THE USE OF TIME AND FINANCIAL VALUE IN PROJECT DECISION TREES - A SPECIFIC MODEL AND AN ALGORITHM FOR ROLLING BACK THE TREES

GODINHO, P.C.; Faculty of Economics of the University of Coimbra and INESC; Av. Dias da Silva, 165; 3004-512 Coimbra; Portugal; Phone number: +351+239790571; Fax number: +351+239403511; E-mail:pgodinho@sonata.fe.uc.pt

COSTA, J.P.; Faculty of Economics of the University of Coimbra and INESC; Av. Dias da Silva, 165; 3004-512 Coimbra; Portugal; E-mail: jpaulo@ sonata.fe.uc.pt

Keywords: Project Analysis and Evaluation; Multicriteria Decision Making; Real Options

ABSTRACT

This paper aims to present an algorithm for an efficient evaluation of very large bicriteria decision trees, considering a specific model. We begin with a presentation of our general approach for representing investment projects using real option decision trees, when time and financial value are considered the relevant criteria. We assume that the decision maker may want to use either the expected value approach or the binomial model for option valuation to aggregate the financial value and we propose an approach for the aggregation of time. We also discuss the identification of non-dominated strategies in such trees. Next, we present a more specific model that allows the use of a set of rules to generate the corresponding tree. The trees generated by the model will usually be very large and calculations may take a long time even within computational systems. With that in mind, we present an algorithm for a faster identification of the non-dominated strategies, without actually building the tree. This algorithm starts with sub-trees corresponding to small project portions (the ones that can be executed by the leaves of the tree), and considers consecutively larger project portions until the non-dominated strategies for the whole tree are identified. We present an illustration example of the use of the algorithm. Then, we compare the performance of an implementation of the algorithm with the performance of an implementation of the basic methodology (that consists on building and evaluating the tree).

1. INTRODUCTION

Decision trees provide a way of representing sequences of decisions and uncertain events through time, so that decisions made today take proper account of what can be done in the future. These characteristics make them particularly useful in the representation and evaluation of investment projects (see Magee, 1964 and Brealey and Myers, 1991). It is now acknowledged that the classical decision tree analysis treats project risk incorrectly (Trigeorgis and Mason, 1987, Trigeorgis, 1996, Godinho and Costa, 1999), so most authors recommend the use of option pricing theory in the evaluation of project decision trees (Trigeorgis and Mason, 1987, Brealey and Myers, 1991, Trigeorgis, 1996, Herath and Park, 1999, Godinho and Costa, 1999, for example).

The traditional use of decision trees in project evaluation only considers the financial perspective. However, there are often certain factors that cannot be incorporated in the financial value of a project and that are very important in deciding whether or not the project should be undertaken. Time is one important criterion that is often overlooked in the project evaluation literature (Godinho and Costa, 1999). In a construction project, for example, there will usually be a deadline. If the company does not meet the deadline, it may have to pay some compensation, and its image may be damaged in a way that is hard to quantify. Other times there may exist some benefits from an early conclusion of the project, like the possibility of undertaking other projects or seizing other opportunities. Competitive interaction will often provide other important reasons to use time as a criterion. In order to deter competitive entries, to gain a competitive advantage or to avoid losses resulting from an early competitive entry, companies may want to undertake a project as soon as possible, while trying to maximise its financial value. In such circumstances, companies may want to use both time and financial value in the definition of their strategies.

The efforts for the use of multiple criteria in project trees evaluation have been based on multi-attribute utility theory (MAUT). Hertz and Thomas (1983) present an overview of some methodologies for the evaluation of multicriteria decision trees, based on utility theory. Godinho and Costa (1999) discuss some drawbacks of these approaches. Smith and Nau (1995) also use utility theory to evaluate project decision trees in incomplete markets.

Godinho and Costa (1999) present a new approach that incorporates both time and financial value in decision trees for the evaluation of investment projects. It focuses on the use of time and financial value, but we think it can be easily extended to other criteria. This new approach allows decision-makers to identify all the non-dominated[1] strategies, letting them use any multicriteria method to choose among them, including methods that do not assume the existence of a utility function. It also allows the decision-maker to use either real option trees or classical decision trees analysis. We outline this approach in section 2, and we follow it in the remainder of this paper.

We acknowledge that this approach will usually lead to very large decision trees, and that the required calculations may take a long time even within computational systems. So, it is useful to develop specific models for particular problems, and algorithms to efficiently identify the non-dominated strategies. Section 3 of this paper presents a particular model that

[1] An alternative is non-dominated if none of the other alternatives is better or equal in all the criteria and better in at least one criterion.

allows the use of some simple rules to build the corresponding tree, based on the general approach that we are following. This model assumes that some different processes may be used to undertake an homogeneous task, each process having a constant cost and requiring a constant time per utilisation, and that there are costs and setup times for changing the process being used. By using a process, the task will advance by one of two different portions in a process utilisation, with corresponding probabilities. Section 4 presents an algorithm that allows a faster identification of the non-dominated strategies, without requiring the definition of the corresponding tree. This algorithm is particularly efficient when the number of non-dominated strategies is small, not only because it cuts the non-interesting branches, but also because it does not even consider most of them 'a priori'. Also, the algorithm is able to avoid the repetition of calculations for similar branches of the tree. This is also an effort in the way of developing models for particular situations and efficient methods for evaluating them. Section 5 presents an illustration example of the use of the algorithm. In section 6 we compare the performance of an implementation of the algorithm with the performance of an implementation of the basic methodology (that consists on building and evaluating the tree). We can conclude that the algorithm performs much better than the basic method, both in terms of speed and in terms of memory usage, when the number of non-dominated strategies is small and the trees are large. Finally, we present our conclusions in section 7. The tables with the most important parameter distributions we used and the most important results are shown in the appendix.

2. THE GENERAL APPROACH FOR THE USE OF TIME AND FINANCIAL VALUE IN PROJECT DECISION TREES

This section presents the general approach for the use of time and financial value in project decision trees that was proposed in Godinho and Costa (1999). We assume that we are maximising financial value in the form of the Net Present Value (NPV) and minimising the time. This approach focuses on the identification of the non-dominated alternatives[2], allowing the decision-maker to choose one of these alternatives using any multicriteria method.

The evaluation of the decision tree will be a two-step process. In the first step, time increments and cash flows (or other value increments) are forwarded to the leaves, in order to calculate the criteria values for each leaf. In the second step, event probabilities are adjusted (if necessary) and the tree is rolled back. The rolling back process will differ from the one usually used, since it must allow the identification of all the non-dominated alternatives, and not only the one best alternative.

The aggregation of criteria values across event nodes is based on the adjustment of probabilities. Time and financial value may follow different aggregation rules, so the adjusted probabilities for these criteria may differ. We may even end up with three different probabilities in the same branch: the basic (initial) probability, a value-adjusted probability and a time-adjusted probability.

The financial value that corresponds to a given event node will always be the sum of the values corresponding to its branches, weighted by their value-adjusted probabilities,

[2] An alternative will be non-dominated if none of the other alternatives has both a shorter or equal time and a larger or equal financial value, with a strict inequality for at least one of these criteria.

regardless of whether classical or real option evaluation are being used. The adjustment of probabilities will depend on the kind of evaluation being used. If, on the one hand, the classical evaluation of decision trees is being used, then there is no need to adjust probabilities (value-adjusted probabilities will be equal to the initial probabilities). Conversely, if the binomial model for option valuation is being used, probabilities must be adjusted to risk-neutral value-adjusted probabilities according to the twin security pricing process[3].

The approach followed for the aggregation of time across event nodes relies on the use of certainty equivalents. We start by asking the decision-maker for a certainty equivalent of a specified uncertain time. Suppose that the situation shown in figure 2.1 is found on the tree being evaluated. Assuming linearity, we can calculate certainty equivalents for all event nodes with the same probabilities. If the decision-maker provides a certainty equivalent of 2.5, then he or she is implicitly adjusting probabilities to (50%, 50%). So, each time the pair of probabilities (60%, 40%) comes up, these will be adjusted to (50%, 50%).

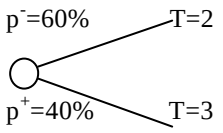

Figure 2.1: Example of a binomial node with uncertain time.

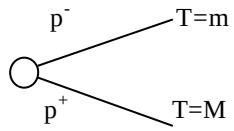

Figure 2.2: A generic binomial node with uncertain time.

If we consider the generic process shown in figure 2.2, with m being the shortest time, M the longest time, p^- and p^+ their corresponding probabilities, and CE the certainty equivalent time provided by the decision-maker, then the adjusted probabilities will be p_T^- =(M-CE)/(M-m) and p_T^+=(CE-m)/(M-m). Assuming linearity, we can replace every instance of (p^-, p^+) by the time-adjusted probabilities (p_T^-, p_T^+). This approach for the aggregation of time across event nodes is quite general, since some common approaches, like the use of the maximum time to completion, the minimum time to completion and the average time to completion are particular cases of this approach.

The evaluation of decision nodes must provide all the non-dominated alternatives. This means that all decisions involving non-dominated alternatives must be made at the root node. Three different rules are used in order to accomplish this.

The first rule is that two consecutive decision nodes are merged. This means that, when there are consecutive decision nodes, we will consider one choice among all the alternatives represented by these nodes, and not consecutive choices among some of those alternatives. Figure 2.3 shows the use of this rule.

[3] In the case of event nodes that correspond to events that do not influence the value of the twin security, these risk-neutral value adjusted probabilities will be the same as the initial probabilities.

This rule allow us to choose directly among alternatives A, B and C, without being forced to choose between A and B before considering alternative C.

