Indian Institute Of Technology Delhi

Major Project Thesis

Application of Lifted inference technique on Graph Min Cut & MaxWalk-SAT algorithms

A thesis submitted in fulfillment of the requirements
for the Dual Degree in Computer Science Engineering

Author:
Manav Goel

Supervisor:
Dr. Parag Singla

Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

June 2013

Declaration of Authorship

This is to certify that the thesis titled **Application of Lifted inference technique on Graph Min-Cut & MaxWalk-SAT algorithms** submitted by **Manav Goel(2008CS50215)** to the Indian Institute of Technology, Delhi for the award of Dual Degree in Computer Science and Engineering, is a bonafide record of work carried out by him under my supervision and has fulfilled the requirement of this thesis. To the best of my knowledge the contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Parag Singla
Department of Computer Science & Engineering
IIT Delhi

Signed: _____

Date: _____

Acknowledgements

I take this opportunity to express my deep sense of gratitude towards my research supervisor Dr. Parag Singla. It has been my privilege to work under his esteemed supervision. I am very grateful to his for timely advice, valuable suggestion, deep insight into problems and continuous encouragement.

I would also like to extend special thanks to Abhinav, Arunim Samat, Dipanshu Agarwal, Harsh Prasad, Hitesh Kumar, Jagjeet Singh, Kritarth Anand, Kshiteej Mahajan and Mohit Meena for their inspiration and enthusiasm they brought into this project.

Last but not least I would like to thank my parents and elder sister whose love and blessings brought me up here and gave me the strength and determination to complete my work with utmost sincerity.

INDIAN INSTITUTE OF TECHNOLGY DELHI

Dr. Parag Singla
Department of Computer Science Engineering

Dual Degree in Computer Science Engineering

Application of Lifted inference technique on Graph Min-Cut & MaxWalk-SAT algorithms

by Manav Goel

Abstract

Statistical relational models such as Markov logic networks [1], bring the power and compactness of first-order logic to probabilistic graphical models. However, inference in them is generally still at the level of propositional logic, creating all ground atoms and formulas and applying standard probabilistic inference methods to the resulting network. Inference in the most ideal case, should be lifted similar to that being done in first order logic, handling whole sets of indistinguishable objects together, irrespective of their cardinality. In this piece of work I have worked with one of the algorithms [2] for lifting, and using this technique presented the Lifted-MaxWalkSAT algorithm for infering the most probable state of the network. Apart from this we also worked on proposing a lifted version of the Graph Cut algorithm. We then show by performing experiments over various domains that using the lifting technique, more accurate results can be achieved while siginificantly improving upon the running time as well.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Probabilistic inference algorithms are employed very widely in the field of artificial inteligence. In a majority of algorithms for inference that have historically been presented [3], [4] involve their processing at propositional level. In domains with a large number of objects this may be both costly and unnecessary, especially if there exist symmetry in the behaviour of the objects i.e. having the same interactions in an abstract sense. Over the previous few years, combining first order logic with probabilistic inference has been a major area of research and there have been some proposals and algorithms developed that essentially try to combine the objects and instances with same behaviour during the course of the algorithm. Poole (2003) [5] and Braz et al. (2005, 2006) [6] developed a lifted version of the variable elimination algorithm, but it is extremely complex, generally does not scale to realistic domains, and has only been applied to very small artificial problems. [2] worked on lifting Belief Propogation technique and showed experimentally its advantages.

Taking inspiration from this approach, we worked to create a lifted version of the MaxWalkSAT algorithm, we show that with lifting the network by combining the entities that behave in an identical manner, the inference regarding the most probable state can be acheived with significant advantages in terms of time taken. We present the first experimental results for lifted probabilistic inference for MaxWalkSAT on real data. These, and systematic experiments on synthetic problems, show that lifted MAxWalkSAT can greatly outperform the standard propositionalized version. We also worked on creation of Lifited Graph min cut algorithm and theoretically proved its correctness. There has been some recent work on lifting WalkSAT [7] which has progressed independently from ours.

1.2 Background and Related Work

1.2.1 Markov Logic:

First-order probabilistic languages combine graphical models with elements of first-order logic, by defining template features that apply to whole classes of objects at once. A simple and powerful such language is Markov logic (Richardson Domingos 2006). A Markov logic network (MLN) is a set of weighted first-order clauses. Together with a set of constants representing objects in the domain of interest, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by $P(x) = 1/Zexp(\sum_i w_i * g_i(x)) = 1/Z(\prod_i f_i(x))$, where w_i is the weight of the ith clause, $g_i = 1$ if the ith clause is true, and $g_i = 0$ otherwise.

1.2.2 Lifted Inference:

A majority of the probabilistic inference algorithms are specified and processes on a propositional leve i.e. they must be specified variable by variable. This in turn increasing the processing time of the entire process as the size of such a network is pretty large. The concept of lifted inference is simple and intuitive in nature. Lifted inference algorithms avoid repeated computations by essentially treating indistinguishable group of objects/nodes as a single entity, thereby avoiding the redundant computations and reducing the computational time and cost. There has been a lot of research over the past decade for recognizing how exactly to recognize those set of features and thereby combining the nodes for reduction in computational time

1.2.3 WalkSAT algorithm

Satisfiability (abbreviated SAT) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. In other words, it establishes if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE.
WalkSAT is a local search algorithm to solve the boolean satisfiability problem. Once the problem is converted into a conjunctive normal-form, it starts assigning a random value to each variable. If the initial algorithm satisfies all clauses, then the algorithm

terminates. Otherwise, pick a random unsatisfied clause. With probability p flip a random atom in the clause, with 1-p pick an atom (greedily) which gives most benefit in terms of reducing the unsatisfied cost.

