Modeling Resource Limitation in Decision Networks

Barker, Hayden
University of Alberta
32 Athabasca Hall, Edmonton, AB
hsbarker@ualberta.ca

Rosevear, Cody
University of Alberta
32 Athabasca Hall, Edmonton, AB
rosevear@ualberta.ca

Abstract

Decision networks are designed to help people make optimal decisions given probabilities of different factors and the desire of certain outcomes. The average decision network assumes that all decisions, while still connected, are independent in the sense that choosing to perform one action doesn't stop the availability of a future action. However, it is common in real world circumstances for there to be a limit on the number of actions that can be performed. This is the idea of resource limitation. While being a large part of daily life, there is no clear representation of this concept in decision networks to date. We have developed a method capable of modeling resource limitation in decision networks through the use of an additional utility node. We developed our networks using UnbBayes, and tested them using BNT in Matlab. Measured using mean expected utility and mean strategy similarity, our model outperformed the baseline model and did not outperform the network without resource limitations, indicating a successful outcome. This paper will focus on explaining the development of our method and the results of our tests.

1 Introduction

1.1 The problem

On a day to day basis, every person has a limit on the number of actions he/she can perform due to time, money, or other constraints. This is an example of resource limitation. Since it is impossible for a person to perform every action he/she desires, priorities need to be made. As a result, a strategy is usually created in order to choose the best action to achieve a goal. However, these strategies are often less than optimal. An example would be a doctor testing a patient for a disease at every check-up. It would be better for the doctor to only test the patient when the probability of an issue is high enough to make it a real possibility. By performing these unnecessary tests, the doctor is using precious time that could have been used to test the patient for other, possibly more dangerous, diseases. This scenario led to the purpose of this project - to find methods to model resource limitations in decision making contexts.

For our research, we decided to focus on developing a resource limitation method for decision networks. Decision networks are a type of directed acyclic graph, comprised of a collection of probabilistic chance nodes, decision nodes, and utility nodes, connected together with edges. They are used in many different decision tasks, ranging from expert medical decision making [1] to assisting in environmental impact assessments of new engineering projects [2]. The goal of this project is to provide a viable and effective method for modeling resource limited decision making so that future researchers will be able to implement these methods for their own tasks. Essentially, we are not looking to solve any specific real world problem; rather, we are looking to find a method to help others solve real world problems.

2 Related Work

The task of monitoring a subject over time with the intention of determining how best to treat them has been studied before within various medical contexts, ranging from efforts to determine how best to administer cancer treatment therapies [3], to planning long term hemodialysis treatment for kidney failure [4]. Many of these papers, however, are not explicitly decision theoretic in nature; instead, they take as their starting point a treatment that has already been predetermined to be the most suitable, and are primarily concerned with optimizing how to best administer that treatment over a long period of time rather than choosing how to optimize treatment from among a multitude of different options. Moreover, there is no focus on the value of information gathering actions, such as testing for the state of one particular disease versus another, since there is only one issue under consideration.

Other work in this vein [5] does take an explicitly decision theoretic approach to the problem, and incorporates the cost of performing a treatment or test at a given point in time versus not performing that treatment, rather than assuming that a certain treatment should be applied and that it is simply a matter of determining the timing. However, the problem is not framed in such a way that accommodates any type of resource limitation; it is taken as given that, once a decision is made, the resulting plan can be carried out.

Finally, [6] describe a few ways of representing memory within the context of influence diagrams, where decisions need to be made regarding whether or not to test or treat for a given disease. Again, the context is not one with explicit resource limitation, but it nevertheless provides a valuable example on how to represent and keep track of decisions throughout time, which, when combined with the general ideas of the above work concerning long term patient monitoring, provides a basis on which to build a decision network model with explicit resource limited monitoring.

