UNIVERSITY OF CALIFORNIA SAN DIEGO

An Empirical Examination of Planted Clique Heuristics

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Computer Science

by

Krytopher Wayne Ashley Weeton

Committee in charge:

Professor Russell Impagliazzo, Chair
Professor Shachar Lovett
Professor Ramamohan Paturi

2022

The Thesis of Krytopher Wayne Ashley Weeton is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Impagliazzzo for his support and help throughout my entire time at UCSD. In both this research and in my broader academic goals, his advice continues to be irreplaceable.

I would also like to thank Pawan Pajela, Sam Mcguire, and Jiazheng Liu for their indispensable help with background material and ideas. Additionally, I want to acknowledge and thank my family for their continued support, without which I would not have been able to finish this work.

ABSTRACT OF THE THESIS

An Empirical Examination of Planted Clique Heuristics

by

Krytopher Wayne Ashley Weeton

Master of Science in Computer Science

University of California San Diego, 2022

Professor Russell Impagliazzo, Chair

Local search heuristics provide generalized ways of solving difficult computational problems. In this research we examine a few simple heuristics applied to the Planted Clique Problem, a problem for which there are a number of lower bounds for sophisticated algorithmic techniques. The heuristics run were able to solve the problem at a well known threshold, providing support for the efficacy of simple heuristics.

Introduction

The question of how to deal with NP-Hard problems in practice is a well studied one in the topic of theoretical computer science. A common approach is to use techniques geared towards average case complexity, using the fact that the same instances which make a problem NP-Hard are often instances which almost never arise in practice.

As one specific technique geared towards solving average case problems in general, heuristics present a simplistic, generalized, and, often, highly effective approach towards developing real world algorithms to problems we can not solve efficiently in the worst case.

Although the term 'heuristic' is fairly ambiguous, in this paper it will be used in reference to the algorithms studied below. These algorithms represent different techniques of accomplishing the same intuitive goal: visualizing the possible solutions to a problem as a 'search space', and then efficiently traversing this search space in order to find the optimal solution.

The goal of this research is to work towards developing concrete and unified ways of demonstrating and explaining the efficacy of heuristics that are commonly used every day. Specifically, this paper will present the results gathered so far towards understanding the conditions where, and extent to which, different heuristics are able to efficiently solve the Planted Clique Problem.

Chapter 1

Definitions and Notation

1.1 Cliques and Random Graphs

For a graph $G = (V, E)$ a *clique* is defined to be a subset of vertices $S \subseteq V$, where every possible edge between vertices in S is present in G. Formally for each $u, v \in S$, $\{u, v\} \in E$. Further, for the remainder of this paper we sample random graphs from two distributions. The first distribution is the standard Erdos-Renyi random graph model, $G_{n,p}$ where a graph with a fixed number, n, of vertices is generated by connecting each pair of vertices with an edge with an independent probability p.

For notation purposes, we denote the number of edges in a subset $S \subseteq V$ as $E(S)$, and the degree of some vertex $v \in V$ in the graph as $d(v)$. We also use the notation $d_S(v)$ to denote the number of edges between a vertex v and a subset S. Formally,

$$d_S(v) = \sum_{u \in S} \mathbb{I}_{\{u,v\} \in E}$$

1.2 Planted Clique Problem

We define the planted clique problem similarly to [5]. The inputs we consider are generated through a two-step process. Firstly, a graph is sampled from $G_{n,p}$. Then a uniformly chosen subset of vertices of some size k is fixed as a clique in the graph. We denote the probability

space from which these graphs are sampled as $G_{n,p,k}$. Specifically, for the remainder of this paper we make use of the probability space $G_{n,1/2,k}$.

We denote the planted clique placed in the sample as K.

1.3 Search Graphs

Before we begin to explain heuristics, we first need to rigorously define the term 'search graph', which is fundamental to the heuristics for which we present analytical results. Abstractly, a *search graph* is a graph whose vertices represent either solutions or near solutions to a problem, and whose vertices are connected based on a well defined condition of 'adjacency' for solutions.

