Charlie Cabot
CyberPoint International
August 2012

This report contains data on the runtimes and memory usage of the core functions of the PGM library. These statistics may be useful in deciding whether to use this library for a probabilistic computing task. They also may reveal room for improvements in efficiency. The library is currently found in the `bayesian` git repository, in the directory `/v3_bayesian/PGMlibrary2.0`.

FORWARD SAMPLING

These tests measure the runtime of the `randomsample` method in the `DiscreteBayesianNetwork` class.

Test 1: Runtime as a function of number of vertices.

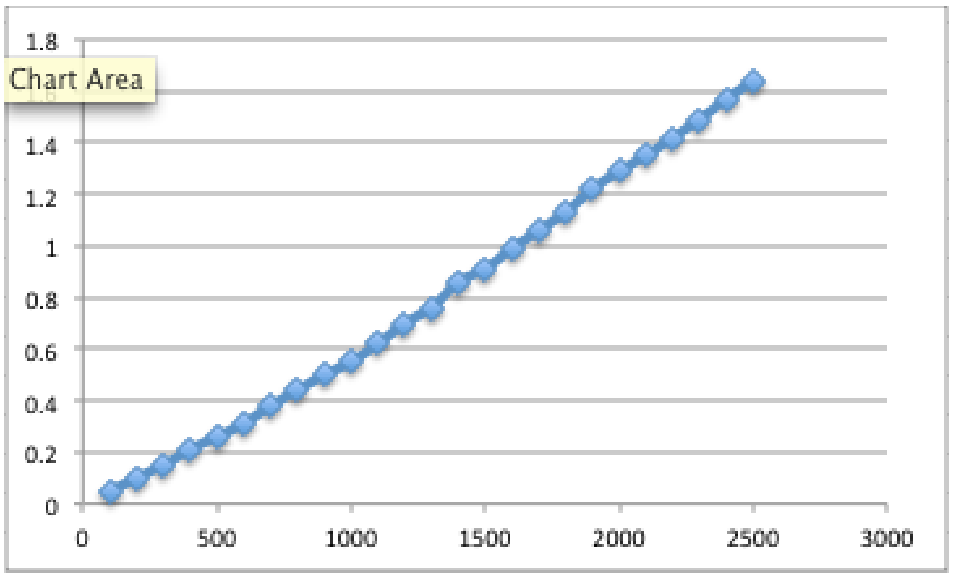

Figure 1: This graph shows the runtime, in seconds, of the `randomsample` method running on graphs with varying numbers of vertices. The following were held constant: 100 samples taken. 3 parents per node (henceforth 3 is the“indegree”). 3 outcomes per node.

Test 2: Memory usage as a function of indegree.

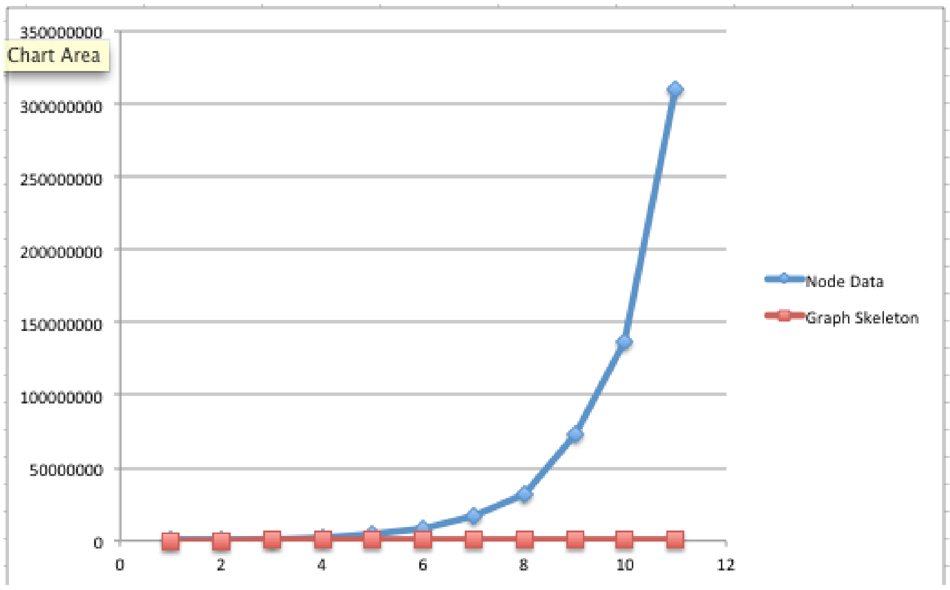

Figure 2: This graph shows the size, in bytes, of the node data and graph skeleton as a function of indegree. 300 vertices. 2 outcomes per vertex.

