TECHNISCHE
UNIVERSITÄT
WIEN

Problem Solving and Search in A.I.

Minimum Vertex Cover

by Dominic Cervicek & Peter Rjabcsenko

Solution Data Structure

- Vertex Cover - boolean list
- Cost - integer
- Adjacency List - 2D list of vertices containing their neighbours

The Algorithm

- Randomized Construction Heuristic
- Simulated Annealing Metaheuristic
- Manual Parameter Selection

Construction Heuristic

Implemented 3 approaches for constructing an initial solution:

- **Worst:** select all vertices for the vertex cover
- **Randomized:** select vertices randomly until the resulting set is a valid vertex cover
- **Greedy:** iteratively select the vertex that covers the maximum number of uncovered edges until the resulting set is a valid vertex cover

For our experiments we decided to work with the “Randomized” approach

Simulated Annealing (Neighbourhood)

Implemented 2 neighbourhood structures:

- **“Strict” move:** flip random vertex, if result is a valid solution - accept move otherwise reject move and try again
- **“Relaxed” move:** flip random vertex, if result is a valid solution - accept move, otherwise include all neighbouring vertices of chosen vertex into the vertex cover (results in a valid solution)

For our experiments we decided to work with the “Relaxed” approach

We used incremental evaluation when flipping vertices to calculate the overall cost of the vertex cover

Simulated Annealing (Parameters)

- **Temperature:** f-max - f-min, which is the number of vertices in the graph
- **Cooling Rate:** we used a static cooling rate that we adjusted depending on the instance (e.g.: 0.5, 0.65, 0.8, 0.95)
- **Equilibrium Coefficient:** when multiplied with vertex number, results in number of moves attempted before the temperature is adjusted
- **Stopping Criterion:** additionally to the 1 min. time limit, if the temperature dropped below 0.01 we also terminated the search (no meaningful improvements reached after this threshold)

Results (over 10 runs)

#	Init. cost	Best cost	Mean cost	Std. dev.	Cool. Rate	Eq. Coefficient
001	6132	2607	2616.9	10.94	0.8	78
003	60051	12234	12293.2	58.24	0.5	2
005	198	129	130	1.18	0.95	14
007	8753	4435	4443.4	6.62	0.5	93
009	38423	21442	21451.9	7.18	0.5	5
011	9806	4990	4996.5	5.1	0.65	50
013	44962	8632	8642.3	4.56	0.65	2
015	53073	10699	10710	6.79	0.5	2
017	23406	12139	12153.7	9.43	0.65	10
019	197	130	131.4	0.8	0.95	14

Conclusions / Remarks

- We observed that we consistently achieved better results than a Local Search version of the algorithm (w/o metropolis criterion)

  Simulated Annealing 1 : 0 Local Search

- We selected the parameters so that there would be at least 20 temperature adjustments per run and the last 20% of temperature levels would be nearly deterministic search (yielded best results)
- Only used moves that yielded valid solutions, so it could be interesting to see “destroy / repair” type moves for this problem as well

Instructions

The algorithm can be run as a .jar from the command line, e.g.:

java -jar vertexcover.jar 005 random relaxed 0.95 0.01 14

where:

005: instance number

random: construction heuristic (random / greedy / worst)

relaxed: neighbourhood type (relaxed / strict)

0.95: cooling rate (optional, default: 0.95)

0.01: termination condition (optional, default is 0.01)

14: equilibrium coefficient (optional, default is square root of vertex number)

Instructions contd.

- Instances should be placed in an “instances” folder on the same level as the .jar file.
- Additionally a “solutions” folder needs to be created and placed on the same level as the .jar file
- Alternatively the algorithm can be compiled from its source in the “program” folder (it is a java project with no external dependencies)

TECHNISCHE UNIVERSITÄT WIEN

THANK YOU

FOR YOUR ATTENTION!