Development of Topics in Electrical Engineering 1 - Reproducing “Valadarsky, Asaf, et al. Learning to Route 2017 “Results

Report

Written by:

Ido Yehezkel 204397368

Table of Contents

Introduction

The main purpose of this report is about reproducing the achieved results from the paper: *Valadarsky, A., Schapira, M., Shahaf, D., & Tamar, A. (2017, November). Learning to route*. In *Proceedings of the 16th ACM workshop on hot topics in networks* (pp. 185-191).

The authors interduce a new approach of how to use machine learning techniques in order to solve one of the fundamental network control challenges, namely traffic routing.

Both deep supervised learning and deep reinforcement learning techniques are presented in the paper for handle this challenge. Supervised learning is used for predicting the next traffic demands matrix based on the past compared to focusing on solving the routing control problem directly using reinforcement learning agent. The supervised learning model is a good starting point but did not achieve noticeable results, on the other hand, the performance of the reinforcement learning agent is better, moreover, for some traffic patterns even beats known routing schemes.

The reproduction of the results starts with generating traffic patterns as the authors described and continues with training a reinforcement learning agent.

Another motivation for this work is gain new knowledge about routing algorithms (Optimal Routing, Oblivious Routing), new heuristics methods, useful programming packages, yet the main goal is to develop a new direction for a continuing research.

Definitions

Network topology- following the paper, for the control routing problem the network topology is represented by a directional graph which is based on unidirectional graph with edges' capacity function (in Mb/s).
The directional graph is simply made by considering each edge as independent bidirectional link.

Synthetic Traffic Generation – to examine the developed techniques a synthetic traffic generation is necessary.
The traffic demand is represent by a square matrix where the cell i, j is the flow demand from node i to node j.
Another property of the matrix is sparsity, which is the percentage of source, destination pairs that demand traffic, for example, a topology with 10 nodes includes 90 difference pairs of source destination, with sparsity of 30% only 27 randomly chosen pairs are included in the traffic.
Two different types of traffic are generated for evaluation:

- **Gravity Traffic**: traffic with correlation to the links' capacities connected to the source node and the destination node, calculated by the formula:
  $$\frac{[\text{source out links' capacity}] \times [\text{destination out links' capacity}]}{\text{total nodes out links' capacity}}$$
- **Bimodal Traffic:** each flow demand of the traffic is sampled by some probability (biased coin flip) from two independent Gaussian distributions. One distribution represents mice flows and the other elephant flows.
  For all the evaluations the Gaussians distributions are: $N(400, 20)$ for elephant flows and $N(150, 20)$ for mice flows.

The Objective: finding a routing scheme for the traffic demands with the objective of minimization of maximum link utilization, , also referred to as the minimax problem and load balancing flows in the network.

Baselines - Optimal Routing

To evaluate the models and techniques that were developed by the authors they define a reference baseline that is based on **load balancing**.

The optimal routing criterion is defined as the **minimization of maximum link utilization**, i.e., minimize the most congested link, for which the mathematic expression is:

$$\min_{e\in E}\left\{\max\frac{f_e}{C_e}\right\}$$

C_e – capacity of link e

f_e – total flow in link e

This can be formulated as an optimization problem for a traffic demand matrix with flow in the i, j cell as follows:

$$Objective: Minimize\ r$$

$$\sum_i\sum_{j\neq i} g_e(i,j) \le C_e \cdot r\ \ \forall e \in Edges$$

where $g_e(i,j)$ is a fraction of the flow $i \to j$ that link e carries

The routing scheme constartions:

For each existing demend $i \to j$:

$$\sum_{e\in IN(v)} g_e(i,v) - \sum_{e\in OUT(v)} g_e(i,v) = demend(i,v)\Big|\, v = j, \text{destination constraint}$$

$$\sum_{e\in OUT(v)} g_e(v,j) - \sum_{e\in IN(v)} g_e(v,j) = demend(v,j)\Big|\, v = i, \text{source constraint}$$

$$\sum_{e\in OUT(v)} g_e(i,j) - \sum_{e\in IN(v)} g_e(i,j) = 0\Big|\, v \neq i, j, \text{transit constraint}$$

$$g_e(i,j) \ge 0\ \ \forall i, j$$

$$r \ge 0$$

By the definition of this optimization problem, all the constraints are linear expressions therefore, a linear programing solver (like "IBM – CPLEX" or "Gurobi") can be used to solve it, ("Gurobi" has been used, because it easy to set multiple objectives much easier so as to prevent the tool from creating unnecessary loops of flows).

The linear programming problem includes $O\left(|edges| \times |nodes|^2\right)$ variables and $O\left(|edges| + |nodes|^2\right)$ constrains.

Baseline - Optimal Oblivious Routing

Another baseline that the authors used is **optimal oblivious routing**. As its name implies, oblivious routing **does not traffic patterns depended on traffic patterns but only on the topology**. This routing technique was presented in several papers in the early 2000s.

The oblivious performance ratio of a routing scheme f is define as follows:

$$CONGESTION(f,D) = \max_{e\in E} \frac{Flow(e,f,D)}{C_e}\Bigg|$$

Most congestion edge w.r.t routing scheme f, demend D and capacity C.

$$OBLIV-PERF-RATIO(f) = \sup_D \frac{CONGESTION(f,D)}{OPT(D)}\Bigg|$$

Worst demends matrix congestion w.r.t routing scheme f normalized by optimal routing congestion.

$$OBLIVE-OPT(G) = \min_f OBLIV-PERF-RATIO(f)\Big|$$

Best routing scheme f of topology G regrdless the traffic demends

Using the result of the paper *Applegate, D., & Cohen, E. (2006). Making routing robust to changing traffic demands: algorithms and evaluation. IEEE/ACM Transactions on Networking, 14(6), 1193-1206,* the optimal oblivious routing problem can be formulated as a single optimization problem as follows:

$Objective : Minimize\ r$

f is valid routing scheme

$\forall edges\ e:$

$$\sum_{h\in E} C_h \cdot \pi_e(h) \le r$$

$$\forall \text{ pairs } i \to j: \ \frac{f_e(i,j)}{C_e} \le p_e(i,j)$$

$$\forall \text{ node } i, \ \forall \text{ edge a=}(j,k):$$

$$\pi_e(a) + p_e(i,j) - p_e(i,k) \ge 0$$

$$\forall \text{ edge h} \in \text{E}: \pi_e(h) \ge 0$$

$$\forall \text{ node } i: \ p_e(i,i) = 0$$

$$\forall \text{ node } i,j: \ p_e(i,j) \ge 0$$

As one can see, because all the constrains are linear, this is also a linear programming problem with $O\left(|edges| \times |nodes|^2\right)$ variables and $O\left(|edges|^2 \times |nodes|\right)$ constrains, where $\pi_e(h)$ is a variable that represents an existing weight such that for every pair of edges e,h: $\sum_{h \in E} C_h \cdot \pi_e(h) \leq r$ and $p_e(i,j)$ represents the length of the shortest path from node i to node j according the edge weights $\pi_e(h)$.

