Computing Tutte Polynomials

Michael Monagan

Department of Mathematics,
Simon Fraser University
Outline

- Reliability polynomials and Tutte polynomials.
- Examples using Maple’s GraphTheory package.
- Haggard, Pearce and Royle’s TOMS paper.
  An example: the truncated icosahedron.
- Edge selection heuristics.
- Maple implementation and some benchmarks.
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$.

$$R_p(\begin{array}{cc} & \bullet \\ \bullet & \end{array} ) =$$
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$. 

$$R_p\left( \begin{array}{c} \bullet \\ p \\ \bullet \end{array} \right) = 1 - p$$
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$.

$$R_p(\bullet \overset{p}{\longrightarrow} \bullet) = 1 - p$$

$$R_p(\bullet \overset{\bullet}{\longrightarrow} \bullet) =$$
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$.

$$R_p(\quad \quad \quad ) = 1 - p \quad \quad \quad R_p(\quad \quad \quad ) = (1 - p)^2$$
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$.

$$R_p(\begin{array}{c}
\bullet \\
p \\
\bullet
\end{array}) = 1 - p$$

$$R_p(\begin{array}{c}
\bullet \\
\bullet
\end{array}) = (1 - p)^2$$

$$R_p(\begin{array}{c}
\bullet \\
p \\
\bullet
\end{array}) =$$
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$.

- $R_p(\bullet\bullet\bullet) = 1 - p$
- $R_p(\bullet\bullet) = (1 - p)^2$
- $R_p(\bullet\bullet\bullet) = 1 - p^2$
Reliability Polynomials

Let $G$ be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network $G$ remains connected when each edge fails with probability $p$.

$$R_p(\quad p \quad) = 1 - p \quad R_p(\quad g \quad) = (1 - p)^2$$

$$R_p(\quad p \quad) = 1 - p^2$$

$$R_p(\quad p \quad) = p R_p(\quad g \quad) + (1 - p) R_p(\quad g \quad)$$
The edge deletion contraction algorithm.

\[ R_p(G) = p R_p(G - e) + (1 - p) R_p(G/e) \]

\[ R_p(\bullet) = 1 \]

\[ R_p( ) = R_p( ) \]
The edge deletion contraction algorithm.

\[ R_p(G) = p R_p(G - e) + (1 - p) R_p(G/e) \]

\[ R_p(\bullet) = 1 \]

\[ R_p(\text{triangle}) = R_p(\text{triangle}) \]

\[ R_p(\text{triangle} - e) = p \times 0 + (1 - p) R_p(G/e) \]
Reliability Polynomials cont.

\[ R_p(\text{e}) = R_p(\text{triangle}) \times R_p(\text{triangle})^2 \]

[1971 Hopcroft and Tarjan]
Computing biconnected components is \( O(n + m) \).
Tutte Polynomials

For a connected graph $G$, the Tutte polynomial $T(G, x, y)$ is a bivariate polynomial defined by

1. $T(\bullet) = 1$
2. $e$ is a cutedge $T(G) = x \ T(G/e)$
3. $e$ is a loop $T(G) = y \ T(G - e)$
4. otherwise $T(G) = T(G - e) + T(G/e)$
Tutte Polynomials

For a connected graph $G$, the Tutte polynomial $T(G, x, y)$ is a bivariate polynomial defined by

1. $T(\bullet) = 1$
2. $e$ is a cut edge $T(G) = x \cdot T(G/e)$
3. $e$ is a loop $T(G) = y \cdot T(G - e)$
4. otherwise $T(G) = T(G - e) + T(G/e)$

Complexity:

$C(n + m) \leq C(n + m - 1) + C(n - 1 + m - 1)$
For a connected graph $G$, the Tutte polynomial $T(G, x, y)$ is a bivariate polynomial defined by

1. $T(\bullet) = 1$
2. $e$ is a cut edge $T(G) = x \cdot T(G/e)$
3. $e$ is a loop $T(G) = y \cdot T(G - e)$
4. otherwise $T(G) = T(G - e) + T(G/e)$

Complexity:

$C(n + m) \leq C(n + m - 1) + C(n - 1 + m - 1) \in O(1.618^{n+m})$

Can we do better?
Tutte Polynomials cont.

