IMPROVING THE EFFICIENCY OF GRAPH-BASED DATA MINING WITH APPLICATION TO PUBLIC HEALTH DATA

By

YAN ZHANG

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

DECEMBER 2007

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of YAN ZHANG find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

IMPROVING THE EFFICIENCY OF GRAPH-BASED DATA MINING WITH APPLICATION TO PUBLIC HEALTH DATA

Abstract

by Yan Zhang, M.S.
Washington State University
December 2007

Chair: Lawrence B. Holder

Relational data are most naturally represented as a graph, with the entities as nodes and the relations between them as edges. Graph-based data mining looks for patterns that can best compress and represent the dataset and thus extract useful information from the data.

An important topic in graph-based relational learning is its efficiency. A suitable search algorithm can largely improve the efficiency of substructure mining by finding better patterns in less time. In the thesis, performance of different search algorithms for graph-based relational pattern learning is studied. A complete graph space search algorithm, an efficient depth-limited search, and heuristic searches including beam search, hill climbing, stochastic hill-climbing, and simulated annealing (SA) are designed and implemented for pattern search in graph-based space. We also designed two new algorithms, SA-Greedy and Hill-Climbing with Stochastic Escape (HCSE). All seven algorithms are evaluated and compared by running with several depth limits on several datasets with Subdue, a graph-based data mining tool. The experimental results show that SA-Greedy finds best substructures in less time than the other search algorithms.

The application of graph-based data mining in public health domain is also conducted. The Pandemic dataset is represented as a graph. Its intrinsic pattern is explored by graph-based data mining. Different search algorithms are run on it and show results consistent with the previous experiments.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

Large amounts of relational data are being generated in various domains nowadays. It has been far beyond human being's ability to observe patterns inside the data. Data mining aims to automatically discover patterns in the data and finally apply the patterns to predict new data.

Many datasets have relations between the data entities, either explicitly or implicitly. For example in the internet, although the existing search engines use a linear feature match [13], the link between the web pages defines a relation to them. Data entity relations can also be found to exist in protein structure [15], web search [3], criminal networks [6], and credit fraud [12]. In this research, the pandemic dataset is also a structural dataset with relations: activities between people relate them together; infections that occur between people also add relationship to the people entities in the dataset.

The relational data are most naturally represented as a graph, with the entities as nodes and the relations between them as edges. With the graph representation, looking for patterns in data can be accomplished by finding the subgraph that best represents the graph. In searching for the best pattern in the graph, several graph data mining systems have been developed, either searching for the most frequent patterns (eg., gSpan [16]), or patterns that best compress the graph (eg., [2]).

One critical problem in graph-based mining is the huge amount of running time when mining large graphs. A way to avoid exploring the exponential space is to use heuristic search. Subdue [2] employs beam search. Beam search tries to retain the beamwidth

number of best substructures in each depth of searching. But this greedy approach faces the difficulty of finding only the local maxima. Global maxima may be missed in beam search so it may miss the best pattern.

Several search algorithms have been well studied, including depth-limited, hill climbing, simulated annealing, and beam search. But only beam search has been applied in graph-based data mining. Also no comparison of these different search methods has been conducted in the context of graph-based data mining. Such a study is important for finding the most efficient search techniques for identifying the best-valued substructure in a graph.

In this thesis, we designed and implemented the search strategies in the space of substructures in a graph. To study the performance of the search algorithms in graph-based data mining, we conducted experiments using search variants of Subdue, a compression-based graph mining system. However, the results of this search algorithm study will add insight to other graph mining approaches in searching for the best patterns in a graph mining task. The search algorithms are run on various datasets from different domains, including the new pandemic dataset. Results show that the SA_ greedy approach finds best substructures in less time than other search strategies.

In the following chapters, chapter two introduces the concept and basic procedures of graph-based data mining. Evaluation methods used in graph-based data mining including MDL heuristic are also explained. Chapter three presents details of search algorithms we studied in graph-based data mining. The complexities of the search algorithms are discussed. Experiments are conducted on datasets from various domains. The conditions of the experiments are stated in Chapter four. The results are also given in Chapter four.

Chapter five summarized three datasets in the public health domain. Data relations are discovered and graph representations are shown. Both graph representation for unsupervised learning on the pandemic dataset and graph representation for supervised learning on it are explored. And chapter six gives the results of the graph-based search algorithms on the pandemic dataset. Chapter seven uses the results to compare the different search algorithms. And conclusions are drawn in chapter eight.

CHAPTER TWO

GRAPH-BASED DATA MINING

Graph-based data mining addresses the need to discover knowledge in large relational datasets. Some of the datasets have a characteristic of structural components, either temporal, spatial, or various relations. The structure can be naturally represented with a graph, with nodes as entities or attributes, and edges as relations between the two entities. To identify the common substructures in the data would be essential to discover knowledge in such a relational dataset.

Search algorithms employed in existing graph-based data mining systems

The gSpan [16] algorithm explores depth-first search in frequent graph mining. CloseGraph [17] which introduces mining closed frequent graph patterns in large graph datasets, is built on gSpan. SiGraM [11] attempts to find the subgraphs most frequently embedded within a large sparse graph with two algorithms: HSIGRAM explores the nodes in a breadth-first fashion, whereas VSIGRAM explore the nodes in a depth-first fashion. DSPM [1] mines all frequent subgraphs in a large set of graphs, exploring the search space in a reverse depth-first fashion. SPIN [8] introduces an efficient maximal subgraph mining algorithm based on mining all frequent trees.

The search used can be either breadth-first or depth-first depending on the particular tree mining algorithm, though depth-first is preferred since it requires much less memory utilization. Mining the complete set of substructures in a graph will require mining an exponential number of substructures.

A way of reducing the search space of sub-graphs is introduced by Subdue. Subdue employs beam search which keeps a beam-width number of best substructures in each extension. Another way to reduce the exploration space is attempted using evolutionary programming (EP) [19]. The EP method randomly selects one mutated substructure from the mutated Childlist during the point mutation of a chromosome. The process of point mutation is similar with the extension process in Subdue by extending the parent instances in every possible way. Then the EP method selects the chromosome based on its fitness. A number of copies of the chromosome are made according to its relative fitness, and selection to a chromosome is performed stochastically. So the better chromosome has less chance of disappearing from the population.

In the following work, we compare performance of beam search, depth-limited, stochastic based search and simulated annealing based new search algorithms.

Graph-based data mining using Subdue

Subdue is a graph substructure discovery system based on compression. Subdue aims to find substructures that best compress the graph, and thus extract useful information from the data. It has been successfully used in various domains, including bioinformatics, social networks and web structure [2].

Subdue can perform both unsupervised learning and supervised learning. In unsupervised learning, the structural data is represented as a labeled graph, and then Subdue discovers substructures in the graph that best compress the graph. It starts from initial substructures of single vertices that have at least two instances in the graph; and then extends them in every possible ways according to the graph. In supervised learning, Subdue

will try to find the substructure that best compresses the positive graph(s), and not compress the negative graph(s).

The search process in Subdue graph-based data mining starts with an initial set of substructures consisting of the uniquely labeled single vertices. With the initial set as parent, the instances in the substructures of the set are extended in all possible ways by one edge. By doing graph isomorphism and graph inexact matching, a new set of substructures based on the extended instances is produced. This new set of substructures is now set as the parent list, and used to generate another iteration of extensions. The total number of iterations to extend the substructures is denoted as *depth_limit*. This extend-and-evaluate search process is terminated upon exhaustion of the extension space or upon reaching a user-specified limit on the number of extensions. The algorithm returns the list of substructures with the best compression values. In the next chapter we describe the different search algorithms for exploring the space of substructures for those that maximally compress the input graph.

In a complete space search, all generated substructures are kept on the child list, and set as the new parent list for the next-iteration extension. While in heuristic search algorithms, not all generated substructures are kept on the child list. The substructures are evaluated and only the selected ones will be retained and further extended. There are three evaluation methods used by Subdue. They are MDL, size, and setcover. We discuss the three methods in the following section.

We should also note that, unlike beam search that will always keep the best-value substructures, other search algorithms in our research use different selection rules.

Simulated annealing based search approaches select the substructure to be kept based on a "time schedule"; greedy stochastic based search methods randomly select the next one based on value distributions among the substructure list.

The MDL heuristic

The Minimum Description Length (MDL) principle states that the best theory for describing the dataset is the theory that minimizes a dataset's description length [14]. In graph-based substructure discovery, the MDL principle can be applied as: the best pattern for representing the dataset is the one that best compresses the graph. So under MDL principle, the value of a substructure is evaluated based on how well the substructure compresses the dataset. The formula is given in formula 1.

$$value(S,G) = \frac{DL(G)}{DL(S) + DL(G \mid S)} \qquad (1)$$

In formula 1, G is the graph to be explored, and S is the substructure under evaluation. $DL(S)$ is the "description length", or number of bits required to encode S, and $DL(G|S)$ is the number of bits required to encode G after being compressed with S. If a negative graph G_n is present, the value of the substructure can be evaluated based on how well the substructure compresses the positive graphs, and at the same time does not compress the negative ones. The formula is shown in 2.

$$value(S,G_p,G_n) = \frac{DL(G_p) + DL(G_n)}{DL(S) + DL(G_p \mid S) + DL(G_n) - DL(G_n \mid S)} \qquad (2)$$

Evaluation based on size

The value of a substructure based on size can be expressed as formula 3

$$value(S,G) = \frac{size(G)}{size(S) + size(G \mid S)} \qquad (3)$$

where $size(G)$=#*vertices*(G)+#*edges*(G). ($G|S$) still denotes the graph G compressed with S.

The size method is faster to compute than the MDL method. But the size method is less consistent than MDL in measuring the true compression because size method does not take into account the labels on vertices and edges, or the directions on the edge [18].

Set Cover Evaluation

In set cover, the value of a substructure S is computed by adding the number of positive examples containing S and the number of negative examples that do not contain S, divided by the total number of examples.

Conclusion

In this chapter, search algorithms employed in existing graph-based data mining systems are summarized. The search process of the Subdue graph-based data mining system is described. Subdue starts pattern search with an initial set of substructures consisting of the uniquely labeled single vertices. The substructures in the set are extended and evaluated. Then the selected children are kept to begin another iteration of extension. Three evaluation methods that are available in Subdue are also introduced in this chapter. Different ways of selecting the children for further extension are explored in the following chapter.

CHAPTER THREE

DESCRIPTION OF GRAPH-BASED SEARCH ALGORITHMS

Search algorithms are developed based on the need to find the best substructures in the least amount of time. In this chapter, an efficient depth-limited search, and heuristic searches are designed and implemented for pattern search in graph-based space.

The depth-limited search algorithm keeps the complete extension set, and thus is guaranteed to find the best substructure within the depth limit. But the depth-limited method suffers from the exponential time complexity which prevents it from finding the best patterns that can be reached only with a large depth of extension.

In heuristic search algorithm design, the substructure extension space can be searched with a much larger extension limit. So the fundamental consideration in heuristic graph search is how to select and retain child substructures from the list of extensions. Greedy approaches retain the substructures with best values, while simulated annealing based approaches give other substructures still some chance, since a worse current substructure may still be able to lead to a better substructure with further extension in compression-based graph mining. In the thesis, heuristic search approaches including beam search, hill climbing, stochastic hill-climbing, and simulated annealing (SA) are implemented. We also designed two new algorithms, SA-Greedy and Hill-Climbing with Stochastic Escape (HCSE). All seven algorithms are evaluated and compared by running with several depth limits on several datasets.

The time complexities of algorithms are described below in terms of the size of the input graph (which is its number of edges $|E|$) and the user-specified extension depth limit d.

Since every search algorithm involves extension of substructures, to simplify our expression and to compare the search algorithms, we represent complexity of extension using the symbol o_{ext}.

Beam Search

Beam search uses a heuristic function to evaluate substructures and keeps a beam-width number of the most promising substructures in each step. Beam search is the original method used by Subdue [2]. It starts with substructures consisting of all vertices with unique labels that appear more than once in the graph. The substructures are extended in all possible ways by adding an edge. The extended substructures are evaluated according to the *MDL* heuristic as described earlier, and only the beam-width numbers of best substructures are retained for the next extension step. This extend-and-evaluate process is repeated until the substructures can be extended no further or a limit on the number of extensions is reached. Pseudocode for this algorithm is given in Algorithm 1.

Since in each iteration, a constant number of beam-width substructures are set as parent, and only the beam-width number of extension is made, there are a total of *d* numbers of such iterations. So the time complexity of beam search is given as in formula 4.

$$T=beamwidth*o_{ext}*d \qquad (4)$$

The default parameter of beamwidth in Subdue is 4. In the following experiments, we only compare the performance of algorithms for different graphs and different depth limits. So we can say *beamwidth* is constant, and the complexity of beam search is $O(o_{ext}*d)$. We should note here that the term o_{ext} is a function of the input graph. So the time complexity is

still a function of the size of the input graph. Though the time is linear to the depth limit, the o_{ext} term might actually be more than linear in terms of the input graph size. But since every algorithm has the same term of o_{ext}, leaving the term as o_{ext} does not affect the comparison of time complexities between algorithms.

Algorithm 1: Graph-based Beam Search

Input: Graph, beam width, limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with unique vertex labels;

While parentList not empty **and** limit >0 **do**

childList ={};

For each parentSub in parentList **do**

extendList = extend each instance of parentSub in all possible ways;

For each extendSub in extendList **do**

Evaluate extendSub;

Insert extendSub into childList in order by value;

If Length(childList) > beamwidth **then**

Delete the last child in childList;

Insert parentSub into discoverList in order by value;

limit = limit - 1;

parentList = childList;

Return discoverList;

Efficient Depth-limited Search

The depth-limited search also starts with substructures consisting of all vertices with unique labels that appear more than once in the graph. The substructures are extended in all possible ways by adding an edge. The extended substructures are evaluated according to the *MDL* heuristic as described earlier. But in extension, the substructure at the head of the parentList is popped and evaluated. If its extension is not null and its depth does not reach the depth_limit, then it is extended. Otherwise, either it is kept in the discovered list if its value is among the best substructures seen, otherwise discarded. After extension, every extended substructure is pushed on the front of the parentList. And the next pop-evaluate-extend process starts until the parentList is empty.

