The Use of Cost and Time in Project Decision Trees: A model and an application[1]

Pedro Cortesão Godinho / João Paulo Costa
Faculty of Economics of the University of Coimbra and INESC

resumo

Neste artigo, apresenta-se um modelo bicritério para análise de projectos baseado em árvores de decisão, e uma aplicação deste modelo a um problema de planeamento de produção. Começa-se por apresentar um modelo que permite a utilização do tempo e do custo na análise de projectos. Este modelo é simples de usar, e pode ser aplicado a um grande número de situações reais em que os principais objectivos são a minimização do tempo e do custo. A construção das árvores de decisão correspondentes a este modelo pode exigir tempos de cálculo elevados, e assim tornar-se impraticável. Este problema é abordado de duas formas: primeiro, define-se um algoritmo para a geração das estratégias, e depois introduz-se um parâmetro de erro que permite evitar a geração das estratégias que estejam muito próximas de outras. Finalmente, apresenta-se uma aplicação do modelo a um problema de planeamento de produção. Define-se o problema, utiliza-se o algoritmo para gerar as estratégias eficientes, e analisam-se os resultados.

résumé / abstract

Cet article présente un modèle bi-critère d'analyse de projets ayant pour base des arbres de décision et qui est ensuite appliqué à un problème de planification de la production. On commence par présenter un modèle qui permet d'utiliser le temps et les coûts dans l'analyse de projets. Ce modèle s'emploie facilement et peut être appliqué à un grand nombre de situations réelles dont le but principal est la minimisation du temps et des coûts. La construction des arbres de décision qui correspondent à ce modèle peut exiger des temps de calcul très longs, ce qui peut le rendre impraticable. Le problème est abordé en deux étapes: on commence par définir un algorithme pour générer les stratégies et on introduit ensuite un paramètre d'erreur qui va permettre d'éviter la génération de stratégies très proches les unes des autres. Finalement, on présente l'application du modèle à un problème de planification de la production. On définit le problème, on emploie l'algorithme pour générer des stratégies efficientes et on analyse les résultats.

In this paper we present a useful bicriteria model for project analysis based on decision trees, and an application of the model to a production planning problem. We start by presenting a model that allows the use of time and cost in project analysis. This model is easy to use, and it can be applied to a large number of real-life situations when the main objectives are the minimisation of cost and the minimisation of time. The construction of the decision trees for this model may require large computational times, and thus become impracticable. We deal with this problem in two ways: first, we define an algorithm for generating the strategies, and then we introduce an error parameter, which allows us to avoid generating all the strategies that are very close to each other. Finally, we apply the model to a production planning problem. We define the problem, use the algorithm to identify the efficient strategies and analyse the results.

1 This research was partially supported by FCT, FEDER, project POCTI/EGE/58828/2004.

1. Introduction

Decision trees are often used in project analysis and appraisal (see Brealey and Myers, 2000; Magee, 1964), usually considering only the financial perspective. In many cases, there are certain factors that cannot be incorporated into the financial value of a project and that are very important in deciding whether or not it should be undertaken, so it is often useful to include multiple criteria in the analysis. The efforts to use multiple criteria in the evaluation of project decision trees have been based on multi-attribute utility theory (MAUT). Hertz and Thomas (1983) present an overview of some MAUT-based methodologies for the evaluation of multicriteria decision trees. These methodologies require either the definition of the criteria aggregation rule or the definition of some trade-offs between criteria before rolling back the tree, and they only allow the identification of the "best" alternative. Smith and Nau (1995) also use utility theory to evaluate project decision trees in incomplete markets.

Time is one important criterion that is often overlooked in the project evaluation literature. Godinho and Costa (2002) outline a new approach for the incorporation of time and financial value in project decision trees, which is detailed and analysed in depth in Godinho (2003). This approach is based on the identification of the non-dominated strategies[2]. After all the non-dominated strategies are identified, the decision-maker may use any multicriteria method to choose from them. We will present this approach in Section 2.

This approach may lead to very large trees, whose construction and evaluation may take a very long time, or even be impractical. Therefore, it is useful to build particular models that allow the automatic definition of the tree using a limited set of parameters. Godinho (2003) proposes one such model, which will be presented in Section 3. This model assumes that some different processes may be used to undertake a homogeneous task, each process having a constant cost and requiring a constant time per utilisation, and that there are switching costs and set-up times for changing the process being used. In this model, project advance follows an additive binomial process. After the processes are defined, the complete decision tree can be automatically built, and the non-dominated strategies can be identified. This model is easy to use, because it only requires a limited number of inputs: the parameters of the processes. Also, it can be used as an approximation to some continuous-time problems, and applied to a large number of real-life situations when the main objectives are the minimisation of cost and the minimisation of time, and when the considered task is homogeneous, or can be treated as such. Possible applications include the analysis of some construction projects and production planning.