Test 3: Memory usage as a function of number of outcomes per vertex.

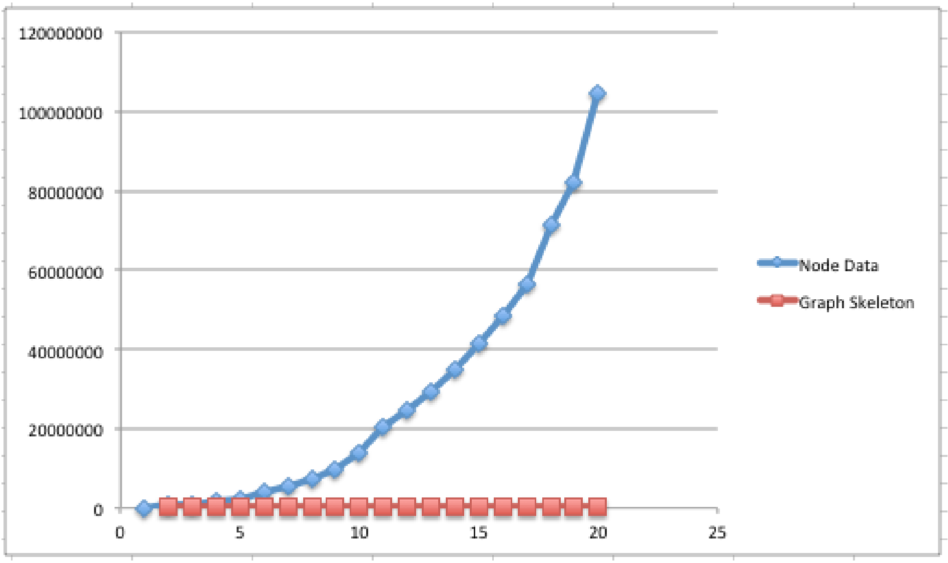

Figure 3: This graph shows the size, in bytes, of the node data and graph skeleton as a function of the number of outcomes per vertex. 300 vertices. Indegree 2.

QUERYING

These tests measure the runtime of the `specificquery` method in the `TableCPDFactorization` class.

Test 4: Runtime as a function of number of vertices (querying root node).

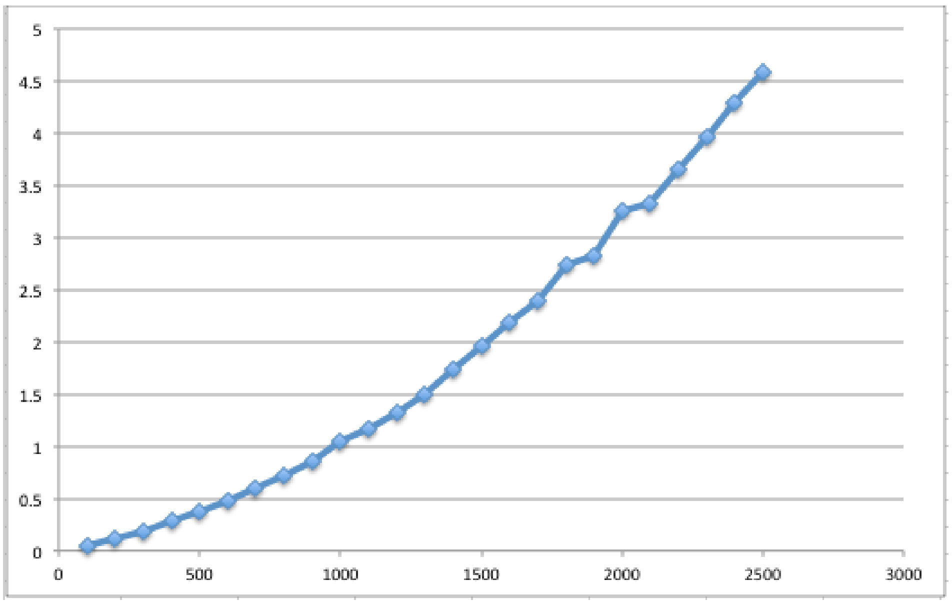

Figure 4: This graph shows the runtime, in seconds, of the `specificquery` method running on graphs with varying numbers of vertices. The following were held constant: Indegree 3. 3 outcomes per node. No evidence. The node queried was at the root of the tree.

Test 5: Runtime as a function of number of vertices (querying leaf node).