The depth-limited search strategy [4] explores a complete substructure space within the limit number of extensions. Pseudocode for depth-limited search is given in Algorithm 2.1. The time complexity of depth-limited search is the sum of the numbers of substructures at each depth, which is

$$O(o_{ext} * |E|^{d+1}) \qquad (5)$$

in terms of graph size $|E|$ and extension depth d.

Algorithm 2.1: Nonefficient Depth-limited search

Input: Graph, Limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with non-unique vertex label;

While parentList not empty **do**

parentsub = head of parentList;

Insert parentsub into discoverList in order by value;

If depth of parentsub **less than** Limit **then**

extendList = extend parentSub in every possible way;

Push every extendSub in extendList into parentList;

Return discoverList;

An improved depth-limited strategy was developed to enhance the performance and hence reduce the running time. Since a particular substructure may be produced repetitively by extensions from all the initial nodes that are contained in the substructure, an improvement can be made by eliminating the duplicate generation of a substructure.

In the Efficient Depth-Limited (EDL) search, every vertex in the graph is given a number in the graph representation, only vertices with a number larger than the smallest label in the current substructure are allowed to be extended toward. In this way, a particular substructure can only be generated from the initial substructure of its smallest labeled vertex, and no other vertices in it will generate this substructure again. So, the repetitive extension is avoided, and the number of substructures in the extension set is reduced significantly. Pseudocode of the algorithm is given in Algorithm 2.2. The complexity of EDL is the sum of the substructure numbers at each depth of extension, that is,

$$O(o_{ext}*(|E|)^{d}) \qquad (6)$$

And since a particular structure can still be guaranteed to be reached from the initial substructure of its smallest labeled vertex, Efficient Depth-Limited search still explores a complete search space within the extension limit. Even though the complexity is still exponential, but it reduces the time proportional to $|E|$. So in large dataset, the practical running time of Efficient Depth-limited search is much smaller than the non-efficient depth-limited.

Algorithm 2.2: Efficient Depth-limited search

Input: Graph,Limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with unique vertex label;

While parentList not empty **do**

parentSub = head of parentList;

Insert parentSub into discoverList in order by value;

If depth of parentSub **less than** Limit **then**

For each vertex to be added via an extension **do**

If vertex number **greater than** smallest vertex number in parentSub **then**

extendList = extend parentSub by adding the vertex and the edge between them;

Push every extendSub in extendList onto parentList;

Return discoverList;

Hill-climbing Search

The hill-climbing search strategy tries to choose the best successor extension at each step. However it is easy to be stuck on a local maximum. In graph-based search, graph-based hill climbing also starts with the unique labeled vertices, and then it selects the best substructure among the extensions from the parent as its successor. The search then repeats the extend-and-select best process until exhausting the possible extensions or a local maximum is met where no value of a substructure in the extension list is larger than the value of current parent substructure.

Graph-based hill climbing is very efficient at going down the substructure extension space. Like beam search that holds a constant number of extended substructures, hill-climbing search holds only one. Time complexity of hill-climbing search is also

$$O(o_{ext} \times d) \qquad (7)$$

The hill climbing search moves by following the steepest path, and stops at the local maxima. So it will fail to find the global best substructure pattern when the problem is complex and has several local maxima on which hill climbing will easily get stuck. Hill climbing returns the pattern at the first local maxima. So in this sense, hill climbing in graph-based data mining is able to reveal how complex is the structure space of the graph by how good is the pattern that hill-climbing can find. If hill climbing can find the best pattern, it shows the graph is simple in structure.

Algorithm 3: Hill-climbing search

Input: Graph;

Output: The best pattern found.

Create parentList from initial substructures with unique vertex labels;

parentSub = pattern in initialList with best value;

childSub = best extension from parentSub;

while value of childSub **greater than** value of parentSub **do**

    parentSub = childSub;

    childSub = best extension from parentSub;

Return parentSub;

Stochastic Hill-climbing (Stochastic HC)

Different from hill-climbing search, which follows the steepest path, Stochastic Hill-climbing selects a random successor from the extensions of substructures; but rather than purely random, the probability that a certain extension is chosen is proportional to its compression value for the graph. In the Stochastic HC search strategy, any extension is possible to be chosen as successor, though the child substructure with a larger compression value has more probability. This extend-and-select process goes on until exhaustion of possible extensions or the user-specified limit of extension steps is reached. Stochastic HC, given in Algorithm 4, has a computation complexity of

$$O(o_{ext} \times d) \qquad (8)$$

Algorithm 4: Stochastic Hill-climbing search

Input: Graph, Limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with unique vertex labels;

parentSub = choose substructure from parentList with probability proportional to substructure values;

While Limit > 0 **do**

extendList = extend parentSub by adding an edge in all possible ways;

childSub = choose a substructure from extendList with probability proportional to substructures' values;

Insert parentSub into discoverList in order by value;

parentSub = childSub;

Limit = Limit - 1;

Return discoverList;

Simulated-annealing Search

Simulated-annealing [9] approaches the global maxima by analogy to the annealing process in metallurgy. The annealing process allows atoms to wander between high energy states and slowly cooling down to give them more chance to finally settle to a state of minimal internal energy.

In graph mining, we adopt this approach by choosing successors as outlined in Algorithm 5. The parent substructure can be seen as the current while the extensions as candidate successors. The successor is randomly selected from the extensions based on value as in Stochastic Hill-climbing; but if the value of the successor is larger than the current compression value, then we choose it. If not, we can still choose it with probability

that decreases as the process goes on; otherwise, we consider another extension substructure. The probability varies according to both the systematic scheduler *T* and the value difference between the successor and current pattern. As time goes on, the possibility to choose a temporary worse move will decrease with the scheduler. This extend-and-evaluate process continues until exhaustion of possible extensions or the user-specified extension limit is reached.

In the algorithm we set *T* to be equal to the total depth limit minus the extension iterations that have been done. The average complexity of simulated-annealing search is given in 9.

$$O(o_{ext} \times d) \qquad (9)$$

In the simulated annealing method, the child substructure is selected based on its value. We select it if the value of the child substructure is larger than its parent. We assign the probability to choose the selected child substructure only when the value of the substructure is less than that of its parent. The probability can be expressed as P = exp(deltaE/T), in which deltaE denotes the difference of values between the child substructure and the parent substructure, and T denotes the time schedule which is initialized to be the depth limit and is decremented with each depth of extension. So T will be decreasing from the depth limit to 1 until the depth limit is reached and the program exits successfully. The probability will be kept within range of [0, 1], because T is always greater than 0, and the condition of reaching the probability selection makes sure that the difference of values between the child and the parent will always be negative or 0. When the value of the child substructure approaches the value of parent substructure, the deltaE factor goes to 0, and the probability is 1. In the other case, e to a factor of negative will

always give a value ranging from 0 to 1. As the time goes on, T is decreased, so the probability P will becomes smaller. In this way, the algorithm gives more chance to select a smaller value child substructure at the beginning, and tries to stick to the best child substructure as time goes on. This condition holds for all the following simulated annealing based search algorithms.

Algorithm 5: Simulated-annealing search

Input: Graph, Limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with unique vertex label;

parentSub = get random structure in parentList with probability proportional to value;

While Limit > 0 **do**

Insert parentSub into discoverList;

T = schedule(t);

If T = 0 **then**

return discoverList;

extendList = extend parentSub by one edge in all possible ways;

childSub = get random structure in extendList with probability proportional to value;

ΔE = value of childSub - value of parentSub;

If $\Delta E > 0$ **then**

parentSub = childSub;

Else

parentSub = childSub only with probability $e^{\Delta E/T}$;

Limit = Limit -1;

Return discoverList;

SA-Greedy

To overcome problems of local maxima for hill climbing and also utilize the advantage of simulated-annealing, SA-Greedy is proposed. In the critical extension selection process, SA-Greedy considers the best pattern (childSub) in the extension list and compares its value to the current parent value. If the childSub is better, then set it as successor with probability as in simulated annealing. But if not, choose a random extension. The process stops upon exhaustion of the possible extensions or the extension limit is reached. Pseudocode of SA-Greedy is shown in Algorithm 6. Its computation complexity is

$$O(o_{ext} \times d) \qquad (10)$$

We proposed this algorithm both to avoid getting trapped in local maxima as in hill-climbing and better convergence as in simulated-annealing. The value of k should make the jump probability to a random point more easily satisfied at the beginning of the search, and converge to a steady state at the end of the search. The parameter k can be tuned by trying different ks, and selecting the one that finds the best pattern. Then this value of k is consistently applied in exploring different datasets.

Algorithm 6: SA-Greedy

Input: Graph, Limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with unique vertex label;

parentSub = best structure in parentList;

While Limit > 0 **do**

extendList = extend parentSub with one edge in all possible ways;

childSub = substructure in extendList with the best value;

Insert parentSub into discoverList in order with value;

ΔE=value of childSub – value of parentSub

If ΔE>0 **then**

parentSub = childSub only with probability of $k*e^{\Delta E/T}$;

Else

parentSub = choose a random extension from extendList;

Limit = Limit -1;

Return discoverList;

Hill-climbing with Stochastic Escape (HCSE)

In graph mining, another way to escape local maxima in hill climbing is to choose a successor in a less deterministic way and continue the process, rather than stop at local maxima. The HCSE strategy tries to escape local maxima by choosing the successor with probability corresponding to the values of extension substructures. Then, HCSE continues hill-climbing and finally stops when extensions are exhausted or reaching the extension limit. So the computation complexity of HCSE is

$$O(o_{ext} \times d) \qquad (11)$$

on average.

The HCSE keeps the advantage of traditional hill climbing that chooses the promising extension substructure as successor, so it can find a good pattern in a short time. With stochastic escape from local maxima, HCSE is able to explore substructures that lie at further extension depths. Also the probability of selecting an extension to escape the local maxima is set proportional to its value: the larger its value, the more likely it will be chosen as the successor.

Algorithm 7: Hill-climbing with Stochastic Escape (HCSE)

Input:Graph, Limit;

Output: discoverList of best patterns found.

Create parentList from initial substructures with unique vertex labels;

parentSub = get best structure in parentList;

While Limit > 0 **do**

extendList = extend parentSub by one edge in all possible ways;

childSub = best substructure in extendList;

Insert parentSub into discoverList;

ΔE = value of childSub - value of parentSub;

If ΔE> 0 **then**

parentSub = childSub;

Else

childSub = a random structure in extendList with

probability proportional to its value;

parentSub = childSub;

Limit = Limit -1;

Return discoverList;

Conclusion

In this chapter, six search strategies in the space of substructures in a graph are designed and implemented. The depth-limited search strategy keeps every substructure in the search and explores a complete substructure space within the limit number of extensions. To reduce the search space but to still find the best substructure, heuristic based search strategies are designed. Beam search uses a heuristic function to evaluate substructures and keeps a beam-width number of the most promising substructures in each step. The hill-climbing search strategy tries to choose the best successor extension at each step. Stochastic Hill-climbing selects a random successor from the extensions of substructures. The probability that a certain extension is chosen is proportional to its compression value for the graph. In Simulated annealing strategy, the probability to choose a random successor varies according to both the systematic scheduler *T* and the value difference between the successor and current pattern. As time goes on, the possibility to choose a temporary worse move will decrease with the scheduler. SA-Greedy considers the best pattern (childSub) in the extension list and compares its value to the current parent value. If the childSub is better, then it becomes the successor with probability as in simulated annealing. But if not, choose another random extension. And the HCSE strategy simulates

hill-climbing but tries to escape local maxima by choosing the successor with probability corresponding to the values of extension substructures. All of the seven search strategies stop upon exhaustion of the possible extensions or the extension limit is reached.

CHAPTER FOUR

SEARCH ALGORITHM EXPERIMENTS

Introduction to experiment conditions

To measure the performance of the search strategies, we conducted experiments on several datasets with different properties from different domains. The experiments are run on a Linux machine with two Intel Pentium 4, 3.40GHz CPUs, 1G Memory; Linux kernel version: 2.6.9-55.ELsmp, Red Hat 3.4.6-3 and with GCC version 3.4.6. The time is recorded as CPU time spent on running the process.

We run the searches with different depth limits: 5, 8, 10, 20, 50, 70 and 100. The running time and best pattern value are recorded and plotted. Also, since the search strategy of Stochastic HC, Simulated-annealing, SA-Greedy and HCSE all make use of a random probability, we applied 100 iterations to get stable results on big datasets. One whole process begins when the initial substructures are selected, extended and evaluated. The process ends when the extensions of substructures are exhausted or the user-specified depth limit is reached.

We choose the number of iterations to be 100 because after tuning this parameter, we found 100 iterations make the results stable during multiple runs of the search algorithms. We also give the comparison of different iteration numbers on one of the datasets in the appendix. The experiment plot (Figure A.1) and tables (Table A.1, A.2 and A.3) show that the algorithms reach a flat stage at iteration of 100. The 100 iterations in an algorithm are executed automatically using a loop in the code of the program. And the running time in the following experiments of the stochastic algorithms is the running time of a program with

100 iterations executed automatically, which roughly equals 100 times a single run of the algorithm.