Inference in Markov logic can be carried out by creating the ground network (by reducing it to the Conjuctive normal Form and creating all instances for all the variables to create a ground network) and applying MaxWalkSAT algorithm over it to get the most probable state of the entire network, but owing to the huge size of the ground network O (d^c) where d is the number of objects in the domain and c represents the highest clause arity. In the following chapters we describe a lifted version of the algorithm for inference and verify the benefits by testing the same over multiple domains.

1.2.4 Graph Cuts

An (s,t)-cut (or just cut if the source and target are clear from context) is a partition of the vertices into disjoint subsets S and T such that s belongs to S and t belongs to T. A minimum cut of a graph is a cut whose cutset has the smallest number of elements (unweighted case) or smallest sum of weights possible. Several algorithms exist to find minimum cuts. A vast number of problems across multiple domains can be modelled as finding the minimum cut/maximal flow across a graph. eg. Image segmentation in vision applications, airline scheduling, bipartite matching.

1.3 Problem Formulation

Taking inspiration from the previous successes of various lifted techniques for inference, we try to come up with solutions to the problems in these two inference related domains

- To design implement lifted technique for Walk-SAT problem by identifying the entities which behave identically
- Working on optimisation of the Graph min-cut algorithm by applying Lifted inference techniques to address minimization/ optimisation problems in multiple domains.

1.4 Our Approach

An overview to the systematic approach that has been followed is described below:

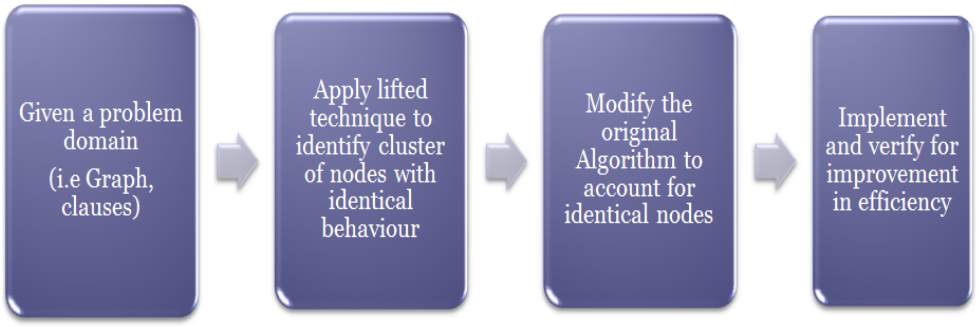

FIGURE 1.1: figure depicting overview of the approach used

The first and foremost work done was to recognize the domains in which the idea is to be applied, following which taking a set of weighted first order clauses and evidence, a Markov Logic Network is generated over which lifting technique is applied as a precursor to be applied towards the original algorithm. The benefits of the lifting technique are then evaluated by carrying out experiments over some real life domains and data-sets and aritificially simulated networks.

1.5 Thesis Outline

The rest of the report is structured as follows:

- Chapter 2 describes the concept of lifted technique for inference.
- Chapter 3 describes in depth the Lifted WalkSAT and Lifted Graph Cut Algorithms
- Chapter 4 descibes our evaluation and the results
- Chapter 5 finally concludes the work and also suggests some future directions.

Chapter 2

Lifted Inference Technique

2.1 Lifting Inference Technique Description

The technique used for lifting that is described below has been taken from [2]. The description and important terminologies are stated as below.

Let us assume that we have an existing MLN **M**, set of constants **C** and set of ground literals (i.e.the evidence database) **E**.

2.1.1 Supernode/Superpredicate

A supernode/ superpredicate is a set of groundings of a predicate that all send and receive the same messages at each step of lifting given M,C and E. The supernodes of a predicate form a partition of its groundings. We shall be using the terminologies supernodes and superpredicates interchangeably throughout this thesis.

2.1.2 Superfeature/Superclause

A superfeature/supeclause is a set of groundings of a clause that all send and receive the same messages at each step of lifting, given M,C and E. The superclauses of a clause form a partition of its groundings. We shall be using the terminologies superclauses and superfeatures interchangeably throughout this thesis.

2.1.3 Lifted Network

A lifted network is a factor graph composed of supernodes and superfeatures. A supernode and a superfeature have an edge between them iff some ground atom in the supernode appears in some ground clause in the superfeature. Each edge has a positive integer weight.

Initially, the groundings of each predicate fall into three groups: known true, known false, and unknown. Atleast one of them is non-empty. Each such group constitutes a supernode initially. Now the process of sending messages start: The groundings of a clause whose atoms have the same combination of truth values (i.e. truth, false or unknown) now send the same messages to the ground atoms in them. This will now essentially start separating the superclauses into finer ones. In turn, all the ground atoms that receive the same number of messages from the superclauses they appear in send the same messages and constitute a new supernode. Now this process gets repeated again and again, thereby creating more refined set of supernodes and superclauses.
If a clause involves predicates $R_1, R_2, ..., R_K$ and $N = (N_1, N_2, ..., N_K)$ is a corresponding tuple of supernodes, the groundings of the clause generated by N are found by joining $N_1, N_2, ..., N_K$ (i.e. selecting tuples in which the corresponding arguments agree with each other and with any corresponding constants in the first order clause). Conversely, the groudings of predicate R_i connected to the elements of a superclase F are obtained by projecting F onto the arguments it shares with R_i. Therefore the lifted network construction essentially proceeds by alternating between two steps:
1 Form superclauses by doing joins of the supernodes.
2 Form supernodes by projecting superclauses down to their predicates, and merging atoms with same projection counts.