For the planted clique problem, the *search graph* we use, with respect to an input $G = (V,E)$ sampled from $G_{n,p,k}$, is a graph whose vertices are arbitrary subsets of V, and an edge exists between two subsets $S,K \subseteq V$ if and only if $|S/K| = 1$ and $|K/S| = 1$. Intuitively if S can be converted to K by removing an element of S and adding an element not in S. Note that this definition if symmetric so the search graph we consider is undirected.

Chapter 2

Heuristics

2.1 Go With the Winners

Go with the Winners as a heuristic was originally introduced independently by Dimitriou and Impagliazzo [4], and Aldous and Vazirani [2]. The original application in [2] was to develop an algorithm to find deep vertices in a tree. To accomplish this the GWW algorithm was defined as an iterative algorithm which generates B particles at the root of the tree, and then moves each particle to a random child, deleting particles which are at leaves. The process then randomly replicates the particles until it has B again and repeats this process until all particles are at a leaf.

Dimitriou and Impagliazzo [4] expand on this algorithm and give a rigorous definition of Go With the Winners that works directly on search graphs, rather than applied to the abstracted problem of finding the deepest node in a tree. This Version of Go with the Winners generates a number of random particles in the search graph, then takes a *thresholded random walk*, a random walk across the subgraph of the solution graph with only the vertices that have at least a specified payoff value.

For this specific problem, we traverse the search graph defined in the prior section, with the payoff value of a subset of vertices S being the number of edges in S. Intuitively, this corresponds to restricting our search space to denser subgraphs as the heuristic runs. We restrict the density of edges in each generation to be the median of the prior generation's particles

densities.

2.2 Successive Augmentation

We consider a *successive augmentation* algorithm, in the context of the planted clique problem, to be an algorithm which begins with a constant size subset of the planted clique and then performs a single pass on all other vertices in the graph, and adds each vertex independently according to a predicate $f : \mathscr{P}(V) \times V \to \{true, false\}$.

Note that we allow our successive augmentation algorithm to begin with a constant size subset of the planted clique because in practice we can achieve this through exhaustive search. We simply loop over all constant size subsets of vertices and run successive augmentation on each, returning the largest clique found in any of the runs. We further define $f(S,v)$ to be,

$$d_S(v) \geq \frac{|S| - |S \cap K|}{2} + |S \cap K| + h = \frac{|S| + |S \cap K|}{2} + h$$

The motivation for this specific predicate lies in the fact that the edges between v and S are only ever considered when this predicate is evaluated for v. This simplifies the probabilistic analysis, as we can consider the graph to be generated as we run the successive augmentation algorithm. Thus for a vertex v we have two possible distributions for $d_S(v)$,

$$d_S(v) \sim \begin{cases} Bin(|S|, 1/2) & v \notin K \\ Bin(|S| - |S \cap K|, 1/2) + |S \cap K| & v \in K \end{cases}$$

It is also worth noting that we make use of the value $|S \cap K|$ in the predicate, which, to compute, requires the use of a powerful oracle that can find the intersection of a given subset with the planted clique. Intuitively, this is essentially the precise problem the successive augmentation algorithm attempts to solve. The goal in using such a powerful oracle for the predicate is to provide a simplistic predicate for analysis. In doing so, we hope to develop probabilistic bounds on precisely the intersection of the partial solution with K as the algorithm runs. These bounds

could then be used to remove the reliance on $|S \cap K|$ in the predicate.

The other feature of note here is the value h. We consider h to be a hyper-parameter provided. To motivate it's usage it is worth considering the following

$$\mathbb{P}[\text{v added to S}] = \begin{cases} \mathbb{P}[Bin(|S|, 1/2) \geq \frac{|S|}{2} + \frac{|S \cap K|}{2} + h & v \notin K \\ \mathbb{P}[Bin(|S| - |S \cap K|, 1/2) \geq \frac{|S| - |S \cap K|}{2} + h & v \in K \end{cases}$$

Using simpler chernoff bounds,

$$\mathbb{P}[\text{v added to S}] = \begin{cases} \leq exp(-(|S \cap K| + 2h)^2/(4|S|) & v \notin K \\ \geq exp(-h^2/(|S| - |S \cap K|)) & v \in K \end{cases}$$

Intuitively h represents the level of confidence we need for a vertex v to add it to our solution. It is formally represented in these exponential chernoff bounds.