For $G$ connected with $n$ vertices and $m$ edges.

$$R_p(G) = (1 - p)^{n-1} p^{m-n+1} T(G, 1, p^{-1})$$

Example: $P(A) = \lambda(\lambda - 1)(\lambda - 2)$

Thus $G$ is not 2-colorable, $G$ is 3-colorable and can be colored in $P(G, 3) = 6$ ways.
Tutte Polynomials cont.

For $G$ connected with $n$ vertices and $m$ edges.

$$R_p(G) = (1 - p)^{n-1} p^{m-n+1} \ T(G, 1, p^{-1})$$

$$P_\lambda(G) = (-1)^{n-1} \lambda \ T(G, 1 - \lambda, 0)$$

Example:
Tutte Polynomials cont.

For $G$ connected with $n$ vertices and $m$ edges.

\[ R_p(G) = (1 - p)^{n-1} p^{m-n+1} T(G, 1, p^{-1}) \]

\[ P_\lambda(G) = (-1)^{n-1} \lambda T(G, 1 - \lambda, 0) \]

Example:

\[ P( ) = \lambda(\lambda - 1)(\lambda - 2) \]

Thus $G$ is not 2-colorable, $G$ is 3-colorable and can be colored in $P(G, 3) = 6$ ways.
Deom Reliability.mw and Demo.mws


David Pearce’s website: 

http://homepages.ecs.vuw.ac.nz/~djp/tutte

Truncated icosahedron demo: T1cos.mw
Edge selection heuristics

\[ T(G - e) + T(G/e) \]

[HPR, 2010] minimum degree heuristic:

And store Tutte polynomials for previously computed graphs and hash on a canonical representation of the graph.
Edge selection heuristics

[HPR 2010] VORDER-pull heuristic:

1 --- 2
\[ \rightarrow \]

1 --- 4

1 --- 5

[HMB 2011] VORDER-push heuristic:

1 --- 5

1 --- 2

1 --- 3

1 --- 4
Edge selection heuristics

[HPR 2010] VORDER-pull heuristic:

[MBM 2011] VORDER-push heuristic:
\[ \text{tp} := \text{proc}(G,x,y) \text{ local n,i,j,Gcon,Gdel,T;} \]
\[ \qquad \# G = [[2,3],[1,3],[1,2,4,4],[3,3]] \]
\[ \text{option remember; \# O(m+n)} \]
\[ \text{n := nops(G);} \]
\[ \text{if n=0 then return 1; fi;} \]
\[ \text{for i to n do} \]
\[ \qquad \text{if G[i] = [] then \ldots \# singleton} \]
\[ \qquad \text{elif member(i,G[i]) then \ldots \# loop} \]
\[ \qquad \text{fi;} \]
\[ \text{od;} \]
\[ \text{i := 1; j := G[1][1]; \# Pick first edge (i,j) with i<j} \]
\[ \text{Gdel := subsop( i=G[i][2..-1], j=G[j][2..-1], G );} \]
\[ \text{Gcon := contract(Gdel,i,j); \# contract i to j} \]
\[ \text{if path(i,j,Gdel) then} \]
\[ \qquad \text{T := tp(Gcon,x,y) + tp(Gdel,x,y); \# O(n+m)} \]
\[ \text{else} \]
\[ \qquad \text{T := expand( x*tp(Gcon,x,y) );} \]
\[ \text{fi;} \]
\[ \text{end:} \]
## Benchmarks: Random cubic graphs