3 Development

3.1 General Model

To cover a large number of situations, we decided to keep the decision network fairly general but complex enough to show that the method was robust. The network was based off the idea of modeling the progression of a patient's disease(s) through time. This involved creating a temporal decision network with two decision nodes, test and treat, for each time-point. Test was a decision node to decide whether or not to test a patient at a given point in order to have a better idea of the state of the disease at the next time-point. Treat was a decision node to decide whether or not to treat a patient at a given point in order to reduce the severity of the disease. The two decisions connected to a utility node which provided both decision nodes with a cost for each action. In addition, there were two chance nodes. One was the observed status of the disease which was designed to represent the way a doctor, or another human being, would observe a disease progressing over time. This node was what fed into the decision nodes. The other was a node that represented the actual progression of a disease through time. Since it was an unobservable node, it was only connected to the utility in the current time-point. The purpose of the connection to the utility was to add the idea of failure to adhere to the overarching task of keeping the person healthy. There was also an edge from the current real disease severity to the next observed disease severity because we believed that the actual progression of the disease would have an effect on the future expected progression due to visible levels of change caused by the disease. The utility was connected to three nodes in total in order to keep track of the cost of performing tests and treatments and to penalize for a patient's suffering/perishing due to their deteriorating condition. Overall, we felt that this model encapsulated the basics of any resource limited decision task. It contained the idea of making a decision between multiple choices, with costs and benefits for each choice, and it included the idea of perception of a situation versus the actual, underlying situation. Figure 1. shows an image of our complete network over a 5 day span.

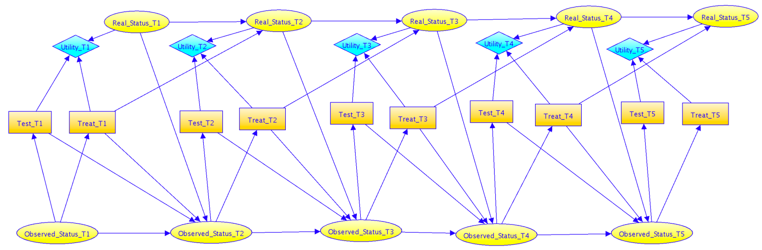

Figure 1: Sample figure caption.

In addition to the model discussed, another more complex model was created to further check our method. As many people have several diseases at the same time, we created a model with 2 diseases, thus doubling the number of decision nodes and chance nodes. The goal of this was to show that the model could do more than a simple ”yes” or ”no” for a couple decisions for a disease. This more advanced model was used to show that the method could handle keeping track of multiple issues and make appropriate compromises in order to maintain the person’s health.

3.2 Scope

Due to the theoretical nature of this paper, work had to go into avoiding scope-creep. The original idea was to create a model which could track an individual through time until the end of his/her life, planning every single decision from now until the end of time. However, this was impractical. First, a method such as this would require a large amount of computational power due to the sheer number of nodes that would be produced. Secondly, it begged the question: what if the person’s situation changed? For instance, the person may develop another disease or the cost of a treatment is changed or any other number of factors. This would severely damage the integrity of the current results and force the user to restart from the beginning. This led us to the idea of limiting the time window of the model. We settled on a 5 day time window because it was long enough to be useful while still being short enough to avoid effects from the issues previously mentioned. In addition, the 5 day time window could be connected through time, if need be, in order to simulate a never-ending model.

3.3 Tools

Due to software limitations, a couple tools needed to be used to help us develop our method. Some compromises needed to be made as well. We used UnbBayes [7]to prototype all our ideas and to make all the networks in this paper. The advantage of this tool was its GUI which allowed us to easily view our ideas in an understandable format. However, the downsides were that the tool could not handle very large networks and it did not explicitly create the decision rule. Instead, to see the decision rule, one would have to walk through the software’s expected utility outputs and infer the decision rule. BNT [8] for Matlab was the tool that we used to truly test our methods. This Matlab package had the ability to handle the calculations for much larger networks and it was able to provide the proper decision rule for a network. However, the package didn’t contain a GUI thus making it difficult to conceptualize the model structure. Also, BNT for Matlab didn’t support continuous nodes in a decision network. This had originally been a key feature in the development of our model of disease progression. We had planned to model progression as a distribution which grew in variance over time but, unfortunately, this was no longer an option. Due to this limitation, we were forced to make all nodes discrete. Between these two tools, we were able to complete all the tasks necessary to develop and test our method for modeling resource limitation.

4 Resource Limitation Method

4.1 Idea

Multiple methods for modeling resource limitation in a decision network were investigated before we found our current method. Originally, we had a model, much like our general model discussed above, that had all decisions connected to every utility node through time. This, we figured, would allow each time-step to be aware of what was possible and not possible, based on the fact that it knew what had been done and what will be done. In order to make this work, the utilities needed to be filled in properly. The user would be required to adjust the utilities so that, whenever more than the desired number of test or treat actions were performed, the utility would become very negative. This would discourage the network from ever making a decision to go over the resource limit. However, this failed for a very obvious reason. The method required that future decisions influence past decisions, which doesn't make sense in the real world. Figure 2. shows an example of this idea.