The psuedocode for the same is as follows:

input : M, a Markov logic network
C, a set of constants
E, a set of ground literals
output: L, a lifted network

foreach *predicate P* **do**
  foreach *truth value t in* {*true, false, unknown*} **do**
    form a superpredicate containing all groundings of P with truth value t
  end
end
repeat
  foreach *clause C involving predicates* $P_1, P_2, ...P_k$ **do**
    foreach *tuple of superpredicates* $(N_1, N_2, ...N_k)$ **do**
      form a superclause SC by joining $N_1, N_2, ...N_k$
    end
  end
  foreach *predicate P* **do**
    foreach *superclause SC it appears in* **do**
      S(P,SC) ← projection of the tuples in SC down to the variables in P.
      foreach *tuple s in S(P,SC)* **do**
        T(s,SC) ← number of SC's tuples that were projected into s
      end
    end
    form a new supernode from each set of tuples in S(P) with the same
    T(s,SC) counts for all SC
  end
until *convergence*;
add all current superclauses and superpredicates to L
return L

Chapter 3

Application of Lifted Inference Technique

3.1 Lifted MaxWalk-SAT Algorithm

3.1.1 MaxWalkSAT

In Markov Logic, the MAP inference problem reduces to finding the truth assignment that maximizes the sum of weights of satisfied clauses. MaxWalkSAT is a weighted variant of the WalkSAT Boolean satisfiability solver [8]. It performs this search by picking an unsatisfied clause at random and flipping the truth value of one of the atoms in it. With a certain probability this atom is chosen at random, otherwise this atom is chosen to maximise the sum of satisfied clause weights when it is flipped. The major advantage of this algorithm is that since it is a combination of random and greedy steps, the probability of the MaxWalkSAT being stuck in local optima while searching decreases.

3.1.2 Lifted MaxWalk-SAT

The approach to the Lifted MaxWalkSAT is to form an MLN, then apply lifting technique and then apply the procedure of MaxWalkSAT on the lifted graph. The psuedocode for the same is as described below:

```
input  : M: Markov Logic Network
         C: set of Constants
         E: set of Ground Literals
         t: maximum number of tries
         f: mazimum number of flips
         target: the target cost for the false clauses
output: The inference based on the Most Probable state of the network

/* output of LiftedInf is a lifted graph of superpredicates and
   superclauses                                                  */
liftednet ← LiftedInf(M,C,E)
foreach superpredicate SP_i in liftednet do
  | create a dummy predicate DP_i and store mapping in a hashmap
end

foreach superclause SC_i in liftednet do
  | create a dummy clause DC_i by replacing superpreds with dummypredicates
  | store the mapping for dummyclause to superclause
  | weight of DC_i = weight SC_i * number of groundings in SC_i
end
network ← Set of dummyclauses DC_1, DC_2, ...DC_n and set of dummypredicates
DP_1, DP_2, ...., DP_k
vars ← Set of variables in network
for i ← 1 to t do
  | soln ← a random truth assignment to vars;
  | cost ← sum of weights of unsatisfied clauses;
  | for j ← 1 to f do
  |   | if cost <= target then
  |   |   | return soln;
  |   | end
  |   | else
  |   |   | c ← randomly chosen unsatisfied clause.
  |   |   | if Uniform(0,1) < p then
  |   |   |   | v ← a randomly chosen variable from c
  |   |   | end
  |   |   | else
  |   |   |   | foreach variable tmp in c do
  |   |   |   |   | compute ChangeInCost(tmp)v̇ ← tmp with lowest
  |   |   |   |   | ChangeInCost(tmp)
  |   |   |   | end
  |   |   | end
  |   | end
  |   | soln ← soln with v flipped;
  |   | cost ← cost + ChangeInCost(tmp)
  | end
  | return soln;
end
```

3.2 Lifted Graph Min Cut Algorithm

3.2.1 Graph-MinCut

In graph theory, a cut is a partition of the vertices of a graph into two disjoint sets. An s-t cut is essentially a partition of the vertices of the graph into two disjoint sets (A,B) such that vertex 's' is in A and 't' in B. The capacity of the s-t cut is defined as the sum of all the edges that go out of the set A. The goal of a graph min-cut algorithm is to find a cut in the graph with minimum capacity. Many of the problems across multiple domains can be reduced to finding the minimum cut across a graph. eg. image segmentation problem can be reduced to the energy minimization problems [9], scheduling problems for airlines etc. Ford et. al, [10] proved that min cut problem in a graph is equivalent to finding the max-flow in a graph.

3.2.2 The Algorithm

Using the lifting technique, we combine the nodes of the graph with the same behaviour, i.e. the same edge weights across its paths from source to destination. Thereafter applying standard graph min-cut algorithms

3.2.3 Proof of Correctness

Under the conditions that there are no edges between the nodes which form a part of the same supernode, the following can be claimed

Theorem 1: Consider two paths form the source to destination, say p1 and p2. If two nodes X belonging to p1 and Y belonging to p2 are part of a supernode, then all the subsequent pair of nodes in the path i.e from X to d and Y to d will also belong to same set of supernodes respectively.

Proof
Induction
Base Case: There is exactly one node along the path to the destination. Then for both path p1 and p2, nodes X and Y belong to the same supernode.
Let us now say that there are (n-1) nodes along the path which pairwise belong to the same set of supernodes. Now we take a path length of n nodes. If the last pair of nodes (nodes A' and B' as shown in figure 3.1) belong to different supernodes, then the previous nodes along the path (i.e. M and N) need to belong to different supernodes as

messages sent by the respective nodes will be different, thereby contradicting the earlier statement.

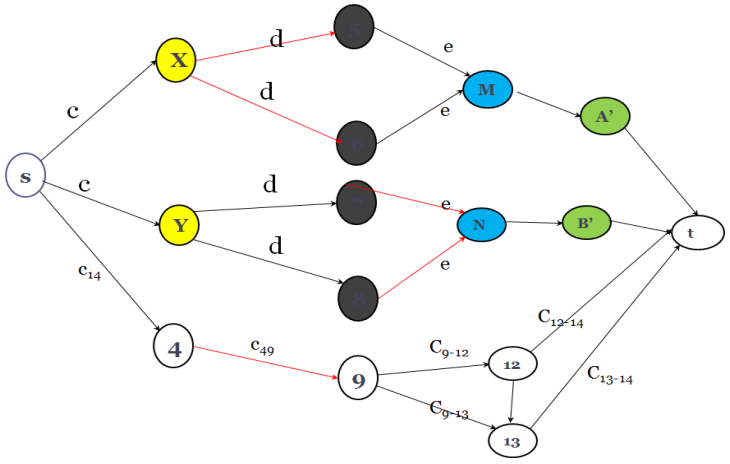

FIGURE 3.1: Figure showing the graph with source 's' and destination 't'

Theorem 2: In atleast one of the optimal solutions, similar behaving nodes (nodes which form a part of the same supernode) must be on the same side of the cut i.e. all the nodes belonging to a supernode will be on the same side of the cut.