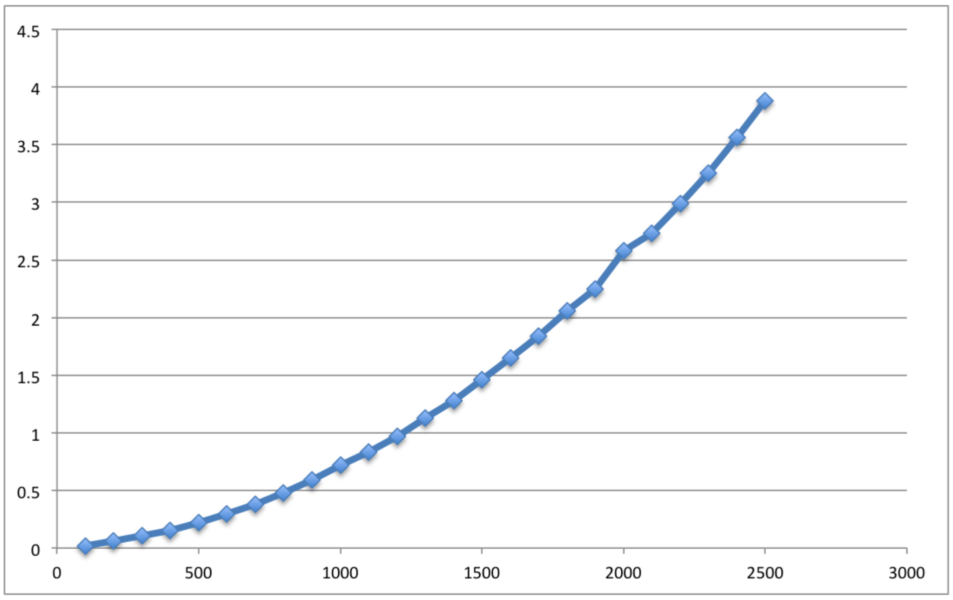

Figure 5: This graph shows the runtime, in seconds, of the `specificquery` method running on graphs with varying numbers of vertices. The following were held constant: Indegree 3. 3 outcomes per node. No evidence. The node queried was at the leaf of the tree.

Test 6: Runtime as a function of indegree.

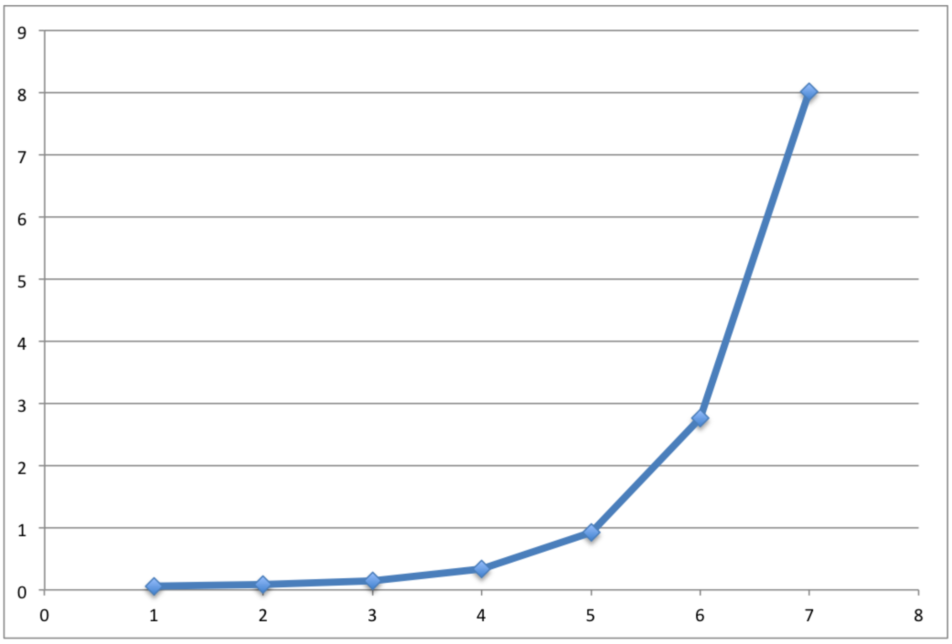

Figure 6: This graph shows the runtime, in seconds, of the `specificquery` method running on graphs with varying indegree. The following were held constant: 300 vertices. 3 outcomes per vertex. No evidence. The node queried was at the root of the tree.

Test 7: Runtime as a function of outcomes per vertex.