In the plots of this chapter, the x-axis represents depth_limit applied in the search algorithms. The y-axis is denoted differently with different plots, but either represents the value of the best substructure found, or represents the corresponding CPU running time with unit of seconds.

Criminal and Social Network Dataset

The Criminal and Social Network dataset contains communication relationships between actors of a group. The dataset is included with the download of Subdue from http://www.subdue.org. Each actor is involved in a communication case as either initiator or respondent. In the graph representation of this dataset, threat groups comprise the positive examples, and non-threat groups comprise the negative examples [7]. We attempt to find pattern substructures to distinguish threat groups from non-threat groups using Subdue in supervised learning mode.

The dataset consists of 3 positive graphs with 118 vertices and 141 edges, and 7 negative graphs with 1406 vertices and 1683 edges, which makes 3348 vertices and edges in total. The average degree of this graph is 2.394. There are 7 unique labels found as initial substructures.

Detailed running results of the best values found with the search algorithms on this dataset are listed in table 1 and table 2. The corresponding running time are given in table 3 and table 4. Running times are plotted in Figure 2, and values of the best substructures found are shown in Figure 1.

From the time plot shown in Fig 2, we can observe that running time plot of all the search algorithms stays flat for depth limits greater than 20. But the time plot of Efficient Depth-limited search goes up quite huge with the growth of depth limits. This huge amount of running time make Efficient Depth-limited search infeasible after the depth limit of 8.

The overall best substructure value in the table 1 and table 2 is found to be 17.6211, which is found by SA_Greedy at the depth limit of 100. The best substructure by beam search is found at the depth limit of 70 and 100. The best value found by beam search is 17.0816, which is less than the value of SA_Greedy. Simulated Annealing finds best structure with value 16.4926. HC with Stochastic Escape finds best with value 16.2524. Stochastic HC finds 16.0191. Efficient Depth-limited can only finds the best substructure with value 15.3578. And Hill climbing performs worst by finding the best substructure with value of only 12.8769.

Table 1: Best Substructure values found by Search algorithms on Groups dataset with various depth limits.

depth_limit	1	5	8	10	20
Beam Search	12.8769	12.8769	12.8769	12.8769	12.8769
Eff Depth-limited	12.8769	13.95	15.3578	–	–
Hill Climbing	12.8769	12.8769	12.8769	12.8769	12.8769
SA_Greedy	12.8769	12.8769	14.3077	16.0191	14.9464
Simulated Annealing	12.8769	12.8769	14.3077	14.8142	15.0
HC with Stochastic Escape	12.8769	12.8769	12.8769	14.6842	16.0962
Stochastic HC	12.8769	12.8769	13.5547	14.3077	14.5565

Table 2: Best Substructure values found by Search algorithms on Groups dataset with various depth limits (Continued).

depth_limit	50	70	100
Beam Search	16.4118	17.0816	17.0816
Eff Depth-limited	–	–	–
Hill Climbing	12.8769	12.8769	12.8769
SA_Greedy	16.0962	16.5743	17.6211
Simulated Annealing	16.4926	15.3578	15.0811
HC with Stochastic Escape	16.2524	16.2524	16.2524
Stochastic HC	16.0191	16.0191	16.0191

Table 3: Running time of Search algorithms on Groups dataset with various depth limits.

depth_limit	1	5	8	10	20
Beam Search	0.06	0.15	0.19	0.33	0.59
Eff Depth-limited	0.63	8.14	281.44	–	–
Hill Climbing	0.06	0.06	0.06	0.06	0.06
SA_Greedy	3.67	14.34	20.62	25.43	31.73
Simulated Annealing	3.91	15.33	23.58	26.70	38.18
HC with Stochastic Escape	3.60	13.36	20.36	22.11	37.21
Stochastic HC	3.87	14.65	21.99	26.45	36.28

Table 4: Running time of Search algorithms on Groups dataset with various depth limits. (Continued)

depth_limit	50	70	100
Beam Search	1.40	1.90	2.04
Eff Depth-limited	–	–	–
Hill Climbing	0.06	0.06	0.06
SA_Greedy	34.02	32.35	30.91

Simulated Annealing	39.85	40.11	40.29
HC with Stochastic Escape	45.97	45.77	48.86
Stochastic HC	45.02	43.61	60.55

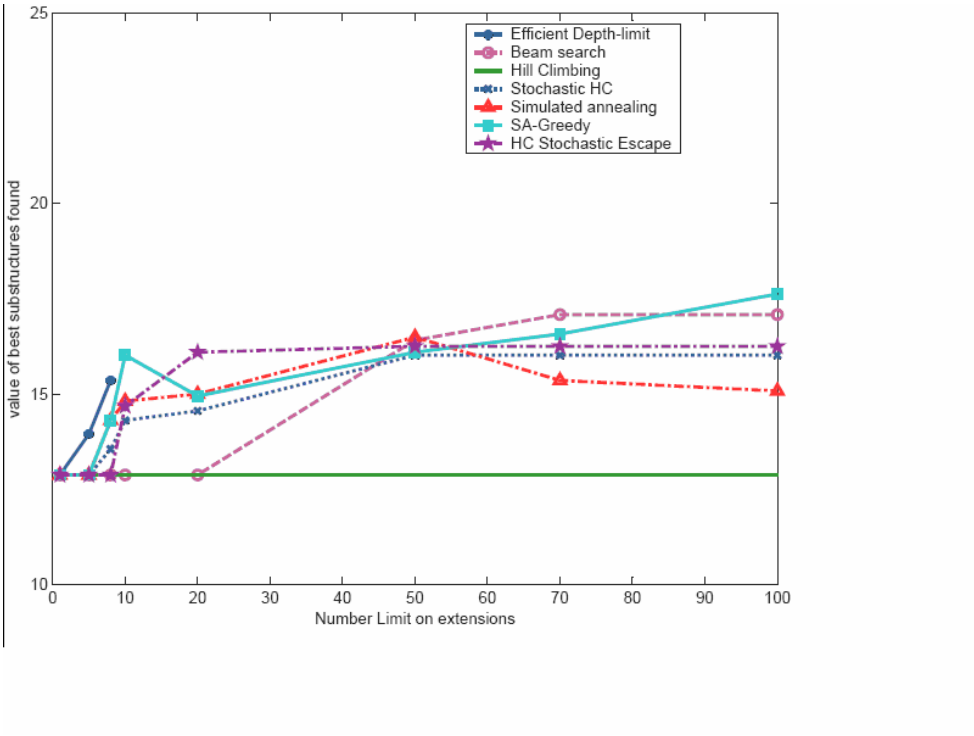

Figure 1: Best Pattern Value plot of Criminal and Social Network

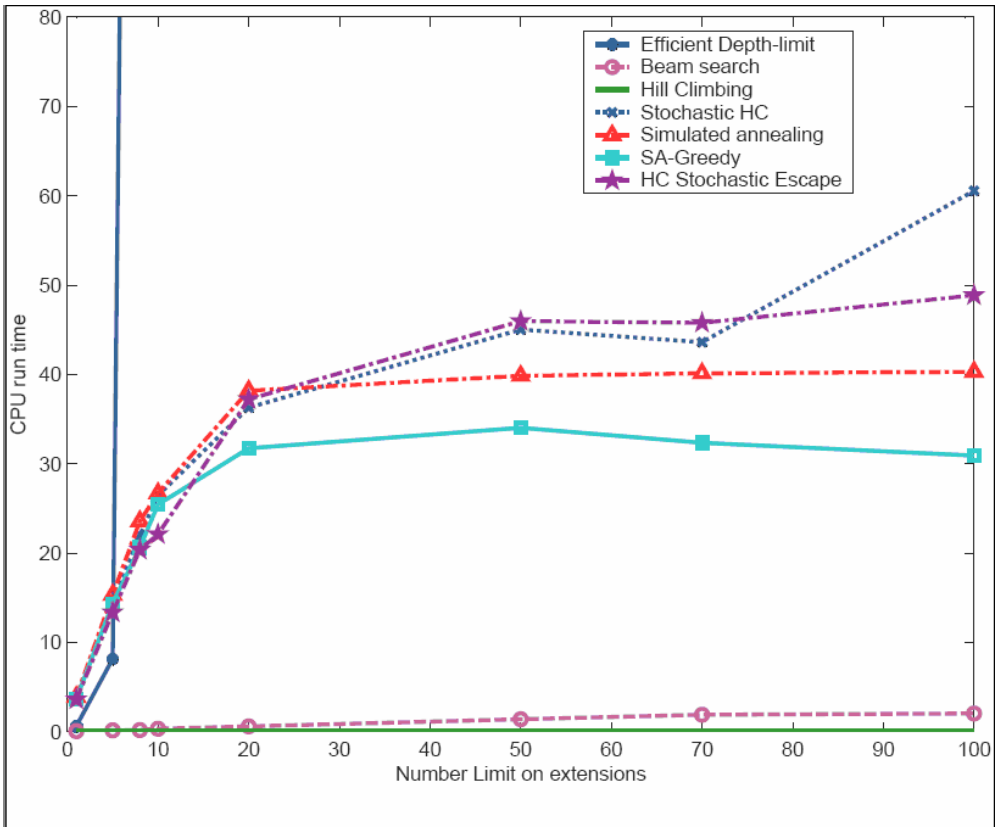

Figure 2: Time plot of Criminal and Social Network

Credit Dataset

The Credit dataset is produced from the credit card application approval database from the UCI repository, which can be downloaded from http://www.ics.uci.edu/ mlearn/databases/credit-screening/. It is also included in website of Subdue from http://www.subdue.org. An entity in this dataset mostly has 20 attributes, and 690 instances are included. The Credit dataset is represented as a graph in the form of a star topology. The credit graph has a total of 28,700 vertices and edges. The average degree of the graph is 1.905. There are 79 unique labels found as initial structures. Running times of the search algorithms on the Credit dataset are listed with detail number in Table 6 and are plotted in Figure 4. The values of the best substructures found are listed in Table 5 and are shown in Figure 3. Note that SA-Greedy and HC with stochastic escape find the same valued substructures, so their curves overlap in Figure 3.

From the time plot shown in Fig 4, we can observe that running time plot of all the search algorithms again stays flat for depth limits greater than 20. But the time plot of Efficient Depth-limited search goes up quite huge with the growth of depth limits. This huge amount of running time makes Efficient Depth-limited search infeasible after the depth limit of 5. We aborted the try of Efficient Depth-limited search with depth of larger than 5 after 5 days. All the trends shown in time plots of credit dataset are very consistent with that of the criminal and social network dataset.

The overall best substructure value in the table 5 is found to be 1.16567, which is found by both SA_Greedy and HC with Stochastic Escape at the depth limit of 10 and 20. The best value found by beam search is 1.12414, which is less than the value found by

SA_Greedy. Efficient Depth-limited can only find the best substructure with value 1.12272. Stochastic HC finds 1.10944. Simulated Annealing finds best structure with value 1.07721. And Hill climbing again performs worst by finding the best substructure with value of only 1.03085. We also did not make the experiment of SA_Greedy and HC with Stochastic Escape with larger depth, because they have shown a very flat trend between depth limit of 20 and depth limit of 70.

Table 5: Best Substructure values found by Search algorithms on Credit dataset with various depth limits.

depth_limit	5	10	20	50	100
Beam Search	1.04284	1.04619	1.04863	1.04863	1.12414
Eff Depth-limited	1.12272	–	–	–	–
Hill Climbing	1.03085	1.03085	1.03085	1.03085	1.03085
SA_Greedy	1.14229	1.16567	1.16567	–	–
Simulated Annealing	1.04863	1.05728	1.06592	1.04909	1.07721
HC with Stochastic Escape	1.14229	1.16567	1.16567	–	–
Stochastic HC	1.08412	1.07503	1.10944	1.08086	1.07988

Table 6: Running time by Search algorithms on Credit dataset with various depth limits.

depth_limit	5	10	20	50	100
Beam Search	0.28	0.28	6.74	6.83	161.31
Eff Depth-limited	3600.44	–	–	–	–
Hill Climbing	2.32	2.32	2.32	2.32	2.32
SA_Greedy	416.34	516.73	854.20	–	–
Simulated Annealing	137.94	215.61	170.75	179.56	157.74
HC with Stochastic Escape	720.02	936.41	963.30	–	–
Stochastic HC	179.01	185.95	208.18	158.75	151.86