<table>
<thead>
<tr>
<th>Random</th>
<th>Minimum degree</th>
<th>VORDER pull</th>
<th>VORDER push</th>
</tr>
</thead>
<tbody>
<tr>
<td>n m</td>
<td>ave med</td>
<td>ave med</td>
<td>ave med</td>
</tr>
<tr>
<td>16 24</td>
<td>0.47 0.50</td>
<td>0.70 0.63</td>
<td>0.22 0.15</td>
</tr>
<tr>
<td>20 30</td>
<td>5.27 4.73</td>
<td>7.31 7.91</td>
<td>2.06 1.75</td>
</tr>
<tr>
<td>24 36</td>
<td>85.58 72.49</td>
<td>136.33 94.60</td>
<td>48.14 58.65</td>
</tr>
</tbody>
</table>
## Benchmarks: Random cubic graphs

<table>
<thead>
<tr>
<th></th>
<th>Random Minimum degree</th>
<th>VORDER pull</th>
<th>VORDER push</th>
</tr>
</thead>
<tbody>
<tr>
<td>n, m</td>
<td>ave, med</td>
<td>ave, med</td>
<td>ave, med</td>
</tr>
<tr>
<td>16, 24</td>
<td>0.47, 0.50</td>
<td>0.70, 0.63</td>
<td>0.22, 0.15</td>
</tr>
<tr>
<td>20, 30</td>
<td>5.27, 4.73</td>
<td>7.31, 7.91</td>
<td>2.06, 1.75</td>
</tr>
<tr>
<td>24, 36</td>
<td>85.58, 72.49</td>
<td>136.33, 94.60</td>
<td>48.14, 58.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Sorted Minimum degree</th>
<th>VORDER pull</th>
<th>VORDER push</th>
</tr>
</thead>
<tbody>
<tr>
<td>n, m</td>
<td>ave, med</td>
<td>ave, med</td>
<td>ave, med</td>
</tr>
<tr>
<td>16, 25</td>
<td>0.23, 0.20</td>
<td>0.41, 0.36</td>
<td>0.03, 0.03</td>
</tr>
<tr>
<td>20, 30</td>
<td>2.32, 2.06</td>
<td>3.94, 4.16</td>
<td>0.08, 0.07</td>
</tr>
<tr>
<td>24, 36</td>
<td>31.88, 31.78</td>
<td>63.68, 52.76</td>
<td>0.51, 0.70</td>
</tr>
<tr>
<td>30, 45</td>
<td></td>
<td>2.63, 2.42</td>
<td></td>
</tr>
<tr>
<td>36, 54</td>
<td>$O(1.287^{(n+m)})$</td>
<td></td>
<td>31.14, 6.80</td>
</tr>
<tr>
<td>42, 63</td>
<td></td>
<td>159.61, 57.96</td>
<td></td>
</tr>
<tr>
<td>46, 69</td>
<td>$O(1.147^{(n+m)})$</td>
<td>463.08, 390.98</td>
<td></td>
</tr>
</tbody>
</table>
The short arc vertex ordering (SHARC).

Show VOrder.mws
Benchmarks: The $P(k, 3)$ - Petersen graphs

VORDER pull (with vertex ordering)

| k | $|V|$ | $|E|$ | time  | #calls  | #identical | #isom |
|---|-----|-----|-------|---------|------------|-------|
| 8 | 16  | 24  | 1.10  | 28641   | 10419      | 0     |
| 9 | 18  | 27  | 1.24  | 30235   | 9818       | 3     |
| 10| 20  | 30  | 4.11  | 90772   | 31049      | 22    |
| 11| 22  | 33  | 24.51 | 434402  | 149286     | 244   |
| 12| 24  | 36  | 32.07 | 471530  | 152284     | 978   |
| 13| 26  | 39  | 162.38| 1668636 | 552034     | 7072  |

