Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

.

USING DISCRETE EVENT SIMULATION IN THE TEACHING OF DECISION ANALYSIS

Ingolf Ståhl

Department of Managerial Economics
Stockholm School of Economics
S-11383 Stockholm, SWEDEN

ABSTRACT

In this paper we discuss how Discrete Event Simulation (DES) was used in a course on Decision Analysis (DA). Against the background of the characteristics of the students and the purpose of the course, we discuss various types of problems and methods that were found suitable to include in the course, in order to show the place of DES in DA. We present a number of simple GPSS programs that have been used in the course and proved effective in promoting the students' understanding of DA.

1 INTRODUCTION

1.1 Types of Students

The background of this paper is a course in Decision Analysis (DA), given annually to around 40 students at the Stockholm School of Economics. The students are either seniors in an undergraduate program or taking electives the last year in an international masters program. While most students are Swedish, at least a quarter are from other countries, like the US and China. The course has been changed very much this year when I took it over. Earlier it was focused mainly on theoretical issues, often of a more behavioral type. The problem examples were then mostly extremely simple and solved by hand. The use of computers was restricted to some use of a decision tree program.

One of my primary goals for the course was that the students, who should soon go out into the real world with evermore powerful PCs, should be able to use the computer also for complex decision problems. Although simple problems are suitable for explaining the general principles of DA, it is also important to show the students tools that are suitable for dealing with problems with many decision alternatives and many possible outcomes.

No one in the class had ever had a course in computer programming. This ruled out teaching the students how to analyze the problems in a General Programming Language (GPL), like Visual Basic (VB), which would have allowed flexibility. Teaching students a GPL from scratch in five hours, the maximum time allowable for such an activity, would be impossible. Hence, we had to choose other types of software. Another limitation in our class was that we wanted to teach the students tools that they would be sure to be able to use in their future jobs. This implies that we could not use expensive packages that a graduate on the first job could hardly persuade management to buy.

In the course we used a mixture of three software products: TreePlan, an inexpensive decision tree software, Excel and WebGPSS, a DES that is available free on the Web. We shall in this paper try to motivate this choice, in particular the choice of a DES package, which is not so common in DA courses. We shall deal with some of the limitations of a decision tree software and Excel as regards important problems in DA and see how a DES tool like GPSS can solve them. To make these points more clear, it appears suitable to first present the different types of problem situations that we wanted to address in our course.

1.2 Types of Problems

The common thing for all problems dealt with in a course in DA is that they are all characterized by the fact that one or several decisions shall be made and that many decisions can lead to different events. Furthermore, the problems are characterized by uncertainty, mainly in the form of risk, in the sense that one can assign probabilities to the different events. We make a distinction between situations that are characterized by a single one-shot decision and those in which there are sequential decisions, in the sense that first a decision is made and then the outcomes of this decision become known, before another decision has to be made.

We also see the problem situations characterized by the **number of decision alternatives** in the decisions, distinguishing between if they are few or many, seeing a continuum as the extreme case of many alternatives. We furthermore see the problem situations characterized by the **number of different events** to which we assign probabilities, distinguishing between just a few events and many events, seeing a continuous probability distribution as the extreme case of many events.

We shall also distinguish between situations with regard to how important time is for the problem. We can mainly distinguish between (*a*) the case when **no time** is involved at all, in the sense that all decisions could be made at the same time, (*b*) the case of **ordinal** time, in the sense that only the relative order in which decisions and events take place matters, but not the actual length of time between them, and (*c*) the case of **cardinal time,** where the time actually passes between decisions, and events really matter for which decisions are selected.

Furthermore, one can characterize problems on the basis of the goals involved, e.g. whether there is mainly one type of goal, like a financial goal, or multiple goals. Due to time limitation, we were in the course forced to focus on the case with only one goal, a financial one. Finally, one can characterize the problem situations on the basis of importance of information availability. We shall here deal mainly with the ordinary case that the information availability will not change over time.

1.3 Types of Methods

Against this background of different types of problem situations, one can discuss what kind of general methodology would be most suitable for our students to learn for dealing with these types of different problems. We here regard **optimization and simulation** as the two main forms of methodologies that can be used in DA.

Optimization is seen as the calculation of expected value and then choosing the alternative that leads to the highest expected value. Using game theory terminology, the analysis of optimization can mainly be made using the Extended form or the Normal form. The Extended form refers to decision trees, either constructed using special decision tree software or made by the modeler directly e.g. in Excel. The Normal form can mainly take the form of a matrix, where all decisions are represented in the rows and all events and their probabilities in the columns. There is here, as discussed below, a great difference between the one-shot case and the sequential decision case.

Simulation can either take the form of static Monte Carlo simulation, e.g. in Excel, or dynamic simulation of the DES type. In a static simulation there is a limited number of time points to which we can assign events and results, while in a dynamic simulation every time point is possible. Excel is mainly limited to static simulation. In case we represent time over the columns, which is commonly done, we cannot have more than 256 time points at the very most. In many cases we would assign the results just to the end of the months. Dynamic simulation is provided by any DES, where the times of different events are determined usually by sampling a floating point number and the clock is then updated to the time of the most imminent event. As we hence move from one event to another we can move to any possible floating-point value of time.

We shall below look a little bit closer at some of the methods outlined above to see how DES is positioned vis-à-vis these methods, to see more clearly the role that DES can play in a course on DA.

