Dual Ascent

Grace LONGO
Timothée DURAND

Technology

- Python 3.6.8
- NetworkX library to handle graphs (https://networkx.github.io/)

Dual Ascent : implementation

- Select a root node among terminals

- While there are terminals left:
    - Pick one terminal with a minimally sized reachable graph (saturation graph G_A).

      Implemented in two different ways:
        - “Full evaluation” : recompute whole reachable graph for all terminals and pick smallest
        - “Lazy evaluation” : terminals stored in a list, ordered by ascending size of the reachable graph. At each iteration, we recompute the reachable graph of the two first terminals, pick the smallest

    - Once we have the reachable set, we pick the arc of minimal cost going into this set
    - If root node reached => remove terminal
    - Update saturation graph G_A, costs and lower bound.

- => We get the saturation graph G_A

Primal heuristic

- Shortest path algorithm:
  - From the root node, compute the shortest path to a terminal *t* using edges in G_A
  - Our solution tree is the Union of all shortest path

- Implemented with Dijkstra

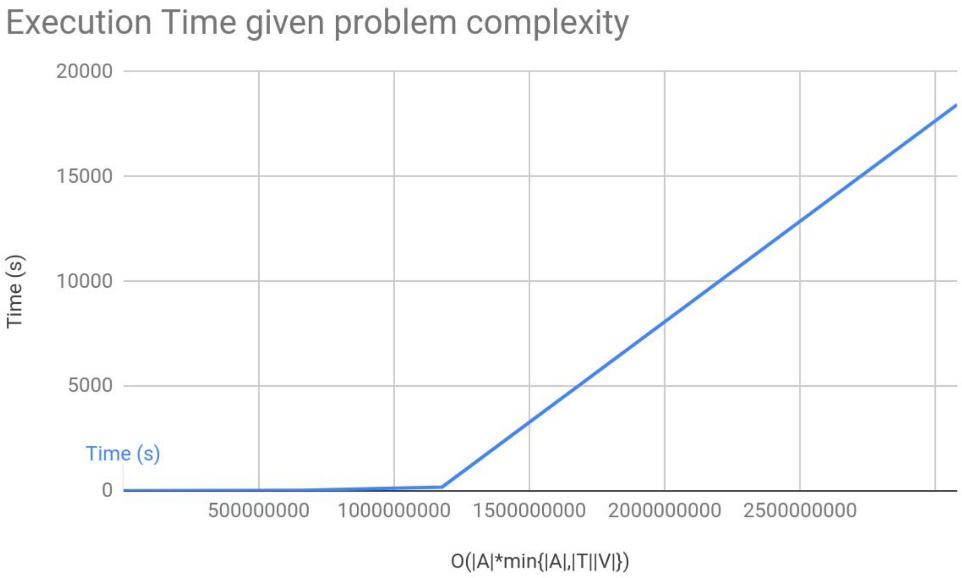
Dual Ascent: results

instance	|V|	|E|	|T|	Complexity	Opt	Result	LB	Time (s)
b01	50	75	9	Ls	82	82	82	0.0018
b11	75	150	19	Ls	88	107	87	0.0077
b12	75	150	38	Ls	174	195	163	0.0088
b13	100	125	17	Ps	165	180	163	0.0055
b18	100	200	50	Ps	218	226	216	0.0095

Dual Ascent: results

instance	|V|	|E|	|T|	Complexity	Opt	Result	LB	Time (s)
i640-145	640	40896	25	NPm	5218	5602	5136	30.97
i640-022	640	204479	9	Pm	1756	1756	1756	181.11
alue2087	1244	1971	34	Ps	1049	1505	954	2.99
alue6179	3372	5213	67	NPs	2452	3097	2325	3.4907
alue7080	34479	55494	2344	NPm	62449	111103	17187	18432

Time complexity



- Explodes with the number of terminals
- Some optimizations possible with higher memory cost
  - Remember reachable nodes
  - Use lower-level programming language