Proof:
Let us take a generic graph with vertices V, edges E. Let us consider two paths in the optimal solution from the source s to destination t. Now atleast one edge along each path will be in the minimum cut. If each super node consists of exactly one node, then the lifted graph and the original graph are exactly the same and hence the lifted graph cut is essentially the same as the original graph cut algorithm. Otherwise, now consider the two nodes (nodes X and node Y) that correspond to the same super node.

Consider one of the nodes say X. Let us now focus on the two cases that can happen:

a) Egde emanating from Node X is the saturating edge along the path for s to t. Now we argue that in the optimal case, edge emanating from node Y has the saturating edge capacity. Consider both the paths from 's' to 't'. Using the result of Theorem 1, the subsequent pair of nodes also belong to the same supernodes, hence the edge capacities of the path will be the same.

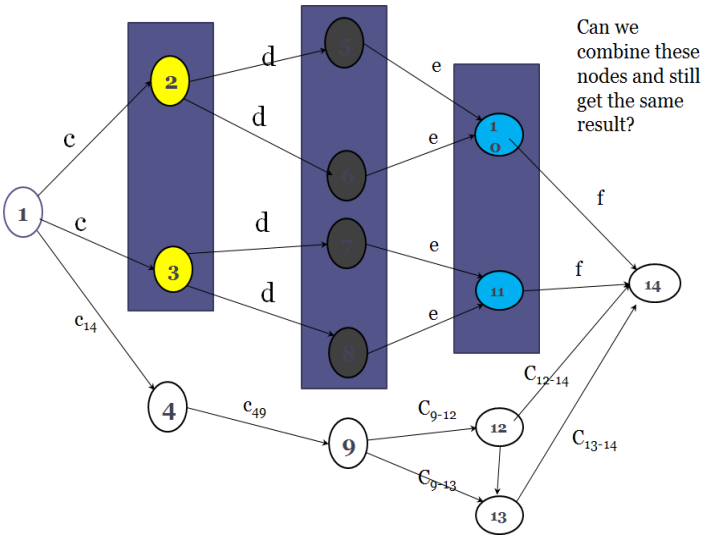

FIGURE 3.2: Figure showing supernodes in the graph

Now as shown in figure, let p1 be the path from source node via node 2 to destination and p2 be the path from source via node 3 to destination. Nodes 2 and 3 belong to the same supernode. Now let 'd' be the saturating edge capacity along p1 and 'e' along p2. The following conditions have to be satisfied

Along path p1: $e - d >= 0$

Along path p2: $d - e >= 0$

$=> d = e$

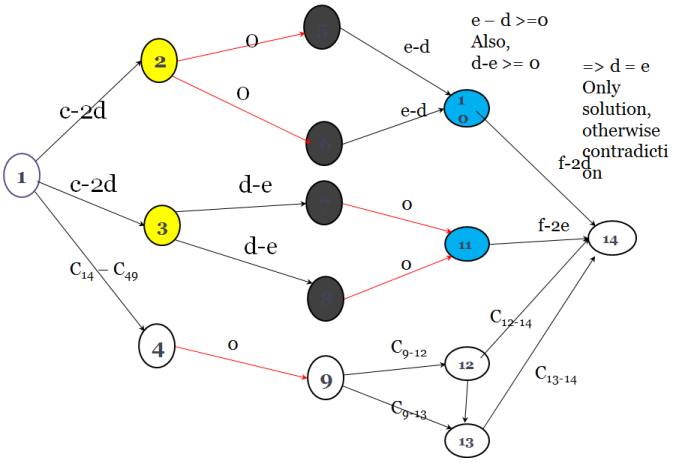

FIGURE 3.3: Figure showing a graph with nodes that can be lifted

Hence proved that the edge emanating from node Y will be the edge with saturating capacity. So in atleast one case we can put the edges from node X and Y to be the saturating edges and hence the nodes X and Y lie on the same side of the cut.

b) Edge emanating from Node X is not the saturating edge along the path form s to t. Here also we make use of the Theorem 1, the subsequent pair of nodes will also belong

to the same set of supernodes. Now some edge along the path is the saturating edge which essentially becomes case 1. Thus there also the nodes lie on the same side of the cut.

Theorem 3: The capacity of the cut in both the graphs remain the same.

Proof Let us examine what exactly the min cut is. It is the minimum across terms, such that (S;V-S)= uv u$\in S; v \in V - S; uv \in E$

We know from Theorem 2, that similar behaving nodes will lie on the same side of the cut, so the terms for comparison can be removed which involved some of the similar behaving nodes in u and others in v. Let us represent the groups of similar behaving nodes as follows:
$SN_1 \leftarrow (N_1, N_2...., N_i)$
$SN_2 \leftarrow (N_(i+1), N_(i+2)...., N_(i+k))$ and so on So essentially the set becomes the following: (S;V-S)=uv u $\in S; v \in V - S; uv \in (SN_1, SN_2....., SN_z)$

This is the exact for of the network with the supernodes and hence the minima of both will be same