Chapter 3

Empirical Results

The experimental results gathered are exploratory and represent an initial dive into the questions presented above. Further work is planned to continue and expand upon the results presented here.

3.1 Experimental Structure

All empirical results were gathered on a Macbook Pro with a 2.2 GHz Quad-Core Intel Core i7 processor. Experiments were written using Python 3.9.6 and all code is accessible at [1]

Note that all experiments were written and carried out for the *planted independent set problem*. This problem is analogous to the planted clique problem for the specific distribution $G_{n,1/2,k}$ as finding a clique in G corresponds to finding an independent set in G^R where an edge is present in G^R if and only if it is not present in G, and performing this operation on samples doesn't change the input distribution.

3.2 Successive Augmentation

Overall, successive augmentation proved to be an effective heuristics for finding planted cliques of size $\sqrt{n}$, for graphs up to size $20,000$. Our results suggested not only that successive augmentation was able to find such planted cliques, but it could do so consistently. Additionally, our experiments suggested that h played a pivotal role in the effectiveness of the

heuristic, int hat it balances the size of the final solution and the size of the intersection in the final solution.

3.2.1 Effectiveness of Successive Augmentation

The first experiment we ran, and the first results presented are simply individual runs of the successive augmentation algorithm described above. For these algorithms we provided the heuristic with a small subset of the planted clique, which could be accomplished through exhaustive testing in practice.

The reasoning for this is successive augmentation in this form fundamentally builds on a partial solution, and as such requires an initial solution to begin with. With these experiments the goal was to understand, at a baseline, how related the size of the partial solution and it's intersection with the planted independent set was. In an ideal situation these values would be the same as the successive augmentation heuristic would only add vertices in the planted independent set.

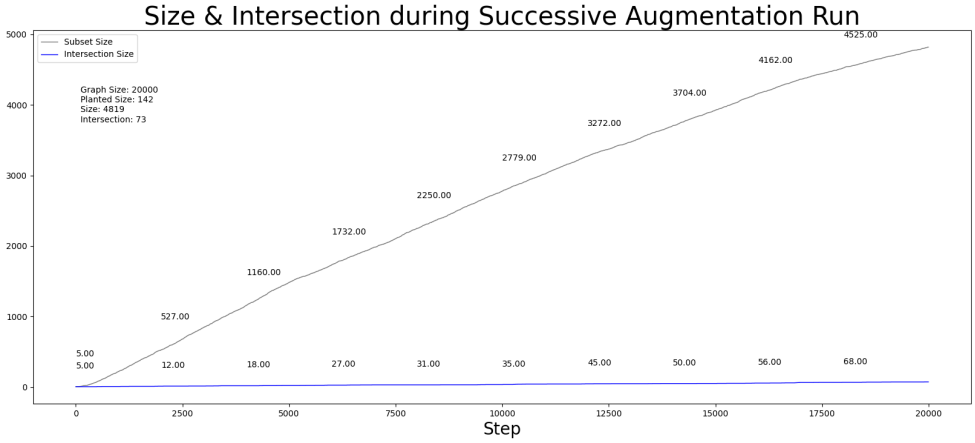

Figure 3.1. Size and Intersection of Successive Augmentation $h = 0$
$n = 20000$ vertices and a planted size of $\sqrt{n} = 142$. Gray line is Subset size of partial solution, blue line is the size of the intersection with the planted independent set.