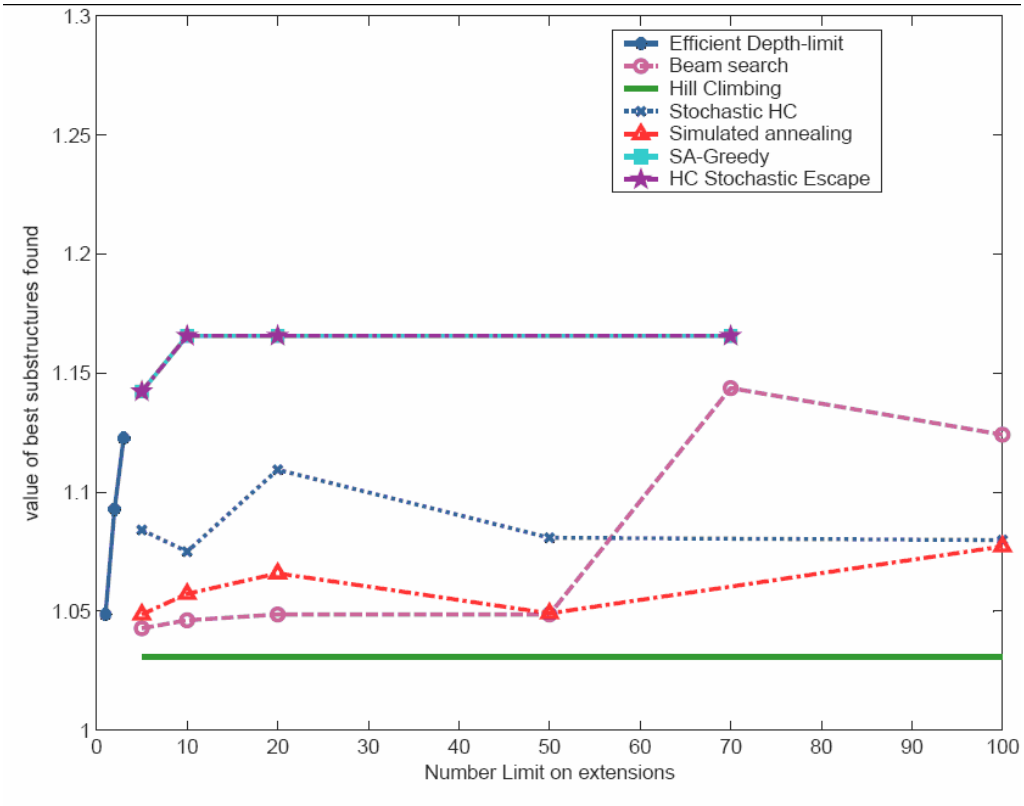

Figure 3: Best Pattern Value plot of Credit dataset

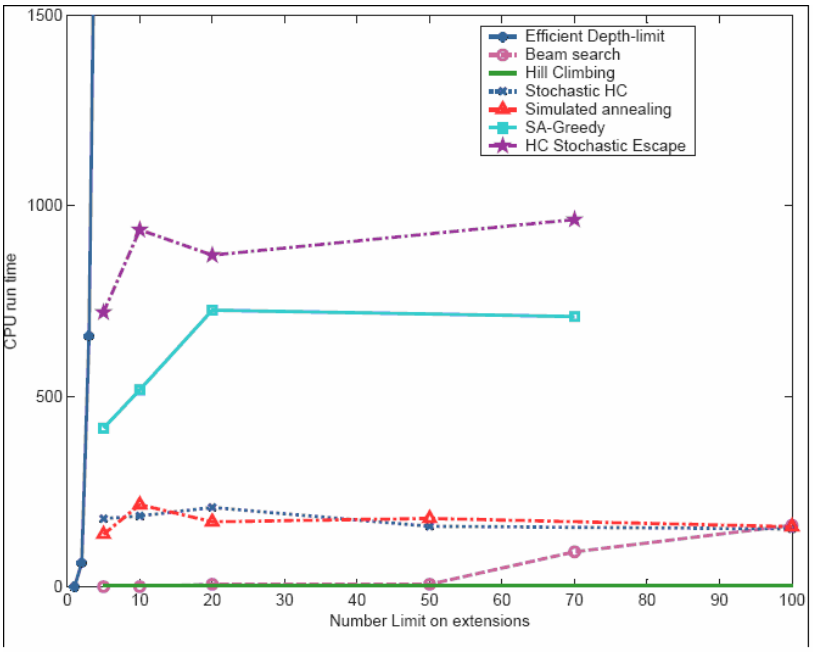

Figure 4: Time plot of Credit dataset

Carbon dataset

The carbon dataset represents a sample structure of a carbon. It consists of 1 positive graph with 79 vertices and 90 edges. The average degree of the graph is 2.28. Two unique labels are found to be the initial substructures. The carbon dataset is included with the download of Subdue from http://www.subdue.org.

The best substructures found in the carbon dataset are given in Table 7 and Table 8. In the following experiments on small datasets in this chapter, only the running result of search algorithms with depth limit 10 and 50 are listed. This is because we observed the same patterns found by search algorithms with various depth limits. Only two sets of results are given to show the trends.

On Carbon dataset, Beam search, SA_greedy, Simulated Annealing, HC with stochastic escape and stochastic HC all find the best substructure, showing the ability of the search algorithms in finding the best substructures on small dataset.

Table 7: Search algorithms running results on Carbon dataset with depth limit 10.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam Search	2.81667	2.81667	2.41429	0.97	10
Eff Depth-limited	2.06098	1.87778	1.87778	7.89	3
Hill Climbing	2.28378	1.76042	0.994118	0.00	N/A
SA_Greedy	2.81667	2.41429	2.28378	1.06	10
Simulated Annealing	2.81667	2.81667	2.41429	0.41	10
HC with Stochastic Escape	2.41429	–	–	1.25	10
Stochastic HC	2.81667	2.41429	2.28378	0.53	10

Table 8: Search algorithms running results on Carbon dataset with depth limit 50.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	2.81667	2.81667	2.41429	1.26	50
Eff Depth-limited	2.06098	1.87778	1.87778	7.89	3
hill climbing	2.28378	1.76042	0.994118	0.00	N/A
HC with stochastic escape	2.81667	2.41429	–	361.40	50
stochastic HC	2.28378	2.06098	1.87778	0.58	50
SA Greedy	2.81667	2.41429	2.41429	647.91	50
Simulated Annealing	2.81667	2.41429	2.28378	1.00	50

Sample dataset

The Sample dataset is a sample structure of object shapes and their positions. It consists of 1 positive graph with 20 vertices and 19 edges. The average degree of the graph is 1.9.

Seven unique labels are found to be the initial substructures. The Sample dataset is included with the download of Subdue.

The best substructures found in Sample dataset are given in Table 9 and Table 10. Only the running result of search algorithms with depth limit 10 and 50 are listed, because we observed the same patterns found by search algorithms with various depth limits, which is the same case with the carbon dataset.

On the Sample dataset, all of the search algorithms find the best substructure, showing the ability of the search algorithms in finding the best substructures on smaller datasets.

Table 9: Search algorithms running results on Sample dataset with depth limit 10.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	1.77273	1.39286	1.39286	0.00	10
Eff Depth-limited	1.77273	1.39286	1.39286	0.00	10
hill climbing	1.77273	1.39286	1.21875	0.00	N/A
HC with stochastic escape	1.77273	–	–	0.00	10
stochastic HC	1.77273	1.39286	1.39286	0.00	10
SA Greedy	1.77273	1.39286	1.39286	0.00	10
Simulated Annealing	1.77273	1.39286	1.39286	0.01	10

Table 10: Search algorithms running results on Sample dataset with depth limit 50.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	1.77273	1.39286	1.39286	0.01	50
Eff Depth-limited	1.77273	1.39286	1.39286	0.01	50
hill climbing	1.77273	1.39286	1.21875	0.00	N/A
HC with stochastic escape	1.77273	–	–	0.00	50
stochastic HC	1.77273	1.39286	1.39286	0.01	50

SA Greedy	1.77273	–	–	0.01	50
Simulated Annealing	1.77273	1.39286	1.39286	0.00	50

Overlap dataset

The Overlap dataset consists of 1 positive graph with 10 vertices and 13 edges. The average degree of the graph is 2.6. Two unique labels are found to be the initial substructures. The Overlap dataset is included with the download of Subdue.

The best substructures found in Overlap dataset are given in Table 11 and Table 12. Only the running result of search algorithms with depth limit 10 and 50 are listed, because we again observed the same patterns found by search algorithms with various depth limits, which is the same case with the carbon dataset and the Sample dataset.

On Overlap dataset, all of the search algorithms also find the best substructure, showing the ability of the search algorithms in finding the best substructures on small datasets.

Table 11: Search algorithms running results on Overlap dataset with depth limit 10.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	1.4375	1.4375	1.27778	0.00	10
Eff Depth-limited	1.4375	1.4375	1.27778	0.00	10
hill climbing	1.4375	0.958333	–	0.00	N/A
HC with stochastic escape	1.4375	–	–	0.01	10
stochastic HC	1.4375	–	–	0.07	10
SA Greedy	1.4375	–	–	0.01	10
Simulated Annealing	1.4375	–	–	0.01	10

Table 12: Search algorithms running results on Overlap dataset with depth limit 50.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	1.4375	1.4375	1.27778	0.00	50
Eff Depth-limited	1.4375	1.4375	1.27778	0.00	50
hill climbing	1.4375	0.958333	–	0.00	N/A
HC with stochastic escape	1.4375	–	–	0.01	50
stochastic HC	1.4375	–	–	0.01	50
SA Greedy	1.4375	–	–	0.00	50
Simulated Annealing	1.4375	–	–	0.00	50

House dataset

The House dataset consists of 4 positive graphs and 4 negative graphs. The 4 positive graphs have 24 vertices and 20 edges. The 4 negative graphs have 24 vertices and 20 edges. The average degree of the graph is 1.67. Eight unique labels are found to be the initial substructures. The House dataset is included with the download of Subdue.

The best substructures found in House dataset is given in Table 13 and Table 14. Only the running result of search algorithms with depth limit 10 and 50 are listed, because we again observed the same patterns found by search algorithms with various depth limits, which is the same case with the carbon dataset, Sample dataset and Overlap dataset.

On the House dataset, all of the search algorithms except hill climbing find the best substructure, showing the ability of the search algorithms in finding the best substructures on small datasets. It also shows that the graph structure could be complex even in small datasets, and hill climbing could get stuck on local maxima even on small datasets. This is consistent with the conclusion that the complexity of a graph structure is the characteristic of the graph itself, and does not depend on the size of the graph.

Table 13: Search algorithms running results on House dataset with depth limit 10.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	4.19048	3.25926	2.66667	0.01	10
Eff Depth-limited	4.19048	3.25926	2.66667	0.01	10
hill climbing	1.95556	–	–	0.01	N/A
HC with stochastic escape	4.19048	–	–	0.00	10
stochastic HC	4.19048	2.66667	2.66667	0.07	10
SA Greedy	4.19048	2.66667	1.95556	0.01	10
Simulated Annealing	4.19048	2.66667	2.66667	0.00	10

Table 14: Search algorithms running results on House dataset with depth limit 50.

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam search	4.19048	3.25926	2.66667	0.01	50
Eff Depth-limited	4.19048	3.25926	2.66667	0.00	50
hill climbing	1.95556	–	–	0.00	N/A
HC with stochastic escape	4.19048	1.95556	–	0.01	50
stochastic HC	4.19048	2.66667	2.66667	0.01	50
SA Greedy	4.19048	1.95556	–	0.01	50
Simulated Annealing	4.19048	2.66667	2.14634	0.01	50

Conclusion

To study the performance of the search algorithms in graph-based data mining, we conducted experiments using search variants of Subdue, a compression-based graph mining system. Experiments are conducted on large datasets including Criminal and Social Network Dataset and Credit Dataset, and on the small datasets including Carbon dataset, Sample dataset, Overlap dataset and House dataset. Results are given in tables and plots for

comparison of search algorithms. The public health dataset is constructed in the following chapter and serves as another large dataset for comparing the search algorithms.

CHAPTER FIVE

PUBLIC HEALTH DATASETS

Graph-based data mining has shown its ability in discovering the patterns in many domains including fraud [12], threat groups [7] Public health datasets also involves complex relations between the entities. Discovering patterns in public health domains will greatly help reveal the intrinsic relations underlying the public health events. For example by extracting the pattern of the spread of a disease from a pandemic dataset using graph-based data mining, we can provide a meaningful guideline in future prevention to the breakout of the disease.

Also, since the public health datasets generally include records from a large population of people, the graph representations of the datasets can grow very large. Together with the natural relations between those people, the graph nodes are highly interconnected. So this kind of complex dataset also provides a very good graph to test search algorithms and compare their performance.

Introduction to Pandemic Dataset

The pandemic dataset is a synthetic data collection on communication and infection within a network of individual activities in the area of Portland, OR. Along with contact activities, it describes an infectious disease outbreak which occurs in that population community. The rich relational structures between entities of the data make it naturally represented as a graph. By applying graph-based data mining, we can analyze, and thus find the intrinsic patterns to the spread of infection, which could provide a meaningful guideline in disease prevention.

The dataset is available at http://ndssl.vbi.vt.edu/opendata/ .

Description of Pandemic Dataset

In the pandemic dataset, original relational information is provided within 6 data files. The "Demographics person" file contains personal information with the following fields:

- Id of the person;
- Id of the household;
- Age of the person;
- Gender;

  1 Male

  2 Female
- Worker status;

  1 Works

  2 Does not work
- Relation to the head of the household.

The "Demographics household" file describes household attributes including the following items.

- Household ID;
- Household income range;
- Household size;
- Home location ID;
- Sub-location ID;

- Numbers of vehicles in the household;
- Number of household members who work.

Location IDs are further related with the location *x* offset from the origin in meters, and *y* offset from origin in meters, provided in the “Locations” file, which gives direct ways of distance calculation between locations.

Activities performed by individuals are recorded with detail in the “Activity” file, which provides critical information on how people interact with each other, and provides a basis for data entity relations and potential patterns associated with how infectious disease propagates. Detailed fields in the activity record file include the following items.

- Identity numbers (as household ID, person ID, activity ID);
- Activity purpose

  0 - Home

  1 - Work

  2 - Shop

  3 - Visit

  4 - Social/Recreation

  5 - Other

  6 - Pick up or drop off a passenger

  7 - School

  8 - College
- Time that activity starts;

- Duration of the activity: Time in seconds of the contact.
- Location where the activity takes place.

Types of contact activities recorded include home contact which basically occurs within the same household, work contact, shop, visit, school contact, etc. Contact purpose and duration between persons are directly represented in the “Contact” data file.

Individuals are connected with communication contacts between them. At the same time with retracing the disease path from the record of 100 initially infected persons from the “Dendro” file, the infection path between the infected person and their infected “parent” enriches the relation complexity between person entities.

Graph-based Data Relation Analysis

Several intrinsic data relations have been observed in the pandemic dataset. For instance, the activities occured between people, people belong to the same household, people with the same household location, people share the same communication activity or a contact, and the infection path between people.