The second rule is to eliminate all dominated alternatives in a decision node. This means that we can eliminate one alternative in a decision node if there is another alternative with a larger or equal financial value and a shorter or equal time, given that one of the inequalities is strict. If, in figure 2.3, alternative B were dominated by alternative C (having a smaller or equal financial value and a longer or equal time, one of the inequalities being strict), then it would be eliminated from the decision node.

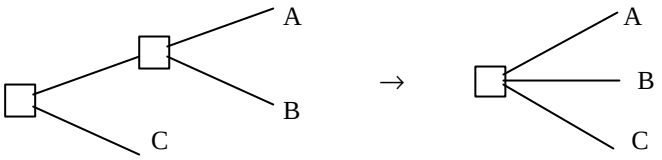

Figure 2.3: Example of the use of rule 1. A, B and C are different alternatives.

The third rule is that if there is an event node before a decision node, then the decision is postponed by considering all possible combinations of decisions. Figure 2.4 shows the use of this rule.

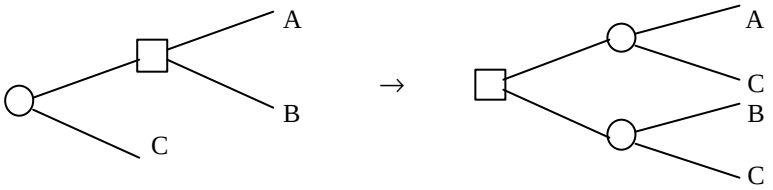

Figure 2.4: Example of the use of rule 3. A, B and C are different alternatives.

Note that, if the lower branch would have a decision node with two alternatives, the total number of alternatives in the resulting tree would be four. This rule may cause a large growth in the number of alternatives. So, it is important to use the second rule in each decision node, in order to prevent the number of alternatives from becoming too large.

By using these rules, all decisions involving non-dominated nodes are delayed until all event nodes are evaluated. Then, any multicriteria method may be used to choose among these non-dominated alternatives.

This section presented the general approach for the use of time and financial value in project decision trees that was proposed in Godinho and Costa (1999). This approach identifies all the non-dominated alternatives, allowing the decision-maker to use any multicriteria method to choose among them. In the next section we will define a particular model based in this general approach.

3. A PARTICULAR MODEL FOR A SPECIFIC CASE

This section presents a particular model, based on the general approach described in the previous section. This particular model is defined for a specific type of problems, and it allows the use of a set of rules to define the corresponding decision tree. This is intended as an effort in the way of developing models for particular situations and efficient methods for evaluating them.

We consider a project that consists on undertaking an homogeneous task. The task may be the construction of a road in an homogeneous landscape or the production of a number of identical items. We define x^0 as the number of "development units" required to complete the task.

We assume that n different processes, P_i, i=1, …, n, may be used to undertake the task. We use event nodes for the representation of the project advance and we consider that we can only change the process being used at the end of a complete process utilisation. Processes may differ by the use of different technology, different human resources, etc. Each process P_i is characterised by:

- a given time, t_i, representing the duration of each utilisation of the process;

- a given cost per utilisation of the process, c_i;

- a set of switching costs, representing the cost of switching to each of the other processes, $c_{i,j}$, j=1, …, n, $j \neq i$ ($c_{i,i}=0$);

- a set of setup times, representing the setup time necessary to switch to each of the other processes, $t_{i,j}$, j=1, …, n, $j \neq i$ ($t_{i,i}=0$);

- a probability distribution for project advance for each utilisation of process P_i, represented by a binomial event node, as shown in figure 3.1.

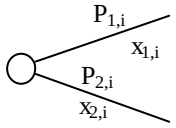

Figure 3.1: Binomial event node representing project advance for each utilisation of process P_i.

$x_{1,i}$ and $x_{2,i}$ ($x_{1,i}>0$, $x_{2,i}>0$) are the possible project advance amounts under process P_i, and $P_{1,i}$ and $P_{2,i}$ are the corresponding probabilities, with $P_{1,i}+P_{2,i}=1$, $P_{1,i}\geq 0$ and $P_{2,i}\geq 0$. Note that this representation assumes the temporal independence of project advances: the probability distribution of project advance under a given process is constant, and consequently independent of previous project advances. We assume, without loss of generality, that $x_{1,i}\leq x_{2,i}$. We will also hereafter assume, for simplicity sake and without loss of generality, that $x^0=1$ and thus $x_{1,i}$ and $x_{2,i}$ can be interpreted as portions of the project. We will use x to represent the unexecuted portion of the project (obviously $x\leq 1$).

This model assumes that the use of the processes is indivisible. This means that we can only change the process being used at the end of a complete utilisation (this is a direct consequence of the use of event nodes for the representation of the project advance). We also assume that even if, at the end of the task, the unexecuted portion of the project (x) is smaller than the project advance in the next use of the process (i.e., $x<x_{b,i}$ for the process i being used and for the considered branch b, b being 1 or 2), the time and cost for a complete utilisation of the process will be required anyway.

According to the general approach that we are following, financial value (in this case replaced by cost) and time are aggregated according to value- and time-adjusted probabilities. We will now discuss the calculation of these probabilities. First we will consider the value-adjusted probabilities, represented by $P^V_{b,i}$, b=1,2, i=1,…,n. The cost per process utilisation is constant for each process, meaning that the cost risk is only related to the project duration risk and to the process choices, and it doesn't seem possible that any financial asset will reflect that risk. The cost risk will thus be unsystematic risk, meaning that no risk premium should be demanded. So, the value-adjusted probabilities will be equal to the initial probabilities, $P^V_{b,i}= P_{b,i}$, b=1,2, i=1,…,n.

Time-adjusted probabilities will be handled differently. For each different pair of initial probabilities occurring in the definition of the processes, the decision-maker will be asked to provide two certainty equivalents, one corresponding to the situation where time is 1 in the upper branch and 0 in the lower, and the other corresponding to the situation where time is 0 in the upper branch and 1 in the lower (see figure 3.2). From these certainty equivalents, two pairs of time adjusted probabilities are calculated, ($P^{T,1}_{1,i}$, $P^{T,1}_{2,i}$) and ($P^{T,2}_{1,i}$, $P^{T,2}_{2,i}$), respectively. The first pair will be used when the time in the upper branch is longer than the time in the lower branch, and the second pair will be used in the opposite situation.

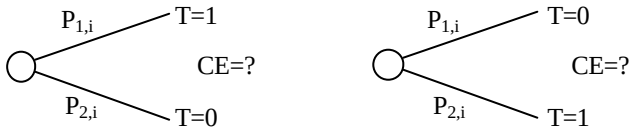

Figure 3.2: Certainty equivalents requested to the decision-maker for the calculation of time-adjusted probabilities.

Some underlying assumptions of this model, particularly the assumptions of binomial event nodes and indivisible process utilisation may, at a first sight, seem too restrictive for the model to be useful. We don't think it will be so. Although in a real life problem it will probably be possible to completely define how much time will a process be used and there will be statistical distributions for project advance, models like this may be used as a proxy to these problems, by defining the processes such that the duration of each utilisation is very small. When very small process durations are considered, the project time and cost distributions (under a given process) will be approximately log-normally distributed, and this log-normal distribution will be an acceptable assumption for many problems. A similar approximation is done in some continuous time real option problems, particularly when multiple options are involved (see, for example, Trigeorgis, 1993).

The assumptions concerning constant costs per utilisation for each process and constant switching costs may also be challenged, since it is reasonable to assume that, at least in projects with a long time span, costs may rise during the project life. We think that, in most situations, it is reasonable to assume that costs rise at a rate similar to the risk-free interest rate. In this case, the present value of the costs will be constant over time, and it will thus be reasonable to use constant real costs to represent the present value of the nominal costs.

The decision trees corresponding to this model will usually be very large. In fact, for each process utilisation and for each branch, there will be a decision node with n branches representing the choice of the process, each of these branches having a binomial decision node. Even in the simple case when we have 3 processes, every non-leaf branch will be divided in 6 different alternatives for each process utilisation. So, particularly when we intend to use this model as a proxy for continuous time problems, having thus to use short process utilisation times, the decision trees corresponding to this model will be very large. Even within computational systems, this trees may require a very large memory space and the corresponding calculations may take a very long time. It is thus very important to develop some algorithms for an efficient identification of the non-dominated strategies. The next section will present an algorithm for an efficient identification of the non-dominated strategies in this particular model.

4. AN ALGORITHM FOR THE GENERATION OF THE NON-DOMINATED STRATEGIES

The previous section presented a particular model based on the general approach that we described in section 2. We argued that the decision trees corresponding to this particular model will usually be very large, and calculations will take a long time even within computational systems. This section will present an algorithm for an efficient generation of the non-dominated strategies corresponding to this particular model. The algorithm takes advantage of the fact that, at least when the number of non-dominated strategies is low, there will be many sub-trees leading to the same set of non-dominated sub-strategies.