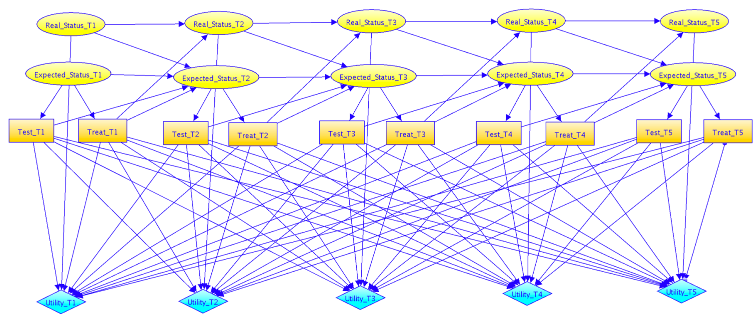

Figure 2: Sample figure caption.

The next method involved adapting the notion of no-forgetting diagrams, as found in [6]. This method involved having each decision aware of the number of previous tests or treatments performed. By doing this, the network would be able to limit the number of test and treatments performed through the entire time window without ever having a future node affecting a past node. Unfortunately, our tools didn't allow for the partial specification of decision rules based on the results of previous decisions (one can either specify ones own decision rue, or allow the system to solve for it). Therefore, we came up with an alternative. We connected the decision nodes in each time-step to an additional counter node for that time-step. These nodes took on discrete values between 0 and the resource limit plus 1, based on what actions had been performed in the time-step. Initially, 0 has a 100 percent probability, meaning that the current count of tests and treatments is 0. However, as test and treatments are performed, a higher discrete value receives a 100 percent probability. This higher discrete number is based on the number of actions taken so far. We can calculate this based on the previous count node and the number of tests or treatments performed at the time-step. The count node is also connected to the utility node for the current time steps, which allows us to use the same very negative utility punishment technique when the network exceeds the resource limit. Figure 3. shows an example of the modified no-forgetting network.

While the no-forgetting method prototyped well, the laborious nature of the additional nodes made us look for a simpler solution. The problem was that the count node needed to grow larger at each time point in order to keep track of the total number of decisions that are possible up to that point. While this is possible in principle it is tedious to keep track of and implement as one's network grows larger. Therefore, instead of creating multiple count nodes throughout the network, we instead added a single non-temporal utility node with edges from all decisions over the 5 point time window feeding into it, and set all rows of that table where the number of decisions exceeded the resource limit to a high negative utility (and 0 to everywhere else), to bias the BNT solver away from treating that decision combination as a viable option. Using a single table also has the added advantage that it

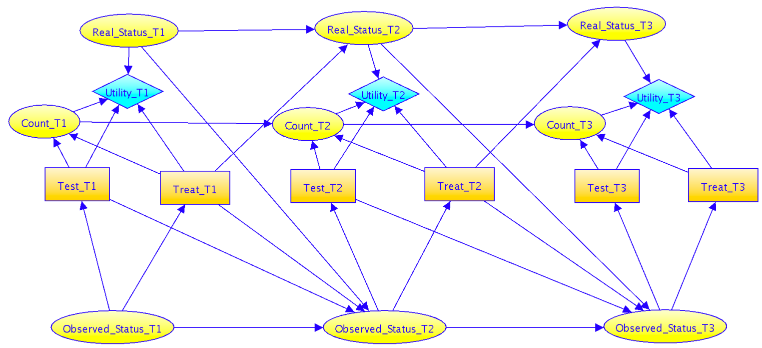

Figure 3: Sample figure caption.

is easier to change one's specified resource limit, as one need only go through the table and add or remove a high negative utility value to those combinations that exceed or fall within one's desired resource limit. Figure 4. shows the finalized model.