Theorem 4: The min cut in lifted graph is the same as the original graph

Proof Let us consider an optimal solution in the original graph satisfying the result as stated in Theorem 2, i.e. a solution where all the similar behaving nodes are on the same side of the cut. Now we need to argue that the supernode will be on the same side of the cut as the node itself. i.e, for an s-t cut, if each of the nodes is on the s side of the cut, then the corresponding supernode will also belong to 's' side of the cut in the lifted graph.
Let us assume that a set of similar behaving nodes lie on the 's' side of the cut and the corresponding supernode in the lifted graph on the 't' side. Now consider the original sub-graph from source to destination which have these nodes in the path from source to destination, since the nodes lie on the s side, the sum of their outgoing capacity is greater than the cut capacity of the cut involving that subgraph. Now consider the subgraph involving only that part of the graph which involves these supernodes, since the supernode is on the 't' side of the cut, sum of its outgoing capacity is greater than that of the cut. But since the capacity of the cut is the same in both the cases as proved in Theorem 3, this means a contradiction. So the similar behaving nodes in the graph in the normal graph and the supernode in the lifted graph will be on the same side of the cut.

Chapter 4

Evaluation and Results

4.1 Experiment Setup

4.1.1 Source Code

The lifted MaxWalkSAT was first implemented by extending the open source software Alchemy [11]. The grounded Max-WalkSAT algorithm had been already implemented as part of the Alchemy software and has been modified absolutely minimally for getting the comparitive time statistics.

4.1.2 Datasets

The testing was done by performing inference on the following three datasets:

- UW-CSE [2]
- Cora [2]
- Smokes -Friends Social Network [2]

The results in terms of timings, number of clauses and accuracy of results was then compared and is summarized in the following sections.

4.2 Determination of the number of flips

The first and foremost task was to determine the number of flips in case of lifted network and the ground network on which the Walk-SAT phase of the algorithm has to be applied.

The number of flips required to converge to an optimum solution i.e leading to the least number of unsatisfied cluases, are bound to be different because of the difference in the structure of the network. If the same number of flips are used, then the advantage of the lifted network to converge faster owing to lesser number of predicates and clauses would be lost. In order to determine the optimal number of flips, we first performed a experiments on the Cora-dataset and also UW-CSE dataset, varying the maximum number of flips and comparing the accuracy of the results and the number of unsatisfied clauses in the network. It was observed that in a majority of cases the number of flips required to converge were directly related to the number of predicates for the flips and on an average they converged at around 50* (number of predicates). Hence in lieu of the deviation in such sceniarios,we took the maximal number of flips in the experiment to be **100*number of predicates**

4.3 Experiments

4.3.1 Link Prediction

Link prediction is a problem that attempts to estimate the likelihood of the existence of a link between two nodes (where nodes are the objects/entities of a network and links represent the relations between the nodes), based on the observed links and the attributes of the nodes. [12]. The experiments for link-prediction were carried out using the UW-CSE database (available publicly on Alchemy's website), which contains around 2678 groundings of predicates like: Student(Person), Professor(Person), AdvisedBy(Person1, Person2), TaughtBy(Course,Person,quarter) etc. The MLN has a total of 94 formulas which specify relationships eg. Each student has at most one advisor, author of a paper has a student and advisor etc. The link prediction task is to predict the student advisor relationship i.e. which teacher is which student's advisor (AdvisedBy (x,y)) predicate.

4.3.1.1 Size statistics

The database W-CSE is divided into five areas (AI, DB etc.) and the results and statistics mentioned below have been compiled using the average of the five subsections of the domain over which the experiments were conducted.

	Grounded	Lifted
Number of (Super) Predicates	5035	2487
Number of (Super) Clauses	193834	130691
Number of false Clauses at solution	17	14
Cost of false Clauses	26.37	25.57

FIGURE 4.1: A graph showing size statistics comparison for UW-CSE

Clearly above we can see that there is a significant difference in the number of (super) clauses and (super) predicates of the lifted and the ground network that have been created for the Lifted and Ground Walk-SAT algorithms. Also at the time of convergence the total number of unsatisfied ground clauses and their total cost is also slightly lower for the lifted network than in ground network.

4.3.1.2 Precision Analysis

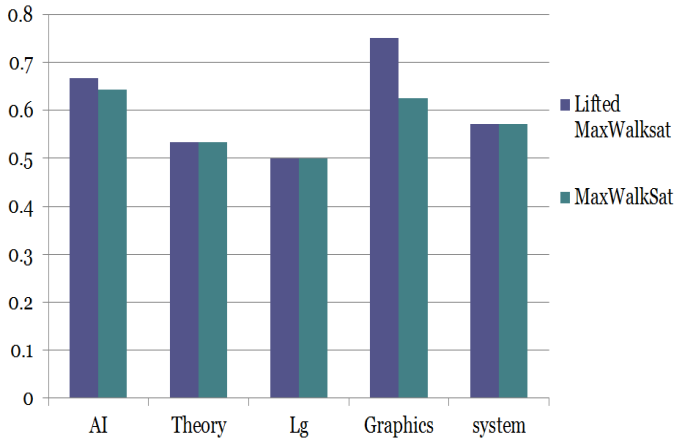

FIGURE 4.2: A graph showing precision statistics comparison for UW-CSE

As can be clearly seen in the graph, the lifted inference gives at least identical precision all of the five divisions of the domain. This is because for reaching the optimal state the lifted network works on the principle that identical entities have same probability of being in a particular state, whether true or false and hence in an optimal state they are most likely to be in the same state. Thus the lifted algorithm flips them all at the same time, while the grounded network starts at a random state and flips them one at

a time and thus it might have either encountered a local minima at stage of the flips to give a slightly less optimal solution as compared to the lifted version at the time of convergence.