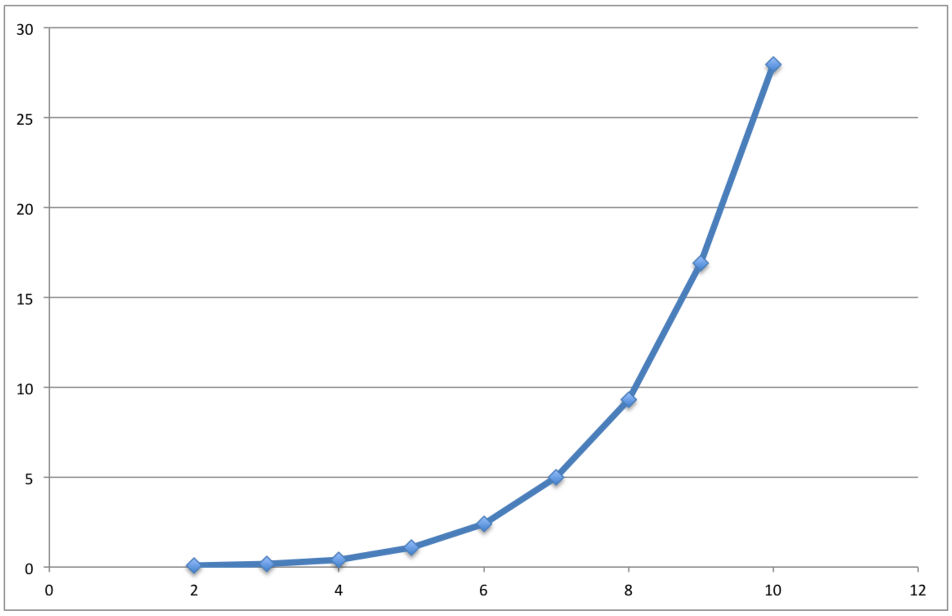

Figure 7: This graph shows the runtime, in seconds, of the `specificquery` method running on graphs with varying indegree. The following were held constant: 300 vertices. Indegree 3. No evidence. The node queried was at the root of the tree.

LEARNING

These tests measure the runtime of the `discrete_mle_estimateparams` and `discrete_constraint_estim` method in the `DiscreteBayesianNetwork` class.

Test 8: Parameter learning runtime as a function of number of vertices.

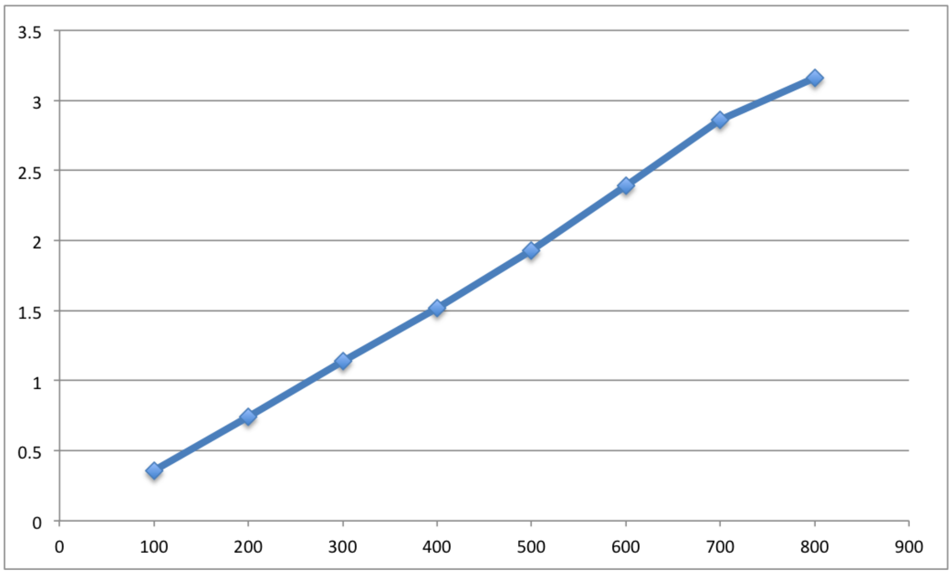

Figure 8: This graph shows the runtime, in seconds, of the `discrete_mle_estimateparams` method running on graphs with varying numbers of vertices. The following were held constant: 2 outcomes per vertex. Indegree 2. 1500 data points.

Test 9: Parameter learning runtime as a function of indegree.