Each sub-tree of the tree, starting at a given level, will correspond to the execution of a portion of the project: the portion of the project that was not executed in the previous branches of the tree. For example, consider two processes, P_1 and P_2, with $x_{1,1}=0.1$ and $x_{2,1}=0.15$, for P_1, and $x_{1,2}=0.11$ and $x_{2,2}=0.14$, for P_2. Let us assume that we start with process P_1 and we advance by branch 1 ($x_{1,1}=0.1$). Then we will be left with an unexecuted portion of the project of x=1-0.1=0.9. The next node in this path will be a decision node, and the sub-tree starting with this node will correspond to the execution of a project portion of x=0.9. If in this decision node, we choose to use once again P_1, we will then have an event node that is the root of a sub-tree that corresponds to the execution of a portion x=0.9 of the project, starting with process P_1. This sub-tree will, in practice, only be used to calculate the non-dominated sub-strategies that allow the completion of a portion x=0.9 of the project (in spite of the fact that they are really sub-strategies, we will hereafter call them strategies, for simplicity sake). Now, let us assume that, in another tree path, we have another sub-tree that corresponds to the execution of a portion x=0.9 of the project, starting with process P_1 (it does not really happen in this tree, but let us assume so for now). We will not have to build both sub-trees, because, after the first one is built, we know that the second will lead to the same set of non-dominated strategies. Moreover, the sets of non-dominated strategies that

allow the completion of a given portion x of the project by starting with a given process, will be the same for the values of x belonging to a given interval. This means that another sub-tree, corresponding to the execution of a portion $x'=0.89\neq0.9$ of the project starting with process P_1, may lead to the same set of non-dominated strategies that we have in our previous sub-tree, if both x=0.9 and x'=0.89 belong to the same interval I of values of x for which the non-dominated strategies beginning with P_1 are equal. In this case, we would only have to calculate one sub-tree, since the other would lead to the same non-dominated strategies. In such a situation, representing by I the interval for which the set of non-dominated strategies is common, and i being the number of the first process being used in the sub-trees (in this case, i=1), we would say that S(i,I) was equal to the common set of non-dominated strategies (the notation will be later detailed).

To see clearly that the sets of non-dominated strategies that allow the completion of a given portion x of the project by starting with a given process will be the same for a given interval of values of x, let us once again use the values of the previous example, and let us analyse the lowest level of the tree. If we have already executed a portion of the project that is larger or equal than 0.9, and we decide to use process P_1 after that, we know that, after using P_1 once, the project will be concluded. Since this is true if the unexecuted portion of the project is positive and smaller or equal than 0.1, we can say that S(1,]0,0.1]) is equal to a set with the strategy of using P_1 once. So, when we have an unexecuted portion of the project $x\in$]0,0.1] and we decide to use P_1 next, we know that our only non-dominated strategy will be the strategy belonging to S(1,]0,0.1]). Similarly, for P_2 we can say that S(2,]0,0.11]) is equal to a set with the strategy of using P_2 once. Notice that, if we have an unexecuted portion of the project of $x\in$]0,0.1] and have not yet decided which process to use next, we also know that the only possible non-dominated strategies for that portion of the project will be those in S(1,]0,0.1]) and S(2,]0,0.11]). For larger values of x, we will also have identical sets of non-dominated strategies for intervals of values of x, thus allowing us to avoid the calculation of sub-trees leading to the same non-dominated strategies.

This algorithm begins with the identification of the non-dominated strategies for small project portions. In our previous example, we would start with the calculation of S(1,]0,0.1]) and S(2,]0,0.11]). Then, the non-dominated strategies for smaller project portions are taken into account in the identification of non-dominated strategies for consecutively larger project portions. In our example, the next set S(1,I) would represent the use of process P_1 first and then the use of a non-dominated strategy in the upper branch. Since the non-dominated strategy to be used in the upper branch could be from either S(1,]0,0.1]) or S(2,]0,0.11]), we would generate two different strategies. Notice that a strategy generated in such a manner would be valid for undertaking a project portion smaller or equal than 0.15, because we are not considering the use of another strategy in the lower tree branch. So, we would have S(1,]0.1,0.15]) equal to the set of two strategies that were generated (if they were both non-dominated), or equal to a set with one of those strategies (if the other strategy is dominated by this strategy). We continue until the complete project is considered. The identification of non-dominated strategies for large project portions takes into account the non-dominated strategies for smaller projects portions, allowing the algorithm to identify and disregard the dominated strategies as soon as they come up, and also allowing it to avoid the repetition of identical calculations that is often performed in decision trees, as was previously explained. In fact, the algorithm starts with sub-trees corresponding to small project portions (the ones that can be executed by the leaves of the tree), and considers consecutively larger project portions until the non-dominated strategies for the whole tree are identified.

In order to define the algorithm, we define S(i) as the set of non-dominated strategies that allow the completion of the project starting with the use of process P_i. Each strategy in S(i) is non-dominated in the sense that there is neither a strategy in S(i) nor in any other S(j), j=1, …, n, $j \neq i$, that dominates it. S(i,x) will be the ordered set of strategies that allow the execution of a portion x of the project by starting with the use of process P_i and are not dominated by any other strategy in this set. This means that a strategy in S(i,x) may be dominated by a strategy belonging to S(j,x), $j \neq i$. We consider S(i,x) to be also defined for intervals of values of x. We define that S(i,I)=S if $\forall\ x \in I$, S(i,x)=S.

The n^{th} strategy of S(i,x) will be identified by $s_n(i,x)$. Each strategy $s_n(i,x)$ will have the form $s_n(i,x)=((j_1,y_1), (j_2,y_2),t,c,x_{max})$, where:

· (j_b,y_b) identifies the next strategy (the next sub-branch) to be followed in the upper branch (if b=1) or in the lower branch (if b=2); the next strategy in that branch will be $s_{y_b}(j_b,x-x_{b,i})$ (if $x-x_{b,i}<0$ then $(j_b,y_b)=(0,0)$);

· t is the process time;

· c is the process cost;

· x_{max} is the largest value of x for which the strategy is valid.

The algorithm will successively calculate S(i,I) for the different processes and for intervals I where the set of strategies does not change. It starts by calculating $S(i,]0,x_{1,i}])$ for each process P_i. Then, assuming that x' is the upper limit of the last interval I for which S(i,I) was calculated, and x'' being the next value that may cause a change in the set of non-dominated strategies, the set $S(i,]x',x'']$) will be calculated. These changes in the set S(i,I) are caused by the fact that, when the value x'' rises above a given value, some strategies may no longer be valid for undertaking a portion x'' of the project. Note that a change in the set of non-dominated strategies may be caused either by the upper or by the lower branch. So, although $x_{1,i}$ will always cause the first change in the set of non-dominated strategies[4], $x_{2,i}$ will also cause a change in the set of non-dominated strategies. In order to efficiently identify $S(i,]x',x'']$), its calculation will be based on the results obtained for previous intervals, and also for the other processes.

The next S(i,I) to be calculated always refers to the process P_i for which the calculations are "less advanced", that is, the process P_i for which the last $S(i,]x',x'']$) that was calculated corresponds to a smaller x''. After S(i,1) is calculated for all processes, S(i) will then be defined as the set of strategies belonging to S(i,1) that are not dominated by any strategy belonging to a set S(j,1), $j \neq i$.

The number of intervals I=]x',x''] that correspond to different sets of non-dominated strategies S(i,I) may increase exponentially as the values of x' and x'' increase. This will be particularly true when the number of non-dominated strategies is large. Also, when x is large, the number of different values of x for which S(i,x) is required will be quite small. Consider, as an example, that there are two processes, P_1 and P_2, with possible advance amounts of $x_{1,1}=0.045$ and $x_{2,1}=0.06$ for P_1 and $x_{1,2}=0.055$ and $x_{2,2}=0.07$, for P_2. In order to calculate the non-dominated strategies for these processes, we obviously need to calculate S(i,1) for both processes. The calculation of S(i,1) for any process P_i will require the use of the sets

[4] Remember that in the previous section we defined that $x_{1,i} \leq x_{2,i}$.

$S(j,1-x_{b,k})$ for both processes (j=1 and j=2) and for the advance amounts, $x_{b,k}$, of both processes (k=1 and k=2) in both branches (b=1 and b=2). This means that, in this case, to calculate S(i,1) for any process we need S(j,0.955), S(j,0.945), S(j,0.94) and S(j,0.93), for both processes (j=1 and j=2). While, for values of x between 0.95 and 1, the number of intervals with different non-dominated strategies may be as large as 10 (for each process), it will only be necessary to know the sets of non-dominated strategies for two different values of x, x=1 and x=0.955, in order to calculate the set of non-dominated strategies that lead to project completion.

To prevent a large growth in the number of different intervals I for which S(i,I) is calculated, we define a threshold τ, we calculate all the values of $x \geq \tau$ for which the sets S(i,x) are needed, and we place them in set V. Then, when we calculate the sets of non-dominated strategies for values of x larger than τ, we compare the upper limit, x'', of the interval I=]x',x''] for which the next S(i,I) could be calculated, with the smallest value in the set V that is larger than x' (we will call it v). If v>x'' we know that, for all the values of x in the interval I, S(i,x) will not be necessary for the calculation of S(j,1), for any process. So, we may proceed with the calculation of S(i,v), since the smallest value of x after x' for which the set of non-dominated strategies will be necessary is x=v. In such a situation we will say that S(i,]x',v])=S(i,v), to keep the interval notation, although this set may not be the set of non-dominated strategies for some values of x belonging to that interval. If $x'' \geq v$, we calculate S(i,]x',x'']) as usual. Once again, if, for a previous interval, we had x''<v, it may happen that this set will not be the set of non-dominated strategies for all the values of x belonging to the interval]x',x'']. However, it will always be the set of non-dominated strategies for all the values belonging both to the interval and the set V, and we will keep the interval notation for coherence.

We can now ask which value should the threshold τ have. On the one hand, if τ will have a large value (near 1), the algorithm may calculate many sets of non-dominated strategies, S(i,I), that are not necessary. On the other hand, if τ has a small value, the calculation of the set V may take some unnecessary time, since the reduction in the number of sets of non-dominated strategies may occur only for large values of x. We have no reason to think that there may be a reduction in the number of sets S(i,I) calculated when the considered values of $x \in I$ are smaller that 0.5, since for x<0.5 the number of intervals I will always grow slower than the number of values in V. So, we do not recommend the use of values of τ lower than 0.5. When the number of strategies in S(i,I) grows fast for small values of $x \in I$ (this will happen when the values of $c_{i,j}$ and $t_{i,j}$ are near 0), so will the number of intervals, and it will be useful to use τ=0.5. When the number of strategies in S(i,I) grows slowly (for large values of $c_{i,j}$ and $t_{i,j}$), a larger value of τ may be used. Since the time necessary for the calculation of V will usually be small when compared with required for the calculation of the sets of non-dominated strategies S(i,I), we recommend the use of τ=0.5 in the general case.