2 THE LIMITS OF OPTIMIZATION METHODS

2.1 Decision Tree Software

The study of the most commonly used text books in DA indicates that the main computer software tools used in DA are decision tree tools, sometimes complemented with a tool for construction of Influence Graphs. Among the more noted decision tree tools one can mention DATA-TreeAge, TreePlan, DPL, Decision Pro and Precision Tree. They have in common that the user constructs the decision tree in a simple manner, e.g. by indicating the number and type of branches at different nodes. Usually there are decision and event branches, emanating from a node of the same type. To each branch one can in most systems assign a specific "marginal" value, influencing the end-nodes, which have the sum of all these marginal values connected to the path leading from the start node to the specific end node.

The decision tree software calculates the optimal decision and the value connected to this by a roll back method, where the roll back procedure starts with the end nodes. To each event node it assigns the expected value as the sum of the products of probability of the event ∗ the value at the node at the end of the event branch. To each decision node it assigns the maximum value of all the values at the nodes at the end of the branches of this decision node.

This is generally a very illustrative, as well as useful, method, provided that the total number of nodes is fairly small. This number is dependent on the number of stages in the case of sequential decisions and the number of event branches at each event node and the number of decision branches at each decision node. For example, in the case of a decision problem with two stages, each with two decision alternatives and with each decision branch in turn leading to two event branches, we would, as seen in Figure 1, get a total of (2*2)*(2*2) = 16 end nodes. Hence, already in this simple case the tree becomes fairly large. If there are three stages, three decision nodes at each stage and three event nodes at each decision, we would have $3^6 = 729$ end nodes. It is very difficult to get an overview of such a large tree. The more decision nodes and event notes, the larger the problems with the decision trees.

In our DA course we used the TreePlan software (a shareware, costing at most $29.00), since we did not want to base the course on expensive software that the students, as newly employed, might have trouble getting their future employer to buy for them. The students found TreePlan quite easy to use, but out of the roughly 10 examples of decision trees used, no one had more than 16 end nodes, thus all allowing for a readable tree on one page. Since the deci-

sion trees are lined up to make the tree look like a real tree, with e.g. in case of two branches emanating from one node, one branch going upwards and one downwards, such a tree still takes a lot of space, as seen in Figure 1.

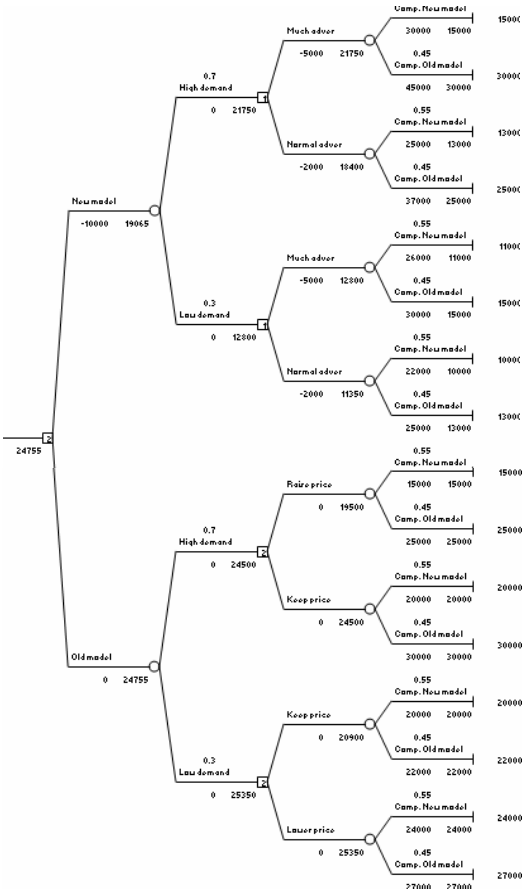

Figure 1: Example of 2*2*2*2 Decision Tree

A special case of many nodes is the case with a continuum of decision alternatives and the case when the outcome of a decision can be seen as a continuous probability function. In these cases other methodologies than this type of decision trees are definitely needed.

2.2 Excel Based Tree Emulation

The tree in Figure 1 no doubt causes a problem by its height. One can save in height by using an Excel sheet where the branches are not written out, but the lines of the rows in the sheet indicate where the nodes are. An example of a problem with the same size as that above is presented in Figure 2, where the 2*2*2*2 tree of Figure 1 is "emulated". As in Figure 1, it is not important to read the text. The only thing of interest is the difference in height between the two figures. We mainly save in height by having all information referring to one end node on the same row, letting the top branch emanating from the node to the left follow directly on the same row and having all other "invisible" branches emanating from this node placed on a lower row.

Decisions	Model	M 1	Val. 1	EVs 1	Demand	Prob	Decis. 2	Alternat 2	M 2	Val 2	EVs 2	Compet	Prob.	T M	End N
Old	New	-10	24.76	19.07	High	0.7	Much ad	Much ad	-5	21.75	21.75	New mod	0.55	30	15
		-10							-5				0.45	45	30
		-10						Normal ad	-2		18.4	Old mod	0.55	25	13
		-10							-2				0.45	37	25
		-10			Low	0.3	Much ad	Much ad	-5	12.8	12.8	New mod	0.55	26	11
		-10							-5				0.45	30	15
		-10						Normal ad	-2		11.35	Old mod	0.55	22	10
		-10							-2				0.45	25	13
	Old	0		24.76	High	0.7	Keep price	Raise price	0	24.5	19.5	New mod	0.55	15	15
		0							0				0.45	25	25
		0						Keep price	0		24.5	Old mod	0.55	20	20
		0							0				0.45	30	30
		0			Low	0.3	Lower price	Keep price	0	25.35	20.9	New mod	0.55	20	20
		0							0				0.45	22	22
		0						Lower price	0		25.35	Old mod	0.55	24	24
		0							0				0.45	27	27

Figure 2: Excel Based Decision Tree Emulation

Even in this case, it becomes difficult to get a good overview of a problem of the 3^6 size with 729 end nodes, since this would require 729 lines, corresponding to several pages. A disadvantage of this "tree emulation" approach, in contrast to the real decision tree case, is that one as user has to input many formulas one self, which can be quite time consuming. The teaching of how to do this involved just around one hour in our DA class.

