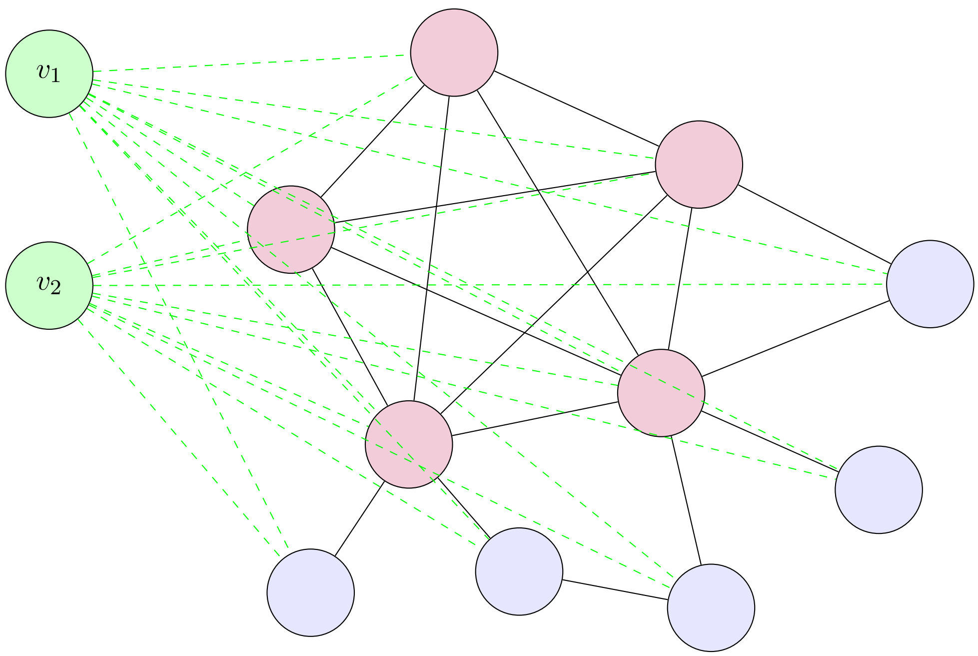
LaTeX Snippet Compilation

February 6, 2014

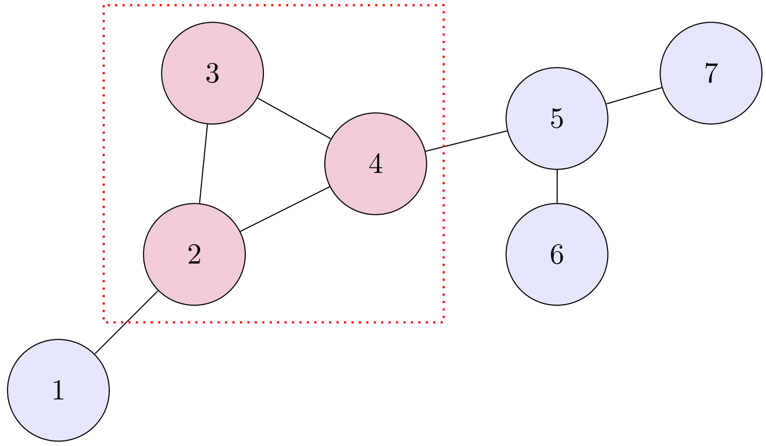
1 TikZ Graphs

Clique



Graph G with clique k_5 highlighted in red and v_1, v_2 connected to every vertex in G. A near clique of size $k + 2$ forms between the red and green colored nodes.

Subgraph



Graph G with clique k_3 highlighted.

Runtime of σ

Add nodes v_1, v_2:	$O(1)$
Connect v_1, v_2 to $v \,\forall v \in V$:	$O(V)$
Total Reduction Runtime:	$O(V)$

2 Matrices

$$M_k = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \quad M_G = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 \end{bmatrix}$$

Runtime of σ

Compute M_k:	$O(k^2)$
Compute M_G:	$O(V^2)$
Total Reduction Runtime:	$O(V^2)$

τ **Reduction:** Construct a function τ that takes the output of σ and converts it to a valid solution of the clique problem:

This is a decision problem with only a boolean output. `True` and `False` map to the same values and the reduction is trivial.

Runtime of τ

Output boolean is equivalent to solution of vertex cover:	$O(1)$
Total Reduction Runtime:	$O(1)$

2.1 Efficient Verifier:

Given a solution set S to the submatrix domination problem, test all values of A into $r(\cdot), c(\cdot)$. If every value of $r(\cdot), c(\cdot)$ matches, then the algorithm should have returned `True`, and if not, then `False`.

Runtime of Verifier

Iterate through m_1 rows and n_1 columns of A:	$O(n_1 m_1)$
Total Runtime:	$O(n_1 m_1)$

3 Party Invitations

3.1 Solution

Prove: *Vertex Cover* $\leq_p$ *Party Invitation Problem*

The Vertex Cover Problem in NP complete reduces to the party invitation problem. If we can solve the party invitation problem, we could solve the vertex cover problem as well. Create this efficient reduction σ as follows:

The party invitation takes inputs:

- a set of lists $L = \{l_1, l_2, l_3, \ldots, l_k\}$
- a set of corresponding values $M = \{m_1, m_2, m_3, \ldots, m_k\}$ of the *minimum* number of elements that can be chosen from $l_i \in L$.
- n - the max number of elements that can be chosen from all lists

The solution to the problem outputs a boolean `True` if there exists a set of elements smaller than n in cardinality that satisfies all the constraints, and `False` otherwise.

Recall that vertex cover has given $G = (V, E)$, and k, the max number of vertices that can be chosen in the cover, with the constraint $\forall e = (u, v) \in E$, e is connected to some $o \in O$. Vertex cover returns a boolean `True` if there exists some set of vertices S such that $|S| \leq k$ and `False` otherwise.