For the definition of the algorithm we will use the following notation:

V: set of values $x \geq \tau$ such that S(j,x), j=1,…,n, may be necessary to calculate S(i,1), i=1,…,n.

l_i, i=1,…,n: upper limit of the last interval I for which the set S(i,I) was determined;

lc_i, i=1,…,n: upper limit of the last interval I for which the set S(i,I) was recalculated (notice that lc_i may be different from l_i since sometimes S(i,1) may be kept from the previous iteration);

u: upper limit of the interval I for which S(i,I) is being calculated;

M_i, i=1,…,n: set of sub-branches (j,b) of S(i,I) that will be modified after the last calculation of this set of non-dominated strategies. Each pair (j,b) means that process P_j is used in the upper (b=1) or lower (b=2) branch of the tree; j=0 means that no other process than the initial P_i is used in the considered branch;

$NDB_{b,i}$, b=1,2, i=1,…,n: set of non-dominated sub-branches ((j,y),t,c) for the upper (b=1) or the lower (b=2) branch of the tree corresponding to the last S(i,I). Each ((j,y),t,c) means that the strategy $s_y(j,x-x_{b,i})$ is used in the branch b of the tree, with a total time for that branch equal to t and a total cost equal to c; j=0 means that no other process than the initial P_i is used in the branch b;

$DB_{b,i}$, b=1,2, i=1,…,n: set of dominated sub-branches ((j,y),t,c) for the upper (b=1) or the lower (b=2) branch of the tree corresponding to the last S(i,I);

B_b, b=1,2: set of non-dominated sub-branches ((j,y),t,c) that are not included in $NDB_{b,i}$ and can be used, after the utilisation of the process under consideration, in the upper branch (for b=1) or in the lower branch (for b=2);

D_i, i=1,…,n: set of dominated strategies formed in the tree corresponding to the last S(i,I) by using branches belonging to $NDB_{b,i}$, b=1,2 (although none of the branches is dominated, the strategy is dominated);

a(i,j,b), i=1,…,n, j=0,1,…,n, b=1,2: next value for which a change in S(i,I) may occur due to the use of process P_j after the initial process P_i in the upper (b=1) or lower branch (b=2); if, for j=0, $a(i,0,b)<+\infty$, then the next change in S(i,I) will occur before the first choice of process in the corresponding branch;

· $x_{min}(s)$, for a given strategy s belonging to S(i,x), for any given i=1,…,n and $x\in]0,1]$: the lower limit of the first interval I for which the strategy was placed in the set of non-dominated strategies S(i,I);

E_i, i=1,…,n: ordered set of the values that caused changes in S(i,I);

RE: set of strategies that will be removed from S(i,I) in the current iteration;

ALL: set of all strategies, both dominated and non-dominated, that were generated for the calculation of the current S(i,I).

By using this notation, the algorithm can be broadly defined as follows:

1. For each process P_i, i=1,…,n, let $S(i,]0,x_{1,i}])=\{((0,0),(0,0),t_i,c_i,x_{1,i})\}$, $x_{min}(s_1(i,]0,x_{1,i}])=0$, $lc_i=x_{1,i}$, $l_i=x_{1,i}$ and initialise $E_i=\{x_{1,i}\}$, $D_i=\varnothing$, $M_i=\varnothing$, $NDB_{b,i}=\{((0,0),t_i,c_i)\}$, $DB_{b,i}=\varnothing$, $a(i,0,b)=x_{b,i}$, b=1,2, and $a(i,j,b)=+\infty$, j=1,…,n, b=1,2.

2. Calculate V, using the auxiliary set V_1: start with $V=\varnothing$ and $V_1=\{1\}$ and, while $V_1\neq\varnothing$, remove from V_1 the largest value $x\in V_1$, add x to V and add to V_1 the values $x_1=x-x_{b,i}\geq\tau$, b=1,2, i=1,…,n.

3. If S(i,1) is calculated for all the processes, then go to step 18.

4. Let P_i be the process with a smaller l_i.

5. For all $a(i,j,b) \leq l_i$, add (j,b) to M_i and update a(i,j,b): if $j \neq 0$, let $a(i,j,b)=x_{b,i}+\min\{e \in E_j: e>l_i-x_{b,i}\}$; otherwise let $a(i,0,b)=+\infty$ and, additionally, let $a(i,k,b)=x_{b,i}+x_{b,k}$, for all k=1,…,n

6. Let u be the minimum value of a(i,j,b) for P_i and let $I=]l_i,u]$.

7. If $l_i \geq \tau$ and $\min\{v \in V: v>l_i\}>u$ then let $i=]l_i,v]$, let u=v and update M_i and a(i,j,b): for all $a(i,j,b)<v$, add (j,b) to M_i and, if $j \neq 0$, let $a(i,j,b)=x_{b,i}+\min\{e \in E_j: e \geq u-x_{b,i}\}$; otherwise let $a(i,0,b)=+\infty$ and, additionally, let $a(i,k,b)=x_{b,i}+\min\{e \in E_j: e \geq u-x_{b,i}\}$, for all k=1,…,n

8. Define RE as the set of strategies that belong to $S(i,l_i)$ and that are not valid for undertaking a project portion belonging to interval I; note that these will be the set of the strategies containing sub-branches that belong to M_i whose $x_{max}<u$.

9. If $RE=\emptyset$ the let $S(i,I)=S(i,l_i)$, remove l_i from E_i and go to step 16.

10. Calculate B_b, b=1,2, the set of non-dominated sub-branches that are not included in either $NDB_{b,i}$ or $DB_{b,i}$ and can be used in the upper branch (for b=1) or in the lower branch (for b=2); the calculation of B_b, b=1,2, involves the use of the set M_i and the x_{min} of the strategies: the sub-branches placed in B_b will be the strategies s belonging to $S(j,u-x_{b,i})$ such that (j,b) belongs to M_i and $x_{min}(s) \geq lc_i$ (the strategies s with $x_{min}(s)<lc_i$ already belong to either $NDB_{b,i}$ or $DB_{b,i}$).

11. Update the sets $DB_{b,i}$ and $NDB_{b,i}$, using B_b: this means removing from $DB_{b,i}$ and $NDB_{b,i}$ the sub-branches that can no longer be used to build strategies to undertake a portion u of the projects (those corresponding to strategies with $x_{max}<u-x_{b,i}$), moving from $DB_{b,i}$ to $NDB_{b,i}$ the sub-branches that are no longer dominated, placing in $DB_{b,i}$ the sub-branches in B_b that are dominated and in $NDB_{b,i}$ those that are not dominated.

12. Define the set ALL of all the strategies belonging to D_i that are still valid for undertaking a project portion belonging to interval I (those with $x_{max} \geq u$).

13. Use the sub-branches belonging to the sets B_b and $NDB_{b,i}$, b=1,2, to generate the new strategies that may be considered for inclusion in S(i,I), and add these strategies to the set ALL; these will be the strategies that start by using process P_i, have an upper sub-branch belonging to either B_1 or $NDB_{b,1} \backslash B_1$, and have a lower sub-branch belonging to either B_2 or $NDB_{b,2} \backslash B_2$ (the combination of an upper sub-branch belonging to $NDB_{b,1} \backslash B_1$ and a lower sub-branch belonging to $NDB_{b,2} \backslash B_2$ does not have to be considered, since such strategies will have been generated in previous iterations and already belong to either S(i,I) or D_i).

14. Let $S(i,I)=S(i,l_i)$, remove from S(i,I) the strategies belonging to RE, add to S(i,I) all the non-dominated strategies belonging to ALL and let D_i be the set of all dominated strategies belonging to ALL; for all the strategies s that were included in S(i,I) and do not belong to $s(i,l_i)$, let $x_{min}(s)=l_i$.

15. Let $lc_i=u$ and $M_i=\emptyset$.

16. Let $l_i=u$ and add u to E_i.

17. Return to step 3.

18. For each process P_i, let S(i) be the set of strategies belonging to S(i,1) that are not dominated by any strategy belonging to other S(j,1), $j \neq i$.

19. Stop.

This section described an algorithm for an efficient generation of the non-dominated strategies corresponding to this model presented in the previous section. The next section will present an illustration example of the use of the algorithm.

5. AN ILLUSTRATION EXAMPLE

This section will present an illustration example of the use of the algorithm described in the previous section. We will consider two process, P_1 and P_2, whose characteristics are shown in figure 5.1.

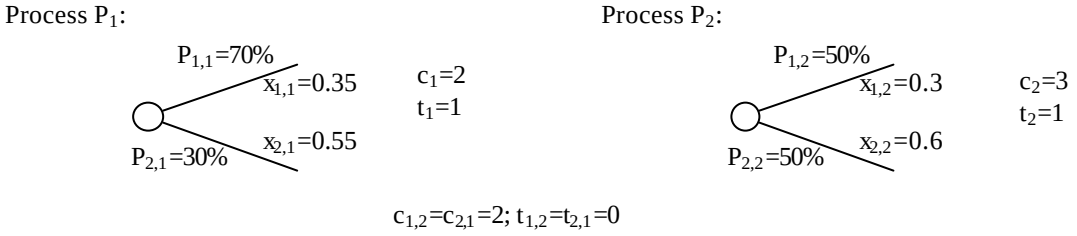

Figure 5.1: Characteristics of processes P_1 and P_2.

In order to calculate the time-adjusted probabilities, the decision-maker would be required to provide certainty equivalents for three different situations. We assume the certainty equivalents shown in figure 5.2.

