Global Min Cut

CS 456: Project 5

1 Introduction

The goal of this project was to analyze and compare the Edmonds-Karp algorithm with Karger's approximation algorithm for finding the global minimum cut of a graph. The global minimum cut problem consists of finding the lowest capacity cut through a graph. Being a NP Hard problem, the runtime of finding the exact solution using the Edmonds–Karp algorithm is $O(|V|^2|E|^2)$, and as such, we aim to experiment with Karger's approximation algorithm to discover its viability, and what trade-offs are made compared to the Edmonds–Karp algorithm.

2 Results

Graph	Exact Cuts	Contract1 Cuts	Contract2 Cuts	Contract3 Cuts	Contract4 Cuts	Contract5 Cuts
celegansneural.graph	1	1	9	4	1	12
Project4Graph2.txt	10	10	10	10	11	11
Project4Graph1.txt	5	5	5	5	5	11
polbooksWeighted.graph	2	5	3	4	7	4
karateWeighted.graph	1	1	2	2	2	2
florida-bay.graph	2	16	204	13	288	12
Graph	Exact (ms)	Contract1 (ms)	Contract2 (ms)	Contract3 (ms)	Contract4 (ms)	Contract5 (ms)
celegansneural.graph	1750.2084	81.0912	18.408	18.5579	17.5412	17.3655
Project4Graph2.txt	0.0775	0.5155	0.4025	0.4386	0.1804	0.4783
Project4Graph1.txt	0.0502	0.4511	0.4779	0.3802	0.4031	0.4454
polbooksWeighted.graph	102.1129	44.3344	21.2961	21.3725	22.2797	21.3039
karateWeighted.graph	3.1357	20.9576	1.2417	0.6967	0.7071	20.8144
florida-bay.graph	513.2117	25.1122	49.1835	24.0553	24.487	24.4457

Table 1: Minimum cut and associated runtimes for 10 iterations

Graph	Exact Cuts	Contract1 Cuts	Contract2 Cuts	Contract3 Cuts	Contract4 Cuts	Contract5 Cuts
celegansneural.graph	1	1	1	1	1	1
Project4Graph2.txt	10	11	11	10	11	17
Project4Graph1.txt	5	13	11	5	5	5
polbooksWeighted.graph	2	4	2	3	3	2
karateWeighted.graph	1	1	1	1	1	1
florida-bay.graph	2	2	13	2	2	13
Graph	Exact (ms)	Contract1 (ms)	Contract2 (ms)	Contract3 (ms)	Contract4 (ms)	Contract5 (ms)
celegansneural.graph	1750.2084	557.0621	503.0906	399.7002	413.8505	406.6934
Project4Graph2.txt	0.0775	0.3823	0.4282	0.4109	0.1151	0.4346
Project4Graph1.txt	0.0502	0.3987	0.369	0.341	0.3666	0.3754
polbooksWeighted.graph	102.1129	73.3894	54.0188	34.3985	53.5005	32.6633
karateWeighted.graph	3.1357	21.8032	21.3198	43.1202	21.163	0.8361
florida-bay.graph	513.2117	87.747	88.7737	84.243	127.649	88.0672

Table 2: Minimum cut and associated runtimes for n iterations

Graph	Exact Cuts	Contract1 Cuts	Contract2 Cuts	Contract3 Cuts	Contract4 Cuts	Contract5 Cuts
celegansneural.graph	1	1	1	1	1	1
Project4Graph2.txt	10	10	10	10	10	10
Project4Graph1.txt	5	5	5	5	5	5
polbooksWeighted.graph	2	2	2	2	2	2
karateWeighted.graph	1	1	1	1	1	1
florida-bay.graph	2	2	2	2	2	2
Graph	Exact (ms)	Contract1 (ms)	Contract2 (ms)	Contract3 (ms)	Contract4 (ms)	Contract5 (ms)
celegansneural.graph	1750.2084	774759.5932	703439.1859	687873.8018	716489.0127	689531.893
Project4Graph2.txt	0.0775	0.9838	1.9534	0.7784	0.7699	0.3623
Project4Graph1.txt	0.0502	0.6279	0.2741	0.6162	0.3017	0.5493
polbooksWeighted.graph	102.1129	6181.3018	6342.346	6557.8713	6485.526	6532.7274
karateWeighted.graph	3.1357	134.6114	110.5004	129.3092	110.1674	130.8909
florida-bay.graph	513.2117	41388.8641	42039.0769	43972.4187	43956.6237	43784.628

Table 3: Minimum cut and associated runtimes for $n^2 \ln n$ iterations

3 Analysis

When looking at the probability of not finding the minimum cut we know that when repeating the contraction algorithm $n^2ln(n)$ times that the probability of not finding the minimum cut is $1/n^2$. However, when running the algorithm only n times there is a rather noticeable change in the probability. Simplifying the probability equation down from $P_n = 1 - [1 - \binom{n}{2}^{-1}]^n$ to $P_n = 1 - [1 - \frac{2}{n^2-n}]^n$ it becomes apparent that while running the algorithm n times is moving toward a low probability of not finding the minimum cut, it is nowhere near as effective as $n^2ln(n)$ times. Where running n times the failure rate never really reaches a sufficiently small probability, whereas $n^2ln(n)$ would reach a point of failure that is sufficient on graphs with very small n values.

When also taking into account that we performed 5 repetitions, the probability of failure for $n^2ln(n)$ drops immediately to almost zero, even at n = 2. Where n still has at least a 1% overall failure rate. This is reflected in our data, where $n^2ln(n)$ did not fail on any repetitions and n failed a majority of the time but at least one of the five repetitions came up with the correct answer.

As for the runtimes, as expected with the Contraction Algorithm being $O(|V|^2 * |E| * log(V))$, which is at least a factor less than the Edmonds-Karp method, the approximation method is much faster even when ran $n^2ln(n)$ times. Therefore, with the accuracy of the $n^2ln(n)$ iteration Contraction Algorithm matching that of the exact method, and its runtime proving to be better, this experiment would seem to conclude that the $n^2ln(n)$ iteration approximation is the preferable choice of all options examined

4 Conclusion

Due to the insurmountable runtimes of the algorithm on a larger graph for $n^2 \ln n$ iterations, it is unfeasible to run the number of repetitions needed for a dataset large enough to conclusively show the probability of not finding the minimum cut converges to zero. Even with parallelization of the main iteration loop the runtime of larger graphs was still excessive. That being said, even five repetitions of the algorithm illustrates the disparity between the different iteration counts that where chosen for this experiment. The level of experimentation was a successfully demonstrated the viability of using Karger's algorithm as a solution to an otherwise difficult problem.