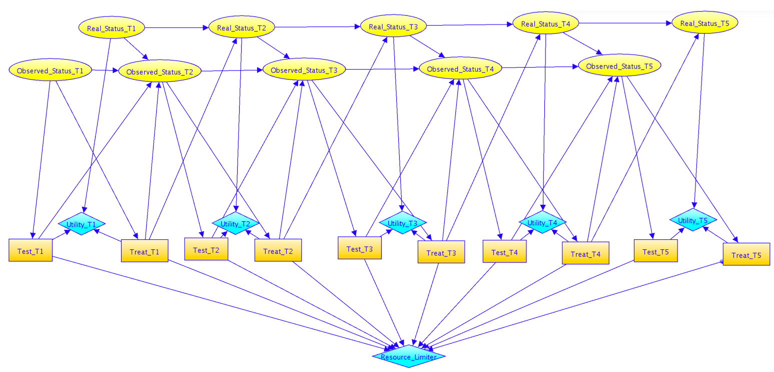

Figure 4: Sample figure caption.

4.2 Simplifications, Assumptions and Data Creation

A few simplifications and assumptions were made in order to simplify the model to a manageable level. First, we decided to focus on limiting the overall number of test and treatments together. We assumed that the average test took just as much resources as the average treatment. This meant that performing a test or a treatment counted the same towards a resource limitation. By doing this, we were able to reduce the problem slightly and make it a bit faster to implement. Also, as mentioned earlier, to ease implementation, we assumed that all node distributions were discrete.

Given the dearth of related work focused specifically on the issue of resource limited decision making, as well as the generic nature of our model, there was little in the way of concrete data that we could find or easily obtain for the purposes of testing our model. As such, we were required to construct our own parameterizations of the model rather than learning it from data. We did this by handcrafting various conditional probability distributions (CPD's) under a set of plausible assumptions that helped characterize the manner in which a disease would progress stochastically over time while interacting with treatments.

These basic assumptions were as follows:

- Diseases come in various stages of severity: minor, moderate, and severe

- The probability of a disease getting worse from one time point to the next is such that a patient's condition will most likely progress linearly through the stages (from minor, to moderate, to severe).
- Skipping stages of severity, is, although possible, less likely.
- Treatment at a given time point generally reduces the chance that one's condition will progress to a worse stage
- Failing to treat at a given time point generally increases the chance that one's condition will progress to a worse stage
- Sometimes treatments can have adverse effects (think of reactions to drugs, for example), although this should be, in general, rare

The above assumptions describe how disease progression is characterized by our model in the most general terms. Note, however, that we did not use a single disease profile in testing our model, as we wanted to avoid the undue influence of pathological situations in evaluating our model. Thus the precise quantitative relationship between how likely a disease is to progress to one stage given the current stage differs according to the different disease profiles used by our model.

Another important facet of the model's parameterization is that of the various utilities used to measure the situation. Normally, one would go through a process of preference elicitation in order to determine how to assign the various utilities to the outcomes and actions of a concrete situation. Given the generic nature of our model, however, we ignored this step, and instead specified our own utilities. We did this for a variety of different scenarios, each of which was intended to serve as a representative case of specific classes of scenarios. All of these scenarios used the same utility scale, where 0 is regarded as an occurrence of neutral valence (it cost nothing), and -100 was regarded as the most unpleasant set of outcomes.

5 Evaluation

5.1 Metrics

There are three levels along which our method must be evaluated. First, one needs to find a way to represent the general situation, absent resource limitation, in a plausible way; if the general situation is representationally flawed then any attempts to apply resource limitation to the situation may provide inaccurate results. Secondly, one must find a way to represent resource limitation itself in a fashion that allows Bayesian networks to do their work. Finally, if any such model is to be useful in practice, one must ensure that the implementation is correct. We concerned ourselves primarily with the first two issues, operating under the assumption that the inference algorithms provided in the tools we used to represent the situation were implemented correctly.

In order to judge the effectiveness of our representations with respect to the first two issues we used two metrics: the mean maximum expected utility and the mean strategy similarity, both of which were defined for each network representation and computed over various CPD and utility parameterizations of that network.

The mean maximum expected utility is the arithmetic average of the maximum expected utility of the network over its different parameterizations, where the maximum expected utility of a network, given a particular parameterization, is the largest expected utility across all possible strategies for that network. A strategy is itself comprised of the assignment of a policy to each decision node in the network, where a policy is a decision rule that specifies which decisions will be taken under what circumstances (where circumstances are described in terms of the network's chance nodes). Given that all of our decisions are binary, all of our policies, and hence strategies, can be represented as binary valued vectors whose percentage of element wise equality serves as a simple, intuitive metric of strategy similarity. The mean strategy similarity of a network is thus the arithmetic average of each optimal strategy for that network compared to every other optimal strategy, for the different network parameterizations. Observing the changes in strategy similarity across various parameterizations also allows us to perform sensitivity analysis on the networks with respect to parameterizations of their CPD's and utilities, as described in [9].