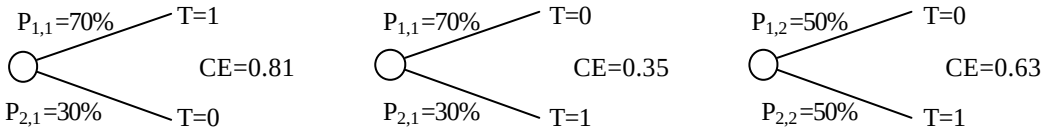

Figure 5.2: Certainty equivalents assumed for the example.

The certainty equivalents will be used for the calculation of the time-adjusted probabilities. We will get $P^{T,1}{}_{1,1}$=81%, $P^{T,1}{}_{2,1}$=19%, $P^{T,2}{}_{1,1}$=65%, $P^{T,2}{}_{1,1}$=35%, $P^{T,1}{}_{1,2}$= $P^{T,2}{}_{2,2}$=63% and $P^{T,1}{}_{2,2}$=$P^{T,2}{}_{1,2}$=37%. We can thus define the time aggregation functions, TAF(t_1,t_2,i), for the two processes, P_1 (i=1 in the function) and P_2 (i=2 in the function). We get:

$$TAF(t_1,t_2,1)=\begin{cases}0.81\cdot t_1+0.19\cdot t_2,\text{if} & t_1\geq t_2\\ 0.65\cdot t_1+0.35\cdot t_2,\text{if} & t_1<t_2\end{cases};\qquad(5.1)$$

$$TAF(t_1,t_2,2)=\begin{cases}0.63\cdot t_1+0.37\cdot t_2,\text{if} & t_1\geq t_2\\ 0.37\cdot t_1+0.63\cdot t_2,\text{if} & t_1<t_2\end{cases};\qquad(5.2)$$

We can also define the value aggregation functions, $VAF(t_1,t_2,i)$, for the two processes, P_1 (i=1 in the function) and P_2 (i=2 in the function):

$$VAF(c_1,c_2,1)=0.7\cdot c_1+0.3\cdot c_2;\qquad(5.3)$$

$$VAF(c_1,c_2,2)=0.5\cdot c_1+0.5\cdot c_2;\qquad(5.4)$$

The complete decision tree for this example would have 63 nodes and 64 leaves. Part of the decision tree is shown in figure 5.3. The complete tree can be easily obtained by combining the processes through the use of the same rules.

We will now use the algorithm described in the previous section to identify the non-dominated strategies without building the tree. We will use τ=0.5.

In step 1, we make the initialisations. The initial sets S(i,I), that correspond to the values of $x\in I$ that only require the process P_i to be used once, and the initial values of l_i, D_i, $NDB_{b,i}$ and $DB_{b,i}$ are shown in table 5.1.

Table 5.1. Initialisations

	I=1	i=2
S(1,]0,0.35])	{((0,0),(0,0),1,2,0.35)}	
S(2,]0,0.3])		{((0,0),(0,0),1,3,0.3)}
lc_i	0.35	0.3
l_i	0.35	0.3
M_i	∅	∅
E_i	{0.35}	{0.3}
D_i	∅	∅
$NDB_{1,i}$	{((0,0),1,2)}	{((0,0),1,3)}
$NDB_{2,i}$	{((0,0),1,2)}	{((0,0),1,3)}
$DB_{1,i}$	∅	∅
$DB_{2,i}$	∅	∅

We define the set of values belonging to a(i,j,b), for i=1,2, as a matrix A(i), with the values of j in the rows and the values of b in the columns. We get:

$$A(1)=\begin{bmatrix}0.35 & 0.55\\ +\infty & +\infty\\ +\infty & +\infty\end{bmatrix}\qquad(5.5)$$

$$A(2) = \begin{bmatrix} 0.3 & 0.6 \\ +\infty & +\infty \\ +\infty & +\infty \end{bmatrix} \qquad (5.6)$$

In (5.5) and (5.6), $a(i,0,b)=x_{b,i}$, for i=1,2, b=1,2, meaning that first change in S(i,I), i=1,2, will be due to the first choice of process in either the upper or the lower branch of the tree. Since the next change in S(i,I), i=1,2, will be due to the first choice of process in one the branches, we cannot yet consider the sub-branches of either the upper or the lower branch of the tree, so the other elements of A(i), i=1,2, are $+\infty$.

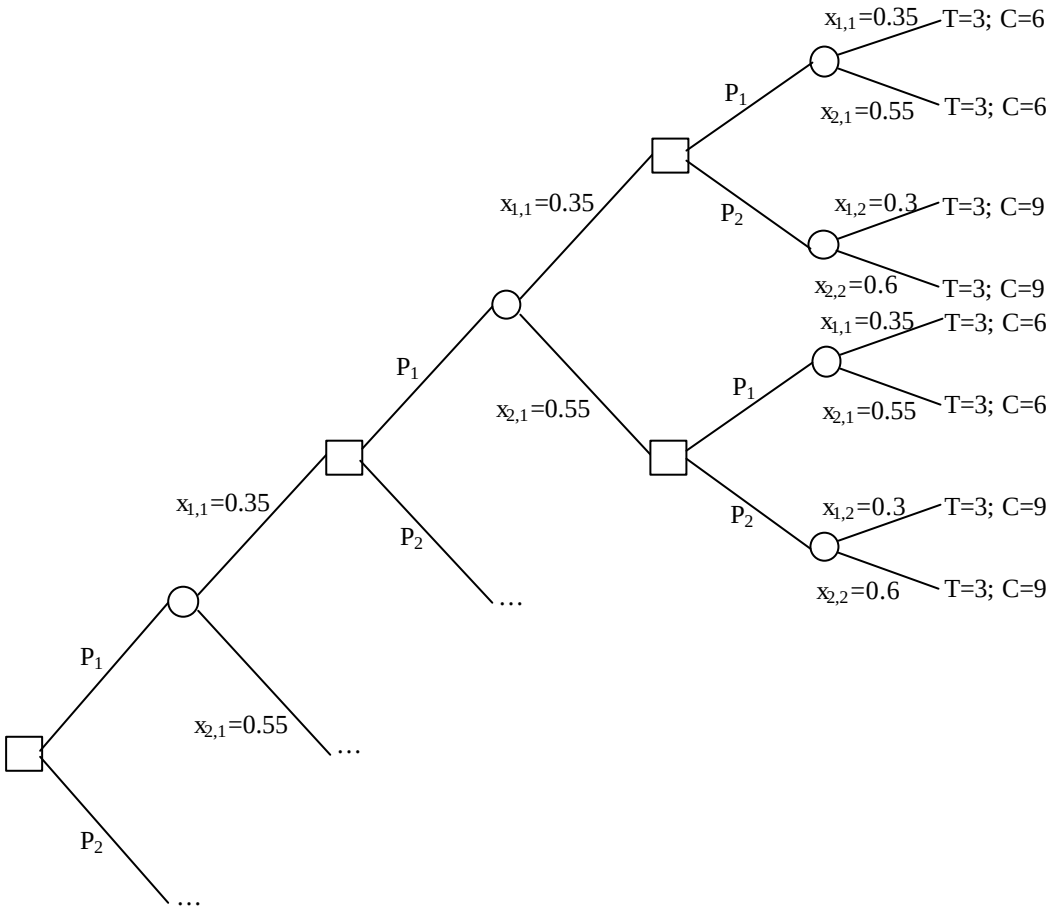

Figure 5.3: Part of the decision tree of the example.

Then, we calculate the set V, using an auxiliary set V_1. We start with V=∅ and V1={1}. Then we move the largest value, x=1, from V_1 to V, and we add to V_1 all the values $x_1=x-x_{b,i}$, b=1,2 and i=1,2, that are larger or equal than $\tau=0.5$. The values of $x-x_{b,i}$ will be 0.7, 0.65, 0.45 and 0.4. Since both the values 0.4 and 0.45 are smaller than τ, we will have $V_1=\{0.7,0.65\}$, and V={1}. Next we move, first the value 0.7 and then the value 0.65, from

V_1 to V. In both cases all the values of $x-x_{b,i}$ are smaller than τ, so no new values will be included in V_1. After the value 0.65 is moved from V_1 to V, V_1 will be empty, so the final set V is V={1,0.7,0.65}.

In step 4 we define that the next process to be considered is P_2, since it has got the lowest l_i. Since the only value in A(2) that is smaller or equal than l_2=0.3 is a(2,0,1)=0.3, we let M_2={(0,1)} (meaning that the first change in S(2,I) is due to the first choice of process in the upper tree branch) and update A(2). We have now

$$A(2) = \begin{bmatrix} +\infty & 0.6 \\ 0.65 & +\infty \\ 0.6 & +\infty \end{bmatrix} \tag{5.7}$$

Let us now examine the meaning of A(2). a(2,0,1)=+∞ means that the next change of strategy in S(2,I) will not occur before the first choice of process is made in the upper branch, while a(2,0,1)=0.55 means that the next change of strategy in S(2,I) may be due to the first choice of process in the lower branch, that will occur for x=0.6. Since the next change in S(2,I) will either occur before the first choice of process in the lower branch or due to that choice of process, we cannot yet consider the sub-branches of the lower branch of the tree, so a(1,1,2)=a(1,2,2)=+∞. a(1,1,1)=0.65 represents the next possible change in S(2,I) due to the use of process P_1 in the upper branch. When P_2 is the first process to be used, the first change in S(2,I) will occur for x=0.3. Since the first possible change in S(1,I) will occur for x=0.35, the change due to the use of P_1 after P_2 in the upper tree branch will occur for x=0.3+0.35=0.65, so a(2,1,1)=0.65. The same reasons let us calculate a(2,2,1)=0.3+0.3=0.6.