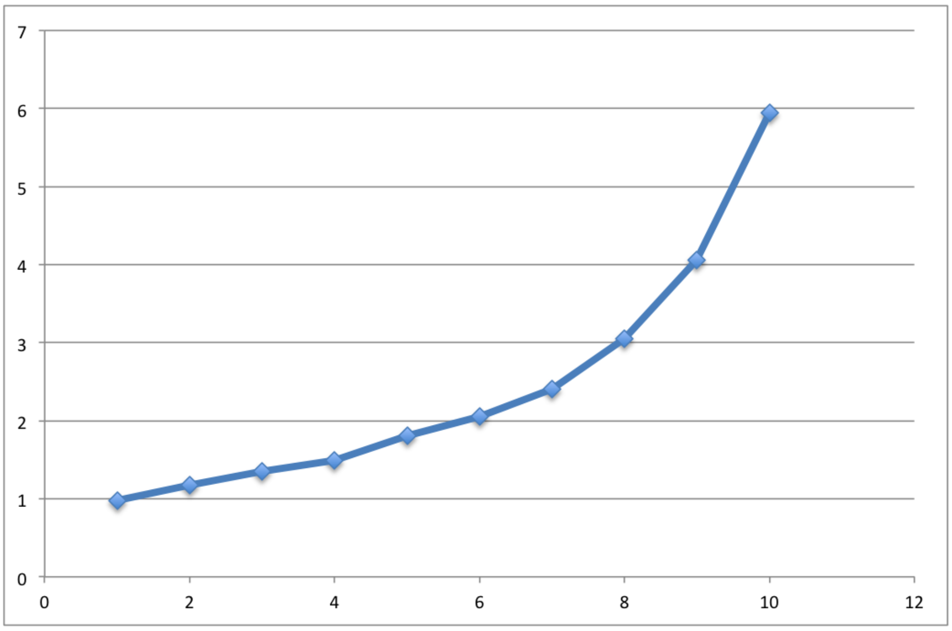

Figure 9: This graph shows the runtime, in seconds, of the `discrete_mle_estimateparams` method running on graphs with varying indegree. The following were held constant: 2 outcomes per vertex. 300 vertices. 1500 data points.

Test 10: Parameter learning runtime as a function of outcomes per vertex.

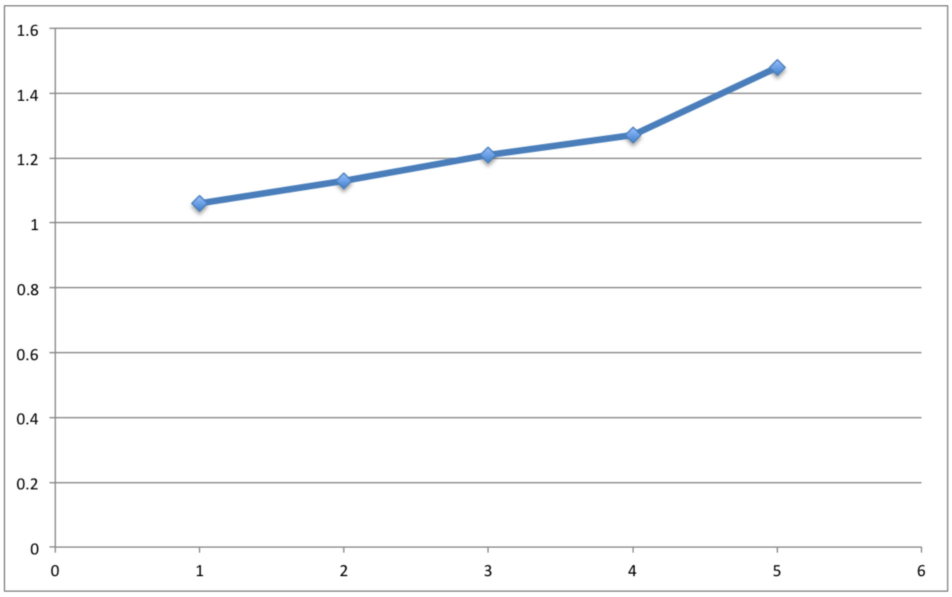

Figure 10: This graph shows the runtime, in seconds, of the `discrete_mle_estimateparams` method running on graphs with varying outcomes per vertex. The following were held constant: Indegree 2. 300 vertices. 1500 data points.

Test 11: Parameter learning runtime as a function of size of data set.