To consider representing the relations with a graph, person, communication, and infection can be seen as entities. Every identical entity was represented with exactly one node. This node could be connected with an edge when this entity (take person as example) is involved in an activity and this needs to be related with other entities in the activity; Other nodes can be defined as attributes of the entities that is, we can just copy relating certain attributes to the node, and so can be copied and linked via an edge to the entity nodes possessing this attribute.

Attributes of a person include id, household id, age, gender, worker status, as well as relationship. Household has attributes of id, household income range, household size, home location ID, sub-location ID, vehicles, and workers. Attributes of an activity are household ID, person ID, activity ID, activity purpose, start time, duration, as well as location. Contact connects two person entities together and has attributes of purpose and duration. Infection also connects two person entities and has attributes of day of infection, location id, and infection generation.

Graph Representation of Pandemic Dataset

A sample sub-graph representing activity and infection relation is given in Fig. 5. In the graph representation, person, activity and infection are the main entities, so they are represented as distinct nodes. An activity node connects the two persons involved in the activity. An infection node connects the person who is the parent of the infection and the person who is the infected one.

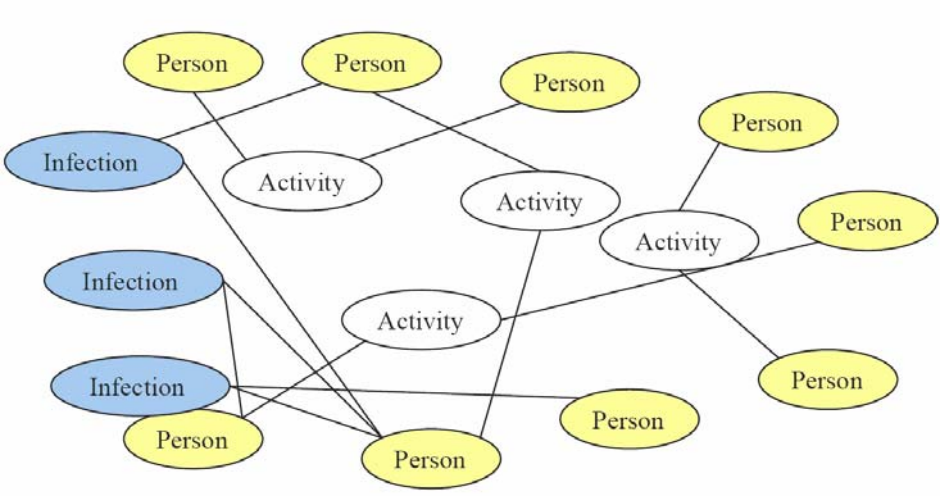

Figure 5: Sample graph representation of Pandemic dataset.

The general graph representation can contain more information by adding attribute values to the nodes. The attributes of a person will include the house hold info, age, gender, worker, and relation to the family. The information can be extracted from the original data file. The whole person node is shown in Fig. 6. As with the person node, the activity node can be augmented with attributes including the purpose of the activity and the duration that the activity takes, as shown in Fig. 7. The infection node can be augmented with attributes including the infection date on which the infection occurs and its generation. The augmented infection node is illustrated in Fig. 8.

It should be noted that the graph is a directed graph in the system. The edges between entity and its attributes come from the entity node and point to the attribute nodes. The edge between the entities comes from the communication node, and points to the person node. For the edge between infection entity and person entity, the parent person of infection points to the infection node and the infection node points to the infected person node.

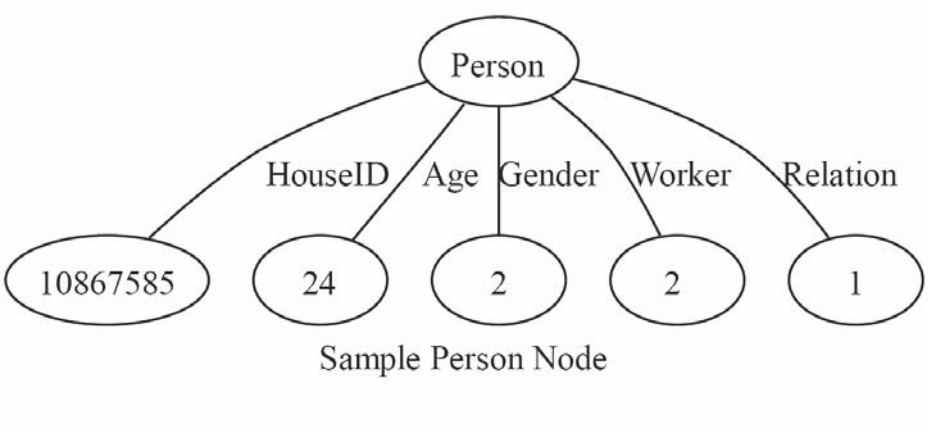

Figure 6: Sample Person Node in Pandemic graph representation.

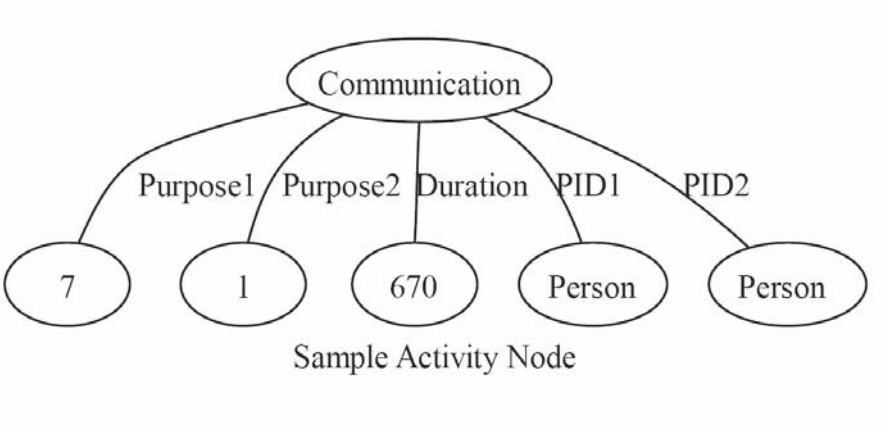

Figure 7: Sample activity Node in Pandemic graph representation.

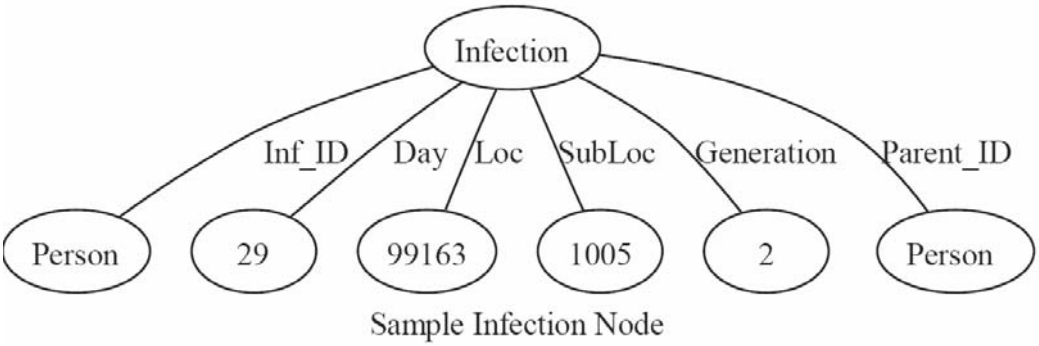

Figure 8: Sample infection Node in Pandemic graph representation.

Improved Graph Representation of Pandemic Dataset

In graph-based data mining, a subgraph containing a different label from the pattern is distinct from the pattern and thus cannot be compressed using the pattern. For example, a subgraph with age of 20 is different with another subgraph that has age of 21. But the age difference between 20 and 21 most likely does not make a difference in whether the persons will be infected or not, because persons at age of 20 and 21 have almost the same immunization ability. Communication that lasts for 100 seconds may not make too much of

a difference with the one that lasts for 101 seconds. So in the original representation, the large number of distinct labels will prevent graph-based data mining from finding common patterns based on these attribute.

This problem can be handled by replacing the many unique labels with fewer common labels. A range mapping is one possibility to do it. So in the person node, the representation is improved by replacing age between [0–18] to be “youth”, and age between (18–50) to “adult”, and >50 to be “senior”. An example is shown in Fig. 9. The duration of communication is mapped into the nearest 10^n according to the value of the duration, as shown in Fig. 10. The day of infection has been mapped to the month within which the infection takes place. The mapping can be done by dividing the day by 30, and keeping the integer part of the division result. A sample of the improved infection node set is shown in Fig. 11.

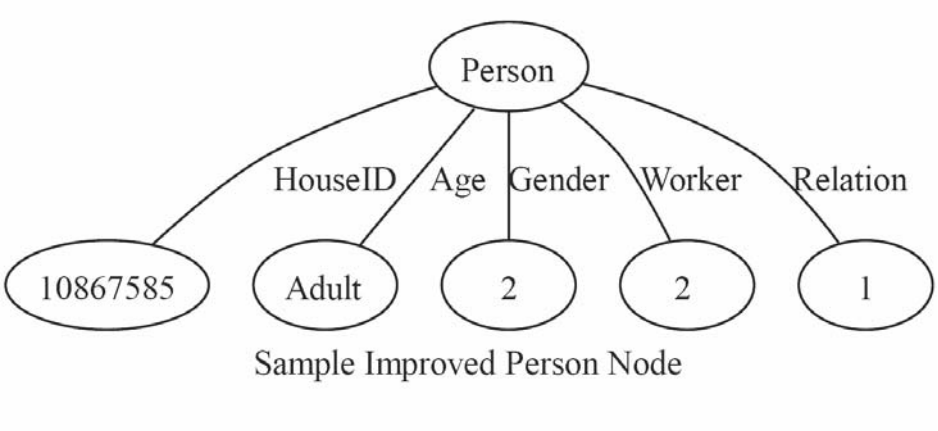

Figure 9: Sample Improved Person Node in Pandemic graph representation.

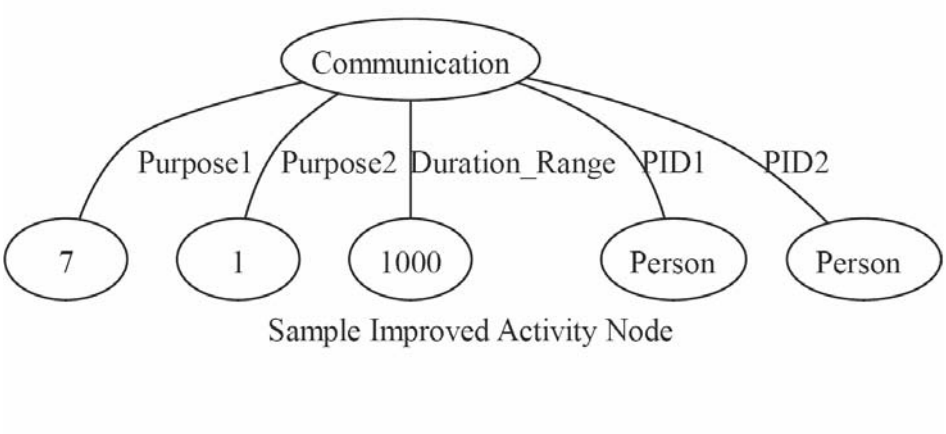

Figure 10: Sample Improved activity Node in Pandemic graph representation.

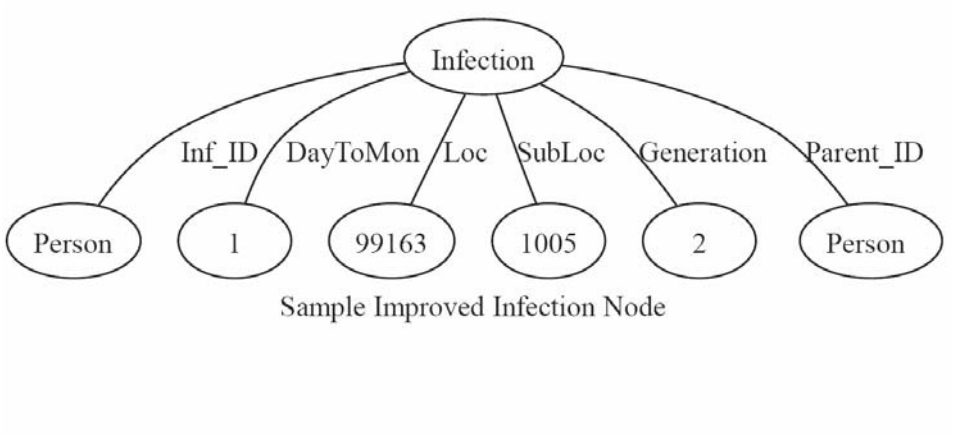

Figure 11: Sample Improved infection Node in Pandemic graph representation.

Graph representation of Pandemic dataset for supervised learning

Another way of representing the graph is to set each person as the origin to develop a surrounding graph, as shown in Fig. 12. To do this, we first extract all the communications involving the person, and collect all the persons at the other end of the communication. Within the "surrounding" persons, we then look for whether there are connections between them. The connections include infection relation and communication relation. And for all the "surrounding" persons, we do another iteration of extension with their communications

and infections. This process continues for a finite number of links away from the original root person. And the nodes in the graph are interconnected with each other.

We define the graph to be positive if the origin person is infected and negative if the origin person is uninfected.

It should also be noted that, the person origin does not include infection information in the graph to avoid the infection node to be the only way of classification, so we discover a pattern that can predict infection based on the person's surrounding information.