2.3 Decision Analysis in the Normal Form

With this limitation of the decision tree approach in mind, whether we have it in its original tree form or in form of ordinary rows in an Excel-sheet, implying the same kind of methodology, one needs to look also at other methods for solving DA problems by optimization. Analysis of the problem in the normal form is then the natural alternative. One would here use a matrix, where one as rows would have the decisions and as columns the events. To each event one can assign a probability that it occurs, provided a certain decision has been made.

We exemplify this with Figure 3, based on a problem from Eppen et al. (1998). We have an outcome matrix, e.g. with the decisions on order quantity as rows (4–7) and the events (=demand) as columns (B-E). The possible profits of e.g. decision alternative 1 are in the array B5:E5, with the values that this alternative can lead to. On row 9, i.e. in the array B9:E9, we have the corresponding probabilities.

	A	B	C	D	E	F	G	H	I
1						Expected		Best	Best
2			Demand			value		value	decision
3	Decision	0	1	2	3				
4	0	0	-50	-100	-150	-85	0		
5	1	-40	35	-15	-65	-12,5	1		
6	2	-80	-5	70	20	22,5	2		
7	3	-120	-45	30	105	7,5	3		
8									
9	Probabilities	0,1	0,3	0,4	0,2			22,5	2

Figure 3: Newsboy Problem in the Normal Form

We here have the simplest case when the same events occur whatever decision is made. To calculate the expected value of a decision we multiply the value of the cell with the probability of reaching this cell. In cell F5 we can hence calculate the expected value of alternative 1 as SUMPRODUCT(B5:E5,B$9:E$9). Having thus the expected value of each alternative in the column array (F4:F7), the value of the best decision can be given in cell H9 as MAX(F4:F7). If we then insert a column G, in which we repeat the values of the decisions of column A, the best value on the decision variable can be given in cell I9 as VLOOKUP(H9,F4:G7,2).

In the case that the probabilities of the events are different for each alternative, we would require a separate probability line for each event and a corresponding change in the SUMPRODUCT formula. This case would still be quite simple and easy to handle for one-shot decisions, as long as the number of alternatives and events do not become very large.

The greatest problems arise when we have a sequential decision. If we would use this standard type of spreadsheet analysis, we would have to combine the decisions of the various stages and in a corresponding fashion combine the events, so that to each path of decision combinations we can assign a unique end value. For example, if we have 3 stages and at each stage 3 decisions, each leading to one of 3 possible events then we have, as noted above, $3^6 = 729$ end nodes and just as many paths. In order to model this problem in the normal form, we would need $3^3= 27$ decisions and $3^3= 27$ events. With more decisions and events and hence an even larger matrix, we would soon reach the limit of feasibility of an ordinary spreadsheet analysis. A continuum of decision alternatives and/or continuous probability functions is not at all possible to handle in this way. Hence, analysis in the normal form suffers from almost the same kind of weakness as the analysis in the extensive form described above. Complex decisions are difficult to solve directly using the mentioned methods.

There are then two other alternatives, both involving approximations. One is simulation to be discussed later. If we want to stay within optimization, we can in the case where the problem lies in a continuous probability function, approximate this function with a few strategically chosen data points. This would allow us to use a small number of event nodes, coming out of each decision node. An example is the Pearson-Tukey method, where three points are selected, namely the points for which the cumulative probability is 0.05, 0.5 and 0.95, respectively (Keefer and Bodily 1983). These three points are given the discrete probabilities 0.185, 0.67 and 0.185. One hence reduces the problem to deal with three events instead of a continuum.

There remains the question of how good this approximation is. If we are unable to make an exact analysis, we have to use another method for investigating how good the approximation is. We can also in this case use simulation. In the course, we dealt with several examples, which were solved both by optimization with the Pearson-Tukey method and by simulation.

3 SIMULATION

The main focus of simulation is, however, that it can handle not only the case with continuous probability distributions, but also the more general case of many alternatives and, perhaps even more importantly, the case of uncertainty over time, i.e. dynamic uncertainty, e.g. when there is uncertainty about the time a certain job will take.

If we start with the simpler static case, we can in many cases use either Monte Carlo simulation or DES.

3.1 Monte Carlo Simulation

For Monte Carlo simulation, we used in our course only ordinary Excel, without any external add-in. An alternative, used in many DA courses at other schools, is to use an add-in like @Risk or Crystal Ball. One reason for not doing this was that we, as mentioned, wanted the course to rely on software that the students could be certain would be available in their future work. Another reason was that the Monte Carlo simulations that we did in this class are, as explained below, quite simple to do in Excel, with the students in the process learning important features of Excel.

Instead of calculating the expected value of each decision as the product of the probability of the event and the value at the node, one will in the Monte Carlo simulation make several runs and in each run sample a probability and then on the basis of this probability determine which alternative will be selected. Each specific run will imply that at each decision node only one alternative, out of the many alternatives emanating from this node, will be selected, leading in turn to one specific value for each run. This value is likely to change from run to run.

We illustrate this by the simple Newsboy problem in Figure 3. We can very easily change this spreadsheet so that it becomes suitable for a Monte Carlo simulation. We have in Figure 4 only included the rows below row 9, since the first 9 rows are the same as in Figure 3.