VORDER push (with vertex ordering)

| k | $|V|$ | $|E|$ | time  | #calls  | #identical | #isom |
|---|-----|-----|-------|---------|------------|-------|
| 8 | 16  | 24  | 0.11  | 2980    | 1181       | 0     |
| 10| 20  | 30  | 0.23  | 4739    | 1889       | 7     |
| 12| 24  | 36  | 1.26  | 18644   | 7454       | 31    |
| 14| 28  | 42  | 4.50  | 41706   | 16691      | 184   |
| 16| 32  | 48  | 11.47 | 66086   | 25975      | 687   |
| 18| 36  | 54  | 22.48 | 93584   | 36495      | 1294  |
| 20| 40  | 60  | 37.58 | 122869  | 47766      | 2002  |
| 22| 44  | 66  | 53.46 | 151954  | 58873      | 2746  |
| 24| 48  | 72  | 81.56 | 181918  | 70346      | 3487  |
| 26| 52  | 78  | 114.26| 211681  | 81767      | 4240  |
| 28| 56  | 84  | 156.69| 241364  | 93134      | 4995  |
| 30| 60  | 90  | 210.17| 271434  | 104649     | 5740  |
Benchmarks: Large girth is harder: $P(14, k)$

<table>
<thead>
<tr>
<th>$k$</th>
<th>girth</th>
<th>time(s)</th>
<th>#calls</th>
<th>deg</th>
<th>time(s)</th>
<th>#calls</th>
<th>deg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>6.12</td>
<td>54040</td>
<td>6.48</td>
<td>0.16</td>
<td>693</td>
<td>2.10</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>209.33</td>
<td>1362412</td>
<td>5.19</td>
<td>0.65</td>
<td>4727</td>
<td>2.30</td>
</tr>
<tr>
<td>35</td>
<td>6</td>
<td>806.92</td>
<td>4035615</td>
<td>4.32</td>
<td>3.82</td>
<td>40142</td>
<td>2.47</td>
</tr>
<tr>
<td>46</td>
<td>7</td>
<td>2273.75</td>
<td>8430139</td>
<td>4.61</td>
<td>7.71</td>
<td>88579</td>
<td>2.49</td>
</tr>
<tr>
<td>57</td>
<td>6</td>
<td>1218.51</td>
<td>6208087</td>
<td>4.49</td>
<td>5.62</td>
<td>71717</td>
<td>2.50</td>
</tr>
<tr>
<td>68</td>
<td>6</td>
<td>979.73</td>
<td>5524084</td>
<td>4.44</td>
<td>6.43</td>
<td>71054</td>
<td>2.47</td>
</tr>
</tbody>
</table>
Isomorphism doesn’t help. BFS doesn’t work.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>SHARC + ISOM</th>
<th>SHARC - Isom</th>
<th>BFS Order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave</td>
<td>med</td>
<td>ave</td>
</tr>
<tr>
<td>22</td>
<td>33</td>
<td>0.32</td>
<td>0.26</td>
<td>0.18</td>
</tr>
<tr>
<td>26</td>
<td>39</td>
<td>0.78</td>
<td>0.42</td>
<td>0.43</td>
</tr>
<tr>
<td>30</td>
<td>45</td>
<td>2.63</td>
<td>2.42</td>
<td>1.35</td>
</tr>
<tr>
<td>34</td>
<td>51</td>
<td>8.59</td>
<td>4.93</td>
<td>3.84</td>
</tr>
<tr>
<td>38</td>
<td>57</td>
<td>76.09</td>
<td>7.86</td>
<td>9.07</td>
</tr>
<tr>
<td>42</td>
<td>63</td>
<td>159.61</td>
<td>57.96</td>
<td>56.00</td>
</tr>
<tr>
<td>46</td>
<td>69</td>
<td>463.08</td>
<td>390.98</td>
<td>120.76</td>
</tr>
</tbody>
</table>
Conclusion

- VORDER-push + SHARC ordering is MUCH faster for sparse graphs!
- Found by trying all possibilities and some good luck.
- It finds polynomial time constructions for some graphs.
- Graphs with large girth appear to be more difficult.
- An explicit graph isomorphism test is unnecessary.
- Is there a better heuristic or ordering?