Reproducing the Baseline Results

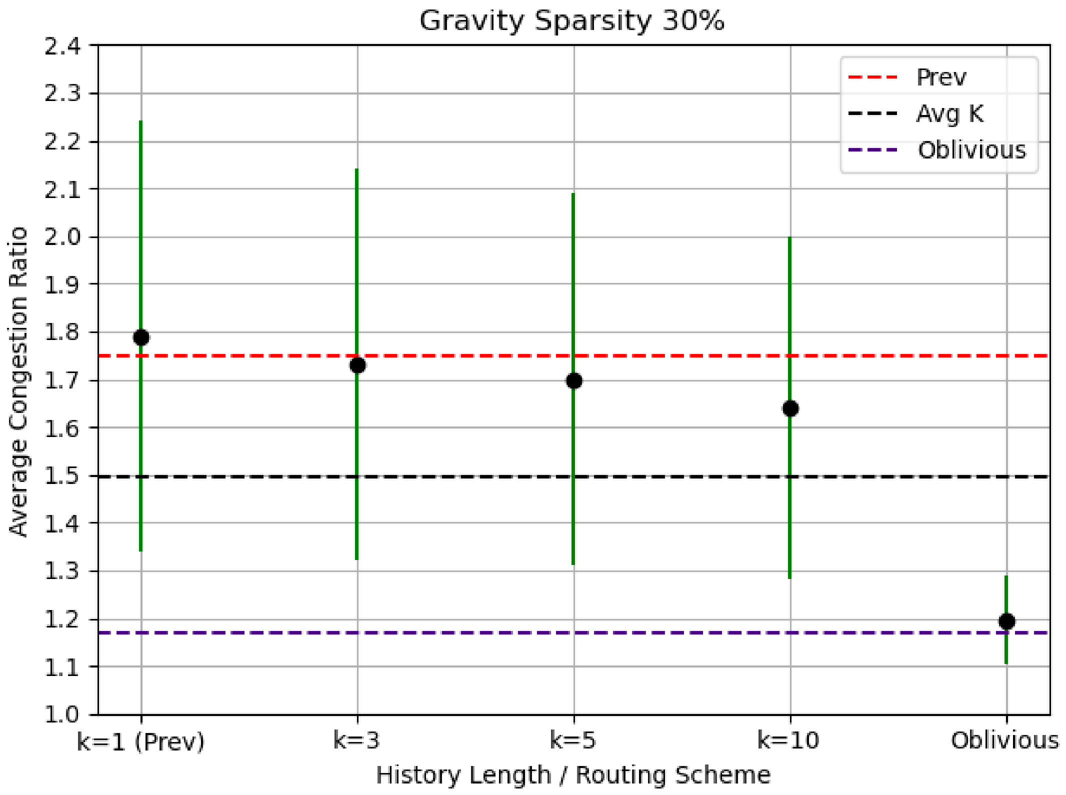
The reproduction process starts with reproducing of all the baselines, such as optimal routing base on averaged history and Oblivious Routing.
Motivated by the reinforcement learning agent that takes a decision by observing the traffic history the authors created a similar referenced baseline that based on traffic history.
By calculating the last K traffic matrices average (component wise), creating a routing scheme for it by solving the optimal routing optimization problem and used it for route the future (next to be routed) traffic matrix (create the routing scheme in advance, before new traffic arrives).
One of the critic challenges that needed to be considered is that a new flow can arrives i.e. it can be happens that a flow exists in the current routed traffic matrix but not in the average one (based on history, this is more common in low matrix sparsity), the solution is to use an ECMP policy with equal weights, so the first appears flows are equally divided between all shortest paths between the source and destination.
The result are normalized with the most congested link utilization when applying the optimal routing scheme.

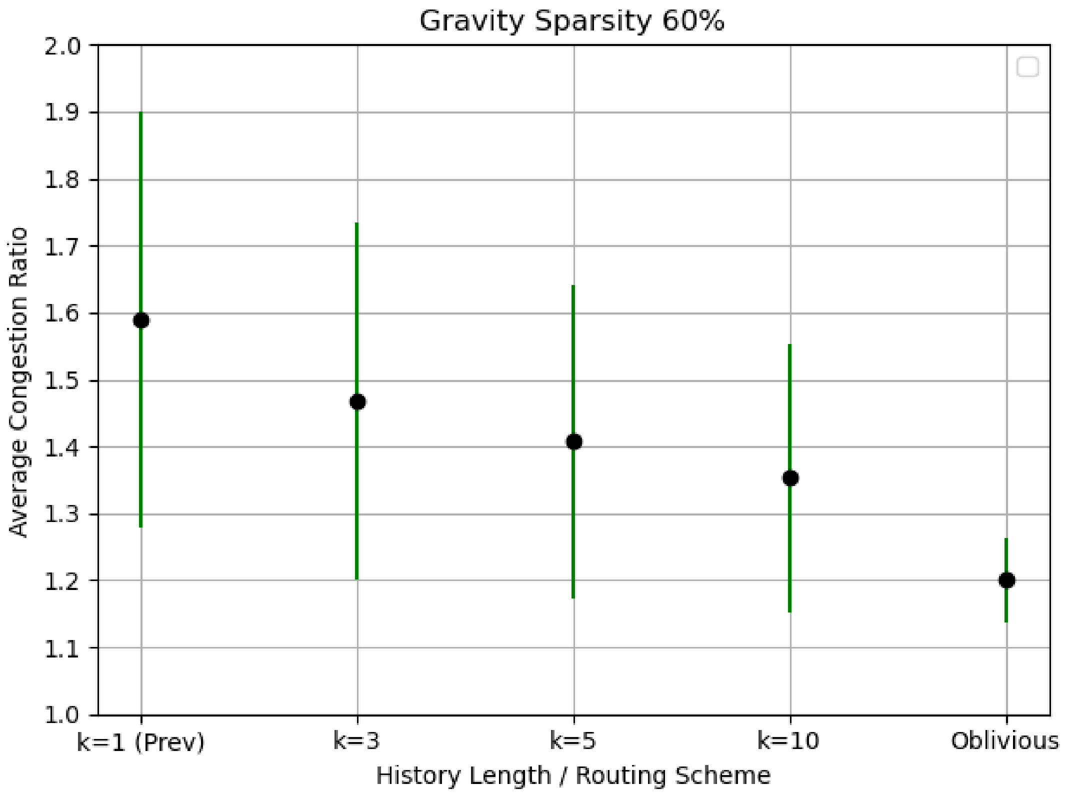
Evaluation

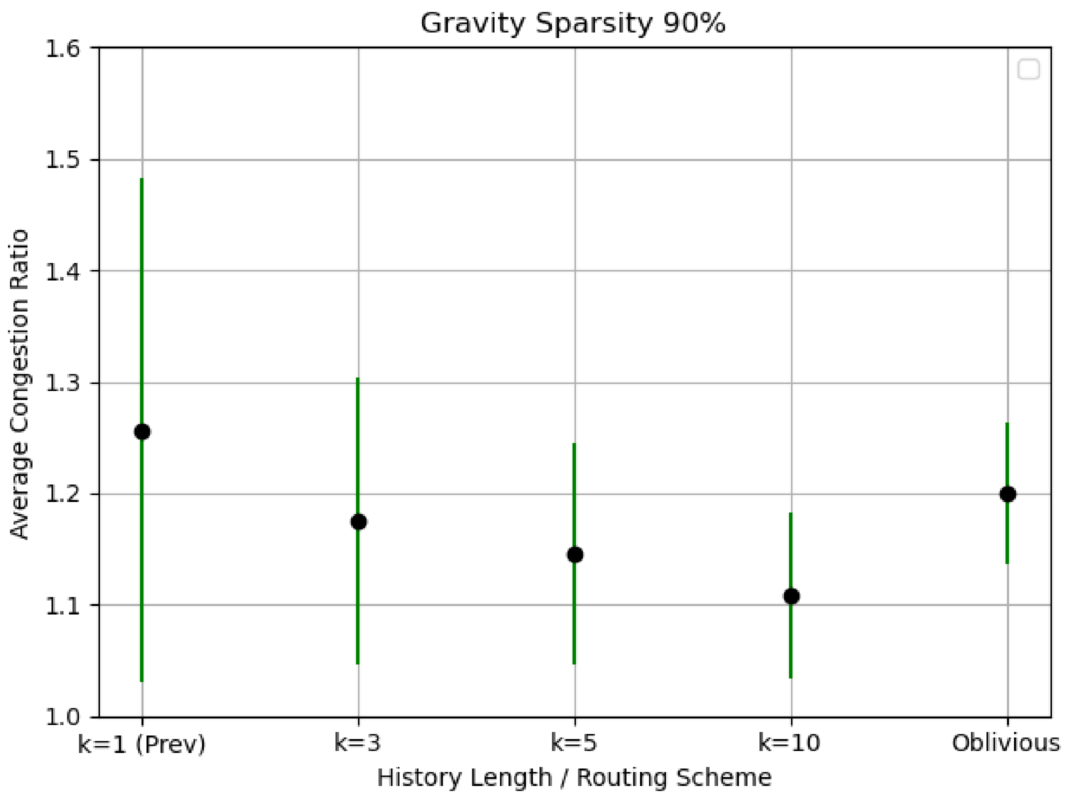
Similar to the paper, a 12-node topology with 26 edges (the original topology includes duplicate edges, this represented as double the capacity for those edges) and constant link capacity of 10,000 Mb/s.
20,000 traffic matrices dataset has been used in order to get the results.
For example, for K=5 matrices 1,2,3,4,5 are used for calculate an average traffic matrix then finding the optimal routing scheme for it and apply it on matrix 6.