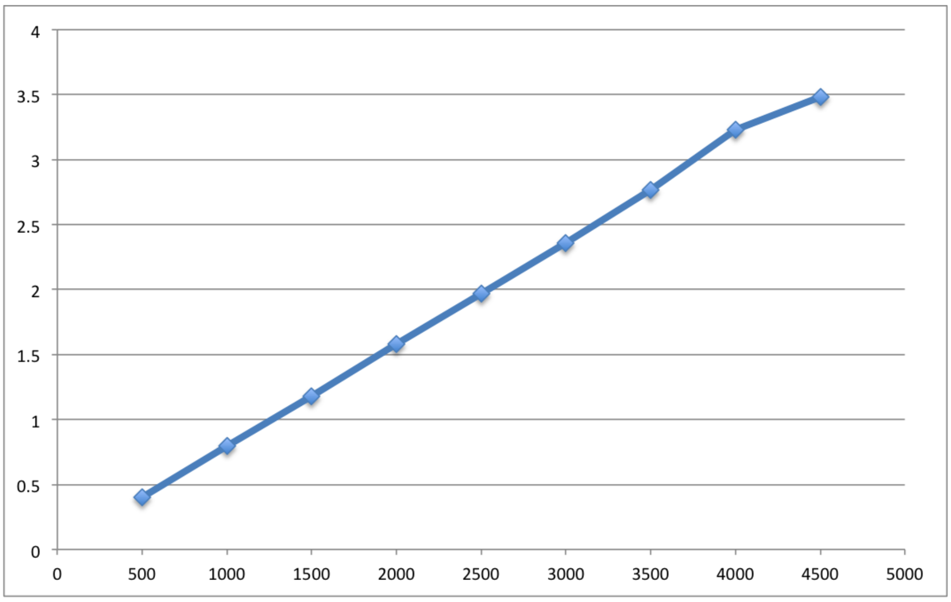

Figure 11: This graph shows the runtime, in seconds, of the `discrete_mle_estimateparams` method running on graphs with a varying data set size, measured in number of data points. The following were held constant: Indegree 2. 2 outcomes per vertex. 300 vertices.

Test 12: Structure learning runtime as a function of number of vertices.

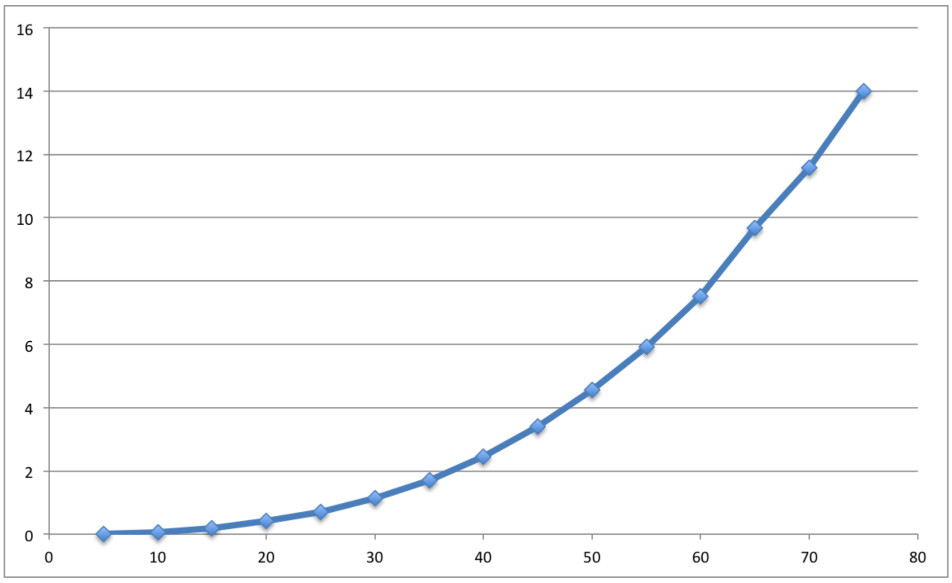

Figure 12: This graph shows the runtime, in seconds, of the `discrete_constraint_estimatestruct` method running on graphs with a varying number of vertices. The following were held constant: Indegree 2. 2 outcomes per vertex. 150 data points. Maximum 1 witness (this means each pair of vertices was checked for conditional independence given sets of nodes of size ≤ 1).

Test 13: Structure learning runtime as a function of indegree.

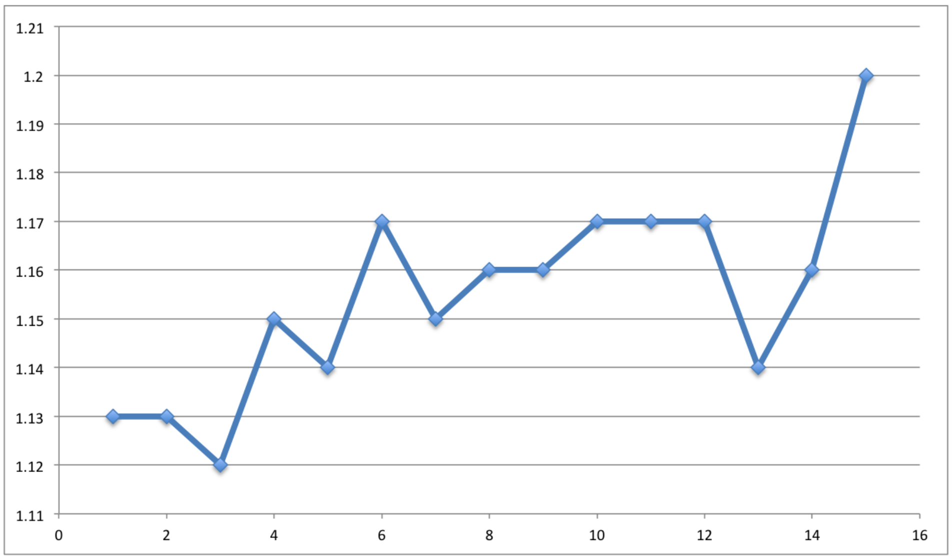

Figure 13: This graph shows the runtime, in seconds, of the `discrete_constraint_estimatestruct` method running on graphs with a varying indegree. The following were held constant: 30 vertices. 2 outcomes per vertex. 150 data points. Maximum 1 witness (this means each pair of vertices was checked for conditional independence given sets of nodes of size ≤ 1).