	A	B	C	D	E	F	G	H	I
10	Cum. Prob.	0.1	0.4	0.8	1				
11	Random	0.4988							
12						Value	Sum of	Average	Best
13		Event=Demand				in run	values		value
14	Decision	0	1	2	3				
15	0	0	0	-100	0	-100	-8E+05	-84.408	
16	1	0	0	-15	0	-15	-1E+05	-12.151	
17	2	0	0	70	0	70	225000	22.5023	
18	3	0	0	30	0	30	66000	6.60066	22.5
19	Run	1							
20	Nr of runs	10000							

Figure 4: Monte Carlo Simulation of the Newsboy problem

We have here introduced cumulative probabilities in row 10 to determine the event of this run, based on the random number sampled in cell B11, in this case 0.4988. Since this random number lies above the upper limit of the cumulative probability for event 1, i.e. 0.4, but below the upper limit of the cumulative probability for event 2, 0.8, we have sampled the occurrence of event 2. This leads to the values for the decisions being the values of event = demand 2, as seen earlier in Figure 3. This provides the value of each decision for this specific run as shown in column F, which is the sum of all the values from columns B to E, with all the cells except those of column D being 0. This provides the value of a specific run.

We must now in order to have a great many simulation runs be able to sum the results of a decision from these runs. This is done in column G. We here have a special Excel formula in e.g. cell G15, namely as follows, =IF(run=0,0,G15+F15). *run* is the name given to the cell B19, where we at present see a value 1. If we set this cell to 0, the cell G15 will become 0. If we then set the cell *run* to 1, we start the addition. We would here normally get a Circular reference error message, since cell G15 refers to itself. Since we on purpose want this to happen, we have to click on *Tools*|*Option* and then on the tab *Calculation*. Here we must get a check mark in the box *Iterations* and in the field *Maximum iterations* we write 1. We should also in the field *Maximum change* write 0. The idea is that we shall just make one iteration at a time, where an iteration implies that Excel determines the values from cell A1 down to the lowest row with a non-blank cell, on each row to the right-most non-empty cell. When in this iteration the determination of values passes though the cell B11, with =RAND(), a new random number is generated and then a new value is given to each decision for this run. Consequently a new value would be added to the sum in cell G15.

To start the sum, we have to reset G15 to 0. This is done by setting the cell *run* to 0. If we next write 1 in cell *run,* B19*,* (and then e.g. press Enter), a new iteration will be made and a new result will be added to the sum in cell G15. If we then again write 1 (or in principle any other value than 0) in the cell *run*, we will make yet another run and add yet another value to cell G15. A copy of the formula in G15 to G16:G18 allows a similar addition in these cells.

Below the cell *run*, we have cell B20 with the number of runs. This contains the formula =IF(run=0,0,B20+1). This also implies a summing up. When we set run to 0, we set the number of runs to 0. When we then set *run* to 1, we add 1 to the present number of runs, i.e. cell B20 is first set to 1. The second time we give *run* the value 1, the number of runs is increased to 2, and so on. We can now use this number of runs to calculate, in H15:H18, the average as the sum of values, in G15:G18, divided by the number of runs thus far. Thus each run will give us a new set of averages. The more runs we make, the closer the averages should get to the true average, i.e. the expected value.

It is cumbersome to do the simulation manually, by writing 1 and pressing Enter, a great number of times. It is here natural that the students learn to record a macro, by *Tool*|*Macro*|*Record New Macro*. It is suitable to assign a short cut letter, like *M*, to the macro in the dialog then opened and click on OK. We then start by moving to the *run* cell, write 0 here and press Enter. Next we write 1, followed by Enter, a number of times. Finally we click on *Tools*|*Macro*|*Stop recording* (or the stop recording button). We can next edit the macro, by *Tools*|*Macro*|*Edit macro*. We might then see a code looking a follows:

```
Sub Macro1()
' Macro1 Macro
' Keyboard Shortcut: Ctrl+m
  Range("B21").Select
  ActiveCell.FormulaR1C1 = "0"
  ActiveCell.FormulaR1C1 = "1"
…..
  ActiveCell.FormulaR1C1 = "1"
  ActiveCell.FormulaR1C1 = "1"
  End Sub
```

It is obviously also cumbersome to have a manually constructed macro like this, if we want to run the simulation e.g. 10000 times. We can then change the program by deleting all but one of the lines with ActiveCell.FormulaR1C1 = "1", and instead write *For j*=1 *To* 10000 in front of it and *Next j* after it, so that the program looks as follows, with the comments (starting with ‘) excluded:

```
Sub Macro1()
Range("B21").Select
ActiveCell.FormulaR1C1 = "0"
For j= 1 To 10000
    ActiveCell.FormulaR1C1 = "1"
Next j
End Sub
```

When we return to the Excel-sheet (e.g. by Alt Q), we can run the simulation 10000 times by just typing *Ctrl M*. After these 10000 runs we should obtain an average for the value of decision 2 (= the best value) that is close to the expected value of 22.5.

3.2 Discrete Event Simulation

We hence see that we can for some decision problems build up a Monte Carlo simulation simply. It should be mentioned that we could build up a model in a discrete events simulation language like GPSS roughly as easily. We used WebGPSS as the language in our course for DES, because some students in the course had studied WebGPSS in an earlier course and the other students could in four extra class room hours, using Ståhl (2003) and Born and Ståhl (2003), learn enough GPSS to be able to write the programs in this paper. Finally, since WebGPSS is available free of charge on the Web, our students could be sure of being able to access GPSS also in their future work.