5.2 Tests

There are a number of dimensions along which to test/explore our models, including the different parameterizations of CPDs and utilities, the number of co-occurring issues we are monitoring, and the degree and scope of resource limitation.

With respect to the parameterizations, we tested the following scenarios:

- a disease that progresses at a slow, medium or fast rate (each one has it's own CPD)
- uniform utility costs for all actions/outcomes
- utility assignments where the cost of the disease>cost of the treatment>cost of the test
- a scenarios with the same utility assignments above, except the disease had a higher cost (it cost 10 more utility)
- utility assignments where the cost of the test>the treatment
- utility assignments where the cost of the disease is lower than the treatment cost and the test cost
- utility assignments where the cost of the disease remains constant over time

We tested all of the above scenarios on three different networks: a network with no resource limitations, a baseline network which performed the treat action at every second time-step, and a network with resource limitations (a maximum of 3 actions throughout the 5 day span). Each of these scenarios are all modified such that they deviate from a common baseline. The quantitative details of each scenario can be found in the appendix.

In order to perform sensitivity analysis with respect to CPDs and utilities, we held one of the two constant while iterating through all of the scenarios of the other. We did this for each network, and we ensured that for each network the fixed CPD or utility was always the same across networks, or, in the case of baseline networks, that the CPD or utility profile, which was held constant, was equivalent to the subset one would get if the original CPD/utility were conditioned on the action that defined the baseline network. With respect to the dimension of the number of issues we considered two cases: a network with a single issue to monitor and one with two issues.

5.2.1 Implementation

All of the source code used for our experiments can be found at: `https://github.com/ZerkTheMighty/PGMProject-Resource-Limited-Monitoring`. All of our tests rely on the BNT influence diagram solver to find the strategy and maximum expected utility of the current network.

5.3 Results

Given the above metrics and test cases, we are now in a position to predict how we should expect our models to perform in terms of mean maximum expected utility (MEU) relative to one another, on average, across different parameterizations of the networks. We should expect the networks that represent the baseline strategy to have the lowest mean MEU since it has the same maximum number of actions as the resource limited model, but it should have the highest degree of strategy similarity (since their is only one baseline strategy being represented in the network). On the other hand, we should expect the entirely unconstrained network to perform significantly better in terms of mean MEU, given that it is not being forced to conform to a baseline strategy and suffers no resource limitations. In between these two extremes, we should expect the resource limited networks to perform better than the naive baseline approach, but still worse than the completely unconstrained network.

Tables 1 and 2 show the results of our tests. RLM stands for 'resource limited monitoring'. The values 'adjusted disease' or 'adjusted utility' within the Model Test column signify whether the disease or utility parameterization was adjusted on each iteration when calculating test statistics. The 'Baseline', 'RLM', and 'No RLM' signify the different networks that were parameterized and solved for by BNT.

Table 1: 5 day time-span results for a single disease

Model Test	Mean of Expected Utility	Mean Strategy Similarity
Adjusted Disease Baseline	-269	100
Adjusted Utility Baseline	-157	100
Adjusted Disease RLM	-257	100
Adjusted Utility RLM	-132	77
Adjusted Disease No RLM	-242	78
Adjusted Utility No RLM	-127	57

Table 2: 5 day time-span results for multiple diseases

Model Test	Mean of Expected Utility	Mean Strategy Similarity
Adjusted Disease Baseline	-580	100
Adjusted Utility Baseline	-346	100
Adjusted Disease RLM	-579	90
Adjusted Utility RLM	-341	84
Adjusted Disease No RLM	-533	67
Adjusted Utility No RLM	-309	74

Tables 1 and 2 show that the results generally match our earlier expectations. Based on our observations, our model outperformed the baseline and it didn't outperform the network with no resource limitation. This means that our model seems to be better than the simple naive baseline strategy, and it doesn't seem to violate any constraints that allow it to outperform the network with no resource limitations. While these results are encouraging, Table 2 shows that our model is not performing much better than the baseline. Further research may be useful in discovering the cause of this small difference in the multiple issue case.