The next possible change in S(2,I) may occur for u=0.6, since 0.6 is the smallest value belonging to A(2). This means that I=]0.3,0.6]. The only strategy belonging to S(2,0.3), that is ((0,0),(0,0),1,3,0.3), will not be valid for S(2,]0.3,0.6]), since the value x_{max} for the strategy (0.3) is smaller than the values in the interval, so RE={((0,0),(0,0),1,3,0.3)}.

The possible sub-branches that can be used in the upper tree branch are ((1,1),2,7) and ((2,1),2,6). Since the former sub-branch is dominated by the latter, B_1={((2,1),2,6)}, and we have $DB_{1,2}$={((1,1),2,7)} and $NDB_{1,2}$={((1,1),2,6)}. There will not be any change in the lower branch, since for $x \in$]0.3,0.6] there is not any choice of process in the lower branch, so $B_2=\varnothing$, and we have no changes in $NDB_{2,2}$ and $DB_{2,2}$.

Since D_i is initially empty, the set ALL will only contain the new strategies generated from sub-branches belonging to B_b and $NDB_{b,2}$, b=1,2. $B_2=\varnothing$, so the only new strategies will be generated by the combination of sub-branches belonging to B_1 with sub-branches belonging to $NDB_{2,2}$. We get one new strategy, ((2,1),(0,0),1.63,4.5,0.6), and ALL={((2,1),(0,0),1.63,4.5,0.6)}. The only strategy in S(2,0.3) will not be valid for undertaking a project portion belonging to]0.3,0.6], so it will not be included in S(2,]0.3,0.6]). S(2,]0.3,0.6]) will thus include the non-dominated strategies of ALL, and D_2 will include the dominated strategies, so $D_2=\varnothing$ and

$$S(2,]0.3,0.6]) = \{((2,1),(0,0),1.63,4.5,0.6)\} \tag{5.8}$$

Since the strategy ((2,1),(0,0),1.63,4.5,0.6) is now included for the first time in a set S(2,I), we have x_{min}(((2,1),(0,0),1.63,4.5,0.6))=0.3. We make $l_2=lc_2$=0.6, and E_2={0.3,0.6}

The next iterations are similar, and provide the following sets of strategies:

$S([1,]0.35,0.55])=\{((1,1),(0,0),1.81,3.4,0.55)\}$ (5.9)

$S([1,]0.55,0.65])=\{((1,1),(1,1),2,4,0.7)\}$ (5.10)

$S([2,]0.6,0.65])=\{((1,1),(2,1),2,6.5,0.65)\}$ (5.11)

At this point we have $l_1=l_2=0.65$. Since $l_1=l_2$, we may choose either P_1 or P_2 for the next iteration. We choose P_1, and get $M_1=\{(2,1)\}$ and

$$A(1)=\begin{bmatrix} +\infty & +\infty \\ 0.7 & 0.9 \\ 0.95 & 0.85 \end{bmatrix}. \quad (5.12)$$

So, u=0.7 and I=]0.65,0.7]. We now have $l_1 \geq \tau$, so we will now compare u with the smallest value in V that is largest than $l_1=0.5$. That value is 0.7, and it is equal to our initial u, so we will proceed without changing u and A(1). Since the only strategy belonging to S(1,0.65) is still valid for undertaking a project portion in the interval]0.65,0.7] (its x_{max} is 0.7), we have $RE=\varnothing$, $S(1,]0.65,0.7])=S(1,0.65)=\{((1,1),(1,1),2,4,0.7)\}$ and $l_1=0.7$. The value $x_{min}(((1,1),(1,1),2,4,0.7))=0.55$ will not be altered, since it always remains the same after the strategy is included in a S(2,I) set for the first time.

In the next iteration we get

$S(2,]0.65,0.85])=\{((2,1),(2,1),2.4,6.25,0.9)\}$ (5.13)

At this point, the smallest l_i is $l_1=0.7$, so we choose P_1. We will have $M_1=\{(1,1);(2,1)\}$

$$A(1)=\begin{bmatrix} +\infty & +\infty \\ 0.9 & 0.9 \\ 0.95 & 0.85 \end{bmatrix}. \quad (5.14)$$

The lowest value in A(1) is now 0.85, so u=0.85. Since $l_1 \geq \tau$, we will now compare u with the smallest value in V that is largest than $l_1=0.7$. That value is 1, meaning that we can now calculate S(1,]0.7,1]). We get, in the next iterations,

$S(1,]0.7,1])=\{((1,1),(1,1),2.96,5.82,1.05);((1,1),(2,1),2.93,6.75,1.05)\}$ (5.15)

$S(2,]0.85,1])=\{((1,1),(2,1),2.86,8.25,1)\}$ (5.16)

Now, we go to step 16. Since none of the strategies belonging to S(i,[1,1]), i=1,2, is dominated by a strategy belonging one of these sets, they are all non-dominated strategies for the project. So:

$S(1)= S(1,]0.7,1])=\{((1,1),(1,1),2.96,5.82,1.05);((1,1),(2,1),2.93,6.75,1.05)\}$ (5.17)

$S(2)= S(2,]0.85,1])=\{((1,1),(2,1),2.86,8.25,1)\}$ (5.18)

We can now build the trees corresponding to these non-dominated strategies. We show these trees in figure 5.4.

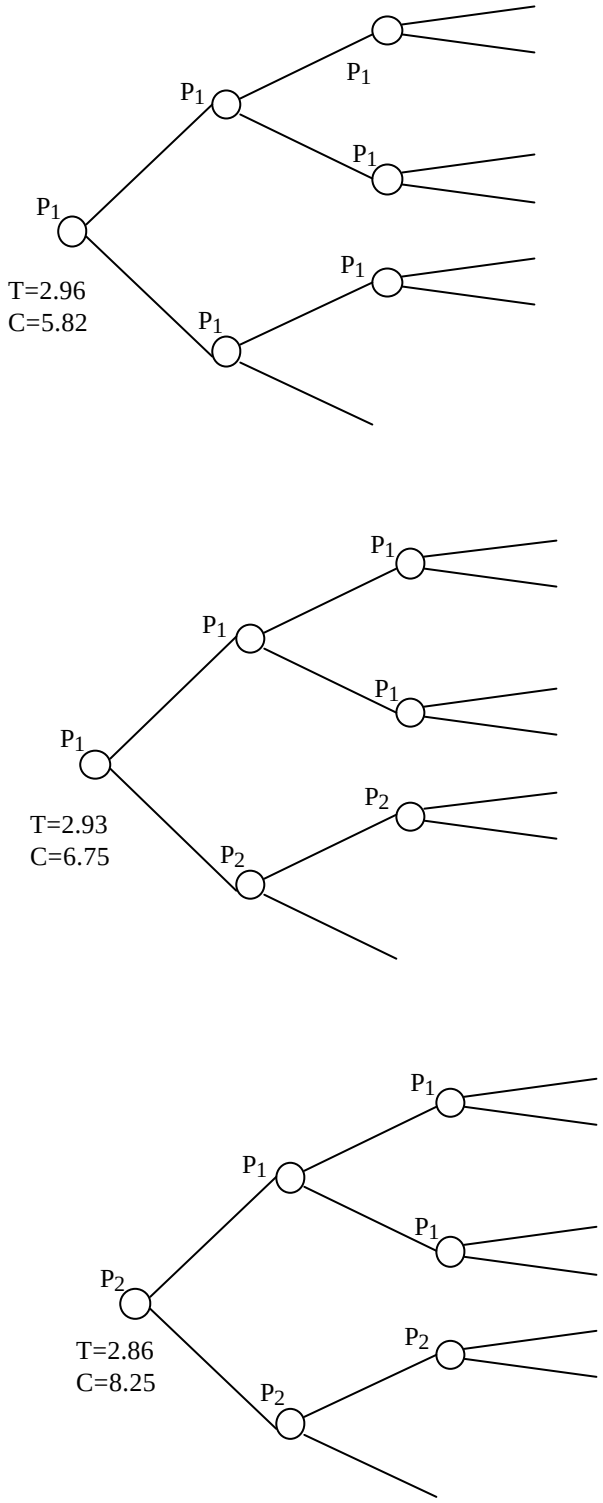

Figure 5.4: Non-dominated strategies of the example

This section presented an illustration example of the algorithm. In the next section we present the results of some tests to the implementation of the algorithm.

6. SOME TESTS TO THE ALGORITHM

This section presents the results of some tests to the performance of the algorithm. We made an implementation of the algorithm and an implementation of the basic methodology (building and rolling back the whole tree), and we describe our most important results in this section.

The implementations were made in Borland Delphi 4, and the tests were performed in a 350MHz Pentium II Personal Computer with 64MB RAM memory and about 1GB of disk space being used as virtual memory.

For these tests, some sequences of files were generated. In every case, the first element of the sequence is generated by defining some parameters as constants, and the remaining parameters as samples of given uniform distributions. The others elements of each sequence are defined through sequential changes in given parameters.

For each set of parameter distributions, 20 sequences were generated. In order to capture the typical behaviour of both the algorithm and the basic methodology for each parameter distribution, the two highest times and the two lowest times were removed, and the average of the remaining 16 times was calculated. The tables with the most important parameter distributions used and the most important results are shown in the appendix. We broadly call to the set of sequences generated for each set of parameter distributions, a "sequence". Each table in the appendix has the same number as its corresponding sequence.

In all the tests we have used τ=0.5. We always generate only the different non-dominated strategies (alternative strategies with identical cost and time are not generated). Most sequences were generated using 2 processes, and only sequence 6 used 3 different processes.

Since the probabilities will not, in general, have a significant effect in either the size of the trees or the number of non-dominated strategies, we always used the same distributions for the probabilities: the basic probabilities for both branches belong to [0.49,0.51], the probability for the branch with the longer time belong to [0.55,0.6], and the probability for the branch with the shorter time belong to [0.4,0.45].