4.3.1.3 Time Analysis

As the figure below shows, the lifted Walk-SAT takes on an average slightly higher time as compared to the grounded Walk-SAT primarily because here the bottneck is the time taken for the creation of the lifted network, which takes about 65

	Grounded	Lifted
Avg Time for Network Construction (sec)	11.4	16.22
Avg Time for running Walk SAT (sec)	9.35	4.19
Avg. Total time (sec)	20.75	21.61

FIGURE 4.3: A table showing time statistics comparison for UW-CSE

4.3.2 Entity Resolution

Entity resoution is problem of identifying and linking/grouping different manifestations of the same real world entity. In terms of a dataset it is the task of finding records that refer to the same entity across different data sources. Nowadays, entity resolution is of utmost importance in many real life domains like Linking Census Records, spam detection, in large scientific projects etc. For the entity resolution experiments, the dataset chosen was Cora [13], which comprises computer science research papers. It includes the full citation graph as well as labels for the topic of each paper (and potentially sub- and sub-subtopics). This contains 46 first-order clauses stating regularities such as: if two fields have high TF-IDF similarity, they are (probably) the same; if two records are the same, their fields are the same, and vice-versa; etc. The inference task here was to de-duplicate citations (SameBibliography), same authors, same venue and same title of the research. The following results were obtained :-

4.3.2.1 Network Size Analysis

The Cora-dataset consists of The crux of the advantage for the lifted network is the reduced number of clauses and predicates. In Figure H we can clearly see that the number of predicates are typically close to a ratio of 1:5 for both the predicate count for the lifted network to the original ground network. This influences the running time for the lifted network over the ground network. Similarly from Figure 2, it can be clearly seen that the number of clauses are also significantly lesser in number.

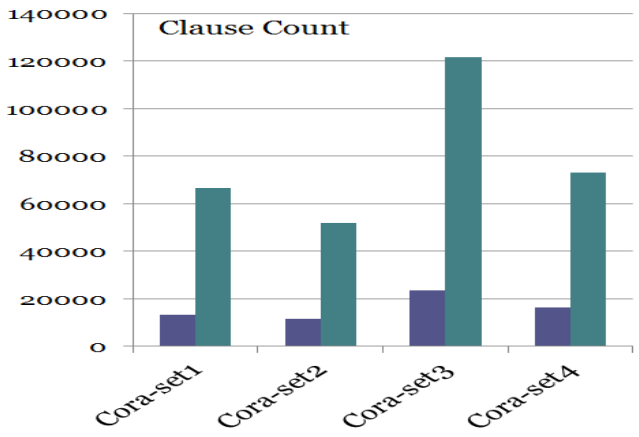

FIGURE 4.4: An image showing (super)clause count for different sub-sets of Cora

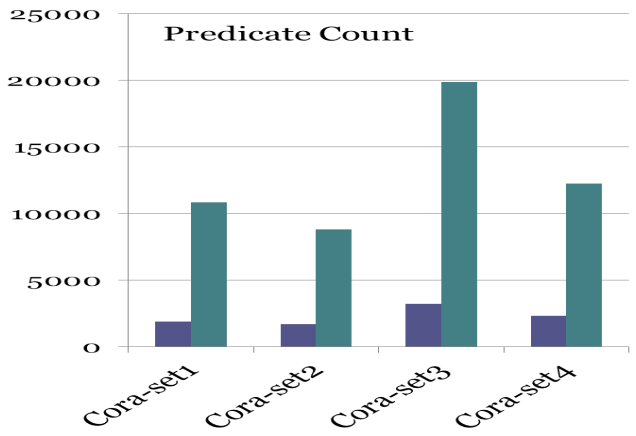

FIGURE 4.5: An image showing (super)predicate count for different sub-sets of cora

4.3.2.2 Time Analysis

One of the major factors that determine the running time of the MaxWalkSAT algortihms is the number of predicates and clauses in the CNF formed of the final ground/super network. As seen from the previous graph, the ratio of the number of predicates and clauses is typically 1:5 for the lifted network vs the ground network and each iteration

involves a maximum of 100*(predicate count) number of flips, it accounts for the huge difference both in absolute and in terms of ratio of the time taken for the two algorithms to complete.

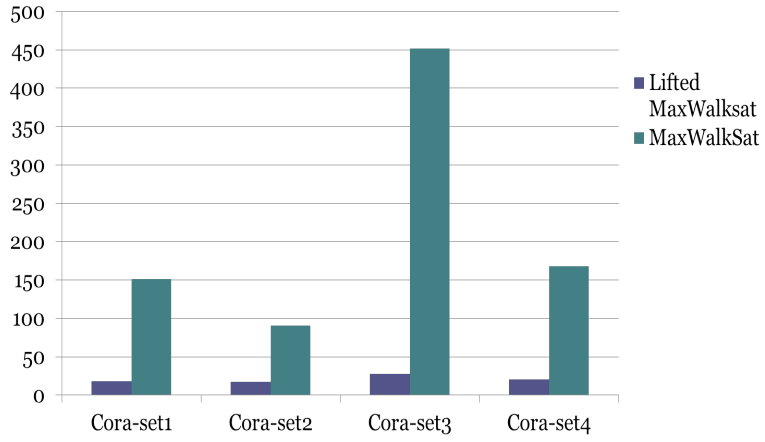

FIGURE 4.6: Graph of time taken (in sec) for the MaxWalKSAT and lifted Max-WalkSAT

4.3.2.3 Accuracy Analysis

The accuracy of the lifted Cora-dataset is significantly higher for the lifted Max Walk-SAT is significantly higher than the standard Max-WalkSAT as shown in Figure below. This can be accounted because given the evidence, we have combined the set of clauses and predicates that behave identically and thus the probability of them being in the same final state is the same. Now given that in the lifted network these are already in the same state (whether true or false) these take a total of 1 flip amongst them to effectively turn them around. On the other hand in the ground network there are 2^n (where n represents the number of grounding of predicates combined to form a superpredicate) combination of states possible and will take higher number of flips. Besides the issue of non-convergence, the Satisfiability algorithms are local search based algorithms that do not necessarily give an optimal solution and can converge to a local minima as well.