We can solve the Newsboy problem with the following WebGPSS program.

```
       simulate
       help      experi,x$order,x$avepro,4,0,3,100
demand function  rn4,r
0 1
1 3
2 4
3 2
run    generate  1
       let       x$demand=fn$demand
       if        x$demand>x$order,miss
*
nomiss let       x$sold=x$demand
       let       x$lost=0
       goto      profit
*
miss   let       x$sold=x$order
       let       x$lost=x$demand-x$order
*
profit let       x$profit=x$sold*75-x$order*40-x$lost*50
       let+      x$sumpro,x$profit
       let       x$avepro=x$sumpro/n$run
       terminate 1
       start     10000,np
       end
```

The program produced among other things the following results as regards the expected outcome of the decisions:

Invalue	Outvalue: Average	Limits with 95 % probab. Lower limit		Upper limit
0.00	-84.98	-85.07	-	-84.90
1.00	-12.45	-12.54	-	-12.37
2.00	22.50	22.40	-	22.59
3.00	7.47	7.34	-	7.61

We here run the program 10000 times in 100 batches, i.e. a million times, to see the variance of the batches. For the decision variable 2, with the expected value 22.5 we get an average value of 22.5, i.e. the expected value, but we can only with 95 percent confidence say that the expected value lies between 22.4 and 22.59.

It should be stressed that running this program, involving 1,000,000 runs, on an ordinary PC took roughly the same time as running just 10,000 runs with the Excel model above. It might, however, be argued that the increase in precision is marginal and might not warrant the extra effort learning some GPSS. In fact, for the problem shown above the direct calculation of the expected value, as shown in Figure 3, is quite straight forward and it is not necessary to use simulation at all. The programs above were, however, shown to, and run by, the students so that they would appreciate the number of runs necessary in order to get a good estimate of the expected value. Such a comparison could be made here, but not in later cases when the direct calculation of the expected value is not possible and simulation is the only feasible method. To these problems, the student could then come with an understanding that a very large number of runs are necessary.

3.3 Simulation of Sequential Decisions

The simple example above dealt with a one-shot decision. In this case, the simulation was quite easy both in the Excel and the DES-GPSS case. When we come to the case of a sequential decision situation, the DES simulation will still be simple, but the simulation in Excel will be much harder, if not outright infeasible. To understand this, we must first notice that in a spreadsheet (where we do not have circular reference or use VBA) all values of each cell will be the same regardless of whether we move them to other cells in the spreadsheet. Excel has its special way of calculating the values of all cells, moving from the upper left-hand corner to the right on the first row until all cells to the right are empty, then moving to the left-most cell on the next row and then moving on this row until the right-most non-empty cell, and so on. This implies that the Excel user cannot determine the order in which the values are calculated. We cannot easily determine that one set of cells shall get their values calculated first when another set of cells have had their values determined. This is of importance when trying to do a Monte Carlo simulation in Excel by a great number of runs.

To fully understand the problem involved in Excel, we shall first look at how a simple multi-stage simulation can be done in DES. In the case of a simulation involving two sequential decisions, where we want to simulate the roll back method involved in the solution of decision trees, the outcomes of the second stage decisions should be completely determined before we turn to the solution of the decision in the first stage. In GPSS this can be done easily, since we can provide ordinal time by using the cardinal timing of the GPSS system.

The easiest way to explain this is by providing an example and a GPSS program. The example is from a textbook by Clemen (1996, p. 105), used in our DA class. This is a two stage decision, though of low complexity. Figure 5 shows the situation. The first decision is whether to accept Texaco's bid of $ 2 Billion for our company or to give a counteroffer of $ 2 Billion. The second decision occurs when Texaco in turn has given a counteroffer of $3 Billion. Shall we then accept Texaco's counteroffer or go to court?

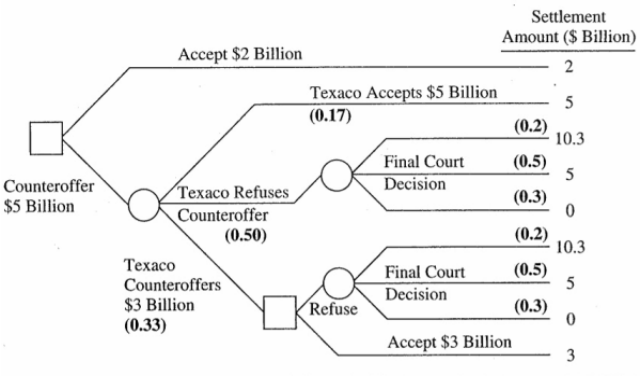

Figure 5: A Two-stage Decision Problem (Figure 4.2 in Clemen 1996)

The expected value of the best decision is 4.635. This can be calculated directly in Excel, but we cannot do a straight-forward simulation in Excel to get an estimate of this result. This can, however, be done in GPSS, as shown by the following program .

```
       simulate  1
       help      experi,x$val,x$eval,1,1,1,100
out    function  rn4,r
win  2
medi  5
loose  3
out1   function  rn3,r
taccep 17
trefus 50
tcount 33
       let       x$accept=2
       let       x$taccep=3
       generate  ,,,10000
       goto      fn$out
win    let       x$countr=10.3
       goto      choice
medi   let       x$countr=5
       goto      choice
loose  let       x$countr=0
choice let+      x$sum,x$countr
let    x$tref=x$sum/10000
       terminate
       generate  ,,1,1
       if        x$taccep>x$tref,accep
ref    let       x$count=x$tref
       terminate
accep  let       x$count=x$taccep
       terminate
       generate  ,,2,10000
       goto      fn$out1
taccep let       x$countr=5
       goto      choic2
trefus let       x$countr=x$tref
       goto      choic2
tcount let       x$countr=x$count
choic2 let+      x$sum1,x$countr
       let       x$countr=x$sum1/10000
       terminate
       generate  ,,3,1
       if        x$accept>x$countr,accept
countr let       x$eval=x$countr
       terminate 1
accept let       x$eval=x$accept
       terminate 1
       start     1,np
       end
```

The program contains four segments, each one starting with a GENERATE block.