(The dashed lines are approximations of the results from the paper, the vertical green lines are the standard deviation).

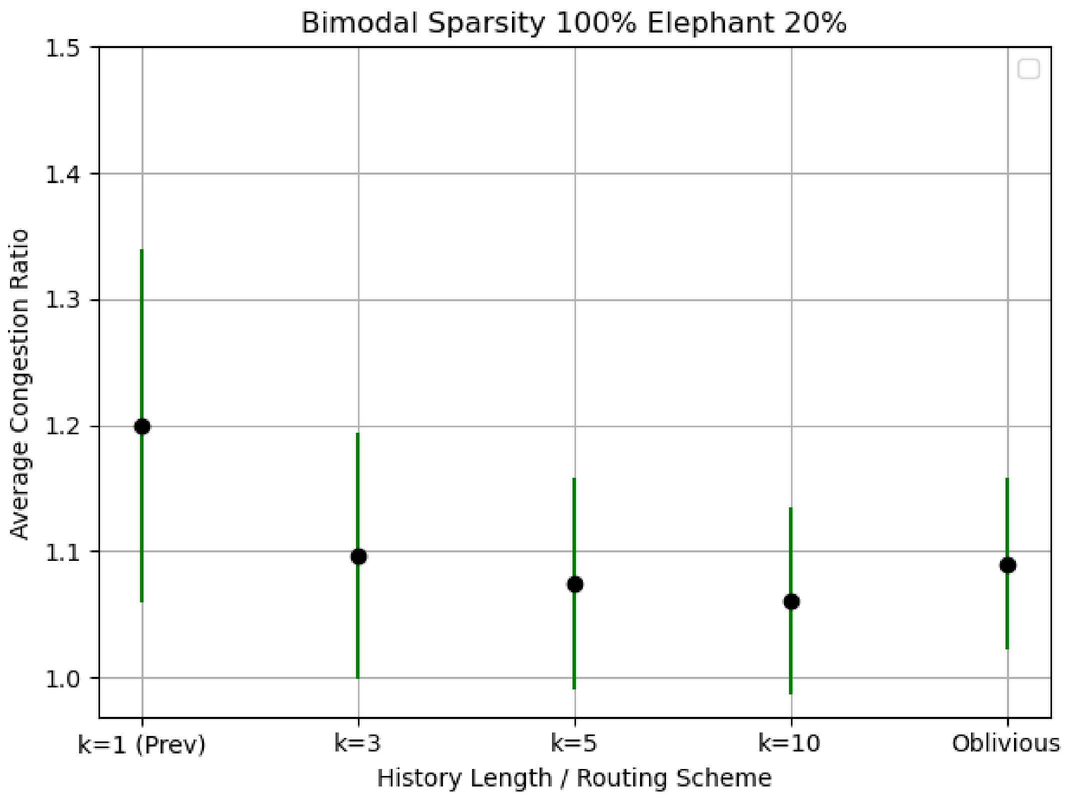
<u>Gravity Traffic:</u>

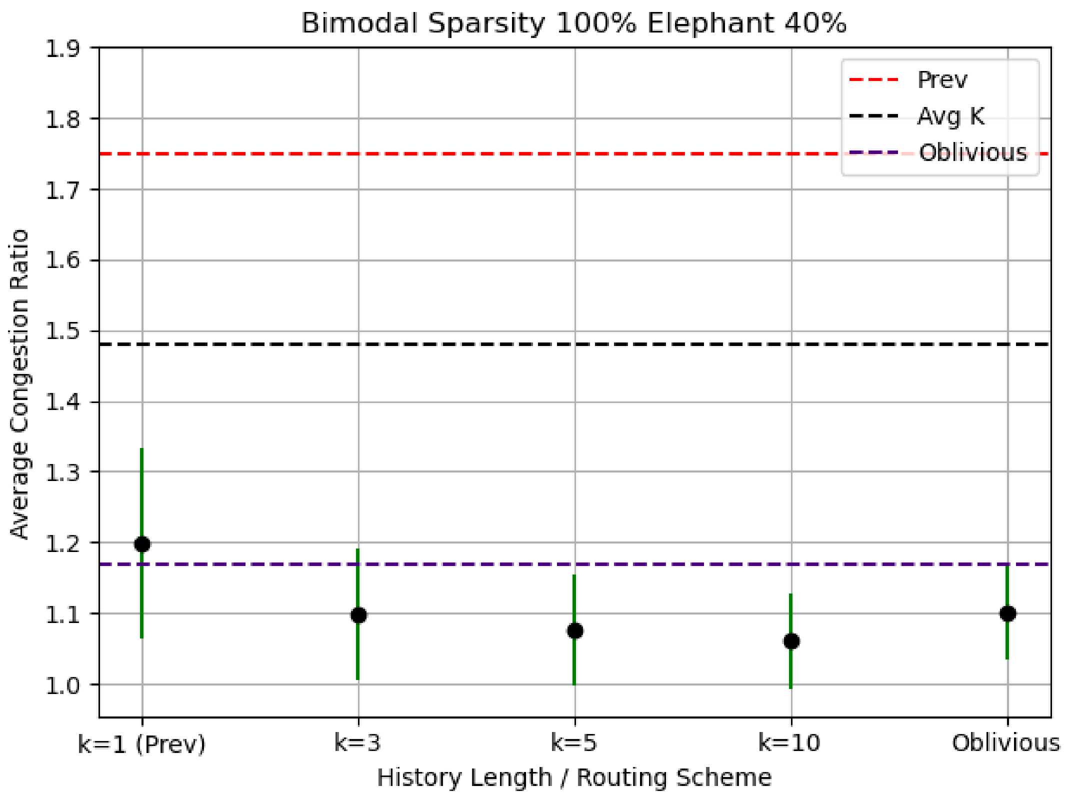




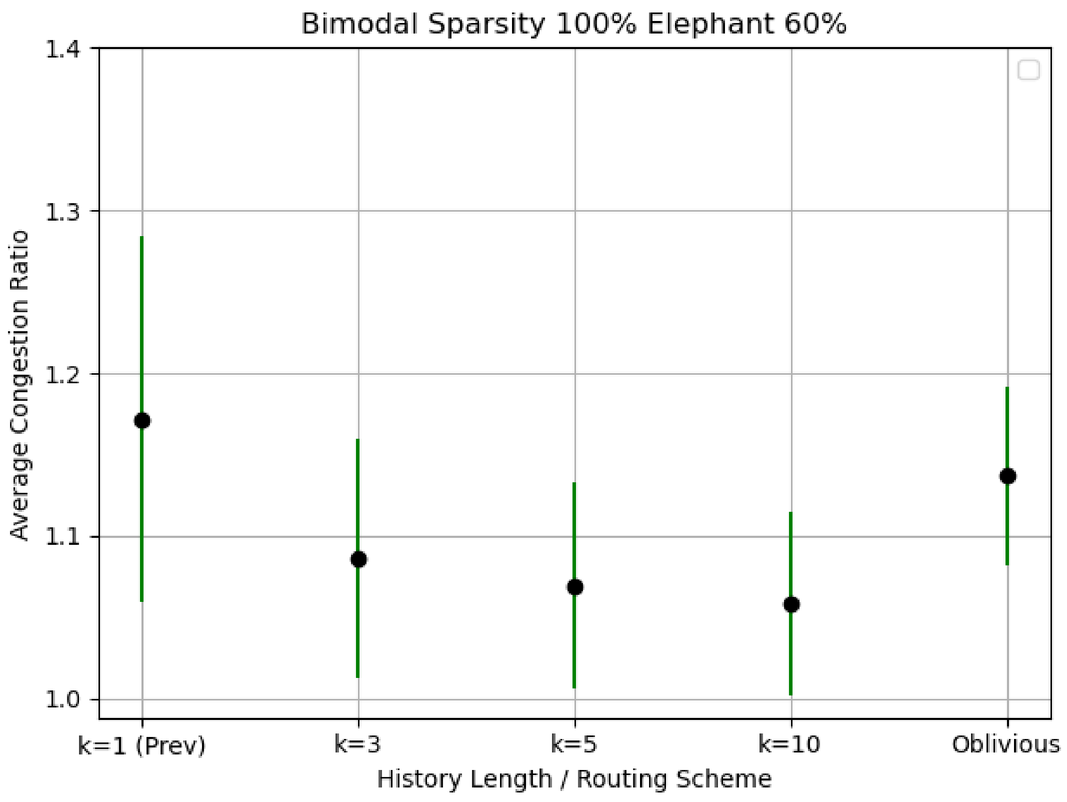


Bimodal Traffic:





Bimodal Sparsity 100% Elephant 40%
Prev
Avg K
Oblivious
Average Congestion Ratio
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
k=1 (Prev)
k=3
k=5
k=10
Oblivious
History Length / Routing Scheme



Bimodal Sparsity 100% Elephant 60%
Average Congestion Ratio
1.4
1.3
1.2
1.1
1.0
k=1 (Prev)
k=3
k=5
k=10
Oblivious
History Length / Routing Scheme

Reproducing the Reinforcement Learning Results

The outstanding result from the paper is the prove of concept that a reinforcement learning agent can be useful to produce a good routing scheme that minimize the congestion ratio as much as other known routing techniques.

The authors set up an environment in order to simulate the network nodes and links and every timestep a new traffic matrix with new demands is routed by the agent.

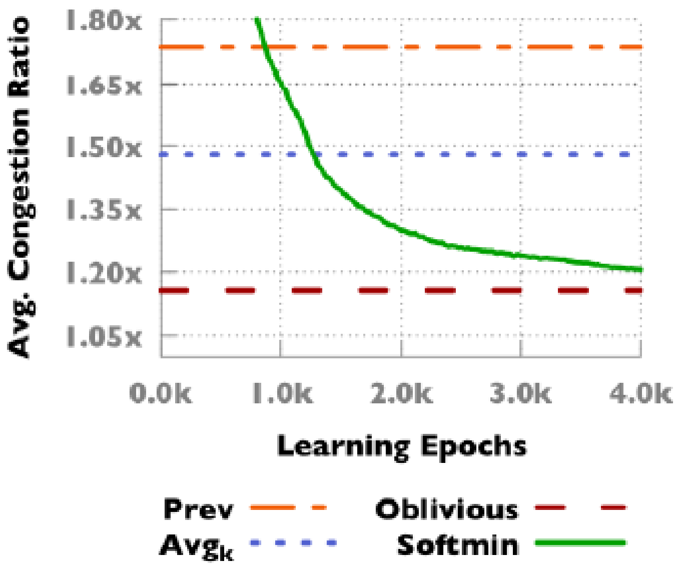
