CSE 6140 Team D

Shengyun Peng, Yuan Ma, Qiang Wang

2015/11/02

Approximation[1]

- select a vertex u of maximum degrees and add it to VC set
- subgraph: uncovered edges and set V without VC set we have chosen
- select a vertex of maximum degree in the subgraph and add it to VC set
- repeat above step until E is empty

Algorithm 1: Maximum Degree Greedy (MDG)

Data: a graph $G = (V, E)$
Result: a vertex cover of G
$C \leftarrow \emptyset$;
while $E \neq \emptyset$ **do**
    select a vertex u of maximum degree;
    $V \leftarrow V - \{u\}$;
    $C \leftarrow C \cup \{u\}$;
end
return C;

Local Search - FastVC[2]

Algorithm 1: FastVC (G, *cutoff*)

Input: graph $G = (V, E)$, the *cutoff* time
Output: vertex cover of G

$C := ConstructVC()$;
$gain(v) := 0$ for each vertex $v \notin C$;
while *elapsed time* $<$ *cutoff* **do**
    if C *covers all edges* **then**
        $C^* := C$;
        remove a vertex with minimum $loss$ from C;
        continue;
    $u := ChooseRmVertex(C)$;
    $C := C \backslash \{u\}$;
    $e :=$ a random uncovered edge;
    $v :=$ the endpoint of e with greater $gain$, breaking ties in favor of the older one;
    $C := C \cup \{v\}$;
return C^*;

Algorithm 2: ConstructVC (G)

Input: graph $G = (V, E)$
Output: vertex cover of G

$C := \emptyset$;
//extend C to cover all edges
foreach $e \in E$ **do**
    if *e is uncovered* **then**
        add the endpoint of e with higher degree into C;
//calculate $loss$ of vertices in C
$loss(v) := 0$ for each $v \in C$;
foreach $e \in E$ **do**
    if *only one endpoint of e belongs to C* **then**
        for the endpoint $v \in C$, $loss(v)$++;
//remove redundant vertices
foreach $v \in C$ **do**
    if $loss(v) = 0$ **then**
        $C := C \backslash \{v\}$, update $loss$ of vertices in $N(v)$;
return C;

Algorithm 3: Best from Multiple Selection (BMS) Heuristic

Input: A set S, a parameter k, a comparison function f
//assume f is a function such that we say an element is better than another one if it has smaller f value
Output: an element of S

$best$:=a random element from S;
for $iteration := 1$ **to** $k - 1$ **do**
    r :=a random element from S;
    if $f(r) < f(best)$ **then** $best := r$;
return $best$;

Local Search - Simulated Annealing[3]

- Each vertex has two states: in the current set (1) or not in current set (0)
- All vertices are selected as an initial solution
- Objective function

$$F = A \sum_{i=1}^{n} v_i + B \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}(v_i v_j - v_i - v_j) + B \sum_{i}^{n} \sum_{j}^{n} d_{ij}$$

- Parameters:
  - A - coefficient for more vertex penalty
  - B - coefficient for uncovered edges penalty
  - T_0 - initial temperature
  - α - temperature decreasing ratio
- When cost function decreases, new solution will be accepted
- When cost function increases, new solution will be accepted with probability

$$p_r = \exp\left(-\frac{\Delta F(1 + Deg(v_i))}{T}\right) \qquad p_a = \exp\left(-\frac{\Delta F(1 - Deg(v_i))}{T}\right)$$

- Temperature updated (decreasing) each iteration

Branch and Bound

- Binary tree style: either add current vertex or all the neighbors connected with it
- Similar as approximation, the vertex with highest degree will be considered, so the initial solution is just an approximation result
- Lower Bound: already used vertex count + number of edges uncovered/highest degree in current graph

Result

graph	Global Optimum	BnB			SA			FastVC			Approximation		
		Sol	Time	Err	Sol	Time	Err	Sol	Time	Err	Sol	Time	Err
jazz	158	158	4.6	0.0000	162	0.169	0.0253	160	0.008	0.0127	159	0.093	0.0063
karate	14	14	0.01	0.0000	14	0.046	0.0000	14	0.002	0.0000	14	0.007	0.0000
football	94	94	0.1	0.0000	95	0.086	0.0106	97	0.006	0.0319	95	0.048	0.0106
as-22july06	3303	3308	0.366	0.0015	NA	NA	NA	3325	0.412	0.0067	3308	26.536	0.0015
hep-th	3926	3945	0.193	0.0048	5275	88.62	0.3436	3942	0.143	0.0041	3945	9.476	0.0048
star	6902	NA	NA	NA	8057	154.17	0.1673	7040	0.188	0.0200	7411	43.149	0.0737
star2	4542	4695	16.2	0.0337	8027	351.35	0.7673	4862	0.257	0.0705	4698	49.914	0.0343
netscience	899	899	0.537	0.0000	1038	2.976	0.1546	899	0.104	0.0000	899	0.463	0.0000
email	594	606	128.6	0.0202	746	1.989	0.2559	613	0.056	0.0320	609	0.416	0.0253
delaunay_n10	703	737	2.369	0.0484	788	1.269	0.1209	747	0.083	0.0626	740	0.398	0.0526
power	2203	2273	7.627	0.0318	3084	26	0.3999	2271	0.096	0.0309	2275	3.429	0.0327

Reference

[1] Fran¸cois Delbot and Christian Laforest. Analytical and experimental comparison of six algorithms for the vertex cover problem. Journal of Experimental Algorithmics (JEA), 15:1–4, 2010.

[2] Fan, Yi, et al. "Exploiting Reduction Rules and Data Structures: Local Search for Minimum Vertex Cover in Massive Graphs." *arXiv preprint arXiv:1509.05870* (2015).

[3] Xu, X., & Ma, J. (2006). An efficient simulated annealing algorithm for the minimum vertex cover problem. *Neurocomputing*, *69*(7), 913-916.

Thank you !