σ **Reduction:** Consider n to be the number of edges, $|E|$. For all $e \in E$, at least 1 vertex must be taken, so $M = \{1, 1, 1, \ldots, 1\}$. For all $e = (u, v) \in E$, construct a list

$L' = \{\{u_1, v_1\}, \{u_2, v_2\}, \{u_3, v_3\}, \ldots, \{u_k, v_k\}\}$ Use L' as the input set of lists for the party invitation problem. This reduction maintains all of the constraints for Vertex Cover - each edge e must have one valid vertex be chosen to cover it where a valid vertex is one that has e as an endpoint.

Runtime of σ

Calculate number of edges:	$O(E)$
Create List L':	$O(E)$
Find M:	$O(1)$
Total Reduction Runtime:	$O(E)$

τ **Reduction:** Construct a function τ that takes the output of σ and converts it to a valid solution of the vertex cover problem:

The construction of τ is trivial - the output of σ is equivalent to if a vertex cover of size k exists. Vertex Cover (VC) will return true if and only if the party invitation problem (PI) returns true because each edge $e \in G$ requires at least one vertex it is connected to to be chosen. This is maintained in PI because each set has a minimum number of elements needed to be picked to be 1, with valid elements being only vertices touching that edge. Therefore, under the constraints of PI, each edge will have at least one vertex that it is connected to be chosen. ■

Runtime of τ

Output boolean is equivalent to solution of vertex cover:	$O(1)$
Total Reduction Runtime:	$O(1)$

3.2 Efficient Verifier:

Given a solution set S of n elements to PI, a polynomial verifier is as follows:

For each element e in list $l \in L$, check every element in S and see if it exists. If every e exists in S, the algorithm should have returned `True`, and if not, then `False`.

Runtime of Verifier

Iterate through every e in list $l \in L$, with m being the total number of these elements: $m = \sum_{l \in L} \lvert l \rvert$	$O(m)$
Iterate through every element of S	$O(n)$
Total Runtime:	$O(nm)$

1: Initialize $t := 0$

Create $g(p, v, time)$ ▷ Returns the current location of Sub
for $s_i = (p_i, v_i) \in S$ **do** ▷ Exhaustively try all possible solutions
  $time := time + 1$
  $location := g(p_i, v_i, time)$
  $hit := f(location)$
  if $hit = 1$ **then** $first\_hit := \{hit, time\}$
  end if
end for
for $s_i = (p_i, v_i) \in S$ **do** ▷ Find Sub location a second time for linear interpolation
  $time := time + 1$
  $location := g(p_i, v_i, time)$
  $hit := f(location)$
  if $hit = 1$ **then** $second\_hit := \{hit, time\}$
  end if
end for
$(p, v) := Interpolate(first\_hit, second\_hit)$ ▷ Linearly interpolate between known points
return (p, v)

4 Code and Syntax Highlighting

```java
// Hello.java
import javax.swing.JApplet;
import java.awt.Graphics;

public class Hello extends JApplet {
    public void paintComponent(Graphics g) {
        g.drawString("Hello, world!", 65, 95);
    }
}
```

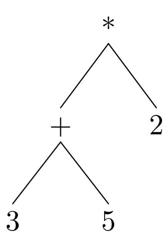
5 Tables

	$<$	$>$	/	$=$	word
TAG	OPENTAG C CLOSETAG				
C	TAG C, ϵ				word C
EQ					word $=$ word
S		ϵ			EQ S
OPENTAG	<word S>				
CLOSETAG	</ word>				

E	2	$2\hat{}3*4+5$
$\to T_1E'$	2	$2\hat{}3*4+5$
$\to T_2T_1'E'$	2	$2\hat{}3*4+5$
$\to NT_2'T_1'E'$	2	$2\hat{}3*4+5$
$\to 2T_2'T_1'E'$	$\hat{}$	$2\hat{}3*4+5$
$\to 2\hat{}T_2T_1'E'$	3	$2\hat{}3*4+5$
$\to 2\hat{}NT_2'T_1'E'$	3	$2\hat{}3*4+5$
$\to 2\hat{}3T_2'T_1'E'$	$*$	$2\hat{}3*4+5$
$\to 2\hat{}3T_1'E'$	$*$	$2\hat{}3*4+5$
$\to 2\hat{}3*T_1E'$	4	$2\hat{}3*4+5$
$\to 2\hat{}3*T_2T_1'E'$	4	$2\hat{}3*4+5$
$\to 2\hat{}3*NT_2'T_1'E'$	4	$2\hat{}3*4+5$
$\to 2\hat{}3*4T_2'T_1'E'$	$+$	$2\hat{}3*4+5$
$\to 2\hat{}3*4T_1'E'$	$+$	$2\hat{}3*4+5$
$\to 2\hat{}3*4E'$	$+$	$2\hat{}3*4+5$
$\to 2\hat{}3*4+E$	5	$2\hat{}3*4+5$
$\to 2\hat{}3*4+T_1E'$	5	$2\hat{}3*4+5$
$\to 2\hat{}3*4+T_2T_1'E'$	5	$2\hat{}3*4+5$
$\to 2\hat{}3*4+NT_2'T_1'E'$	5	$2\hat{}3*4+5$
$\to 2\hat{}3*4+5T_2'T_1'E'$	\$	$2\hat{}3*4+5$
$\to 2\hat{}3*4+5T_1'E'$	\$	$2\hat{}3*4+5$
$\to 2\hat{}3*4+5E'$	\$	$2\hat{}3*4+5$
$\to 2\hat{}3*4+5$	\$	$2\hat{}3*4+5$

6 Trees

$((3+5)*2)$



$(3+(5*2))$