6.338/18.337	**Maximum Indpenedent Sets**	Minty's Algorithm
Andrea Lincoln		December 16, 2011
Implemented in Julia		

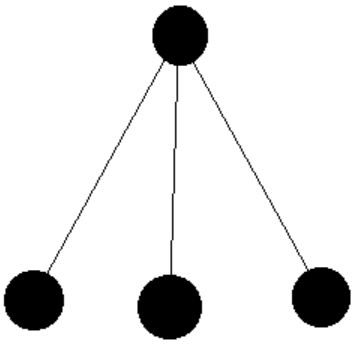
Introduction

The problem of finding maximum indpenedent sets is important to many feilds. For the purposes of my research it is important because with the maximum indepnedent set we can do scheduling wireless ad-hoc networks [3].

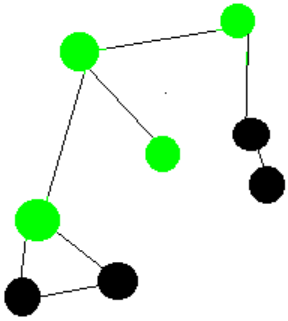
Finding a maximum indpenedent set is useful to many applications including: social networks[4], computational biology[5], additional network communications reserach [6] and circut design [7].

Minty's algorithm [1], with the correction made by Nakamura [2], is important becauase it allows us to find a maximum indepenednet set in polynomial time. However, it only finds the maximum stable set in polynomial time on claw free graphs. In general finding a maximum independent set is NP hard and the added constraint is what allows the algorithm to run in polynomial time.

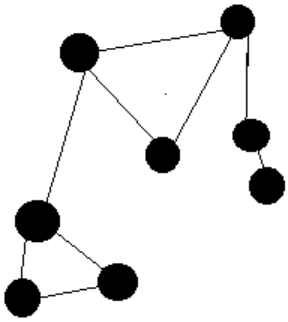
What does it mean to be a claw free graph? It means that no subgraph forms a claw. A subgraph of a graph is a graph formed by taking some subset of the origial nodes and ALL of the edges between any nodes in that subset. A claw is one node connected to three others where none of those three nodes share edges.



Here is an example of a graph with a claw (the nodes in the claw are in green):



Here is an example fo a graph without a claw:



Minty's Algorithm and Nakamura's correction

A white node: A node not in the given independent set.
A black node: A node in the given independent set.
A free node: A white node connected to only one black node.
A super free node: A white node connnected no black nodes.

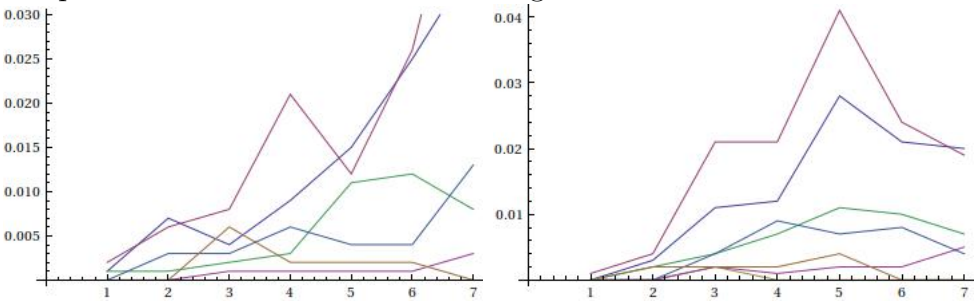
Scetch of Minty's algorithm[1][2]:
If an indepenedent set is maximum then no augmenting path would exist. If a given independent set is not a maximum then an augmenting path will exist.
Minty's algorithm works on a macro scale by starting with an independent set of one node and then finding augmenting paths. This process can be sped up by adding nodes until every given set is maximal.
For a given augmenting path we add the white nodes into the indepenedent set and remove the black nodes. We feed this new independent set of increased cardinality back in to get a new independent set. The process stops when no more augmenting paths can be found.
If the input indpenedent set is I then we look at all nodes that could be end nodes for an augmenting path. In this case this is white nodes that are connected to exactly one black node.
For each pair of possible end nodes we remove 'useless' nodes from the graph (as an example white nodes adjacent to the end nodes chosen).
Next we partition the white nodes into two groups for the two black nodes they are adjacent to (unless they are an end node in which case they are only adjacent to one black node).
Using the partitions we determine the irregular white augmenting paths (IWAPs) where an irregular black node is only connected to two black nodes through its white nodes.
Using the IWAPs and the paths formed by white nodes create a graph where black nodes are represented by edges and paths or white nodes are represented by edges.
Edmonds algorithm is called as a way to determine an augmenting path in this altered graph.