The first one with GENERATE ,,,10000 will at time 0 generate 10000 transactions. Twenty percent of these transactions are sampled to go to WIN, where the outcome of the court settlement is determined as 10.3; 50 percent go to MEDI, for a settlement of 5 and 30 percent to LOOSE for a settlement of 0. All transactions in this segment then meet at CHOICE, where we add up and take the average of these court settlements to calculate one expected value, x$tref. This segment, which refers to the right parts of the tree in Figure 5, is thus guaranteed to be calculated first.

The second segment with GENERATE ,,1,1 implies that a single transaction is generated at time 1, i.e. after the expected value of the court settlement has been calculated. Here we calculate x$count as the highest value of x$taccept and the just calculated value of the court settlement.

The third segment with GENERATE ,,2,10000 will at time 2, i.e. after x$tref and x$count have been calculated, generate 10000 transactions. Seventeen percent are sampled to go to TACCEP for Texaco accepting 5, 50 percent go to TREFUS, implying that one gets the value of the court decision x$tref, and 33 percent go to TCOUNT, where Texaco counters leading to x$count, just calculated in segment 2. Finally all transactions meet in CHOICE for the calculation of the expected value x$countr of a counter offer.

The fourth segment, with GENERATE ,,3,1 generates one single transaction which determines the best value of the problem as the highest of what we get by accepting, x$accept, and by counter offering, x$countr, just calculated. After this the simulation is stopped by TERMINATE 1.

This is a simulation that cannot be done easily in Excel. The reason is that we cannot in Excel first run the calculations corresponding to the first segment 10,000 times; then with the average from these runs determine the value of the second decision and then run the events in the first stage 10,000 times. In Excel we cannot, without resorting to programming in VBA, outside the scope of most of our students, first run only specific parts of the spreadsheet many times and later other parts many times. All cells in the spreadsheet get their values determined in each run.

We have studied this program mainly to see what can be done in DES and not in Excel, but we also want to see how well we approximate the exact solution. We get an expected value, printed with 2 decimals, as 4.64, to be compared with the true result of 4.635. We should mention that the simulation takes only a few seconds on an ordinary PC.

However, we can easily make the problem more interesting, e.g. by assuming that in the 50 percent case when the court settlement is assumed to be medium, we instead have an exponential distribution with the average of 5. We replace the block `medi let x$countr=5` with the block `medi let x$countr=5*fn$xpdis`. The solution is now no longer so easy to calculate without simulation. It is the use of many alternatives and continuous probability distributions in the case of multi-stage decisions that makes simulation with a DES like WebGPSS of interest in the static case.

4 PROBABILITY CALCULATIONS

We have in our DA course, just like in many other DA courses, included parts on how different forms of joint probabilities are calculated. We want to show that by using the computer, e.g. with DES software, one could for more difficult probability calculations, do these not only faster, but also with greater chance of a correct answer, at least as regards the most significant decimals. One can well pay the price of getting the answer wrong as regards the fourth or fifth decimal, if one thereby substantially reduces the risk of the students getting an answer wrong by say 20 percent. In the course we illustrated this use of DES for probability calculations with two examples found in the literature.

The first example deals with a **Car-Trip Example,** presented by Barton (2002). "D. Event owns two cars, one old, the other older. Each day, he uses a car to get to school and back. Car 1 is old and has probability 0.79 of starting on any day, while Car 2, even older, starts with probability 0.71. For either car, once it is started, the chance that it completes a trip to or from school is 0.95. Compute the probability that D. Event will make it to school and back on any particular day." As pointed out already by Barton, most students assume that the system consists of two cars in parallel, and they perform calculations leading to a probability of 0.762. The correct value can, with some effort, be calculated as 0.65878. A simpler and safer way of getting the correct solution is by DES. Barton used ARENA; our students WebGPSS. The program requires only very basic WebGPSS, as seen in the block diagram in Figure 6.

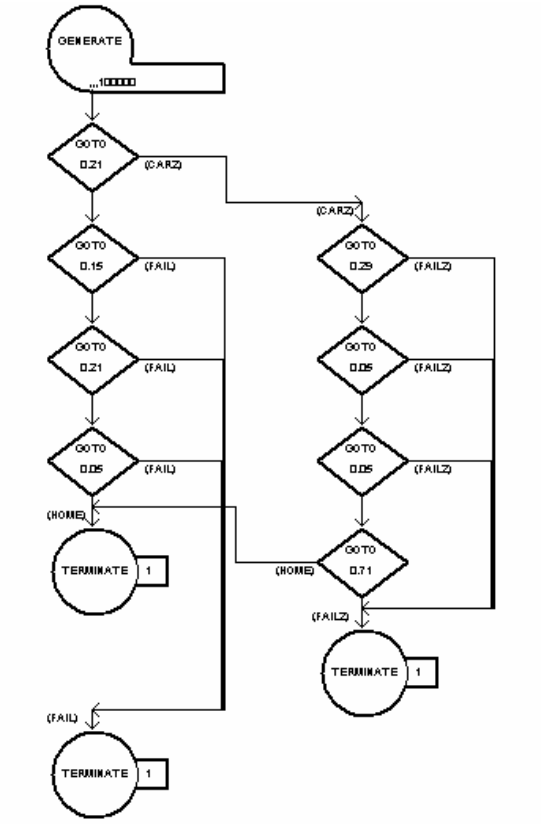

Figure 6. Block Diagram of Car Trip Model

In the block statistics we find that 65979 have come to the block with the address HOME, i.e. the calculated probability is 0.65979, implying an error of only 0.00101.