The main goal of the process of learning is the agent learns a **map** from a **history of traffic matrices to weights** for each network link which used for routed the future traffic matrix. Using those weights, the environment calculates shortest path to each destination of demands from every other node. Finally, plug these paths' costs and each link weight to Soft-Min function in order to calculate for each node the percentage of flow carried by each leaving link.
The final step is to simulate each demand of flow around the network links until all of it (99.99%) reaches to its destination and calculate the most congested link for the final congestion ratio of the traffic matrix.

Evaluation

Similar to the paper, a 12-node topology with 26 edges (the original topology includes duplicate edges, this represented as double the capacity for those edges) and constant link capacity of 10,000 Mb/s.

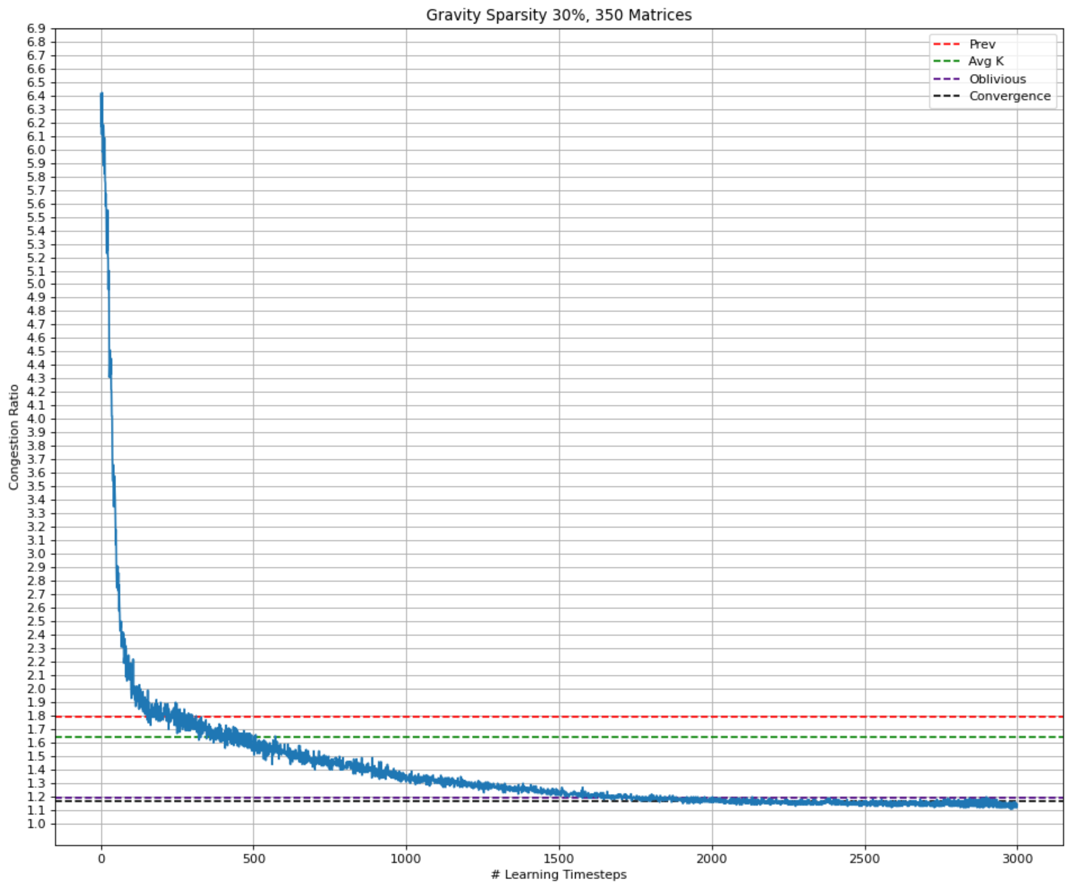
The evaluated agent parameters are: fully connected neural network with 3 layers of [128, 64, 64], the state is the history of traffic with length of 10 (like the paper), discount factor equal to 0 and episode length of 1. Because flows are not continued more than one timestep (flows can't start in time t and continue to time $t+1$), therefore the assumption that a myopic approach to minimize the congestion of the current traffic matrix is a good start because agent current decisions has no effect on future ones.