Serial vs Parallel in Theory

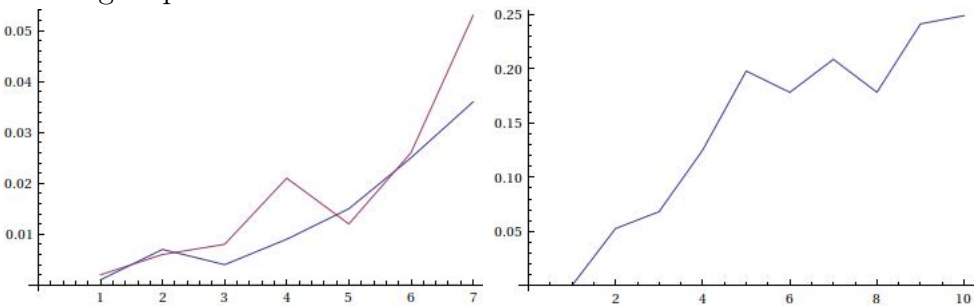
The serial version runs in $O(n^7)$[8]. Minty's algorithm has a step run $O(n^2)$ times where each step takes $O(n^5)$.
If there are $f(n) = O(n^2)$ processors then we can improve the assamptotic running time of the algorithm by $f(n)$ that is to say the new running time will be $O(n^7/f(n))$.

Serial vs Parallel in Implementation (Results)

First I needed to determine the model by which I would produce random graphs. I use the Erdos model of random graph formation and remove claws however the image on the right shows what happens if you keep a constant probability: lots of claws form and the size of maximum independent set is very small (cardinality 2). This means that on large graphs the computationaly expensive task of Minty's algorithm is only run once. In graphs we care about the maximum indepnedent set grows as the graph grows. To model this we set the probability for an edge to be $1/n$ where n is the cardinality of the graph. We see the time computation takes on this model as n grows on the left:



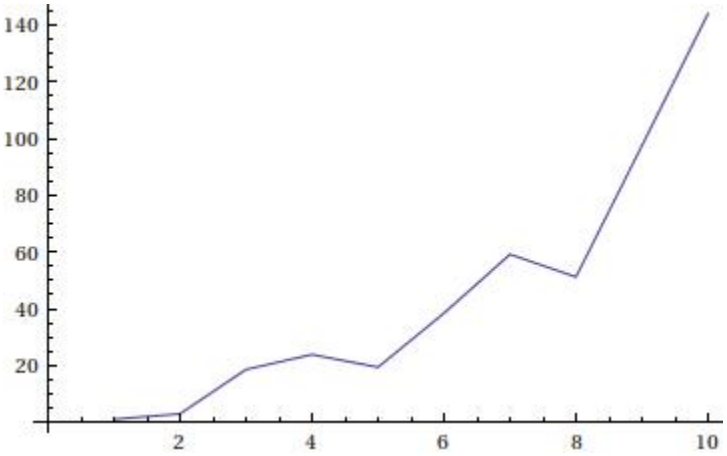
It is intersting to note that the steps that are most compuationaly expensive are the graph cleaning steps. The required communication for parallel algorithms I came up with were too great to make this a useful task to parallelize. However, parallelizing these taks would be an interesting direction for future work. Below is a graph of graph size (x) by time (y) of the cleaning step:



Next I wanted to determine the practical growth of the number of trivially parallelizable tasks. The following shows the average number of calls to the trivially parallized task that could be done in parallel (the average number of tasks that could be done at one time in parrallel).