The next example is by Kim (2004). "There is a small theater. The ticket price is $10 each. There are 2N people waiting in line to buy tickets. Among 2N people, N people carry only one $10 bill and the rest carry one $20 bill. The theater has zero money. So they need to give $10 bills they receive as change to those with $20 bills. People waiting in line can buy tickets from the booth only and the theater sells tickets on first come first serve basis. What is the probability that the theatre will be able to sell 2N tickets without any problem when 2N people line up randomly?"

The students were asked to write a program, which calculates the probability of not running out of money, not only when one starts with zero, but also with any starting amount. A GPSS program, for the case of 100 ticket buyers (N=50), run 500 times, would look as follows:

```
       simulate  1
       help      experi,x$cash,x$prob,1,0,0,500
start  generate  ,,,100
       goto      dol10,0.5
       let-      x$cash,10
       if        x$cash<0,fail
       goto      finish
dol10  let+      x$cash,10
finish if        n$start=100,succ
       terminate 1
succ   let       x$prob=100
       terminate 1
fail   let       x$prob=0
       terminate 100
       start     100,np
       end
```

At the end of the run, we get the following report:

```
 Result in run 500      0.00
 After 500 runs:
 Average:      8.20   Standard deviation:     27.46
 With 95 percent probability:
 Average lies between      5.74 and     10.66
```

We see that we had failure in run 500, but of the 500 runs 8.2 percent resulted in success. The average success probability would in the long run lie between 5.74 and 10.66 with 95 percent probability. We can run the same program with a starting cash of 50, replacing the second statement in the program with `help experi,x$cash,x$prob,1,50,50,500.`

5 A BIDDING EXAMPLE

We included in our course also other examples where DES could be important. Samuelson and Bazerman (1985) provided a bidding example, for which DES provides a powerful and illustrative solution. The problem is as follows:

Company A is considering acquiring Company T by means of a tender offer in cash for 100% of T's shares. The value of T depends on the outcome of an oil exploration project that T is currently undertaking. If the exploration fails completely, T will under current management be worth $0 per share, but in the case of a complete success, a share of T will be worth $100. All values between $ 0 and $ 100 are equally likely. Regardless of outcome, T will be worth 50% more under A's management than under its current management. The price of the offer to T must be determined before A knows the outcome of the drilling project, but T will know the outcome when deciding whether or not to accept A's offer. T is expected to accept any offer from A that is greater than the per share value of T under its current management. What price should A offer?

A typical student view is that the expected value of T to its current owner is $ 50 and that is worth 50 % more to A. Hence, A should bid in the interval of $ 50 to $ 75. To demonstrate that this is wrong, we simulate the expected value for A for a given price offer by the program in Figure 7 with two segments. The first segment starts at times 1, 2, ..,100, in the case of 100 contract cases being investigated. We determine the value of T as a value lying between 0 and 100. In order to illustrate the results, we also print the time and the value computed. In the second segment, we generate the bids which come at times 1.5, 2.5, .., 100.5, i.e. always a little later than when the value was determined. We here leave with no contract if the bid is lower than the just determined value. Otherwise, we set the profit of A to 1.5 times the value minus the bid. We print that the value has been accepted and the resulting profit. We sum up all the profits and calculate the average of all profits.

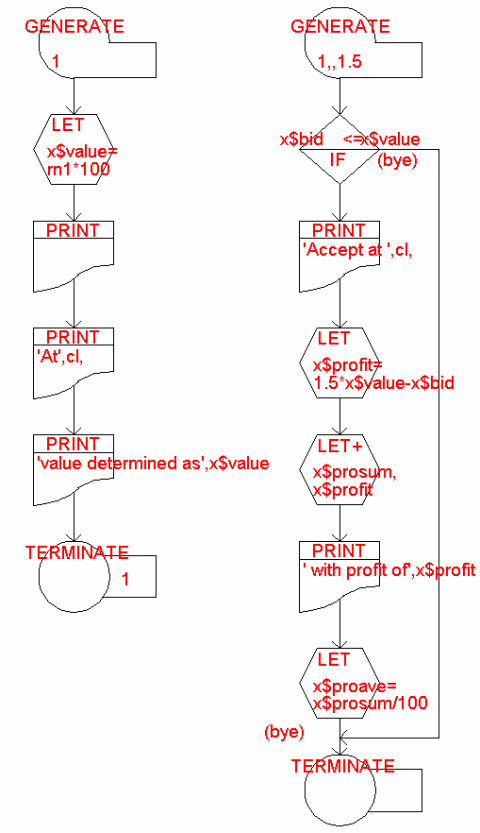

Figure 7: Block Diagram of Bidding Price Model

The program provides an output of the following sort for the case when we have input 55 as A's bid. We have just included a few of the lines for illustration.

```
At      1.00 value determined as      3.79
Accept at       1.50  with profit of     -49.32
At     38.00 value determined as     14.96
Accept at      38.50  with profit of     -32.57
At     39.00 value determined as     74.44
```

This output explains why there is a negative expected profit. In the cases when the value is larger than the bid, there is no acceptance. In the cases when 1.5*value is less than the bid there is a loss. There is a profit only in the cases when 1.5*value > bid > value.