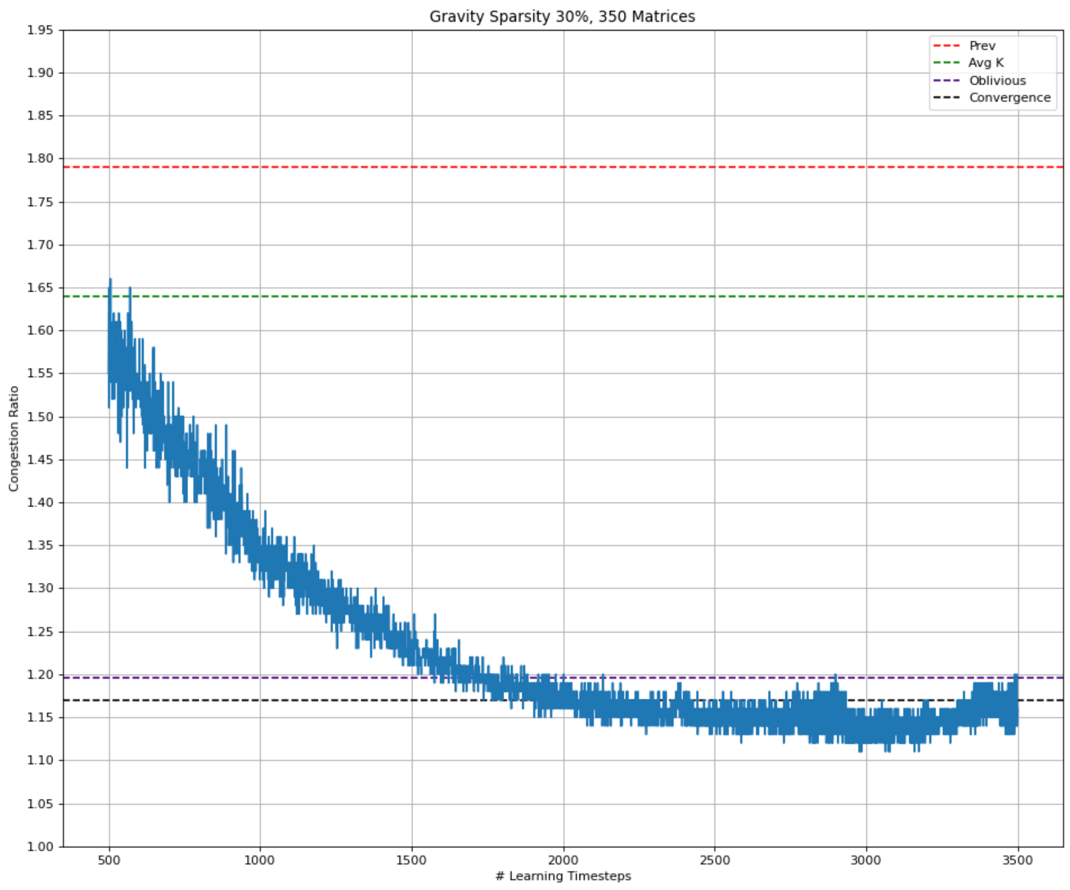
Gravity Traffic Paper results:



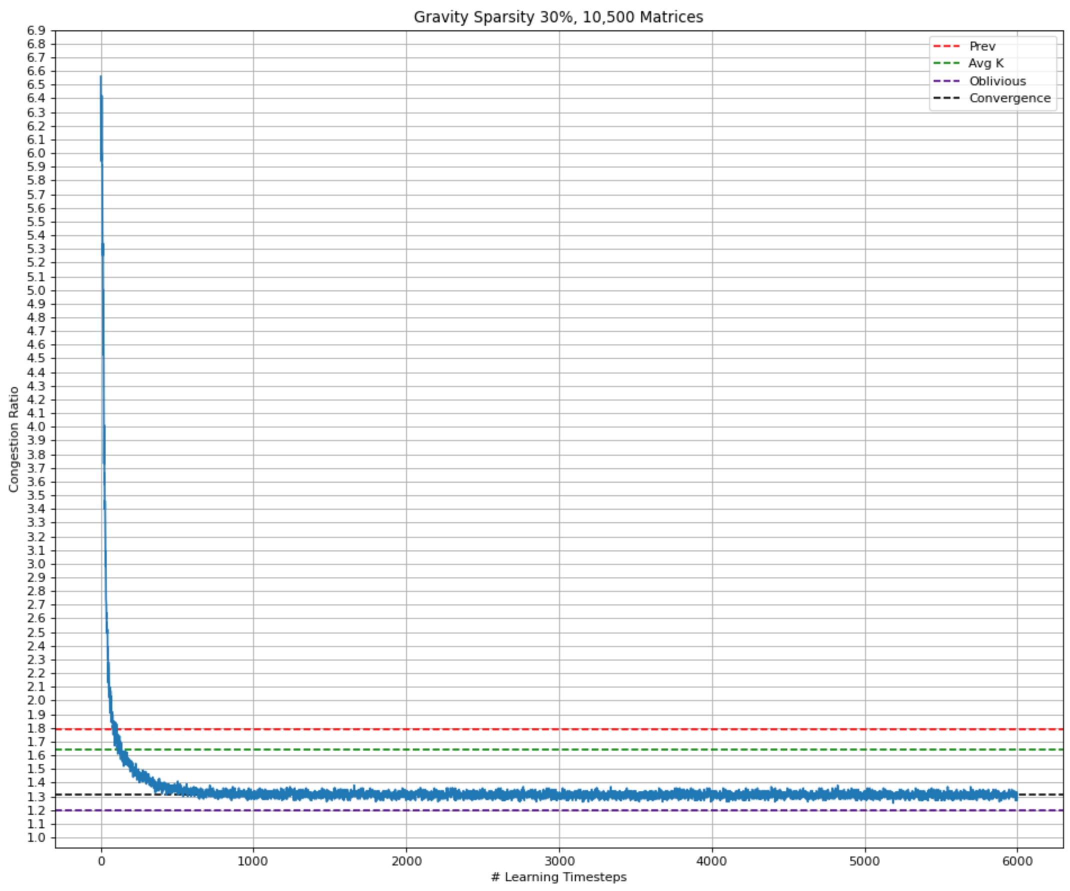
(a) *Congestion ratio for sparse ($p = 0.3$) gravity DM sequences*

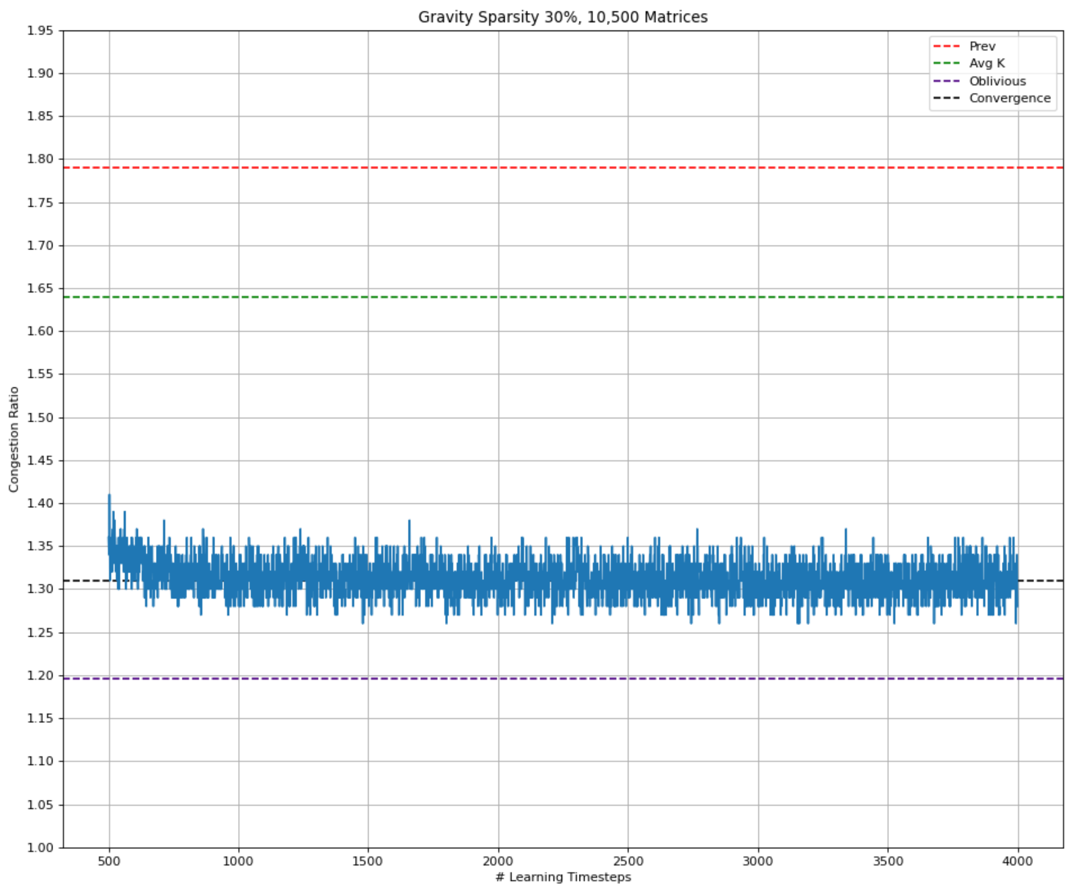
Gravity Traffic, Sparsity of 100% and 350 different TMs:



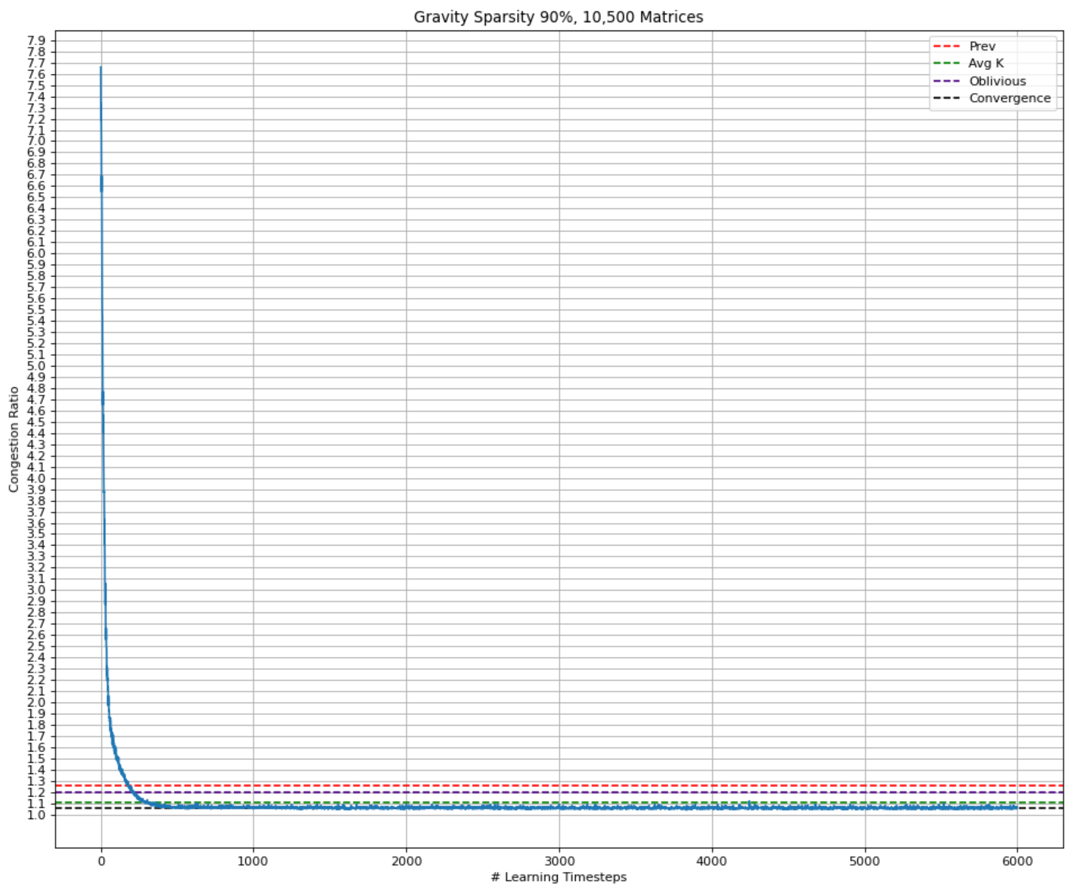

Gravity Sparsity 30%, 350 Matrices
Prev
Avg K
Oblivious
Convergence
Congestion Ratio
# Learning Timesteps
1.95
1.90
1.85
1.80
1.75
1.70
1.65
1.60
1.55
1.50
1.45
1.40
1.35
1.30
1.25
1.20
1.15
1.10
1.05
1.00
500
1000
1500
2000
2500
3000
3500

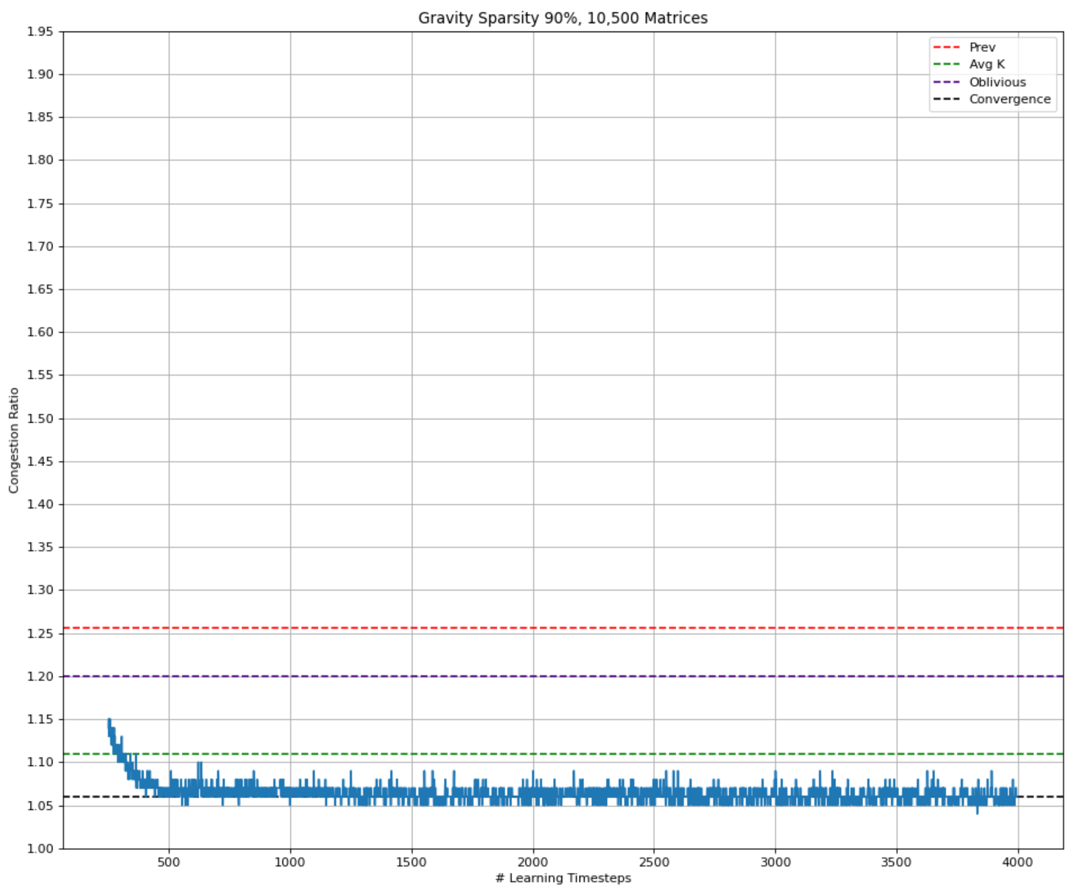
Gravity Traffic, Sparsity of 30% and 10,500 different TMs:



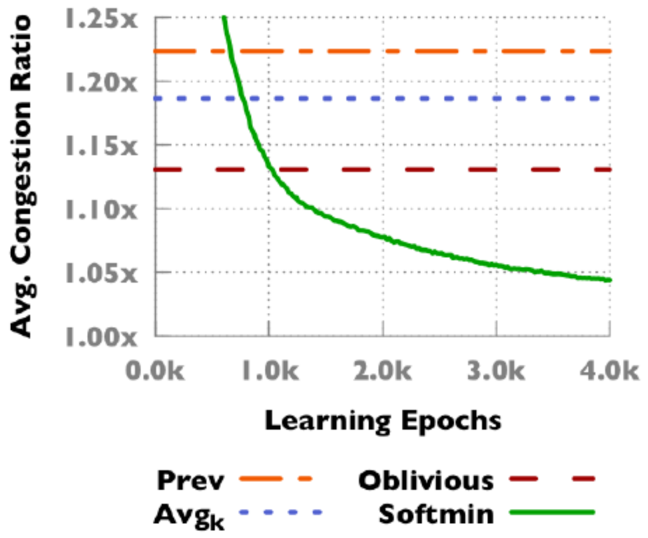


Gravity Traffic, Sparsity of 90% and 10,500 different TMs:



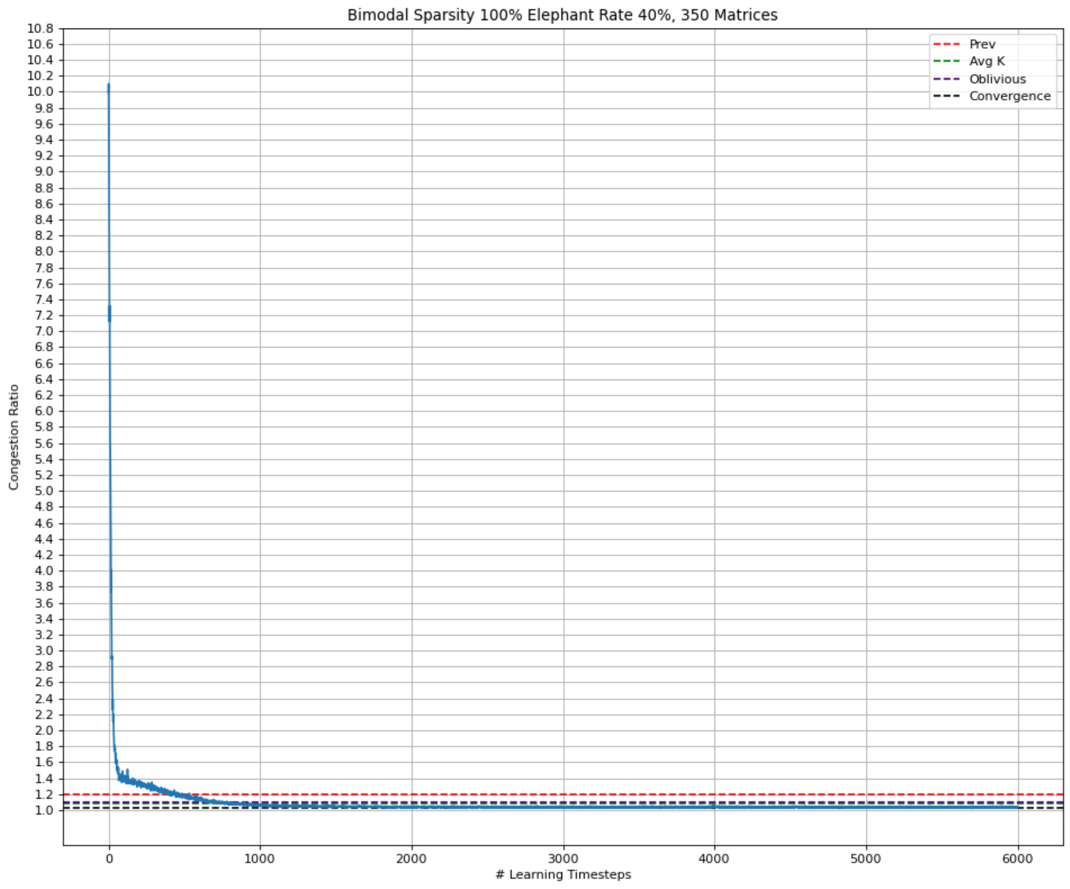


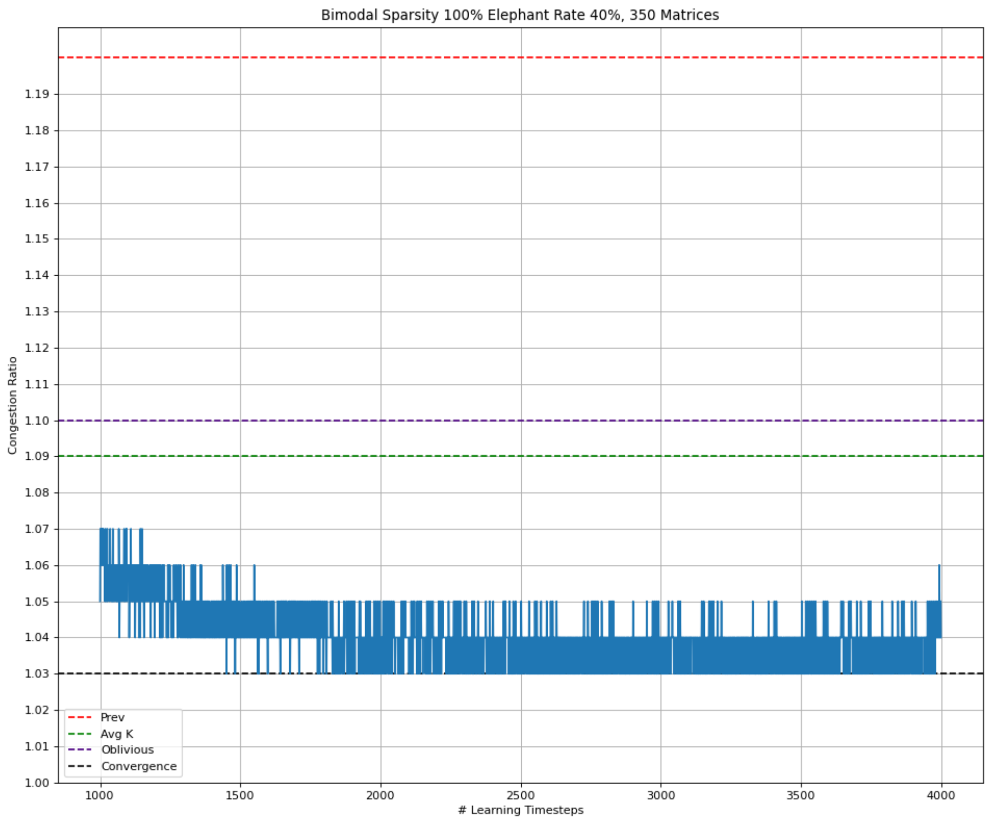
Bimodal Traffic Paper results:



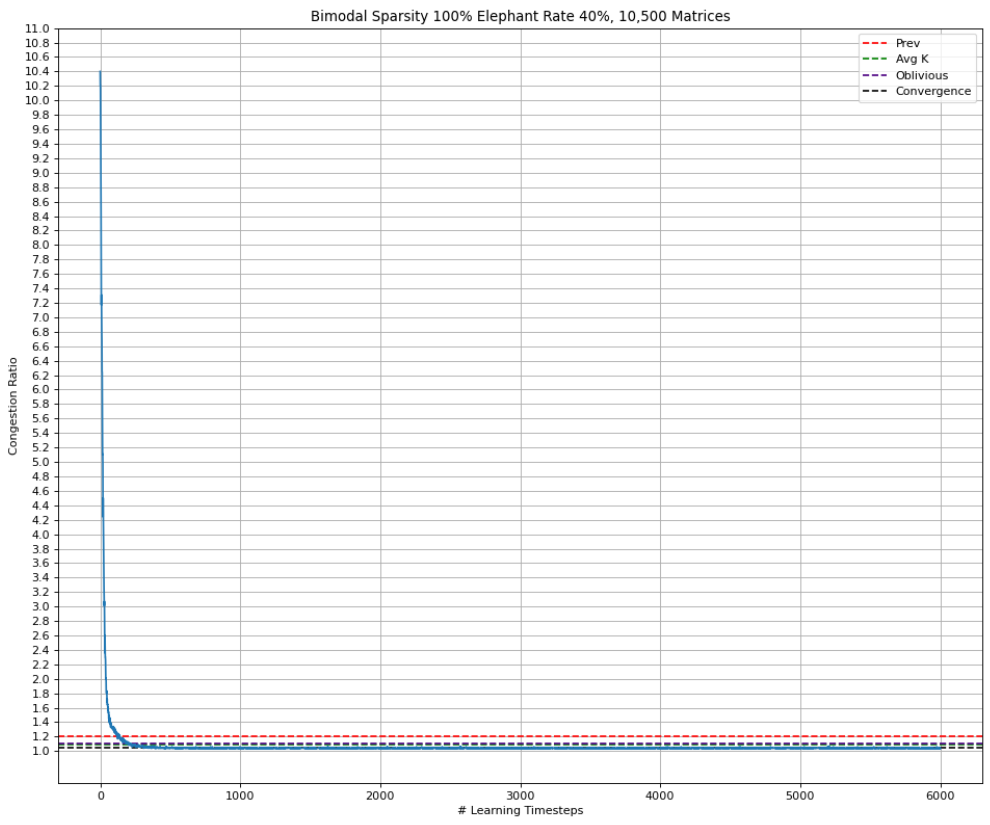
(b) *Congestion ratio for non-sparse (p = 1.0) bimodal DM sequences with 40% elephant flows*

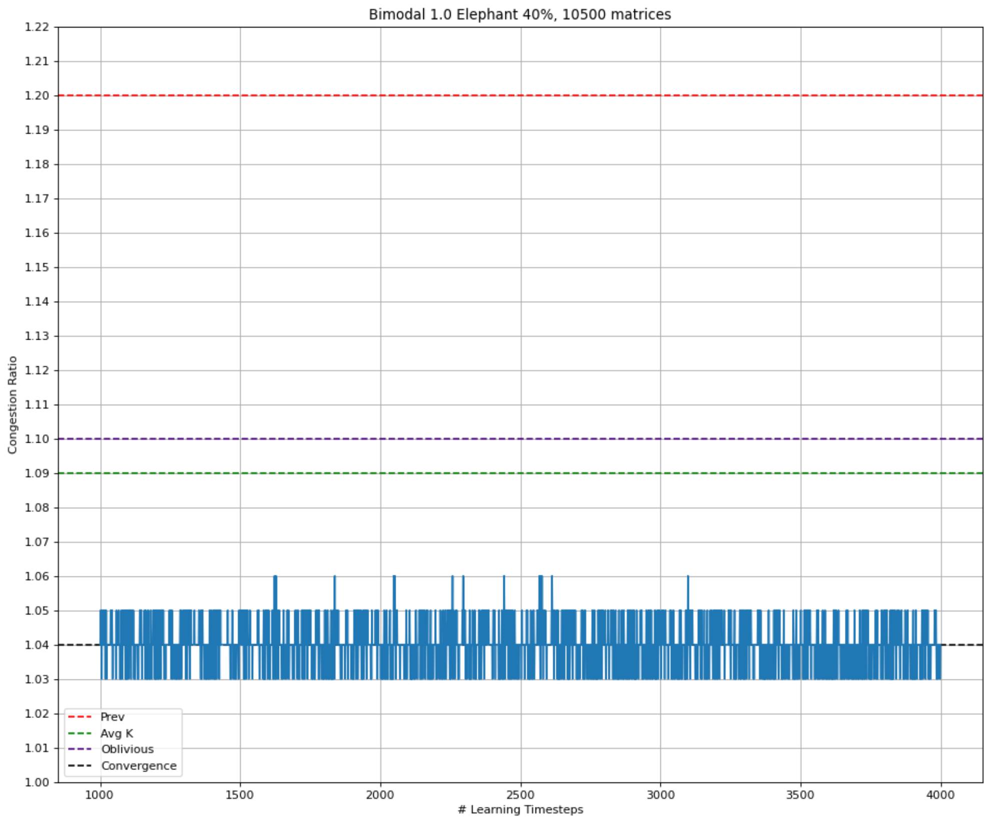
Bimodal Traffic, Sparsity of 100% and 350 different TMs:




Bimodal Sparsity 100% Elephant Rate 40%, 350 Matrices
Congestion Ratio
# Learning Timesteps
1.00
1.01
1.02
1.03
1.04
1.05
1.06
1.07
1.08
1.09
1.10
1.11
1.12
1.13
1.14
1.15
1.16
1.17
1.18
1.19
1000
1500
2000
2500
3000
3500
4000
Prev
Avg K
Oblivious
Convergence

Bimodal Traffic, Sparsity of 100% and 10,500 different TMs:





Conclusions and Summary

First, I must say the results reproduction process help me, as a young researcher, to understand much better the algorithms and methods of evaluation used in the paper that I am about to use in my own research.
Moreover, the practical experience with the Python packages for this task help me to improve my machine learning programming skill especially for reinforcement learning, doe example, how to write a RL environment, how to configure an agent such as choosing the hyperparameters.

While the reproduction process and after examination the outputs figures I notice some conclusions that may be helpful for next research directions.
As seen in the baselines results, the oblivious routing scheme is becoming less effective when the sparsity of the matrix is higher, I think the reason for that is because with higher sparsity the average matrix includes more relevant flows that appears in the next routed traffic matrix, so this make the average one a better approximation of the next one.
The first seen flows (as described earlier) have a big impact on the performance because they are routed by static ECMP with equal link weight, therefore, more sophisticated way can be useful for allocating the weights, for example each link weight is $\frac{1}{Capacity}$, the motivation behind it is to preferred paths with high link capacities.

By inspection of the Bimodal traffic matrices evaluations specifically, the effectiveness of the percentage of an elephants flows on the performance is low (the reason may be because the means of the Gaussians distributions for mouse and elephant flows should be difference with factor scale, like 100).

The reinforcement learning agent achieves results similar to presented in the paper and it seems the convergence of the congestion ratio is faster; the reason is probably because the change in the learning algorithm from TRPO to PPO which is good as much as TRPO.

The same trends the paper shows can be seen also in the reproduced results, for gravity traffic with low sparsity (30%) the oblivious is still better than the agent, but for the bimodal traffic with sparsity of 100% and for gravity traffic with sparsity of 90% the agent beats all the baselines. The reason for better performance is that the agent learning from most of pairs all the time, and because the low standard deviation for the distributions the values of demands are not changing too much, also the demand value for a source destination pair in gravity traffic is constant the history of matrices are more precise approximation of the next routed traffic matrix. Another evidence for that statement is that there is no change in the performance when number of different traffic matrices to be learned from is increasing.

To summarize, I enjoyed examining and reproducing the results from the paper, that helped me a lot to start gain some knowledge about the topics my research is about

to handle with.
Also, I would like to thank Asaf Valadarsky (one of the paper authors) that help me and contribute parts of his code.

References

- Valadarsky, A., Schapira, M., Shahaf, D., & Tamar, A. (2017, November). Learning to route. In Proceedings of the 16th ACM workshop on hot topics in networks (pp. 185-191).
- Applegate, D., & Cohen, E. (2006). Making routing robust to changing traffic demands: algorithms and evaluation. IEEE/ACM Transactions on Networking, 14(6), 1193-1206.
- Azar, Y., Cohen, E., Fiat, A., Kaplan, H., & Räcke, H. (2004). Optimal oblivious routing in polynomial time. Journal of Computer and System Sciences, 69(3), 383-394.
- Roughan, Matthew, et al. "Experience in measuring internet backbone traffic variability: Models metrics, measurements and meaning." Teletraffic Science and Engineering. Vol. 5. Elsevier, 2003. 379-388.