Test 14: Structure learning runtime as a function of outcomes per vertex.

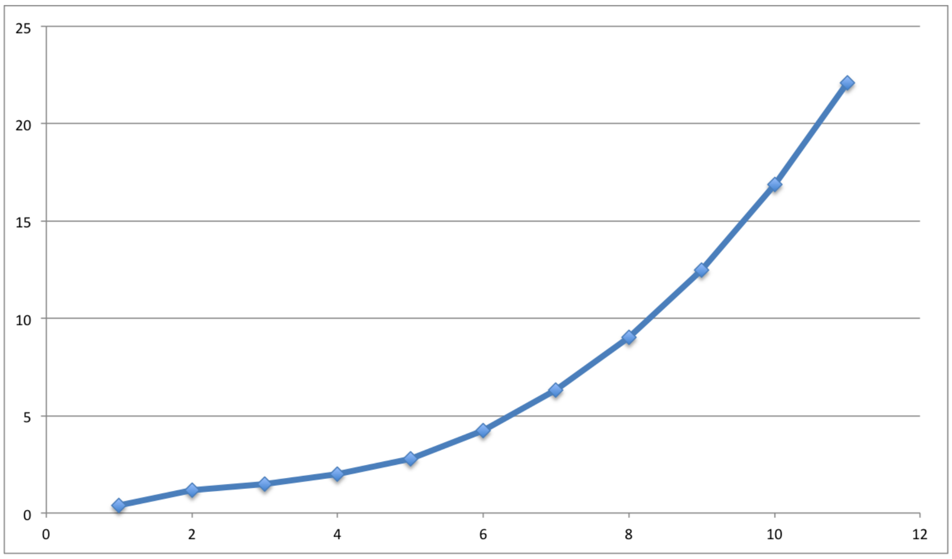

Figure 14: This graph shows the runtime, in seconds, of the `discrete_constraint_estimatestruct` method running on graphs with a varying outcomes per vertex. The following were held constant: 30 vertices. Indegree 2. 150 data points. Maximum 1 witness (this means each pair of vertices was checked for conditional independence given sets of nodes of size ≤ 1).

Test 15: Structure learning runtime as a function of size of data set.

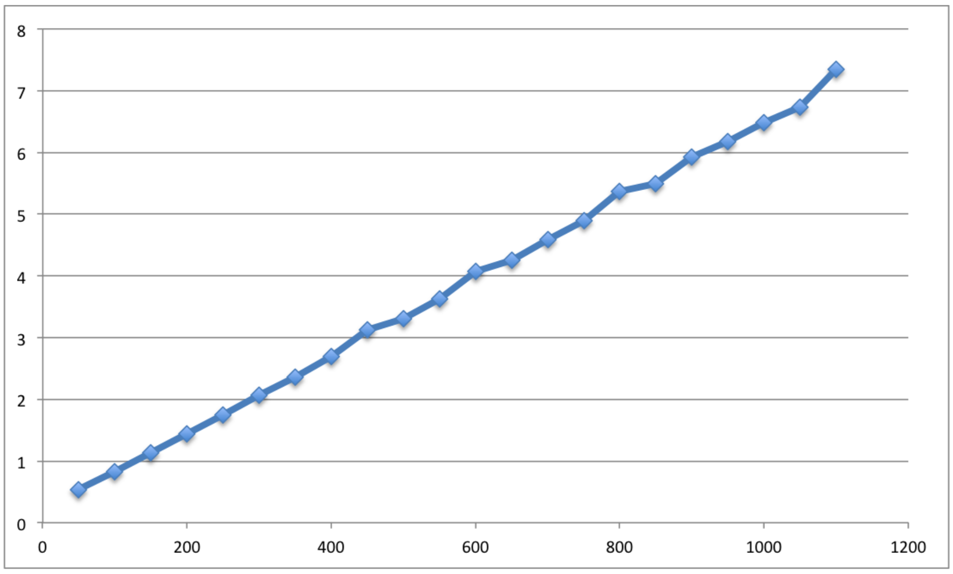

Figure 15: This graph shows the runtime, in seconds, of the `discrete_constraint_estimatestruct` method running on graphs with a varying data set size, measured in number of data points. The following were held constant: 30 vertices. Indegree 2. 2 outcomes per vertex. Maximum 1 witness (this means each pair of vertices was checked for conditional independence given sets of nodes of size ≤ 1).