6 Conclusion

Through our research we found many ways of potentially modelling resource limitation. Every method discussed had its uses. However, our recommendation is our final model (in figure 4.) due to its simplicity and ability to handle temporal decision networks. While initial results are encouraging, further work needs to be done. It would be beneficial to find real data-sets to perform tests on in order to further validate correctness, and compare the network's recommended strategies against real word strategies as currently used by physicians. We would also like to gather/simulate a larger sample size of disease and utility distributions, in order to perform a t-test between the baseline method and our resource limited representation. The goal of this would be to show that our representation allows fort the discovery of strategies statistically outperform the baseline, across a wide range of parameterizations. Overall, more work needs to be done to fully validate our method of representing the problem, but initial results suggest that our model is a good basis for future research.

Acknowledgments

The authors thank Russell Greiner, Roberto Vega, and all of our peers for the comments and constructive feedback on earlier versions of this work.

References

[1] S. Sadeghi, A. Barzi, and N. Zarrin-Khameh, "Automated medical decision making utilizing bayesian network knowledge domain modeling," Feb. 3 2004. US Patent 6,687,685.

[2] J. Zhu and A. Deshmukh, "Application of bayesian decision networks to life cycle engineering in green design and manufacturing," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 2, pp. 91–103, 2003.

[3] C. Berzuini, R. Bellazzi, S. Quaglini, and D. J. Spiegelhalter, "Bayesian networks for patient monitoring," *Artificial Intelligence in Medicine*, vol. 4, no. 3, pp. 243–260, 1992.

[4] C. Rose, C. Smaili, and F. Charpillet, "A dynamic bayesian network for handling uncertainty in a decision support system adapted to the monitoring of patients treated by hemodialysis," in *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, pp. 5–pp, IEEE, 2005.

[5] N. B. Peek, "Explicit temporal models for decision–theoretic planning of clinical management," *Artificial Intelligence in Medicine*, vol. 15, no. 2, pp. 135–154, 1999.

[6] S. L. Lauritzen and D. Nilsson, "Representing and solving decision problems with limited information," *Management Science*, vol. 47, no. 9, pp. 1235–1251, 2001.

[7] S. Matsumoto, R. N. Carvalho, M. Ladeira, P. C. G. da Costa, L. L. Santos, D. Silva, M. Onishi, E. Machado, and K. Cai, "Unbbayes: a java framework for probabilistic models in ai," *Java in academia and research*, p. 34, 2011.

[8] K. P. Murphy, "The bayes net toolbox for matlab," *Computing Science and Statistics*, vol. 33, p. 2001, 2001.

[9] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 ed., 2003.

Appendices

Appendix A: Resources

Network Source Code: `https://github.com/ZerkTheMighty/PGMProject-Resource-Limited-Monitoring`
Bayes Net ToolBox: `https://github.com/bayesnet/bnt`
UnBayes: `https://sourceforge.net/projects/unbbayes/`

Appendix B: Sample CPD's And Utility Assignments

```
DISEASE PROFILES
treat_d = 1 = don't treat
treat_d = 2 = treat

Uniform progressive
S_true(t - 1) treat_d(t - 1) S_true(t)
         1             1             1 0.33
         2             1             1 0.33
         3             1             1 0.33
         1             2             1 0.33
         2             2             1 0.33
         3             2             1 0.33
```

```
          1             1              2 0.33
          2             1              2 0.33
          3             1              2 0.33
          1             2              2 0.33
          2             2              2 0.33
          3             2              2 0.33
          1             1              3 0.33
          2             1              3 0.33
          3             1              3 0.33
          1             2              3 0.33
          2             2              3 0.33
          3             2              3 0.33

Minor progressive
S_true(t - 1) treat_d(t - 1) S_true(t)
          1             1              1 0.85
          2             1              1 0.75
          3             1              1 0.65
          1             2              1 0.95
          2             2              1 0.80
          3             2              1 0.75
          1             1              2 0.10
          2             1              2 0.15
          3             1              2 0.20
          1             2              2 0.04
          2             2              2 0.15
          3             2              2 0.10
          1             1              3 0.05
          2             1              3 0.10
          3             1              3 0.15
          1             2              3 0.01
          2             2              3 0.05
          3             2              3 0.10

Moderately progressive
S_true(t - 1) treat_d(t - 1) S_true(t)
          1             1              1 0.40
          2             1              1 0.10
          3             1              1 0.05
          1             2              1 0.80
          2             2              1 0.70
          3             2              1 0.20
          1             1              2 0.50
          2             1              2 0.40
          3             1              2 0.15
          1             2              2 0.15
          2             2              2 0.20
          3             2              2 0.50
          1             1              3 0.10
          2             1              3 0.60
          3             1              3 0.80
          1             2              3 0.05
          2             2              3 0.10
          3             2              3 0.30

Highly progressive
S_true(t - 1) treat_d(t - 1) S_true(t)
          1             1              1 0.10
          2             1              1 0.05
```