In order to prevent any one process from dominating the other, we always used one process with a shorter duration and a higher cost than the other. When two processes with similar advance amounts were used, the first process has t_1=1 and c_1=2, and the second process has t_2=2 and c_2=1.

The advance amounts are used to increase or decrease the size of the complete tree (larger advance amounts correspond to smaller trees), and the switching costs and setup times are used to increase or decrease the number of non-dominated strategies (larger switching costs or setup times will usually lead to fewer non-dominated strategies).

In the first tests we used similar advance amounts for both branches of each process, and for both processes. We used values of $x_{b,i}$, i=1,2, b=1,2, belonging to [0.09,0.1] (sequence

1) and [0.08,0.085] (sequence 2). The switching costs and setup times for the first element of the sequences were chosen such that there would only be a very small number of non-dominated strategies. In order to achieve that, we have chosen values near $1/x_{b,i}$ for these parameters. Then, we decreased the values of these parameters in order to have an increasing number of non-dominated strategies.

In sequence 1 the algorithm performs better when the number of non-dominated is smaller than 1000, and worse otherwise. For this sequence, the basic method always takes a minimum time (about 3.6 seconds), no matter how small is the number of non-dominated alternatives, while the algorithm takes a time that is indistinguishable from 0 for a small number of non-dominated strategies (less than 174). This is because the basic method always has to build the tree, that has a reasonable size.

For sequence 2, we were unable to run the basic method, since the available disk and memory space (about 1GB) was insufficient to build and evaluate the tree. However, we were still able to run the algorithm. This seems to show that the algorithm is more efficient in memory usage terms than the basic method.

Next we still used processes whose $x_{b,i}$ were similar, but differed between the upper and the lower branch. In sequence 3 we used $x_{1,i} \in [0.14,0.15]$ and $x_{2,i} \in [0.19,0.2]$. Once again we started by defining switching costs and setup times such that there would only be a very small number of non-dominated strategies, and then we decreased the values of these parameters in order to have an increasing number of non-dominated strategies. The performance differences between the algorithm and the basic method are never large in this sequence. However, the algorithm performs better than the basic method, except when we have 285 non-dominated strategies.

In sequence 4 we used processes with different $x_{b,i}$: $x_{1,1} \in [0.07,0.08]$ and $x_{2,1} \in [0.09,0.1]$, for P_1, and $x_{1,2} \in [0.105,0.12]$ and $x_{2,2} \in [0.135,0.15]$, for P_2. Since the advance amounts of the two processes are quite different (the project will advance much more in an utilisation of P_2 than in an utilisation of P_1), we tried to prevent one process from being much better than the other by adjusting the corresponding durations and costs. So, we defined $t_1=1$ and $c_1=2$, for P_1, and $t_1=3$ and $c_2=1.5$ for P_2. We started with large setup times and switching costs, to get a small number of non-dominated strategies, and decreased these parameters to increase the number of non-dominated strategies.

In this sequence, the algorithm performance seems to be worse than the performance of the basic method when the number of non-dominated strategies is large. For a small number of non-dominated strategies, the algorithm always performs better than the basic method. Once again, the basic method always takes a minimum time (about 1.9 seconds), no matter how small is the number of non-dominated alternatives, while the algorithm takes a time that is indistinguishable from 0 for a small number of non-dominated strategies.

In sequence 5 we analyse the performances of the algorithm and the basic method when the size of the tree increases and the setup costs and switching times also increase, keeping the number of non-dominated strategies very small (equal to 2). In these sequence, $x_{b,i}$ initially belongs to [0.25,0.26], for i=1,2 and b=1,2, and is decreased along the sequence.

For this sequence, the algorithm performs much better than the basic method. The algorithm times only start to be distinguishable from 0 when $x_{b,i} \in [0.0311,0.0323]$, while the basic method times are already distinguishable from 0 for $x_{b,i} \in [0.1395,0.1450]$. Also, we

were only able to run the basic method until the tree size reached about 11 million nodes ($x_{b,i} \in [0.0846,0.0879]$), and after that we ran out of disk and memory space. However, we were able to run the algorithm until the last values we tried ($x_{b,i} \in [0.0028,0.0029]$). This clearly shows, once again, that, when the number of non-dominated strategies is small (because of large setup times and switching costs), the algorithm is clearly faster and much more efficient in memory usage terms than the basic method.

Finally, we used a sequence with 3 processes, sequence 6. We used $x_{b,i}$, i=1,2,3, b=1,2, belonging to [0.19,0.2], and $t_1=1$ and $c_1=3$, for P_1, $t_2=2$ and $c_2=2$, for P_2 and $t_3=3$ and $c_3=1$, for P_3. The values of the switching costs and setup times were initially set to 5 and then decreased. The results were consistent with the results we had for 2 processes: the algorithm times are slightly lower for a small number of non-dominated strategies, and slightly higher for a large number of non-dominated strategies.

So, we reached two general conclusions. First, the algorithm shows no clear improvement over the basic method when the number of non-dominated strategies is large, and many times it even seems to perform slightly worse in these situations. Second, the algorithm performs much better than the basic method, both in memory usage terms and in terms of time, when the number of non-dominated strategies is small and the trees are very large. Other tests were performed, both for situations similar to those presented but using different parameters and for other different situations .The results of those tests always supported these conclusions. The new situations considered included:

- increasing the tree size and simultaneously increasing the number of non-dominated strategies at different (slower and faster) rates;

- considering setup times different from the switching costs, and considering asymmetric setup times and switching costs for the projects (the cost and time necessary to switch from P_i to P_j was made different from the cost and time necessary to switch from P_j to P_i, for $j \neq i$).

So, we can say that, while not being able to generate a large number of non-dominated strategies in a short time, the algorithm seems very well fit to be used in some interactive decision methods. A possible approach would be the generation of only a few non-dominated strategies, and asking the decision-maker to compare them or choose some of them. According to the answer of the decision-maker, a new set with some non-dominated strategies would be generated and the decision maker would be questioned again. The process would be repeated until we got a set that surely included the non-dominated strategies preferred by the decision-maker.

This section presented the more important results of some tests that were made in order to assess the performance of the algorithm. The next section presents our conclusions.

7. CONCLUSIONS

This paper focused on the use of two criteria, time and financial value, in project decision trees, and on an efficient identification of the non-dominated strategies for a specific model, both in terms of speed and memory resources.

We started by presenting the general approach for representing investment projects using decision trees, when time and financial value are considered the relevant criteria. This general approach is based on Godinho and Costa (1999), and it focuses on the identification of the non-dominated strategies. It allows the decision maker to use either the expected value approach or the binomial model for option valuation to aggregate the financial value, and suggests a specific approach for the aggregation of time.

Next, we presented a more specific model that allows the use of a set of rules to generate the corresponding decision tree. This model assumes that some different processes may be used to undertake an homogeneous task, each process having a constant cost and requiring a constant time per utilisation, and that there are costs and setup times for changing the process being used. By using a process, the task will advance by one of two different portions, with corresponding probabilities. The decision trees generated by this model will usually be very large and calculations may take a long time even within computational systems. With that in mind, we present an algorithm for a faster identification of the non-dominated strategies without requiring the definition of the corresponding decision tree. This algorithm is particularly efficient when the number of non-dominated strategies is small, not only because it cuts the non-interesting branches, but also because it does not even consider most of them 'a priori'. Also, the algorithm is able to avoid the repetition of calculations for similar branches of the tree. This intends to be an effort in the way of developing models for particular situations and efficient methods for evaluating them. We present an illustration example of the use of the algorithm.

Then, we compare an implementation of the algorithm with an implementation of the basic method (that consists on building and then evaluating the tree). We reach two conclusions. First, the algorithm shows no clear improvement over the basic method when the number of non-dominated strategies is large, and many times it even seems to perform slightly worse in these situations. Second, the algorithm performs much better than the basic method, both in memory usage terms and in terms of time, when the number of non-dominated strategies is small and the trees are very large. So, we can say that, while not being able to generate a large number of non-dominated strategies in a short time, the algorithm seems very well fit to be used in some interactive decision methods. A possible approach would be the generation of only a few non-dominated strategies, and asking the decision-maker to compare them or choose some of them. According to the answer of the decision-maker, a new set with some non-dominated strategies would be generated and the decision maker would be questioned again. The process would be repeated until we got a set that surely included the non-dominated strategies preferred to the decision-maker.

APPENDIX – TABLES WITH THE TESTS RESULTS

This appendix provides the most important results of the tests described in section 6.

The implementations were made in Borland Delphi 4, and the tests were performed in a 350MHz Pentium II Personal Computer with 64MB RAM memory and about 1GB of disk space being used as virtual memory.

For these tests, some sequences of files were generated. In every case, the first element of the sequence is generated by defining some parameters as constants, and the remaining parameters as samples of given uniform distributions. The others elements of each sequence are defined through sequential changes in given parameters. For each set of parameter distributions, 20 sequences were generated. The two highest times and the two lowest times were removed, and the average of the remaining 16 times was calculated. The values shown in the tables are the average values, for these 16 sequence elements, of the algorithm time, basic method time, number of non-dominated strategies and size of the tree (in event nodes). We broadly call to the set of sequences generated for each set of parameter distributions, a "sequence". Each table has the same number as its corresponding sequence.