To establish the optimal bid, we run the following extension of the program (without any print-outs):

```
        simulate  1
        help      experi,x$bid,x$proave,10,0,90,20
        generate  1
        let       x$value=rn1*100
        terminate 1
        generate  1,,1.5
        if        x$bid<=x$value,bye
        let       x$profit=1.5*x$value-x$bid
        let+      x$prosum,x$profit
num     let       x$proave=x$prosum/100
bye     terminate
        start     100,np
        end
```

In this program we run the program for all even 10 dollar bids from $0 to $90, with 20 runs for each bid, and see in the output the expected value and their ranges. We clearly see that 0 is the optimal bid. Every one of the other bids will with at least 97.5 (=95+2.5) probability lead to a negative expected value.

```
   Invalue    Outvalue:     Limits with 95 % probab.
              Average      Lower limit  Upper limit
     0.00        0.00          0.00  -       0.00
    10.00       -0.21         -0.31  -      -0.12
    20.00       -1.03         -1.25  -      -0.81
    80.00      -16.96        -18.43  -     -15.50
    90.00      -21.58        -23.39  -     -19.76
```

6 TIME DEPENDENT SIMULATION

The real advantage when it comes to using DES for DA is when there is uncertainty about the time that certain processes take and time is an important variable. Then we must involve time explicitly and a simulation using WebGPSS becomes very much simpler than doing the simulation in Excel. In the course, the students studied an example dealing with the decision on the manning of a team that shall develop a new software product. The product is expected to be of interest only during the next four years. After that the company believes that the product can no longer be sold. We are hence only interested in the profits during the next four years.

The product can only be put on the market once the development is finished. The expected value of the finishing time varies with the number of developers. The real time will, however, be subject to random fluctuations, which management believes can best be described as an Erlang function with a shape factor 4, corresponding to the average value of four samplings from the negative exponential distribution.

There is a cost per developer per day of $1.6 K. If the software has not been developed within three years, development is stopped. The development cost is to be paid at

the end of each month. Once the development is ready, sales start in the immediately following month. Average sales are expected to be 3,500 units per month. Actual sales vary according to a normal distribution with a standard deviation of 700 units. The foreseen price $ 49. The unit cost of production is $ 6.

All monthly profits are discounted back to the present. The discounting rate is 18 percent, corresponding to 0.05 percent per day. We want to estimate the total expected discounted profit for different numbers of developers, what variations are likely and which number of developers seems most suitable. The expected relationship between the number of developers and the average development time is expressed by a function with 9 data pairs.

The following WebGPSS program solves the problem.

```
        simulate  1
        help      experi,x$man,x$pval,9,1,9,100
        let       x$start=1500
        let       x$daycos=1.6
        let       x$ucost=6
        let       x$price=49
        let       x$sales=3.5
        let       x$drate=0.0005
man     function  x$man,c
1 1000
.
9 125
        generate  ,,,1
        let       x$time=fn$rlng4*fn$man
        advance   x$time
        let       x$start=cl
        if        x$start>1080,tofail
        terminate
tofail  terminate 1
        generate  30
        if        cl>x$start,next
        let       x$devcos=30*x$daycos*x$man
        let       x$mincom=0
        let       x$mtocos=0
        goto      joint
next    let       x$rsales=x$sales*(1+0.2*fn$snorm)
        let       x$mincom=x$rsales*x$price
        let       x$mtocos=x$rsales*x$ucost
        let       x$devcos=0
joint   let       x$mprof=x$mincom-x$mtocos-x$devcos
        let+      x$pval,x$mprof*fn$exp(-x$drate*cl)
        terminate
        generate  1440
        terminate 1
        start     1,np
        end
```

In the first segment we sample x$time as the development time. We give the start of production, x$start, the value of the simulation clock, CL. If the development is finished later than at time 1080 we terminate the simulation.

In the second segment, we do the reporting of each month. If it is a month when the development is still going on, we have a development cost, but the income and production costs are 0. If it is a month when the product is developed, we have sales, income and production costs. Regardless of type of month, we finally calculate the monthly profit, and in x$pval we sum up the present value of all the monthly profits.

It should be mentioned that the students were also allowed to do this simulation in Excel, which many did, but most of them solved this problem using quite large and complex spreadsheets, e.g. with one row for each manning alternative and one column for each of the 48 months. The GPSS solutions were clearly very much simpler.

CONCLUSIONS

The examples above show that there are several cases when a course in Decision Analysis can be improved by the use of Discrete Events Simulation (DES). The use of the traditional decision trees is mainly limited to very small problems. Simulation in Excel works well for many one-shot problems, but for sequential decisions DES is needed. DES is also needed when the decision problem deals with uncertainty regarding the length of different times.

REFERENCES

Barton, R. 2002. Using simulation to teach probability: Panel. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C-H.Chen, J Snowdon and J. Charnes, 1816-7. NewYork: ACM.

Born, R. and I. Ståhl. 2003. *WebGPSS slide presentation*. DeKalb, IL: R. Born (available on request from R. Born: <rborn@niu.edu>)

Clemen, R. 1996. *Making hard decisions*. 2nd edit. Duxbury Press, Pacific Grove, CA.

Eppen, G., F. Gould, C. Schmidt, J. Moore and L. Weatherford. 1998. *Introductory management science*. 5th edit. Upper Saddle River, NJ: Prentice Hall.

Keefer D. and S. Bodily. 1983. Three-point approximation for continuous random variables. *Management Science*, 29, 595-609.

Kim, S-H. 2004. The puzzle: Theater with no money. *Informs College on Simulation Newsletter*. Volume 28. Number 2.

Ståhl, I. 2003. *Simulation made simple with WebGPSS – a tutorial*. Stockholm: Stockholm School of Economics.

Samuelson, W. and Bazerman, M. 1985. Negotiating under the winner's curse. In V. Smith (Ed.*), Research in experimental economics* (Vol. 3, pp. 105-137). Greenwich, CT: JAI Press.

AUTHOR BIOGRAPHY

INGOLF STÅHL is a Professor at the Stockholm School of Economics, Stockholm, and has a chair in Computer Based Applications of Economic Theory.. He has taught GPSS for almost thirty years at universities and colleges in Sweden, Norway and the USA. Based on this experience, he has led the development of the micro-GPSS and WebGPSS educational simulation systems. His email address is <ingolf.stahl@hhs.se>. His WebGPSS system is at <http://www.webgpss.com/>.