3	1	1	0.01
1	2	1	0.20
2	2	1	0.10
3	2	1	0.05
1	1	2	0.50
2	1	2	0.25
3	1	2	0.04
1	2	2	0.60
2	2	2	0.40
3	2	2	0.30
1	1	3	0.40
2	1	3	0.70
3	1	3	0.95
1	2	3	0.20
2	2	3	0.50
3	2	3	0.65

BASELINE STRATEGY NETWORK DISEASE PROFILES

These are acquired by essentially conditioning on the treatment action (such that we always treat

Uniform progressive

S_true(t - 1)	treat_d(t - 1)	S_true(t)	
1	2	1	0.33
2	2	1	0.33
3	2	1	0.33
1	2	2	0.33
2	2	2	0.33
3	2	2	0.33
1	2	3	0.33
2	2	3	0.33
3	2	3	0.33

Minor progressive

S_true(t - 1)	treat_d(t - 1)	S_true(t)	
1	2	1	0.95
2	2	1	0.80
3	2	1	0.75
1	2	2	0.04
2	2	2	0.15
3	2	2	0.10
1	2	3	0.01
2	2	3	0.05
3	2	3	0.10

Moderately progressive

S_true(t - 1)	treat_d(t - 1)	S_true(t)	
1	2	1	0.80
2	2	1	0.70
3	2	1	0.20
1	2	2	0.15
2	2	2	0.20
3	2	2	0.50
1	2	3	0.05
2	2	3	0.10
3	2	3	0.30

