Monte Carlo Search Algorithms for Network Traffic Engineering

C. Dang[1,2], C. Bazgan[2], T. Cazenave[2], M. Chopin[1], P-H. Wuillemin[3]

[1] Orange Labs, Châtillon, France
{chen.dang,morgan.chopin}@orange.com
[2] Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE, F-75016 Paris, France
{cristina.bazgan,tristan.cazenave}@dauphine.psl.eu
[3] Sorbonne Université, CNRS, UMR 7606, LIP6, F-75005 Paris, France
pierre-henri.wuillemin@lip6.fr

Mots-clés : *traffic engineering, policy adaptation, Monte Carlo search*

Introduction. Many telecommunication networks rely heavily on shortest path computation for packets transportation, such as Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). In such routing protocols, every packet is routed from its origin to its destination along the shortest paths induced by so-called administrative weights. In this work, we are interested in the corresponding optimization problem of finding a set of weights such that the demands routed along the induced shortest paths generate a minimum congestion *i.e.,* the maximum ratio value of the total traffic going through an edge over the edge's capacity is minimized. We briefly describe the problem formulation here. Given a bidirected graph $G = (V, A)$, every vertex $v \in V$ corresponds to a router while an arc uv corresponds to a link between routers u and v. Every arc uv is associated a capacity denoted by c_{uv}. Let K denote a set of demands to be routed in G. Each demand $k \in K$ is defined by a pair of vertices s^k and t^k representing the source and the target of k, a traffic volume D^k to be routed from s^k to t^k. Given a metric $w \in Z_+^{|A|}$, each demand $k \in K$ is routed along the shortest paths between s^k and t^k. If there are more than one shortest paths joining the extremities of k, the traffic volume D^k is splitted evenly among those paths according to the so-called ECMP (Equal-Cost Multi-Path) rule. We then define the *load* of an arc uv induced by w as the amount $load(uv, w)$ of traffic traversing the arc uv over its capacity. The *congestion* $cong(w)$ of a given metric w is defined by $\max_{uv \in A} load(uv, w)$, that is the maximum load over all arcs.

The considered optimization problem is called Minimum Congestion Shortest Path Routing(Min-Con-SPR) which aims at finding a metric $w \in Z_+^{|A|}$ and the routing paths induced by these weights such that the network congestion $cong(w)$ is minimum.

Monte Carlo Search on Routing Problem. Monte Carlo Search is a general optimization technique. In our research, we employed NRPA [3] (Nested Rollout Policy Adaptation), which learns a policy by recursively calling to the lower level at each nesting level, searching to improve its current best score. When it succeeds, the best score of the corresponding state *score*(*state*) is updated, and the current action sequence is recorded as the best sequence.

To model the Min-Con-SPR problem using NRPA, we assume that a solution to the Min-Con-SPR problem is represented by a point (i.e. the metric $w = \langle w_1, w_2, ..., w_{|A|} \rangle$) in the discrete space $[1, 65535]^{|A|}$. To reduce the search space, we set the metric's value space as a subspace W of the original space $[1, 65535]$. The metric of the graph is assigned and the objective function $cong(w)$ is evaluated during each playout. An action a is therefore a choice of metric values for an arc. We also propose the technique *force_exploration* which greatly improved the performance of NRPA.

name	|V|	|A|	|K|	Unit OSPF	InvCap OSPF	Local Search	NRPA	LPLB
abilene	12	30	132	187.55	89.48	60.42	**60.412**	60.411
atlanta	15	44	210	3.26	3.37	**2.22**	**2.22**	2.18
france	25	90	300	4.12	4.12	**2.53**	2.56	2.41
nobel-us	14	42	91	37.15	37.15	**24.4**	24.7	24.2
nobel-eu	28	82	378	13.31	13.31	10.68	**10.67***	10.67
brain	161	332	14311	1.415	1.415	0.962	**0.903***	0.903
rand50a	50	132	2450	7.9	7.9	**5.55**	5.77	5.55
rand50b	50	278	2450	**2.88***	**2.88***	**2.88***	**2.88***	2.88
rand100a	100	278	9900	15.71	15.71	10.42	**9.59**	9.35
rand100b	100	534	9900	4.15	4.15	4.38	**3.85**	3.76
wax50a	50	142	2450	6.46	6.46	**4.63**	4.66	4.59
wax50b	50	298	2450	**2.279***	**2.279***	2.284	**2.279***	2.279
wax100a	100	284	9900	17.46	17.46	15.049	**15.048**	15.048
wax100b	100	492	9900	5.51	5.51	4.14	**4.04**	3.44

TAB. 1 – Maximum congestion value of state-of-the-art heuristics and our NRPA. The value is in bold if it is the best one among those returned by the other heuristics. In addition, a value is followed by * if equal to the lower bound LPLB (which is reported in the last column). Each graph is tested in a fixed time and each value is averaged on 5 independent executions.

Experimental results. The algorithms are implemented in C++ and the experiments are done on a server (64-core Intel(R) Xeon(R) Gold 5218 CPU), with 125 GB of memory. The experiments are performed on several graphs from SNDlib [1] of different sizes. In addition to these instances, several random graphs and waxman graphs are also generated using the same setup as in [2], demands are also generated similarly to [2]. The capacity of all arcs is set to 1000. Every generated graph is verified to be connected.

The results in Table 1 demonstrate that NRPA performs very well on all sizes of graphs and is very close to the lower bound. In the vast majority of cases, our approach outperforms local search [2] and only utilizes a small amount of processing time and resources. At the same time different runs can be computed in parallel, which significantly reduces the execution time of NRPA. Further testing shows that on larger graphs, local search does not produce acceptable results in less than 30 minutes. On the contrary, NRPA produces satisfactory results in a relatively short time even on graphs with thousands of nodes.

Conclusion In this work, we applied the Monte Carlo Search approach for solving the Min-Con-SPR problem for the first time. Experiments show that our approach is comparable with the existing ones for instances from the literature. Nonetheless, for larger graphs, our method outperforms local search heuristics and produces results that are near to the lower bound. At the same time, because this approach is not sensitive to the size of the graph or the size of the search space, particularly the size of the search space, it may be simply extended for situations with additional constraints.

Références

[1] Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R. : SNDlib 1.0–Survivable Network Design Library. Networks **55**(3), 276–286 (2010)

[2] Fortz, B., Thorup, M. : Increasing internet capacity using local search. Computational Optimization and Applications **29**, 13–48 (09 2000)

[3] Rosin, C.D. : Nested rollout policy adaptation for Monte Carlo Tree Search. In : IJCAI. pp. 649–654 (2011)