In 3.1, we track the progress of a successive augmentation algorithm throughout a single run on a larger graph, (10000 vertices) by graphing the size of the partial solution, in gray,

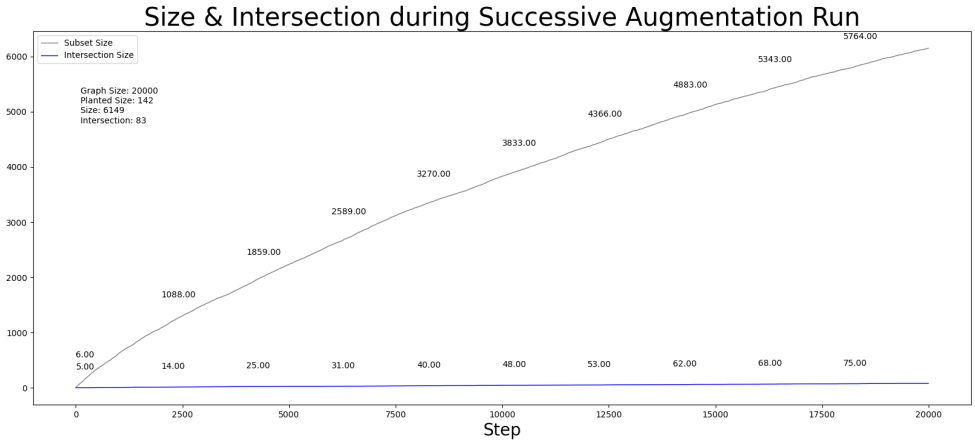

Figure 3.2. Size and Intersection of Successive Augmentation $h = -5$ $n = 20000$ vertices and a planted size of $\sqrt{n} = 142$.

and the size of that subsets intersection with the planted independent set in blue. The horizontal axis of this graph being the step of the algorithm, where each 'step' corresponds to the algorithm considering a single vertex.

Along with these results, we mapped runs that varied the hyper-parameter h attempting to understand if the hyper-parameter would sufficiently cause the successive augmentation algorithm to be more selective about the vertices it chose to add to the final solution.

In figure 3.2 we set h to a negative value, effectively making it so vertices which were in the planted independent set were added with larger than a 50% chance. As expected, the size of the subset increases significantly over the course of the run, where the intersection with the planted subset remains roughly constant.

On the other hand, as we moved h to a small positive constant we began to see the successive augmentation algorithm perform better. In 3.3 we saw a small improvement; however, the primary improvement came from 3.4 where we began to see a significant intersection with the final subset found and the planted independent set in the graph.

One thing worth noting is that these results only run the first 'phase' of a successive augmentation algorithm. The goal in this analysis is to show that the successive augmentation algorithm finds a subset with a sufficiently large intersection with the planted independent set to

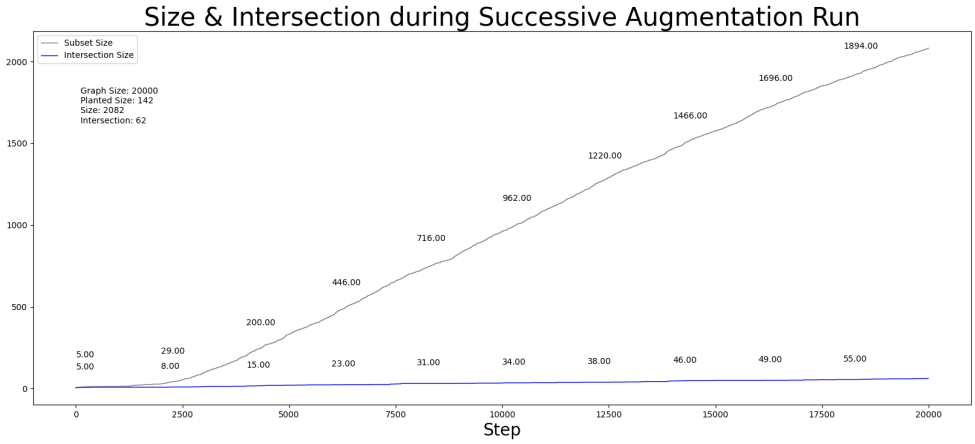

Figure 3.3. Size and Intersection of Successive Augmentation $h = 1$
$n = 20000$ vertices and a planted size of $\sqrt{n} = 142$.

allow for existing techniques to recover the planted independent set from the subset.