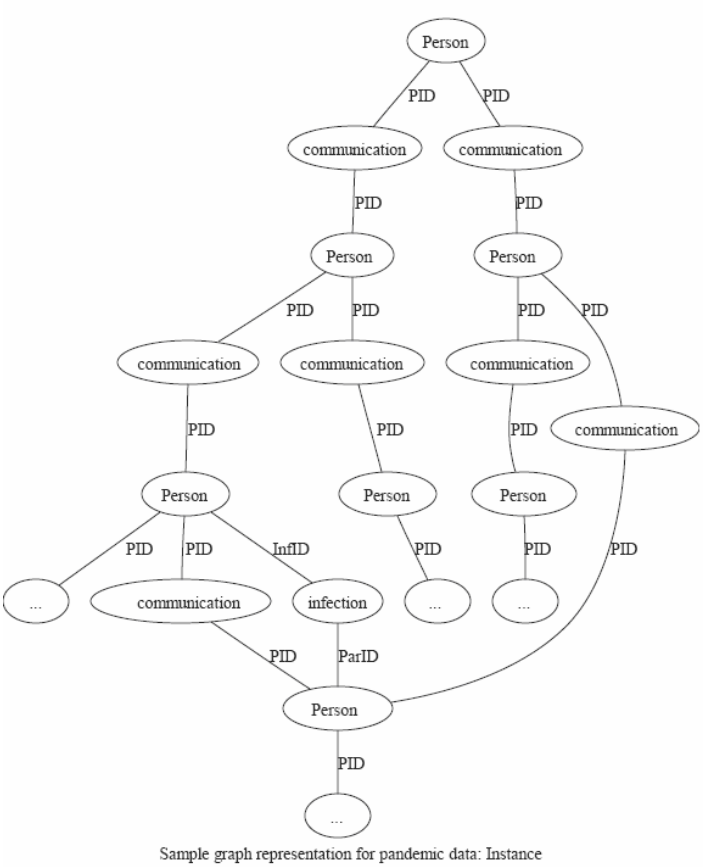

Figure 12: An alternate graph representation of the Pandemic dataset

Other potential good datasets

Besides the pandemic dataset, other good datasets in the public health domain are also collected. The problems in those datasets are also interesting. And graph-based data mining is an excellent way to discover patterns in those problems. Two of the problems are listed in detail in the following sections. We also provide suggestions on how to represent those relational datasets in graph. They provide general insights for readers to understand other potential applications of graph-based data mining.

- **Hazardous Substance Release and Health Effects Database**

  The Hazardous Substance Release and Health Effects Database (HazDat) is provided by the Department of Health and Human Services, Agency for Toxic Substances & Disease Registry. The database provides information on hazardous release from sites and events. The effects of the hazardous substances on human health can also be accessed from the database. The effects are classified in categories based on the severity of the hazard.

  The data can be obtained from http://www.atsdr.cdc.gov/hazdat.html .

- **HazDat dataset Description**

  HazDat data consists of two categories, information about the site and information on chemicals which may be found at some sites. The detailed content is listed in the following.

  Site Activity

In the HazDat dataset, site has attributes including Site ID, Public health threat category, Chems found in the area, site name, city, county, state, zip, latitude, longitude, population, region no, cong_dist, fed_fac, owner text, and fac_text.

Public health threat category has the following 11 options:

1. NULL or blank
2. Poses Public Health Hazard
3. Poses No Public Health Hazard
4. Category Not Reported In Document
5. Not Applicable To This Document
6. No Apparent Public Health Hazard
7. Poses Urgent Public Health Hazard
8. Indeterminate Public Health Hazard
9. Insufficient Data to Reach Conclusion
10. Posed Public Health Hazard Only In The Past
11. Posed Urgent Public Health Hazard Only In The Past

“Region no” is one of the ten ATSDR/EPA regions of the U.S. in which a site or event is located. “Cong_dist” is The number of the U.S. Congressional District within the state where the site or event is located.

fed_fac has the following 11 cases:

1. NULL or blank = Unknown

2. Y = Yes, site is a federal facility
3. N = No, site is not a federal facility " npl_text
4. NULL or blank = Not a Superfund site
5. Final = Site is currently on the NPL
6. Proposed = Site has been proposed for the NPL
7. Removed PreSARA = Site was removed from the NPL before SARA
8. Deleted PreSARA = Site was deleted from the NPL before SARA
9. Removed PostSARA = Site was removed from the NPL after SARA
10. Deleted PostSARA = Site was deleted from the NPL after SARA
11. Non NPL = Site has never been proposed or final on NPL

owner text has 11 types, which are listed in the following:

1. NULL or blank = Not stated in Agency document
2. Private
3. Municipality
4. County
5. District
6. State
7. Federal
8. Indian Lands
9. Mixed Ownership

10. Other
11. Unknown

fac_text has 9 types given in the following:

1. NULL or blank = Not stated in Agency document
2. Waste Storage/Treatment
3. Waste Recycling
4. Government
5. Mining/Extracting/Processing
6. Manufacturing/Industrial
7. Affected Area
8. Residential
9. Other

Chemical

In HazDat dataset, chemicals are recorded with several attributes. The attributes are: basic chemical info, proj_type, media_txt and s_loc_txt. The chemical info includes the following items, in which cas_id is a standard code used to uniquely identify a chemical substance from the Chemical Abstracts Service (CAS) Registry of chemical substances. Proj_type is a code to specify the type of Agency document or project.

1. cas_id
2. substance name
3. substance_rank

4. substance_class
5. NTP cancer classification
6. TSCA regulated
7. FIFRA regulated
8. RCRA regulated
9. CERCLA regulated (example: http://www2.atsdr.cdc.gov/gsql/getsubstance.script? in_cas=000079-01-6)

Media_txt is the specific media sampled during a site/event investigation containing the specified contaminant. There are 24 medias in the dataset:

1. NULL or blank
2. Crops
3. Farm/Domestic Animal
4. Fish
5. Game Animal
6. Shellfish
7. Air
8. Soil Gas
9. Human
10. Waste Materials/Containers
11. Other Media
12. Unknown Media

13. Sediment
14. Hard Surface (wipe)
15. Sludge
16. Subsurface Soil (>3 depth)
17. Surface/Top Soil (<3 depth)
18. Soil (unspecified depth)
19. Unknown
20. Groundwater, Public
21. Groundwater Monitor (including test pit)
22. Groundwater, Unspecified
23. Groundwater, Private
24. Contained Material (drum, tank, etc.)

S_loc_txt denotes the location, on-site or off-site, where the contaminant sample was taken.

1. NULL or blank
2. Offsite
3. Onsite
4. Not Reported = Information not reported in document
5. Max_conc
6. Conc_unit_txt
7. Conc year: The year in which the contaminant sample was analyzed.

- **HazDat Data Relation Analysis**

  Sites have attributes of id, public health threat category, chemicals found, position and its basic information as listed above. Chemicals have attributes of id, contaminant, media type, onsite/offsite, and concentration. It's clear to see that these attributes can be connected with entities of sites or chemicals. But to consider relations between those data, we could connect sites that are nearby together since chemicals in an area may affect public health in several nearby locations. And to define nearby, we could calculate the distance between two sites, and put a threshold on it.

  This way we get a graph with relational data, and with potential patterns to explore. Other representational variants including indexing attribute choices with numbers, so using a number to take the place of long description text. Also, concentrations can be grouped into categories.

- **Graph Representation of HazDat**

  A sample graph of the HazDat data is shown in Fig. 13.

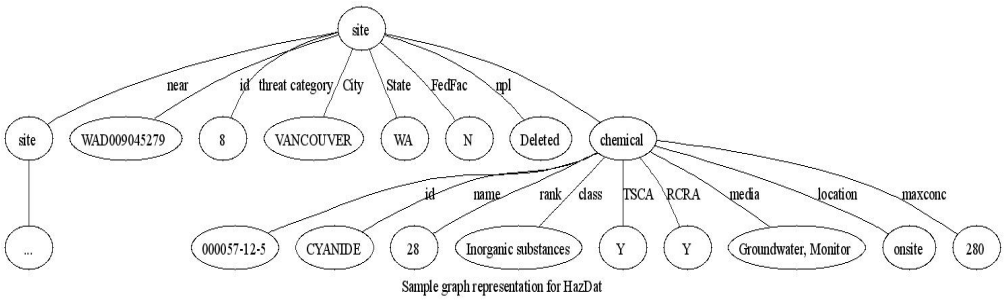

Figure 13: Sample graph representation for HazDat with one chemical present.

- **SEER Data for Cancer**

The Surveillance, Epidemiology, and End Results (SEER) program of National Cancer Institute provides an authorized source of cancer incidence in the United States. The SEER program collects data on patient location, tumor site, tumor morphology, cancer treatment, and vital status etc. The data are further grouped into nine files by sites: Breast, Colon and Rectum, Other Digestive, Female Genital, Lymphoma of All Sites and Leukemia, Male Genital, Respiratory, Urinary and all Other Sites.

This dataset can be accessed via http://seer.cancer.gov/data/ and a signed copy of agreement.

- **SEER dataset Description**

  SEER documentation gives a very informative explanation of the file coding. Some selected important characteristics are list below:

  - Location;
  - Behavior Code: 0 Benign; 1 uncertainty; 2 noninvasive; 3 Malignant.
  - Diagnostic Confirmation: 1 Positive histology; 2 Positive cytology; 4 Positive microscopic confirmation; 5 Positive laboratory test/marker study; 6 Direct visualization without microscopic confirmation; 7 Radiology and other imaging techniques without microscopic confirmation; 8 Clinical diagnosis only (other than 5, 6, or 7); 9 Unknown whether microscopically confirmed; death certificate only
  - Tumor Marker 1: 0 None Done; 1 Positive; 2 Negative; 3 Borderline; 8 Ordered, but results not in chart; 9 Unknown or no information.

- **SEER dataset Analysis**

  SEER provides cancer incidence information in a detailed manner. But it is a fairly flat dataset, since the information basically consists of characteristics of a person. But consider the relations between the HazDat which provides potential hazardous chemical information of a particular location, and the SEER dataset which provides cases of personal cancer incidence of that location. We can relate the two datasets together, and explore the pattern of how the hazardous environment (by HazDat) would affect public health (by SEER) in more detail.

- **Sample Graph of SEER**

  Fig. 14 shows an example of the graph representation for the SEER data.

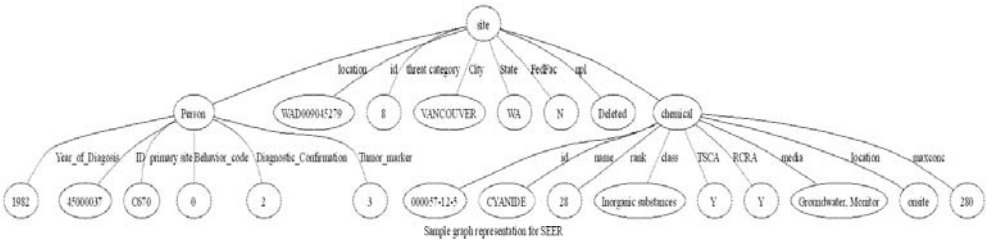

Figure 14: Sample graph of SEER

More Datasets to Explore

There are many other health-related datasets that can be explored for their affinity to a relational graph representation. ICPSR is a large data archive on social science data. It provides datasets in 19 subjects including "Health Care and Facilities", "International

Systems: Linkages, Relationships, and Events", "Community and Urban Studies", and "Economic Behavior and Attitudes"(direct access at http://www.icpsr.umich.edu/ICPSR/access/subject.html).

In the category of "Health Care and Facilities", a poll of 505 sets regarding health issues is provided for download with raw data. This poll constitutes a very promising dataset provider for further exploration of relational datasets.

Conclusion

In this section, three relational datasets which may be explored by Subdue are described and analyzed. By analyzing the dataset into entity-attribute relations, a graph representing the relations in the data is constructed. Subdue is capable of analyzing such relational graphs and find the intrinsic patterns inside the graph.

With those datasets, Subdue can be applied, and the knowledge discovered may be used as reference to health officials. But to focus on studying the search algorithms in graph-based data mining, experiments are only conducted on the pandemic dataset using the search algorithms we studied in chapter three. And the experiment results are described in chapter six. Application of Subdue to the additional public-health datasets is a direction for future research.

CHAPTER SIX

GRAPH-BASED SEARCH ALGORITHM EXPERIMENTS ON PANDEMIC DATASET

The graph representation of the pandemic dataset is discussed in chapter five. And in this chapter, both unsupervised learning and supervised learning will be conducted on the pandemic dataset.

Unsupervised learning experiment results on 100 person's Pandemic dataset

To do unsupervised learning on the Pandemic Dataset, the dataset is represented as a graph as shown in Fig. 5. In the dataset, there are 1,600,880 persons, with ID ranging from 2,000,218 to 3,601,098. In our graph representation, a set of 100 persons are randomly sampled from the pool of population in the dataset. The communication and infection events link the person nodes together. In the resulting one big graph, there are 16,230 vertices, 16,194 edges and 3,661 unique labels.

Running times of search algorithms on the pandemic dataset are plotted in Fig. 16. The best patterns found by the search algorithms are shown in Fig. 15. Table 15 lists the running time and pattern values. Because the best substructures are all found at a depth limit of 70, only the results of search algorithms with depth limit of 70 are listed in the table.

From the time plot shown in Fig 16, we can see that the time plots are again flat with various depth limits. This trend is again very consistent with plots of credit dataset and that of criminal and social network dataset.

From the value chart (Fig. 15) and Table 15, SA_Greedy and HC with Stochastic Escape again find the best substructure with value 1.22358. The best value found by beam

search is 1.0956, which is less than the value found by SA_Greedy. Stochastic HC finds 1.00302. Simulated Annealing finds a structure with value 1.01033. And Hill climbing also finds the best substructure with value of 1.22358.