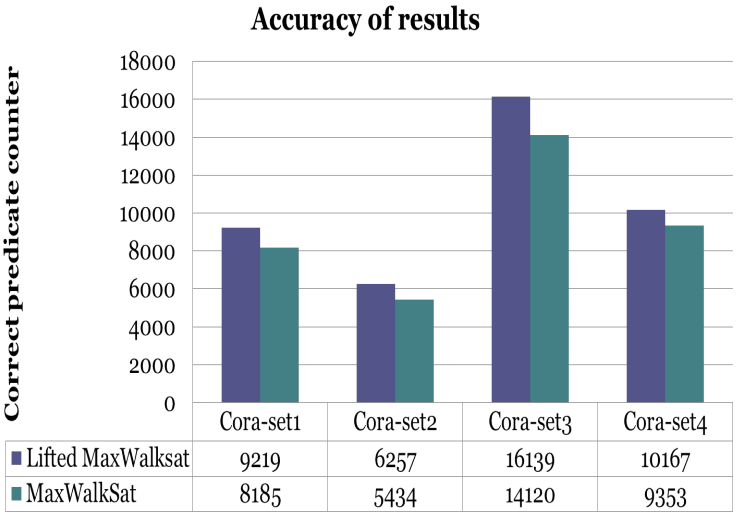

FIGURE 4.7: Graph showing comparison between the number of correct final state predictions by lifted and ground MaxWalkSAT

Now to summarize the entire results are the average statistics for all the four subsets of the data for the Cora-dataset.

	Lifted	Grounded
Number of (Super) Predicates	2279	12927
Number of (Super) Clauses	16178	78230
Number of false (Super) Clauses at solution	1614 (Super) 9467 (Ground)	11885.75
Cost of false Clauses	28361.5	39365.45
Avg. Accuracy	41782/51711 = 0.8077	37092/51711 = 0.7173

FIGURE 4.8: Figure showing the relative statistics

Thus all the above results clearly show the comparitive advantage of the algorithm for the lifted Max-WalkSAT over the standard Max-WalkSAT.

4.3.3 Social Network

The next experiment carried out was with the artificially created simple social network "Friends and Smokers" with a very small set of first order clauses (as listed below). The goal of experimenting with this domain was to see the relative performance of the lifted MaxWalkSAT and normal WalkSAT algorithm with the increasing number of

objects in the domain. The parameters for comparison are the number of grounded / superpredicates and correspondingly the running time of the lifted MaxWalkSAT and MaxWalkSAT algorithms. We varied the number of people from 50 to 500 in increments of 50, and the fraction of known people means that we know for a randomly chosen fraction of all people (a) whether they smoke or not and (b) who 10 of their friends are (other friendship relations are still assumed to be unknown). Cancer(x) is unknown for all x. The people with known information were randomly chosen. The whole domain was divided into a set of friendship clusters of size 50 each. For each known person, we randomly chose each friend with equal probability of being inside or outside the friendship cluster. All unknown atoms were queried.

The predicates for the simple network are:

- !Smokes(x)
- !Cancer(x)
- !Friends(x,y)

- Smokes(x) AND Friends(x,y) $\Rightarrow Smokes(y)$

4.3.3.1 Size Analysis

In such a network where the number of clauses are very less in number and simple, there is a significant difference in the number of (super) predicates, and the difference becomes more and more pronounced with the increment of the number of persons in the network. Same is the case with the number of grounded / super clauses in the network as can be seen clearly with graphs below.

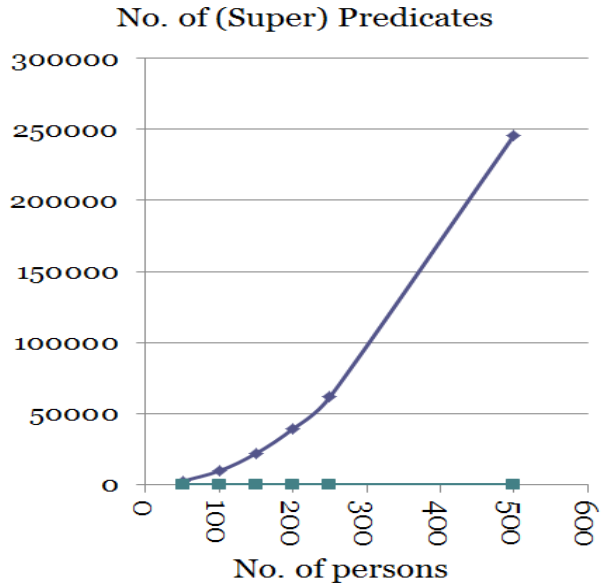

FIGURE 4.9: Graph showing comparison between the number of predicates/super predicates in ground and lifted MaxWalkSAT respectively

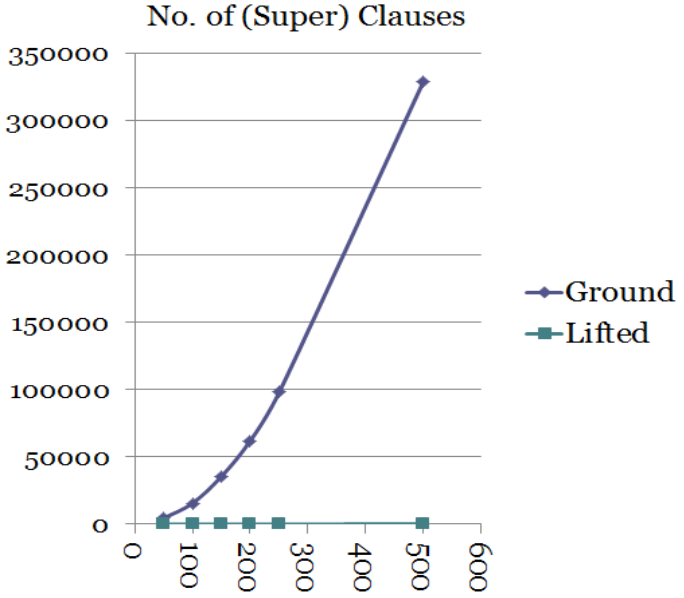

FIGURE 4.10: Graph showing comparison between the number of predicates/super clauses in ground and lifted MaxWalkSAT respectively

4.3.3.2 Timing Analysis

The running of the two algorithms have two major components, the network construction time and once the ground network is created, the time for execution for the same. As described in the previous subsection, since the size of the network for the lifted algorithm is significantly smaller than the grounded network, both the time for construction and

the time for execution show high degree of advantage. When the number of persons are less (i.e 50) the time for construction and execution in sec are almost same owing to the very small query size and hence the ground network also is very small. As the number of persons increase, the time starts to differ and becomes more and more pronounced as the number of persons (as evidence) grow.

