Computing Tutte Polynomials

Michael Monagan

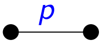
Department of Mathematics,
Simon Fraser University

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.

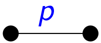
Reliability Polynomials

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.

$R_p($  $) =$

Reliability Polynomials

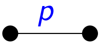
Let G be an undirected graph. The reliability polynomial $R_p(G)$ is the probability that the network G remains connected when each edge fails with probability p.

$R_p($ 

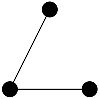
$) = 1 - p$

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.

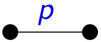
$R_p($ 

$) = 1 - p$

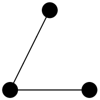
$R_p($  $) =$

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.

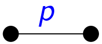
$R_p($ 

$) = 1 - p$

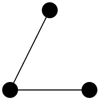
$R_p($  $) = (1 - p)^2$

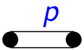
Reliability Polynomials

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.

$R_p($ 

$) = 1 - p$

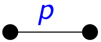
$R_p($  $) = (1 - p)^2$

$R_p($ 

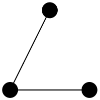
$) =$

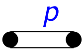
Reliability Polynomials

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.

$R_p($ 

$) = 1 - p$

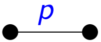
$R_p($  $) = (1 - p)^2$

$R_p($ 

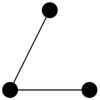
$) = 1 - p^2$

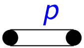
Reliability Polynomials

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.

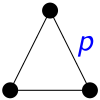
$R_p($ 

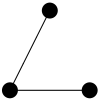
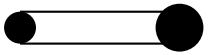
$) = 1 - p$

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

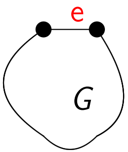
$R_p($ 

$) = 1 - p^2$

$R_p($ 

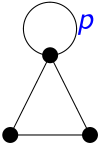
$) = p\, R_p($  $) + (1 - p)\, R_p($  $)$

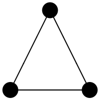
The edge deletion contraction algorithm.

$R_p($ 

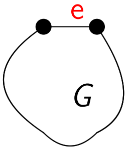
$) = p\, R_p(G - e) + (1 - p)\, R_p(G/e)$

$R_p(\bullet) = 1$

$R_p($ 

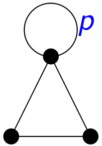
$) = R_p($  $)$

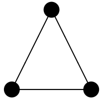
The edge deletion contraction algorithm.

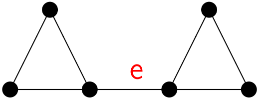
$R_p($ 

$) = p\, R_p(G - e) + (1 - p)\, R_p(G/e)$

$R_p(\bullet) = 1$

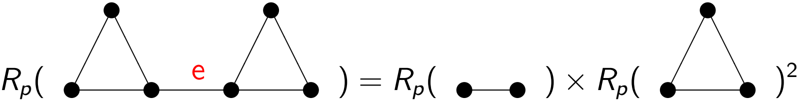
$R_p($ 

$) = R_p($  $)$

$R_p($ 

$) = p \times 0 + (1 - p)\, R_p(G/e)$

Reliability Polynomials cont.



[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 cutedge $T(G) = x\, T(G/e)$
3. e is a loop $T(G) = y\, T(G - e)$
4. otherwise $T(G) = T(G - e) + T(G/e)$

Complexity:

$$C(n + m) \leq C(n + m - 1) + C(n - 1 + m - 1)$$

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 $T(G) = T(G - e) + T(G/e)$

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.

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

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:

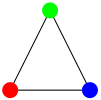
Tutte Polynomials cont.

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

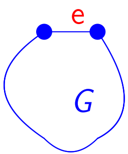
G. Haggard, D.J. Pearce, and G. Royle.
Computing Tutte Polynomials. *TOMS* **37**:3, 2011.