Table 15: Search algorithms results on Pandemic dataset with 100 persons in one graph (Depth 70).

	Best pattern	2nd pattern	3nd pattern	time(s)	depth limit
Beam Search	1.0956	1.0956	1.03775	3.91	70
Eff Depth limit	–	–	–	–	8
Hill Climbing	1.22358	1.0956	0.999962	69.97	N/A
SA_Greedy	1.22358	1.0348	1.01241	489.44	70
Simulated Annealing	1.01033	1.00586	1.00575	295.77	70
HC with Stochastic Escape	1.22358	1.01279	1.01254	403.40	70
Stochastic HC	1.00302	1.00189	1.00138	297.01	70

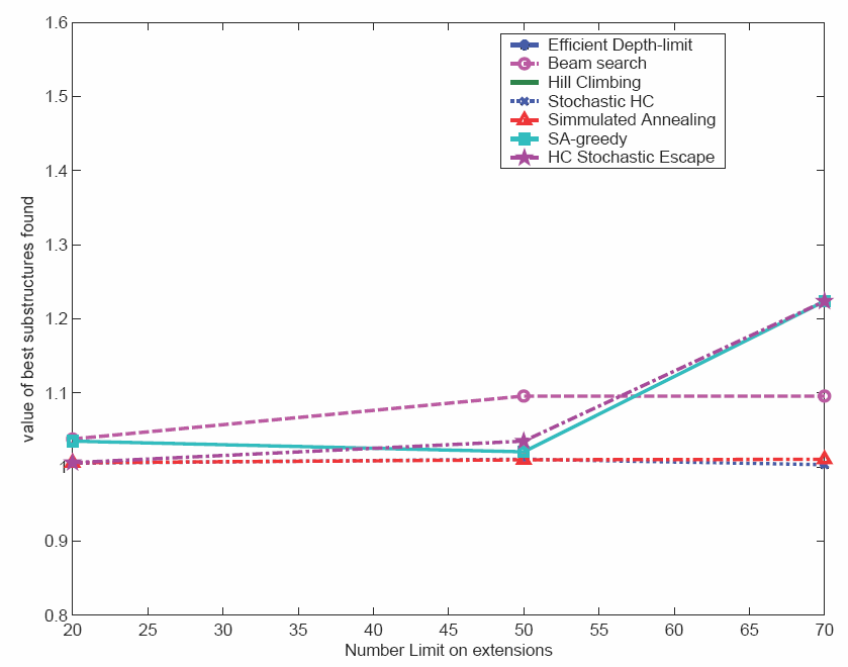

Figure 15: Best Pattern Value plot of Pandemic dataset

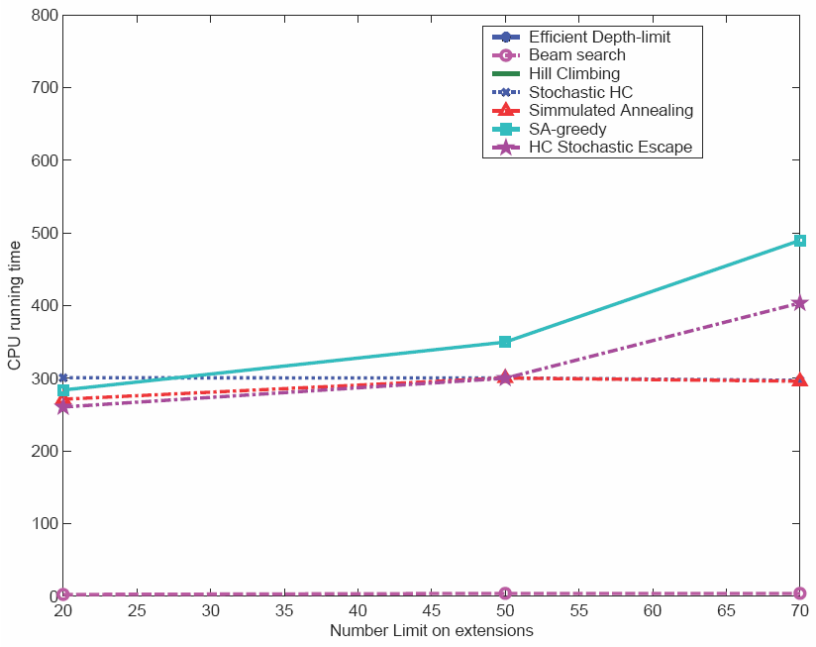

Figure 16: Time plot of Pandemic dataSet

Unsupervised learning experiment results on 300 person's Pandemic dataset

In this experiment, the dataset is represented as a graph as shown in Fig. 5, and 300 persons are sampled from the raw data pool. The communication and infection event links the person nodes together. In the resulting one big graph, there are 46,935 vertices, 46,769 edges, and 9,278 unique labels.

Running times of search algorithms on the pandemic dataset with 300 person samples are plotted in Fig. 18. The best patterns found by the search algorithms are shown in Fig. 17. Table 16 lists the running time and pattern values in number. Because the best substructures are all found at a depth limit of 50, only the results of search algorithms with depth limit of 50 are listed in the table.

From the time plot shown in Fig 18, we can see that the time plots are again flat at various depth limits, which is consistent with previous ones.

From the value chart (Fig. 17) and Table 16, SA_Greedy and HC with Stochastic Escape again find the best substructure with value 1.23919. The best value found by beam search is 1.10209, which is less than the value found by SA_Greedy. Stochastic HC finds 1.00481. Hill climbing finds a substructure with value of 1.01487. And Simulated Annealing performs worst by finding a structure only with value 1.00457.

Table 16: Search algorithms results on Pandemic dataset with 300 persons in one graph (depth 50).

	Best pattern	2nd pattern	3nd pattern	time(s)	depth limit
Beam Search	1.10209	1.10209	1.04134	39.75	50
Eff Depth limit	–	–	–	–	8

Hill Climbing	1.01487	0.99987	–	59.71	N/A
SA_Greedy	1.23919	1.01487	1.01239	6021.36	50
Simulated Annealing	1.00457	1.00382	1.00368	1924.74	50
HC with Stochastic Escape	1.23919	1.02074	1.00459	3670.78	50
Stochastic HC	1.00481	1.00371	1.00217	1876.02	50

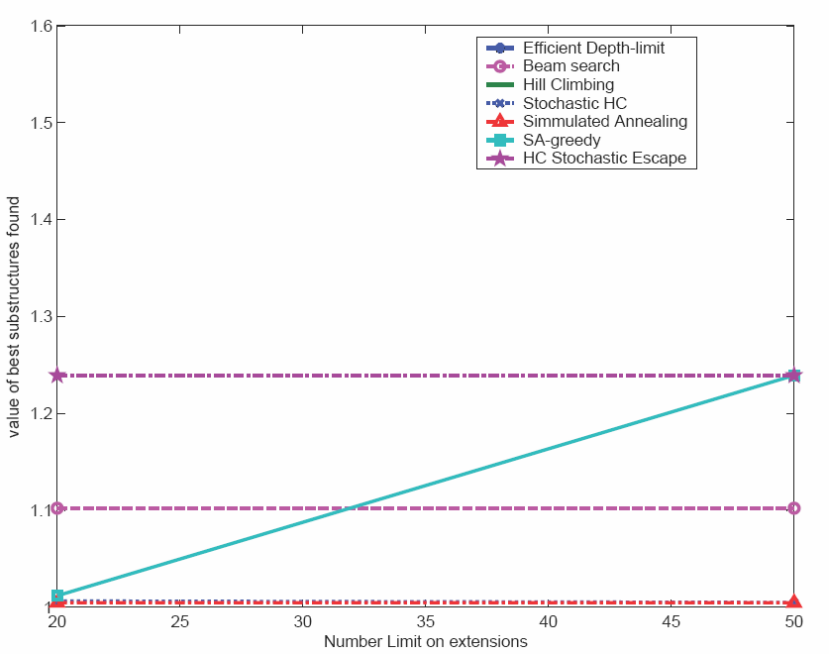

Figure 17: Best Pattern Value plot of Pandemic dataset

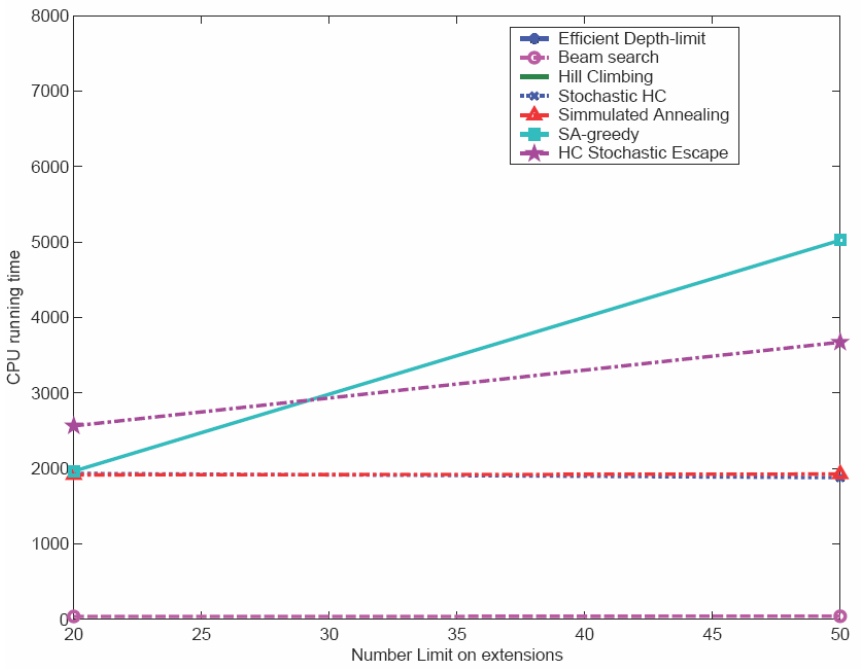

Figure 18: Time plot of Pandemic dataset

Supervised learning experiment results on Pandemic dataset

To do supervised learning on the Pandemic Dataset, the dataset is represented as a graph as shown in Fig. 12. Ten persons are sampled from the pool, each as an origin to develop a relational communication and infection graph. The ten persons just provide the center to develop the graphs. The graph is growing fast by adding the other end of communication to the original person, and then adding the other end of communication to those persons who have been added. In the end each graph contains about 200 persons interconnected with communication and infection.

The graph is designated as a positive example if the origin person is found to be infected; and is designated as a negative example if the origin person is found to be uninfected. There are 4 total positive graphs and 6 total negative graphs. The 4 positive

graphs have a total of 6,704 vertices and 6,740 edges. The 6 negative graphs have 5,159 vertices and 5,143 edges. Thirteen initial substructures are found to start the pattern search exploration.

Search algorithms results on supervised learning in the Pandemic dataset are given in Table 167. SA_Greedy again finds the best substructure with value of 1.96391. Beam search, Efficient Depth-limited and Hill climbing also find the best substructure with value of 1.96391. Stochastic HC finds a substructure with value of 1.89106. And Simulated Annealing again performs worst by finding a structure only with value 1.82342.

Table 17: Search algorithms results on supervised learning in the Pandemic dataset

	Best pattern	2nd pattern	3nd pattern	running time(s)	depth limit
Beam Search	1.96391	1.89106	1.89106	5.15	10
Eff Depth limit	1.96391	1.89106	1.89106	25847.10	8
Hill Climbing	1.96391	1.89106	1.82342	5.49	N/A
SA_Greedy	1.96391	1.82342	1.82342	55.34	10
Simulated Annealing	1.82342	1.72373	1.70169	49.39	10
HC with Stochastic Escape	–	–	–	–	10
Stochastic HC	1.89106	1.72373	1.71728	38.59	10

The best pattern learned by Subdue to distinguish infected from non-infected persons is shown in Fig. 19. Results show that there are 1,536 instances containing this pattern, in which 1,414 are positive instances and 122 are negative instances.

The pattern consists of a person node connecting with a communication node, and the communication node has two attributes showing that both of the two purposes of the communication are “1”, which stands for going to work (by referring to the description of the graph in chapter five). The meaning of the pattern can be expressed as: one person may

be easier to get infected by working with co-workers. Such patterns can give us more insight into how infections propagate and breakout of a community.

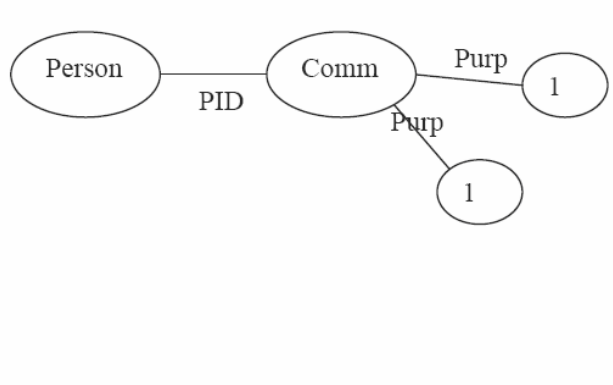

Figure 19: Best patterns found in pandemic dataset

The second best pattern is shown in Fig. 20. Results show that there are 1,433 instances containing this pattern, in which 1,133 are positive instances and 300 are negative instances.

Similar with the best pattern, the second best pattern also consists of a person node connecting with a communication node, and the communication node has two attributes showing that both of the two purposes of the communication are “1”, which stands for going to work. One more attribute shows up in this pattern of the communication: a duration of 100,000. Referring to the description of graph representation of the pandemic dataset in chapter five, the actual duration in seconds is mapped to its closest power of 10. So from “duration of 100,000” it can be understood that the actual duration is between 10,000 seconds to 100,000 seconds (2.78 to 27.8 hours). But in this case, both of the two

parts communicates at work, the duration of the continuous communication at work normally satisfies a value of less than 27.8 hours. So it can be concluded that this pattern shows that a person may be more easily infected by working with co-workers for a duration longer than 3 hours.