In the run provided in 3.4 for example, the size of the planted clique in the entire graph is precisely $\sqrt{n}$; however, the final subset chosen by the successive augmentation algorithm has size 17 with intersection of $12 \approx 3\sqrt{17}$.

The results of these experiments can be summarized as providing a suggestion of the effectiveness of successive augmentation. With larger values of h, corresponding to the heuristic being more selective about the vertices it chooses to add to the partial solution, the size and intersection of the partial solution remain fairly correlated with each other. This suggests that it is reasonable to prove these observed results may be able to be converted to analytical proofs of correlation.

The other important note here is that, expectedly, the hyper-parameter h is fundamental to the behavior of successive augmentation. With h set to a larger value we get a stronger correlation between the size of the final solution and it's intersection with the planted independent set; however, with smaller h we observe a much larger final solution. Because the goal of the heuristic is to find a 'large' subset with high correlation, this suggests that any analytical results will need to balance this hyper-parameter h, finding an optimal range for both strong correlation and provably large final solution.

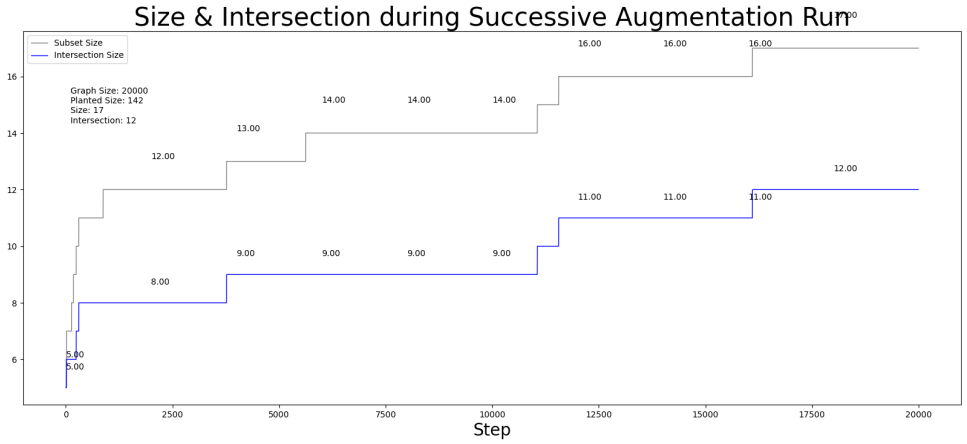

Figure 3.4. Size and Intersection of Successive Augmentation $h = 5$ $n = 20000$ vertices and a planted size of $\sqrt{n} = 142$.

Although these results suggest the broad effectiveness of successive augmentation and begin to answer questions about what values h should take, they left open the important question of consistency. Is successive augmentation able to find a subset with high intersection with the planted clique with high probability, a constant amount of the time, or with vanishing probability? Although the experiments were run with $20,000$ vertices, only a single run was done.

3.2.2 Consistency of Successive Augmentation

To address the question of successive augmentation's ability to consistently recover the planted clique we repeated the simple experiment of running successive augmentation with $h = 5$, an initial subset of only size 5, on $G_{20000,1/2,\sqrt{20000}}$ graphs 100 times and compared these runs to see if, on average, we observed the successive augmentation algorithm successfully finding a subset of vertices that has a large intersection with the planted independent set.

Specifically, we segmented the runs in all permutations of two possible conditions that we expected may be useful in improvements to the heuristic and in analysis of the heuristic. The first of which was those satisfying an *invariant* throughout the run of the heuristic. The invariant we required to include a run in this category is that, at every step of the heuristic, the subset

(with some size s) has intersection at least $\sqrt{s}$ with the planted independent set. Thus setting the requirement that the relative size of the planted independent set in the partial solution not shrink.

The second classification we gave for runs as we analyzed them was that of runs which didn't satisfy a specific *threshold* near the beginning of the run. The intuitive motivation for such a classification was to empirically analyze whether random restarts would be useful for successive augmentation.