David Pearce's website:

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

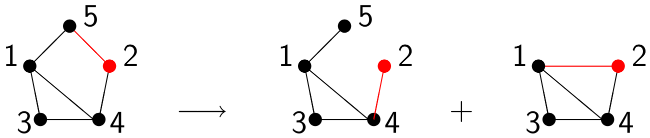
Truncated icosahedron demo: TIcos.mw

Edge selection heuristics



$$T(G) = T(G - e) + T(G/e)$$

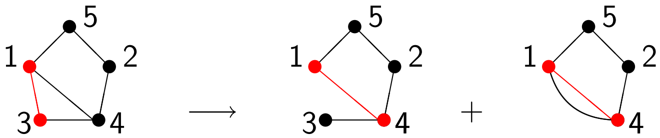
[HPR, 2010] minimum degree heuristic:



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

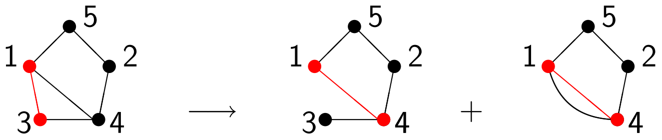
Edge selection heuristics

[HPR 2010] VORDER-pull heuristic:

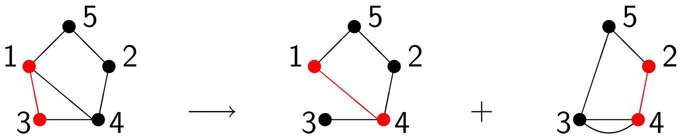


Edge selection heuristics

[HPR 2010] VORDER-pull heuristic:



[MBM 2011] VORDER-push heuristic:



```
tp := proc(G,x,y) local n,i,j,Gcon,Gdel,T;

   # G = [[2,3],[1,3],[1,2,4,4],[3,3]]

option remember; # O(m+n)

   n := nops(G);
   if n=0 then return 1; fi;

   for i to n do
       if G[i] = [] then ...   # singleton
       elif member(i,G[i]) then ... # loop
       fi;
   od;

   i := 1; j := G[1][1];  # Pick first edge (i,j) with i<j

   Gdel := subsop( i=G[i][2..-1], j=G[j][2..-1], G );
   Gcon := contract(Gdel,i,j); # contract i to j

   if path(i,j,Gdel) then
       T := tp(Gcon,x,y) + tp(Gdel,x,y);   # O(n+m)
   else
       T := expand( x*tp(Gcon,x,y) );
   fi;

end:
```

Benchmarks: Random cubic graphs

Random		Minimum degree		VORDER pull		VORDER push	
n	m	ave	med	ave	med	ave	med
16	24	0.47	0.50	0.70	0.63	0.22	0.15
20	30	5.27	4.73	7.31	7.91	2.06	1.75
24	36	85.58	72.49	136.33	94.60	48.14	58.65

Benchmarks: Random cubic graphs

Random		Minimum degree		VORDER pull		VORDER push	
n	m	ave	med	ave	med	ave	med
16	24	0.47	0.50	0.70	0.63	0.22	0.15
20	30	5.27	4.73	7.31	7.91	2.06	1.75
24	36	85.58	72.49	136.33	94.60	48.14	58.65

Sorted		Minimum degree		VORDER pull		VORDER push	
n	m	ave	med	ave	med	ave	med
16	25	0.23	0.20	0.41	0.36	0.03	0.03
20	30	2.32	2.06	3.94	4.16	0.08	0.07
24	36	31.88	31.78	63.68	52.76	0.51	0.70
30	45					2.63	2.42
36	54	O(1.287^(n+m))				31.14	6.80
42	63					159.61	57.96
46	69			O(1.147^(n+m))		463.08	390.98

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: $P(14, k)$

		VORDER pull			VORDER push		
k	girth	time(s)	#calls	deg	time(s)	#calls	deg
1	4	6.12	54040	6.48	0.16	693	2.10
24	5	209.33	1362412	5.19	0.65	4727	2.30
35	6	806.92	4035615	4.32	3.82	40142	2.47
46	7	2273.75	8430139	4.61	7.71	88579	2.49
57	6	1218.51	6208087	4.49	5.62	71717	2.50
68	6	979.73	5524084	4.44	6.43	71054	2.47

Isomorphism doesn't help. BFS doesn't work.

		SHARC + ISOM		SHARC - Isom		BFS Order	
n	m	ave	med	ave	med	ave	med
22	33	0.32	0.26	0.18	0.17	0.87	0.65
26	39	0.78	0.42	0.43	0.33	2.67	0.90
30	45	2.63	2.42	1.35	1.30	9.82	3.94
34	51	8.59	4.93	3.84	1.80	18.71	11.92
38	57	76.09	7.86	9.07	4.63	357.97	185.50
42	63	159.61	57.96	56.00	23.24		
46	69	463.08	390.98	120.76	70.49		

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 unecessary.
- Is there a better heuristic or ordering?