Highly progressive

S_true(t - 1)	treat_d(t - 1)	S_true(t)	
1	2	1	0.20

```
          2             2             1 0.10
          3             2             1 0.05
          1             2             2 0.60
          2             2             2 0.40
          3             2             2 0.30
          1             2             3 0.20
          2             2             3 0.50
          3             2             3 0.65

UTILITY PROFILES
Utility scale ranges from 0 (no cost) to -100 (maximum unpleasantness)

test = 1 = don't test
test = 2 = test
treat = 1 = don't treat
treat = 2 = treat

Standard costs (the different scenarios below modify these in various ways)
Disease: -25, -50, -75 (it cost more as one's condition deteriorates)
Test: -5
Treatment: -10

Utility profile used as a constant when varying different
disease progression distributions
Cost of disease > treatment > test
S_true  test_d treat_d utility
      1       1       1 -25
      2       1       1 -50
      3       1       1 -75
      1       2       1 -30
      2       2       1 -55
      3       2       1 -80
      1       1       2 -35
      2       1       2 -60
      3       1       2 -85
      1       2       2 -40
      2       2       2 -65
      3       2       2 -90

All of following utility profiles are modifications
of the above 'constant' one, to induce certain
specific scenarios

Cost of disease > treatment > test (But more severe disease by -10)
S_true  test_d treat_d utility
      1       1       1 -35
      2       1       1 -60
      3       1       1 -85
      1       2       1 -30
      2       2       1 -65
      3       2       1 -90
      1       1       2 -45
      2       1       2 -70
      3       1       2 -95
      1       2       2 -50
      2       2       2 -75
      3       2       2 -100

Disease has minuscule cost such that cost of treat > disease and cost of test > disease
```

```
S_true  test_d treat_d utility
      1      1       1 -1
      2      1       1 -2
      3      1       1 -3
      1      2       1 -6
      2      2       1 -7
      3      2       1 -8
      1      1       2 -11
      2      1       2 -12
      3      1       2 -13
      1      2       2 -16
      2      2       2 -17
      3      2       2 -18

Cost of Test > cost of treatment
S_true  test_d treat_d utility
      1      1       1 -35
      2      1       1 -60
      3      1       1 -85
      1      2       1 -45
      2      2       1 -70
      3      2       1 -95
      1      1       2 -30
      2      1       2 -65
      3      1       2 -90
      1      2       2 -50
      2      2       2 -75
      3      2       2 -100

Cost of disease remains constant
S_true  test_d treat_d utility
      1      1       1 -25
      2      1       1 -25
      3      1       1 -25
      1      2       1 -30
      2      2       1 -30
      3      2       1 -30
      1      1       2 -35
      2      1       2 -35
      3      1       2 -35
      1      2       2 -40
      2      2       2 -40
      3      2       2 -40

Uniform Cost
S_true  test_d treat_d utility
      1      1       1 -10
      2      1       1 -10
      3      1       1 -10
      1      2       1 -10
      2      2       1 -10
      3      2       1 -10
      1      1       2 -10
      2      1       2 -10
      3      1       2 -10
      1      2       2 -10
      2      2       2 -10
      3      2       2 -10
```

```
BASELINE STRATEGY NETWORK UTILITY PROFILES
These are the utility profiles used in the baseline network.
They are simply all of the above utility profiles
constrained to the case where the monitor always chooses to treat.
  1       1       2 -35
      2       1       2 -60
      3       1       2 -85

      [-35 -60 -85]

      1       1       2 -45
      2       1       2 -70
      3       1       2 -95

      [-45 -70 -95]

      1       1       2 -11
      2       1       2 -12
      3       1       2 -13

      [-11 -12 -13]

      1       1       2 -30
      2       1       2 -65
      3       1       2 -90

      [-30 -65 -90]

      1       1       2 -35
      2       1       2 -35
      3       1       2 -35

      [-35 -35 -35]

      1       1       2 -10
      2       1       2 -10
      3       1       2 -10

      [-10 -10 -10]

RESOURCE LIMITED UTILITY PROFILES
These are utility profiles that bias the
influence diagram solver away from sets of decisions
that exceed the daily resource limit

They are the same as the original profiles,
except whenever two decisions are made at once,
the utility is set so low as to make that
decision pair invalid in the context of maximizing
expected utility

Cost of disease > treatment > test
S_true  test_d treat_d utility
      1       1       1 -25
      2       1       1 -50
      3       1       1 -75
      1       2       1 -30
      2       2       1 -55
      3       2       1 -80
```

```
      1       1       2 -35
      2       1       2 -60
      3       1       2 -85
      1       2       2 -1000
      2       2       2 -1000
      3       2       2 -1000


Cost of disease > treatment > test (But more severe disease by -10)
S_true  test_d treat_d utility
      1       1       1 -35
      2       1       1 -60
      3       1       1 -85
      1       2       1 -30
      2       2       1 -65
      3       2       1 -90
      1       1       2 -45
      2       1       2 -70
      3       1       2 -95
      1       2       2 -1000
      2       2       2 -1000
      3       2       2 -1000


Disease has minuscule cost such that cost of treat > disease and cost of test > disease
S_true  test_d treat_d utility
      1       1       1 -1
      2       1       1 -2
      3       1       1 -3
      1       2       1 -6
      2       2       1 -7
      3       2       1 -8
      1       1       2 -11
      2       1       2 -12
      3       1       2 -13
      1       2       2 -1000
      2       2       2 -1000
      3       2       2 -1000

Cost of Test > cost of treatment
S_true  test_d treat_d utility
      1       1       1 -35
      2       1       1 -60
      3       1       1 -85
      1       2       1 -45
      2       2       1 -70
      3       2       1 -95
      1       1       2 -30
      2       1       2 -65
      3       1       2 -90
      1       2       2 -1000
      2       2       2 -1000
      3       2       2 -1000

Cost of disease remains constant
S_true  test_d treat_d utility
      1       1       1 -25
      2       1       1 -25
      3       1       1 -25
```

```
      1       2       1 -30
      2       2       1 -30
      3       2       1 -30
      1       1       2 -35
      2       1       2 -35
      3       1       2 -35
      1       2       2 -1000
      2       2       2 -1000
      3       2       2 -1000

Uniform Cost
S_true  test_d treat_d utility
      1       1       1 -10
      2       1       1 -10
      3       1       1 -10
      1       2       1 -10
      2       2       1 -10
      3       2       1 -10
      1       1       2 -10
      2       1       2 -10
      3       1       2 -10
      1       2       2 -1000
      2       2       2 -1000
      3       2       2 -1000
```