Throughout the 100 runs of the heuristic, every single run satisfied the invariant throughout the entire run, and 80 out of the 100 satisfied the threshold of having intersection at least $2sqrts$ after analyzing $1/4$ of the vertices, where s is the size of the subset at that point.

It is worth noting here that the decision for this threshold was fairly arbitrary and more specific experiments would need to be done to analyze whether this was an optimal point in the algorithm to consider restarting. This avenue was not pursued as the experiments suggested that successive augmentation was, broadly, successful in recovering a significant subset of the planted independent set.

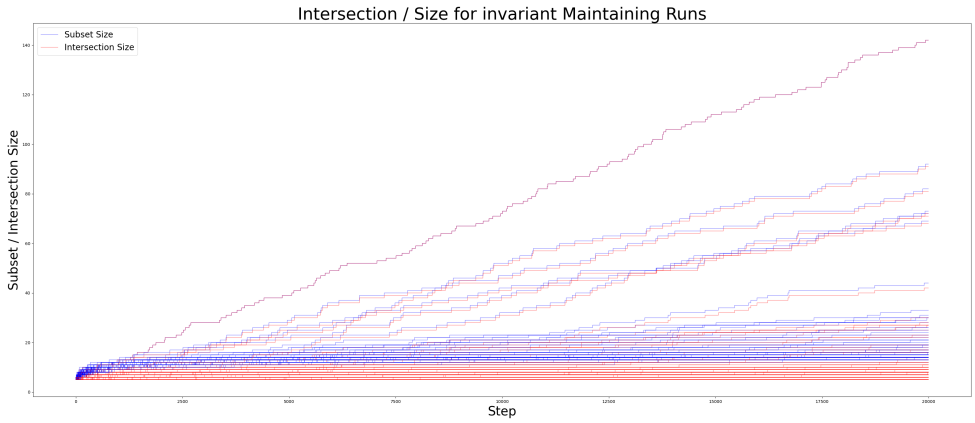

Figure 3.5. Intersection and Size for Invariant Maintaining Successive Augmentation Runs
100 runs with $n = 20000$ vertices and a planted size of $\sqrt{n} = 142$, with $h = 5$

In our analysis of this we began to also observe the fundamental tradeoff of this approach. As h grows larger, the intersection size of our final solution increases. As shown in figure 3.6 the average size our final solution was fairly small, only increasing from an initial size

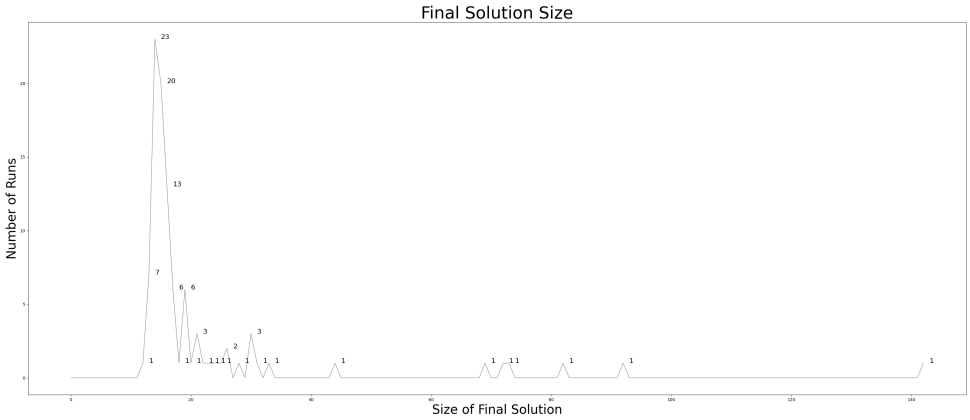

Figure 3.6. Final Solution Sizes for Successive Augmentation Runs
100 runs with $n = 20000$ vertices and a planted size of $\sqrt{n} = 142$, with $h = 5$

of 5 to an average final size of 21.52.

On the other hand, having h be a reasonably large value was fundamental to getting a large intersection with the planted independent set. In these runs when h was set to 5 we ended with, on average, 58.76% of the vertices in our final solution being from the planted independent set. A complete distribution of the relative size of the planted independent set in the final solution is provided in 3.7. As seen earlier in 3.1, setting h too low could decrease this relative intersection size.