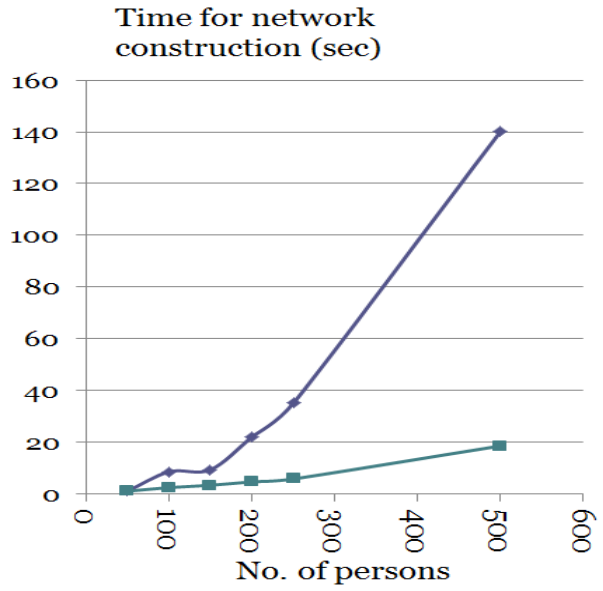

FIGURE 4.11: Graph showing comparison between network creation timings in ground and lifted MaxWalkSAT respectively with increasing number of persons in the network

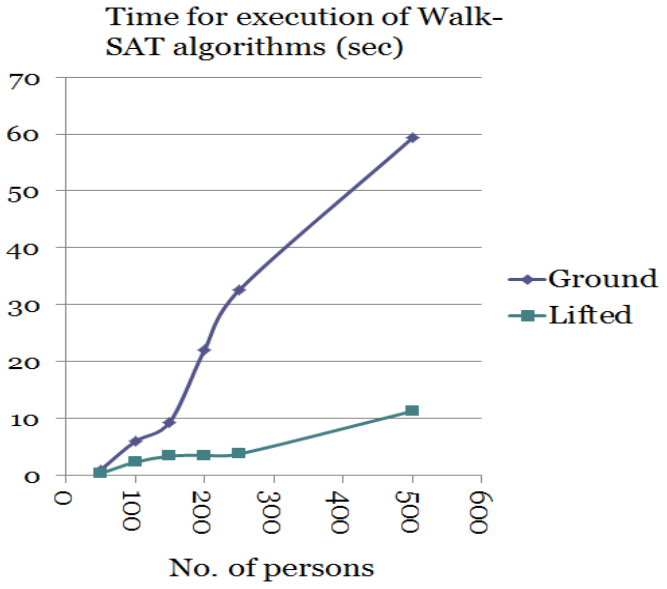

FIGURE 4.12: Graph showing comparison between execution timings of WalkSAT phase for ground and lifted MaxWalkSAT respectively with increasing number of persons in the network.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

To summarise our work, we came up with a modification of the MaxWalkSAT algorithm for MAP inference on Markov Logic networks, using the lifted technique [2]. We carried out experiments across various real life domains like Cora,UW-CSE [2] and artificially created domain Friends-Smokes network [2] to determine the benefits of the lifted MaxWalkSAT algorithm over the original algorithm and found significant improvements in terms of time as well improvement in precision. We also worked on creation of lifted GraphMinCut algorithm and formally proved (under certain assumptions) that the lifted algorithm will give the correct result as the original GraphMinCut algorithm.

5.2 Future Directions

The inference on Graph Mincut has been identified as a problem for lifted inference application and a majority of the theoretical aspects have been thought and proven, this needs to be implemeted and verified for improvement in terms of empirical running time. By applying lifted inference technique to the Max-WalkSAT algorithm and laying the theoretical background for the lifted inference on the graph min-cut problem, a lot of scope for multiple domains has been laid where symmetries within the domains exist and which if can be idetified can lead to significant savings in terms of computational requirements. So the recognition of domains where lifted inference can potentially be applied is also a major work that can be done in the future.

Bibliography

[1] Pedro Domingos and Daniel Lowd. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3 (1):1–155, 2009.

[2] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pages 1094–1099, 2008.

[3] Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993.

[4] Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2):393–405, 1990.

[5] David Poole. First-order probabilistic inference. In *IJCAI*, volume 3, pages 985–991. Citeseer, 2003.

[6] R Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1319–1325. Citeseer, 2005.

[7] Somdeb Sarkhel and Vibhav Gogate. Lifting walksat-based local search algorithms for map inference. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[8] Henry Kautz, Bart Selman, and Yueyen Jiang. A general stochastic approach to solving problems with hard and soft constraints. In *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, 1996.

[9] DM Greig, BT Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

[10] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[11] Stanley Kok, Marc Sumner, Matthew Richardson, Parag Singla, Hoifung Poon, Daniel Lowd, Jue Wang, and Pedro Domingos. The alchemy system for statistical relational {AI}. 2009.

[12] Lise Getoor and Christopher P Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[13] Kristie Seymore, Andrew McCallum, and Roni Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.