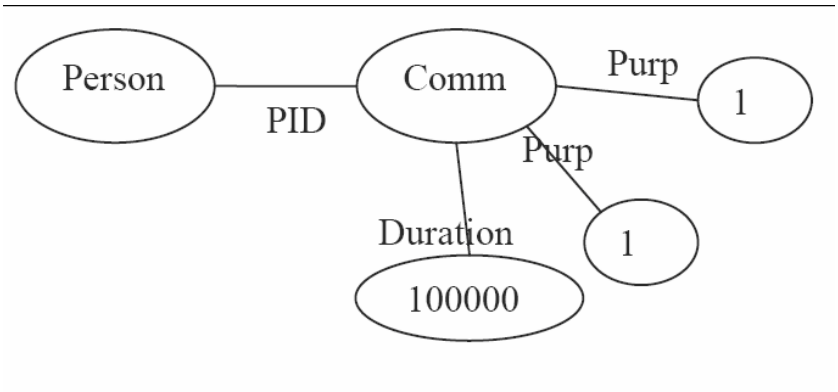

Figure 20: Second best pattern found in the pandemic dataset

Conclusion

In this chapter, both unsupervised learning and supervised learning are conducted on the pandemic dataset. Experiment results are shown both in tables and plots. Interesting patterns are found in the pandemic dataset that indicate people more likely to get infected. Further analysis on comparisons of the search algorithms is presented in the following chapter seven.

CHAPTER SEVEN

SEARCH ALGORITHM EXPERIMENT RESULT ANALYSIS AND COMPARISON

Results on the Criminal and Social Network dataset show that Hill Climbing search has the least running time of 0.06s, but can only find the substructure with the smallest value of 12.8769. The improved Depth-limited search still has a running time of $O(o_{ext}*(|E|)^d)$, which requires much longer time as d increases. So it is impractical to explore substructures that can only be generated with large extension limits. But within the complete search space of a depth of 8, it can still find valuable substructures with a value of 15.3578. Beam search also has a very short running time within 10 seconds, and finds very good substructures after extension limit of 50 though performs not so well for small extension limits. The remaining four search algorithms, Stochastic HC, Simulated-annealing, SA-Greedy, and HC stochastic escape, roughly have the same range of running time and substructure values, but SA-Greedy is distinguished from the other three with the least running time and the best substructure found. Comparing beam search with SA-Greedy, though beam search has less run time, SA-Greedy outperforms Beam search in terms of value of substructure found. SA-Greedy generally finds better substructures or equal substructures for most extension limits, and finds the substructure with the best value in overall results at extension limits of 100.

From the results on the Credit dataset, we can see that Hill Climbing again has the least run time of 2.32 seconds, and still finds the least-valued substructure. The improved depth limited search again has a long running time of $O(o_{ext}*(|E|)^d)$ as depth limit increases,

though finds good valued substructures. Beam search again performs well after an extension limit of 50. In contrast to their behavior on the Criminal and Social Network dataset, SA-Greedy and HC with stochastic escape perform better than the others by finding much better valued patterns in the Credit graph. Though their running times are about 6 times longer, they do not follow an exponential trend in running time as the extension limit increases.

From the performance analysis, the best three algorithms are beam search, SA-Greedy, and HC with Stochastic Escape. But in comparison of the three algorithms, HC with Stochastic Escape finds the least valued substructure on the Criminal and Social Network dataset (Fig. 1), and spends the longest time in the Credit dataset (Fig. 3). So we conclude that beam search and SA-Greedy are the best two.

Comparing the beam search and SA-Greedy, SA-Greedy performs better at finding better substructures in both of the two datasets (Fig. 1 and Fig. 3), though beam search has better running times (Fig. 2 and Fig. 4). But better substructure patterns are our main concern when running time is acceptable and not exponential as the extension limit increases. So we conclude that SA-Greedy is the best search algorithm in graph mining in general for graph mining when searching for highly-compressing patterns. But if speed is a more important issue, then beam search with a depth limit over 50 will be a good choice.

Experiments on the pandemic dataset show the same results with the results of the two datasets we discussed above. Small datasets tend to have the similar plot result as plots of Criminal and Social Network dataset in which the four search algorithms, Stochastic HC, Simulated-annealing, SA-Greedy, and HCSE, roughly have the same range of running time,

and all the search algorithms except hill climbing find the same range of best substructure values. Large datasets tend to have the similar plots as Credit dataset, where SA-Greedy finds significantly better patterns than other algorithms. SA-Greedy and HCSE have longer running times than stochastic HC and Simulated-annealing in large datasets.

There are common characteristics in the results of large datasets and small datasets. Beam search and hill climbing have much shorter running times than the other algorithms; and Efficient Depth-limited runs much longer. The running times are also consistent with the calculated time complexities of algorithms. Efficient depth-limited has the largest time complexity of $O(o_{ext}*(|E|)^d)$; and all the other six search algorithms have the time complexity of $O(o_{ext} \times d)$, which is proportional to depth limit *d*.

CHAPTER EIGHT

CONCLUSIONS

To discover better substructures in less time, seven search algorithms are studied in application to graph-based relational mining. Results show that Hill climbing gets trapped at local maxima at a very early stage, preventing it from exploring further extensions. This shows that the problem of searching for the best compression substructures by extension is complex. The results of Efficient Depth-limited search confirm that the extension space of substructures in a graph is $O(o_{ext}*(|E|)^d)$, causing the complete-space search algorithm to be impractical and inefficient for exploring the substructures at higher extension depths. Simulated-annealing and Stochastic HC cannot find the best substructure suggesting more study on the scheduler function. Beam search may discard the substructure which is not among the best ones at present, but becomes best with further extensions. SA-Greedy integrates Greedy search by following the best one first, and integrates simulated annealing that potentially jumps out to other extension paths leading to the best substructure pattern. To achieve stable results on graph mining, SA-Greedy tries more paths compared to Beam search, causing a cost of several times longer running time than beam search, but SA-Greedy is capable of finding the best valued substructure among all seven algorithms studied.

One of the future directions for this work would be to study different scheduler functions in SA-Greedy. The study of coefficients in the scheduler may result in even better performance of the SA-Greedy algorithm. Also, we plan to try these search algorithms in other graph mining systems in an attempt to make the general graph mining process much

more efficient. We also plan to apply graph mining to the other public-health datasets discussed earlier in order to discover structural patterns that can help us better understand and prevent the spread of disease.

APPENDIX

The appendix lists the experiment results of the best values found by stochastic-based algorithms under different numbers of iterations on Groups dataset. The point is discussed in chapter four, and the supporting data are listed here.

The experiment plot (Figure A.1) and tables (Table A.1, A.2 and A.3) show that the algorithms reach a flat stage at iteration of 100. Running the stochastic-based search algorithm 100 times will produce much stable result during multiple experiments. So we choose the number of iterations to be 100 in the following experiments.

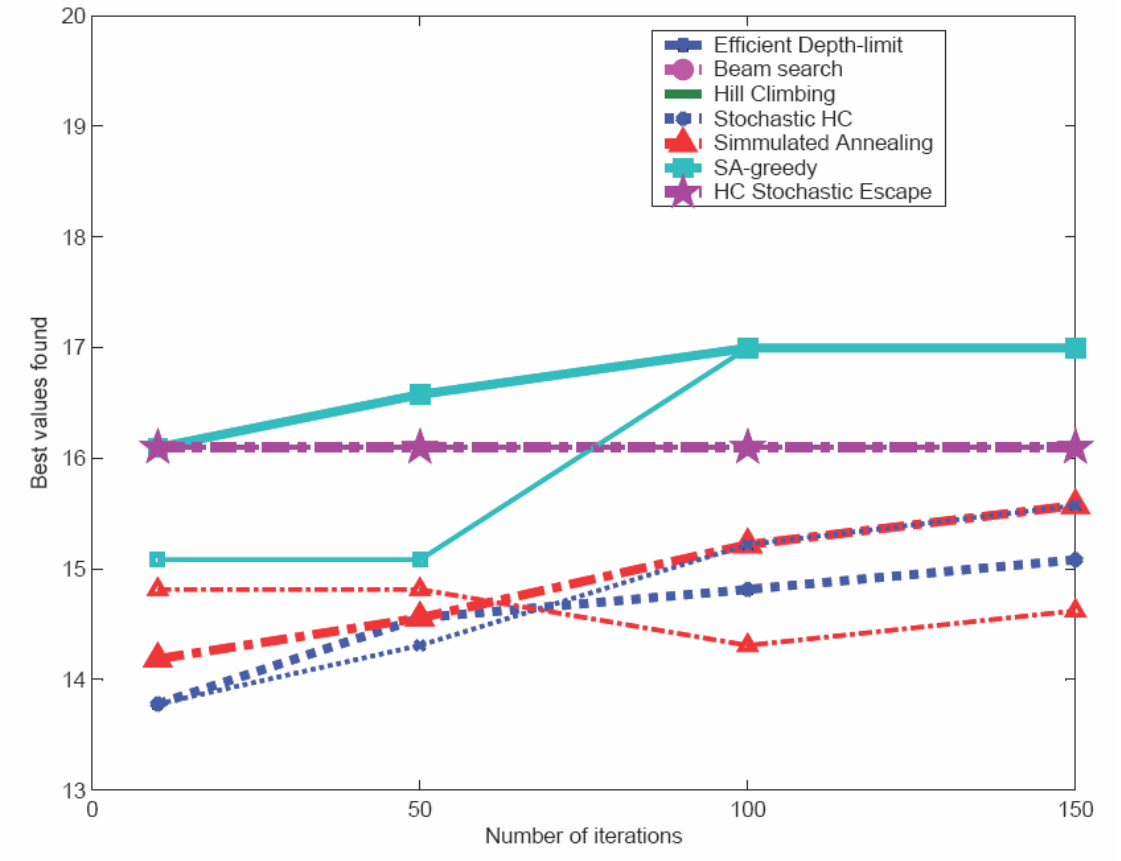

Figure A.1 Plot of best values found under different numbers of iterations on Groups dataset. (The wider line plots the experiment with depth limit of 70. The thinner line plots the experiment results with depth limit of 50)

Table A.1 Best values found under different numbers of iterations on Groups dataset (depth limit 70)

Iteration	10	50	100	150
HC with stochastic escape	16.0962	16.0962	16.0962	16.0962
Stochastic HC	13.7778	14.5565	14.8142	15.0811
Simulated annealing	14.1864	14.5565	15.2182	15.5721
SA-greedy	16.0962	16.5743	16.9949	16.9949

Table A.2 Best values found under different numbers of iterations on Groups dataset (depth limit 50)

Iteration	10	50	100	150
HC with stochastic escape	16.0962	16.0962	16.0962	16.0962
Stochastic HC	13.7778	14.3077	15.2182	15.5721
Simulated annealing	14.8142	14.8142	14.3077	14.6201
SA-greedy	15.0811	15.0811	16.9949	16.9949

Table A.3 Best values found under different numbers of iterations on Groups dataset (depth limit 10)

Iteration	10	50	100	150
HC with stochastic escape	13.7778	14.6842	14.6842	14.6842
Stochastic HC	14.6842	14.5565	13.7778	14.8142
Simulated annealing	12.8769	14.8142	16.0191	16.0191
SA-greedy	14.8142	14.8142	14.8142	14.8142

References

[1] M. Cohen and E. Gudes. Diagonally subgraphs pattern mining. *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 51–58, 2004.

[2] D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.

[3] D. Cook, N. Manocha, and L. Holder, Using a Graph-Based Data Mining System to Perform Web Search, *International Journal of Pattern Recognition and Artificial Intelligence* 17(5), 2003

[4] T. H.Cormen. *Introduction to Algorithms*. MIT Press, 2001, 2nd edition.

[5] J. Rissanen, Stochastic Complexity in Statistical Inquiry, World Scientific Publishing, Singapore, 1989.

[6] L. Holder, D. Cook, J. Coble and M. Mukherjee, Graph-based Relational Learning with Application to Security, *Fundamenta Informaticae Special Issue on Mining Graphs, Trees and Sequences*, 6(1-2):83-101, March 2005

[7] L. Holder and D. Cook. Graph-based relational learning with application to security. *Fundamenta Informaticae Special Issue on Mining Graphs, Trees and Sequences*, 66(1-2):83–101, 2005.

[8] L. Huan and W. Wang. Spin: Mining maximal frequent subgraphs from graph databases. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–586, 2004.

[9] S. Kirkpatrick and C. D. Gelatt. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[10] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1038–1051, September 2004.

[11] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery*, 11(3):243–271, November 2005.

[12] C. Noble and D. J. Cook, Graph-Based Anomaly Detection, Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003

[13] A. Rakhshan, L. Holder, and D. Cook, Structural Web Search Engine, *Proceedings of the Sixteenth International Conference of the Florida AI Research Society*, May 2003.

[14] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, 1989.

[15] S. Su, D. Cook, and L. Holder, Application of Knowledge Discovery to Molecular Biology: Identifying Structural Regularities in Proteins, *Proceedings of the Pacific Symposium on Biocomputing*, p190–201, 1999.

[16] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 721–724, 2002.

[17] X. Yan and J. Han. Closegraph: Mining closed frequent graph patterns. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 286–295, 2003.

[18] Subdue manual, version 1.3, http://www.subdue.org

[19] S. Bandyopadhyay, U. Maulik, D. J. Cook, L. B. Holder and Y. Ajmerwala, Enhancing Structure Discovery for Data Mining in Graphical Databases Using Evolutionary Programming, *Proceedings of the Fifteenth International Conference of the Florida AI Research Society (FLAIRS)*, May 2002