	1	9		
Equal			9	0	22	0	14	0	5	0	7	0		
Smaller			6	7	1	8	2	7	0	0	4	3		

Table: Span and time requirement of different algorithms on benchmark instances

Questions?

Thank you