All of these results suggest some initial trends for the behavior of successive augmentation. Broadly, the simple heuristic seems to be able to find a significantly sized subset of the planted independent set. Additionally, fundamental to this approach is the balancing of the hyper-parameter h. Balancing h between low values, which lead to larger subset, and higher values, which lead to higher relative intersection with the planted independent set, is crucial to the success of successive augmentation.

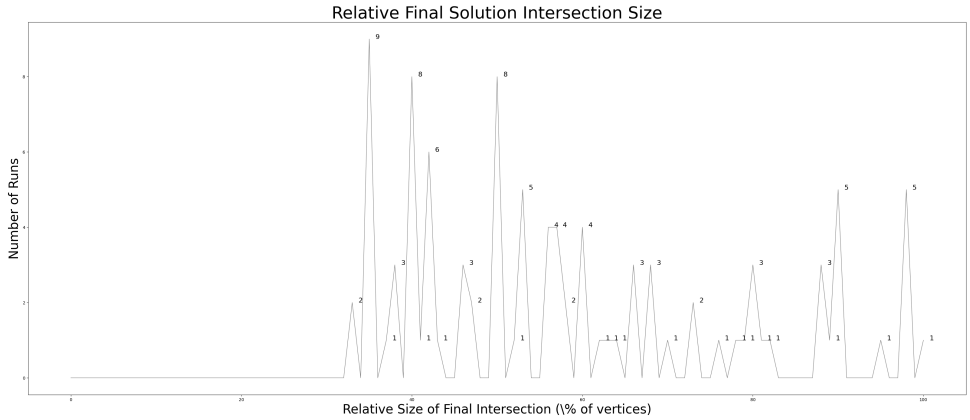

Figure 3.7. % of Vertices in Final Solution also in Planted Independent Set
100 runs with $n = 20000$ vertices and a planted size of $\sqrt{n} = 142$, with $h = 5$

3.3 Go With the Winners

In addition to analyzing the efficacy of successive augmentation, we began to run experiments to analyze how effective Go With the Winners was at solving the equivalent problem. We thought it important to include these results for completeness, although the results are only of an initial experiment running Go With the Winners for a variety of graph sizes n, and broadly analyzing the final intersection and sizes of the solutions.

As an initial guess, Go With the Winners was run with $2\sqrt{n}$ particles and considered the solution graph to be all subsets of size $n^{2/3}$. Each generation removed the $1/2$ of particles with the lowest *density*, and took a random walk of length $\log(n)$. The process stopped when the density of all solutions was below 0.1.

Very simply, the results in figure 3.9 and 3.8 broadly suggest Go With the Winners is successful in recovering a significant portion of the planted clique. The results are on fairly small graphs and any complete justification for the efficacy of Go With the Winners requires more empirical and analytical justification.