In all the tests we have used τ=0.5. We always generate only the different non-dominated strategies (alternative strategies with identical cost and time are not generated). The process parameters are shown in the tables or in the table legends, except the probabilities. The probabilities we used were $P_{1,i} \in [0.49,0.51]$, $P_{2,i}=1-P_{1,i}$, $P^{T,1}{}_{1,i} \in [0.55,0.6]$, $P^{T,1}{}_{2,i}=1-P^{T,1}{}_{1,i}$, $P^{T,2}{}_{1,i} \in [0.4,0.45]$ and $P^{T,2}{}_{2,i}=1-P^{T,2}{}_{1,i}$, for all the processes P_i and all the sequences.

Table A.1 - Sequence 1

Parameters $t_{1,2}=t_{2,1}=c_{1,2}=c_{2,1}$	Algorithm Time (sec.)	Basic method Time (sec.)	Non-dominated strategies	Size of the tree (event nodes)
10.0	0.00	3.57	2	2797633
9.0	0.00	3.56	5	2796211
8.0	0.00	3.58	14	2797633
7.0	0.00	3.71	55	2796210
6.0	0.12	3.68	197	2796211
5.5	3.76	6.56	470	2797624
5.0	5.81	8.19	569	2797624
4.5	77.68	75.84	1151	2797633
4.0	137.98	126.85	1268	2797633

Other parameters: $x_{b,i} \in [0.09,0.1]$, i=1,2, b=1,2; $t_1=1$, $c_1=2$, $t_2=2$, $c_2=1$.

Table A.2 - Sequence 2

Parameters $t_{1,2}=t_{2,1}=c_{1,2}=c_{2,1}$	Algorithm Time (sec.)	Basic method Time (sec.)	Non-dominated strategies	Size of the tree (event nodes)
12.0	0.00	-	3	20000000*
10.0	0.00	-	16	20000000*
9.0	0.00	-	40	20000000*
8.5	0.03	-	88	20000000*
8.0	0.07	-	117	20000000*
7.5	0.53	-	228	20000000*
7.0	1.43	-	314	20000000*
6.5	8.72	-	586	20000000*
6.0	23.12	-	772	20000000*
5.5	149.11	-	1373	20000000*

Other parameters: $x_{b,i} \in [0.08,0.085]$, i=1,2, b=1,2; $t_1=1$, $c_1=2$, $t_2=2$, $c_2=1$;
-: it was not possible run the basic method for these parameters due to lack of memory;
*: a low estimate of the size of the tree is shown, since it was not possible to build the complete tree for these parameters.

Table A.3 - Sequence 3

Parameters $t_{1,2}=t_{2,1}=c_{1,2}=c_{2,1}$	Algorithm Time (sec.)	Basic method Time (sec.)	Non-dominated strategies	Size of the tree (event nodes)
7.00	0.00	0.00	2	5749
5.00	0.00	0.00	6	5749
3.00	0.00	0.00	39	5749
2.00	0.00	0.05	88	5713
1.67	0.09	0.10	143	5599
1.33	0.29	0.30	217	5702
1.00	0.48	0.53	285	5699
0.67	18.04	20.81	778	5695
0.33	173.60	184.87	1916	5695

Other parameters: $x_{1,i} \in [0.14,0.15]$ and $x_{2,i} \in [0.19,0.20]$, i=1,2; $t_1=1$, $c_1=2$, $t_2=2$, $c_2=1$.

Table A.4 - Sequence 4

Parameters $t_{1,2}=t_{2,1}=c_{1,2}=c_{2,1}$	Algorithm Time (sec.)	Basic method Time (sec.)	Non-dominated strategies	Size of the tree (event nodes)
10	0.00	1.85	8	1272559
9	0.00	1.85	18	1272559
8	0.00	1.90	38	1272559
7	0.03	1.88	74	1272559
6.5	0.11	1.93	114	1272559
6	0.54	2.20	186	1266209
5.5	1.95	2.97	268	1266209
5	7.19	6.14	388	1266209
4.5	29.80	21.27	662	1251021
4	77.64	57.83	885	1248462

Other parameters: $x_{1,1} \in [0.07,0.08]$, $x_{2,1} \in [0.09,0.1]$, $x_{1,2} \in [0.105,0.12]$ and $x_{2,2} \in [0.135,0.15]$; $t_1=1$, $c_1=2$, $t_2=3$, $c_2=1.5$.

Table A.5 - Sequence 5

Parameters $t_{1,2}=t_{2,1}=c_{1,2}=c_{2,1}$		Algorithm Time (sec.)	Basic method Time (sec.)	Non-dominated strategies	Size of the tree (event nodes)
[0.2500,0.2600]	10.0	0.00	0.00	2	170
[0.2116,0.2201]	13.5	0.00	0.00	2	682
[0.1791,0.1863]	18.1	0.00	0.00	2	2730
[0.1516,0.1577]	24.4	0.00	0.00	2	10922
[0.1395,0.1450]	28.3	0.00	0.02	2	35133
[0.1283,0.1334]	32.8	0.00	0.03	2	43690
[0.1180,0.1228]	38.0	0.00	0.11	2	174762
[0.1086,0.1129]	44.1	0.00	0.38	2	534012
[0.0999,0.1039]	51.2	0.00	0.50	2	699050
[0.0919,0.0956]	59.4	0.00	3.23	2	2796202
[0.0846,0.0879]	68.9	0.00	20.70	2	11184810
[0.0778,0.0809]	79.9	0.00	-	2	38000000*
[0.0716,0.0744]	92.7	0.00	-	2	1.8E+08*
[0.0606,0.0630]	124.7	0.00	-	2	5.5E+09*
[0.0513,0.0533]	167.8	0.00	-	2	3.2E+11*
[0.0434,0.0451]	225.7	0.00	-	2	4.0E+13*
[0.0367,0.0382]	303.8	0.00	-	2	1.2E+16*
[0.0338,0.0351]	352.4	0.00	-	2	2.9E+17*
[0.0311,0.0323]	408.7	0.03	-	2	9.7E+18*
[0.0286,0.0297]	474.1	0.06	-	2	4.3E+20*
[0.0263,0.0274]	550.0	0.08	-	2	2.7E+22*
[0.0242,0.0252]	638.0	0.17	-	2	2.4E+24*
[0.0223,0.0232]	740.1	0.22	-	2	3.2E+26*
[0.0189,0.0196]	995.9	0.22	-	2	2.0E+31*
[0.0160,0.0166]	1340.0	0.22	-	2	9.8E+36*
[0.0135,0.0140]	1803.1	0.22	-	2	5.1E+43*
[0.0114,0.0119]	2426.3	0.23	-	2	4.3E+51*
[0.0097,0.0101]	3264.8	0.25	-	2	1.0E+61*
[0.0082,0.0085]	4393.2	0.25	-	2	1.2E+72*
[0.0069,0.0072]	5911.4	0.28	-	2	1.4E+85*
[0.0059,0.0061]	7954.4	0.30	-	2	4.1E+100*
[0.0050,0.0052]	10703.5	0.33	-	2	7.4E+118*
[0.0042,0.0044]	14402.6	0.39	-	2	2.8E+140*
[0.0036,0.0037]	19380.2	0.47	-	2	8.5E+165*
[0.0033,0.0034]	22481.0	0.52	-	2	2.3E+180*
[0.0030,0.0031]	26077.9	0.59	-	2	1.1E+196*
[0.0028,0.0029]	30250.4	0.65	-	2	1.2E+213*

Other parameters: $t_1=1$, $c_1=2$, $t_2=2$, $c_2=1$;

-: it was not possible run the basic method for these parameters due to lack of memory;

*: a low estimate of the size of the tree is shown, since it was not possible to build the complete tree for these parameters.

Table A.6 - Sequence 6

Parameters $t_{1,2}=t_{2,1}=c_{1,2}=c_{2,1}$	Algorithm Time (sec.)	Basic method Time (sec.)	Non-dominated strategies	Size of the tree (event nodes)
5	0.00	0.06	34	27993
4	0.00	0.05	38	27993
3.5	0.12	0.15	140	27993
3	0.17	0.19	167	27993
2.5	0.19	0.23	199	27993
2	0.27	0.29	216	27993
1.75	22.34	21.61	701	27993
1.5	40.72	38.94	861	27993
1.25	68.37	62.63	984	27993
1	94.57	91.86	1095	27993

Other parameters: $x_{b,i} \in [0.19,0.2]$, i=1,2,3, b=1,2; $t_1=1$, $c_1=3$, $t_2=2$, $c_2=2$, $t_3=3$, $c_3=1$.

REFERENCES

Brealey, R. and Myers, S., Principles of Corporate Finance (McGraw-Hill, 4th edition, 1991).

Godinho, P.C. and Costa, J.P., "Incorporating Risk in a Decision Support System for Project Analysis and Evaluation". In: Kantarelis, D. (Ed.), Business and Economics for the 21st Century – Volume III, Business & Economics Society International, Worcester, 1999, pp. 90-101.

Herath, H.S. and Park, C.S., "Economic Analysis of R&D Projects: An Options Approach", The Engineering Economist, Vol. 44, nº 1, 1999, pp. 1-35.

Hertz, D.B. and Thomas, H., Risk Analysis and its Applications (John Wiley and Sons, 1st edition, 1983).

Magee, S., "How to use Decision Trees in Capital Investment", Harvard Business Review, 42, September-October 1964, pp. 79-96.

Smith, J.E. and Nau, R.F., "Valuing Risky Projects: Option Pricing Theory and Decision Analysis", Management Science, 41, nº 5, 1995, pp. 795-816.

Trigeorgis, L., "The Nature of Option Interactions and the Valuation of Investments with Multiple Real Options", Journal of Financial and Quantitative Analysis, 28, nº 1, 1993, pp.1-20.

Trigeorgis, L., Real Options: Managerial Flexibility and Strategy in Resource Allocation (The MIT Press, 1996)

Trigeorgis, L. and Mason, S.P., "Valuing Managerial Flexibility", Midland Corporate Finance Journal, Spring 1987, pp. 14-21.