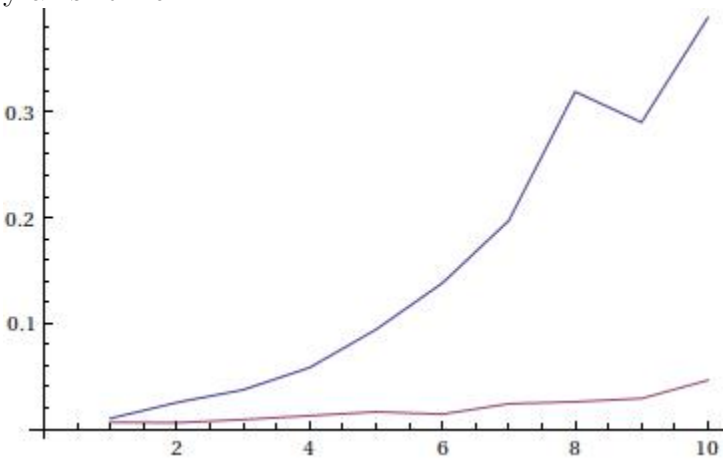
x axis: size of graph

y axis: number of parallizable tasks on average

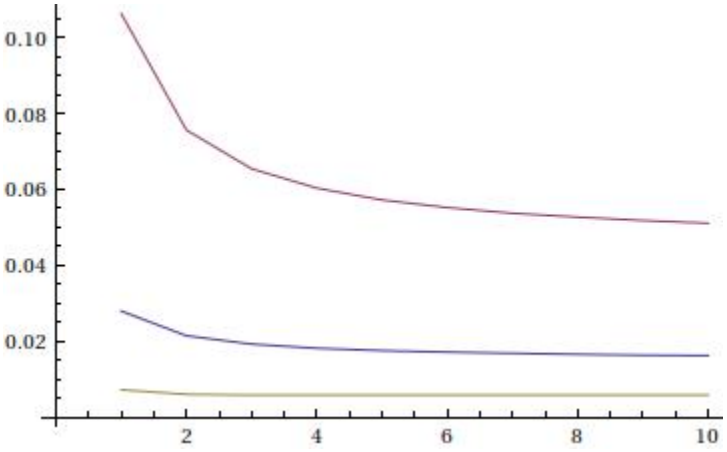


As you can see from the graph there is in fact quite some room for improvement in parallelizing the tasks created by the diffrenet choices of possible start and end nodes of the augmenting path.

If there were no communication cost and as many processors as desired then the following graph shows the best possible time when trivailly parallelizing. In other words the pink curve the theoretical ideal of how long the task will take in parallel and the blue curve is the origional time for refrence.
x axis: size of graph
y axis: time



Next I wanted to see how increased numbers of processors affected the time that computation took. The follwoing has plots for the graph sizes: 20, 50 and 100.
y axis: computation time
x axis: processors/10



Future Work

In order to use this alrogithm the graph must have no claws. So one area of future research is developing good ways of removing claws from graphs (by adding or removing constraints) such that the graph becomes claw free while keeping the size of the maximum independent set as close to the actual size of the maximum independent set.

If you are conserned that graphs have too many claws to reasonably remove them: let me assuage your concerns. As it turns out claws are strictly balanced. In Bolobas' book Random Graphs the properties of strictly balanaced sub-graphs are discussed. One particularly important property (with numbers pluged in for the specific case of claws) is that if $p = cn^{\frac{-4}{3}}$ then $\lim_{n\to\infty} E(X) = \frac{c^3}{6}$. In other words the expected value of the number of claws in the graph as n, the cardinality of the graph, heads to infinity is constant!

Parallelize the graph cleaning steps that are so expensive.

Another area for future research that is of interest if we are finding the maximum independent set of the conflict graph of a wireless ad-hoc network. In that case the network that would be running the algorithm would be related to (or perhaps even equal to) graph that was being analyzed. It would be interesting to look into how the network architecture itself could be leveraged to improve the speed of the algorithm.

Conclution

The problem of finding maximum independent set is important problem in many feilds. In general this problem is NP-hard. However, if the graph contains no claws as subgraphs then the problem is solvable in polynomial time.

When run in serial this problem is $O(n^7)$ where $O(n^2)$ of that time is due to trying every possible set of end nodes. This means that Minty's algorithm is trivially parallelizable. in fact if we have $O(n^2)$ processors Minty's algorithm only takes $O(n^5)$ time.

When run practicaly there is a speed up we get when increasing processors untill there are no more tasks that can be run in parallel. There is a speed up despite the communication cost. Due to the increase in the number of parallizable tasks as n increases the amount of speed up for a given number of processors is depenedent on the size of the problem.

Citations

For Minty's algorithm:
http://www.sciencedirect.com/science/article/pii/009589568090074X

For a corection to the origonal algorithm:
http://ci.nii.ac.jp/naid/110001183942/

[3] Scheduling for Network Coded Multicast: A Conflict Graph Formulation, IEEE, Danail Traskov, Michael Heindlmaier , Muriel M dard , Ralf Koetter, and Desmond S. Lun

[4]R. Duncan Luce and Albert D. Perry, http://www.springerlink.com/content/j7u53t8276412qp7/

[5] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Journal of Computational Biology. October 1999, 6(3-4): 281-297. doi:10.1089/106652799318274.

[6] Prihar, Z.; http://ieeexplore.ieee.org/xpls/abs$_a ll.jsp?arnumber = 4052128tag = 1$

[7] Alberto, A., Simao, A., "Minimization of incompletely specified finite state machines based on distinction graphs", Test Workshop, 2009. LATW '09. 10th Latin American, On page(s): 1 - 6, Volume: Issue: , 2-5 March 2009

[8] A new algorithm for the maximum weighted stable set problem in claw-free graphs; Gianpaolo Oriolo, Ugo Pietropaoli, Gautier Stauffer, http://dl.acm.org/citation.cfm?id=1788822

[9] Random Graphs, 2nd ed, Bela Bollobas pg78-93