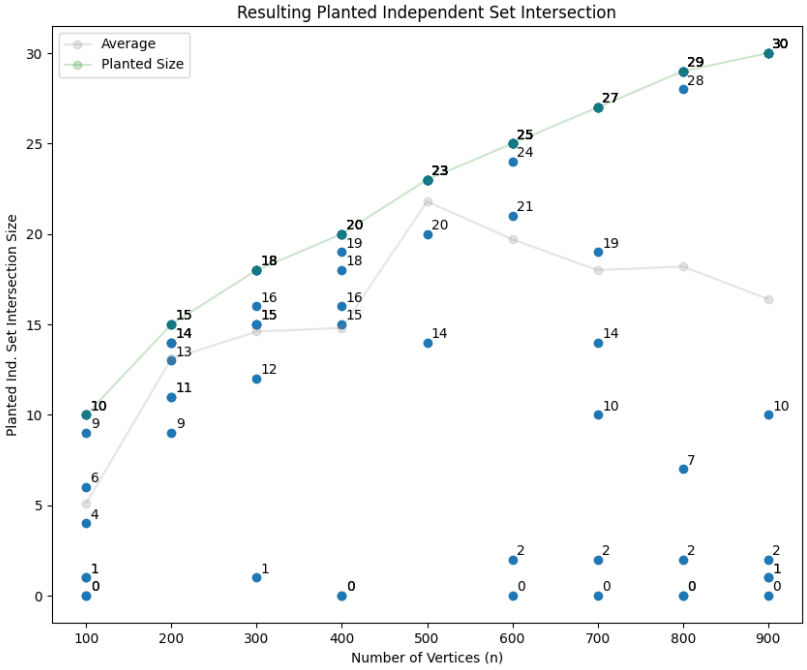

Figure 3.8. Intersection Sizes of final solutions for GWW Heuristic

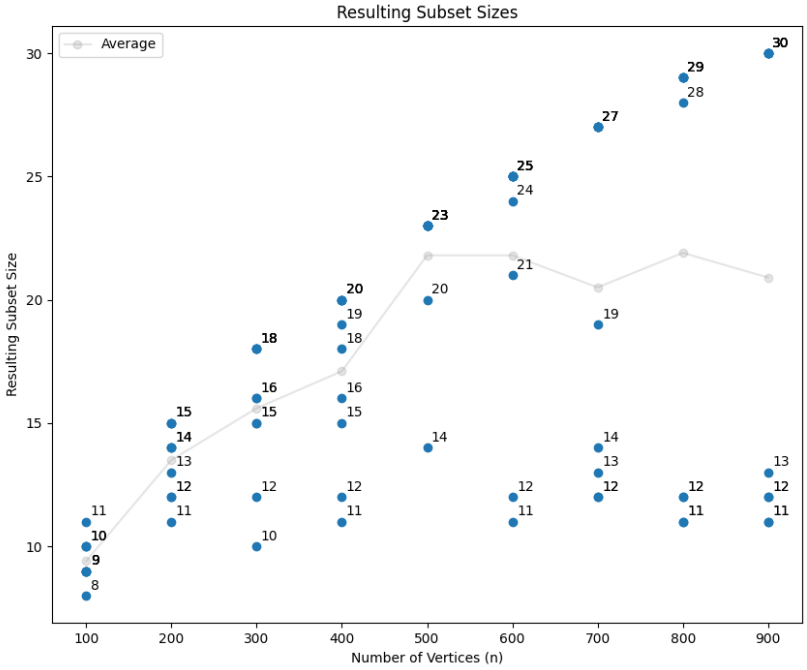

Figure 3.9. Subset Sizes of final solutions for GWW Heuristic

Chapter 4

Next Steps

The next steps for this research are primarily to expand on the empirical results through more extensive and granular testing and to concretely apply these results to the analytical world. Empirically, the goal is to test the Go With the Winners heuristic more through experiments similar to those found in [3]. These experiments would help analyze the expansion properties of the search graphs used, and also begin to map out the conditions necessary for local optimization to be sufficient to find the planted solution.

Analytically, the next steps are to continue analysis of the Successive Augmentation heuristic. Fundamental to this heuristic is the tradeoff between the size and intersection of the final solution, and in attempts to proof of efficacy, this has become an issue. Further work is planned to refine these proofs and begin to map out under what regimes the hyper-parameter h is able to find large subsets of the planted clique.

Also fundamental to further work is expanding the scope of the experiments already done. Much of the experimental work points towards success of these heuristics; however, do not specify under which specific parameters the algorithms succeed. To accomplish this, a more robust code base is required with more extensive computational testing. In this regard, the next steps for this experiment is to expand on the code used for testing to optimize it and run larger and more extensive tests that require more computational power.

Bibliography

[1] Independentset github repository. https://github.com/KrystopherWeeton/IndependentSet.

[2] D. Aldous and U. Vazirani. ”go with the winners” algorithms. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 492–501, 1994.

[3] Ted Carson. Empirical and analytic approaches to understanding local search heuristics. 2001.

[4] Tassos Dimitriou and Russell Impagliazzo. Towards an analysis of local optimization algorithms. 12 2000.

[5] Mark Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3(4):347–359, 1992.