APPENDIX A: Methodology

The function used to generate graphs is found in `/runtime-tests/bn_generator.py`, along with examples of the `timer.py` and `processor.py` methods used to generate the data. The `pympler` library was used to measure size of python objects. The `time.clock()` method was used to measure runtime.

APPENDIX B: Data Tables

Test 1:

number of vertices	runtime
100	0.05
200	0.1
300	0.15
400	0.21
500	0.26
600	0.31
700	0.38
800	0.44
900	0.5
1000	0.55
1100	0.62
1200	0.7
1300	0.76
1400	0.86
1500	0.91
1600	0.99
1700	1.06
1800	1.13
1900	1.22
2000	1.29
2100	1.35
2200	1.42
2300	1.49
2400	1.57
2500	1.64

Test 2:

indegree	ND size	GS size
1	621584	93848
2	821696	165152
3	1420312	237056
4	2196888	307816
5	4443488	378944
6	7792624	448712
7	17281032	520576
8	31969704	590600
9	72569936	660704
10	136750336	727904
11	309613872	797880

Test 3:

outcomes per vertex	ND size	GS size
1	575760	165152
2	821696	165152
3	1501568	165152
4	2200560	165152
5	4125232	165152
6	5554624	165152
7	7379040	165152
8	9642160	165152
9	13951568	165152
10	20329632	165152
11	24571704	165152
12	29467040	165152
13	35058552	165152
14	41389152	165152
15	48501752	165152
16	56439264	165152
17	71468280	165152
18	81935384	165152
19	104384520	165152

Test 4:

number of vertices	runtime
100	0.05
200	0.12
300	0.19
400	0.28
500	0.38
600	0.48
700	0.6
800	0.72
900	0.85
1000	1.04
1100	1.16
1200	1.33
1300	1.5
1400	1.74
1500	1.97
1600	2.18
1700	2.4
1800	2.74
1900	2.83
2000	3.25
2100	3.32
2200	3.65
2300	3.97
2400	4.29
2500	4.58

Test 5:

number of vertices	runtime
100	0.02
200	0.06
300	0.11
400	0.15
500	0.22
600	0.3
700	0.38
800	0.48
900	0.59
1000	0.72
1100	0.83
1200	0.97
1300	1.13
1400	1.28
1500	1.46
1600	1.65
1700	1.84
1800	2.06
1900	2.25
2000	2.58
2100	2.73
2200	2.99
2300	3.25
2400	3.56
2500	3.88

Test 6:

indegree	runtime
1	0.06
2	0.08
3	0.15
4	0.34
5	0.92
6	2.77
7	8.01

Test 7:

outcomes per vertex	runtime
2	0.06
3	0.15
4	0.42
5	1.08
6	2.38
7	4.97
8	9.29
9	16.89
10	27.96

Test 8:

number of vertices	runtime
100	0.36
200	0.74
300	1.14
400	1.52
500	1.93
600	2.39
700	2.86
800	3.16

Test 9:

indegree	runtime
1	0.98
2	1.17
3	1.35
4	1.49
5	1.81
6	2.06
7	2.41
8	3.05
9	4.06
10	5.95

Test 10:

outcomes per vertex	runtime
1	1.06
2	1.13
3	1.21
4	1.27
5	1.48

Test 11:

number of data points	runtime
500	0.4
1000	0.8
1500	1.18
2000	1.58
2500	1.97
3000	2.36
3500	2.77
4000	3.23
4500	3.48

Test 12:

number of vertices	runtime
5	0.01
10	0.07
15	0.2
20	0.41
25	0.71
30	1.14
35	1.71
40	2.44
45	3.4
50	4.56
55	5.91
60	7.52
65	9.67
70	11.59
75	14

Test 13:

indegree	runtime
1	1.13
2	1.13
3	1.12
4	1.15
5	1.14
6	1.17
7	1.15
8	1.16
9	1.16
10	1.17
11	1.17
12	1.17
13	1.14
14	1.16
15	1.2

Test 14:

outcomes per vertex	runtime
1	0.39
2	1.17
3	1.5
4	2
5	2.8
6	4.22
7	6.31
8	9.01
9	12.47
10	16.86
11	22.11

Test 15:

number of data points	runtime
50	0.54
100	0.82
150	1.13
200	1.44
250	1.75
300	2.06
350	2.36
400	2.69
450	3.13
500	3.3
550	3.63
600	4.07
650	4.26
700	4.59
750	4.89
800	5.37
850	5.49
900	5.93
950	6.18
1000	6.48
1050	6.74
1100	7.34