The decision trees generated by this model are usually very large, and building and evaluating the trees in order to identify the non-dominated strategies may be impractical, because it takes a very long time, or even impossible due to memory limitations of the computer systems. In order to handle this problem, an algorithm for the identification of the non-dominated strategies was proposed in Godinho (2003). We will outline this algorithm in section 4. The algorithm does not require the construction of the complete decision tree, since it is able to disregard non-interesting branches as soon as they come up, and to avoid the repetition of calculations for similar branches of the tree. The algorithm also provides a more compact representation of the tree, since identical branches located in different points of the tree are represented only once.

Computational tests were performed by Godinho (2003) to assess the performance of the algorithm and some results are presented in Section 4.2. These tests have shown that the algorithm performs well when the number of non-dominated strategies is small, but it does not perform so well when that number is large. In order to overcome the problem of model usage with a large number of non-dominated strategies, we introduce a new approach: an error

2 We consider a strategy to be the complete set of decisions that will be made until the end of the project. A strategy is non-dominated if none of the other strategies is better or equal in all the criteria, and strictly better in at least one criterion.

parameter that allows us to avoid generating all the non-dominated strategies that are very close to each other. Using this approximation, we are able to reduce the number of strategies that are generated. This improves the performance of the algorithm, and we are able to apply the model to some cases that we would otherwise be unable to handle.

Finally, we describe an application of the model. We consider a production planning problem, and we calculate the model parameters for that problem. Then we use the algorithm to calculate the non-dominated strategies and we analyse the results.

This paper is structured as follows. Section 2 presents a general approach for the use of time and value (or cost) in decision trees. Section 3 presents the particular model that we use in our application. Section 4 outlines and discusses the algorithm for the identification of non-dominated strategies. Section 5 describes an application of the model. Finally, we present our conclusions in Section 6.

2. An approach for the use of bicriteria decision trees in project analysis

This section presents the general approach for the use of time and financial value in project decision trees that was proposed by Godinho and Costa (2002). We assume that we are maximising the financial value, as measured by the Net Present Value (NPV), and minimising the time. This approach focuses on the identification of the non-dominated strategies, allowing the decision-maker to choose one of these alternatives using a multicriteria method.

The evaluation of the decision tree is 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, time- and value-adjusted probabilities are calculated, and the tree is rolled back. Since we are using multiple criteria, the rolling back process is different from the one used in standard decision tree analysis.

The aggregation of criteria values across event nodes is based on the adjustment of probabilities. Since financial value and time may follow different aggregation rules, time-adjusted probabilities may differ from value-adjusted probabilities for the same branch, and they are usually different from the initial probabilities of that branch.

In an event node, the financial value is calculated as the weighted sum of the values corresponding to its branches, using the value-adjusted probabilities as weights. The adjustment of probabilities follows a discrete-time option valuation model: probabilities are adjusted to risk-neutral value-adjusted probabilities according to the pricing process of the securities on which the project value is contingent. In this paper, we use the binomial model, but other models can be used with this approach. One alternative model is the trinomial model (used, for example, by Smith and McCardle, 1998).

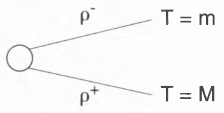
A new problem arises with the use of an option valuation model: some event nodes may not correspond to events that influence the value of the securities on which the option value is contingent – that is, private risks may exist. For example, uncertainty about whether a factory will take 18 or 20 months to be built does not influence the value of a publicly traded company with a similar factory. Such events are unique to the project, and should thus be treated as unsystematic risk. Since this risk is not relevant to the investors, the probabilities associated with such nodes are not adjusted for the calculation of financial value – only the event nodes that correspond to the passage of time require the adjustment of probabilities.

Time may be aggregated across event nodes in several different ways, depending on the situation that the decision-maker is facing. Average time to completion (or average delay relative to a specified completion time) and maximum time may, under some circumstances, be of interest. However, we believe that an uncertain time is, in most situations, considered equivalent to a fixed time between the average and the maximum of the uncertain times. So, a very general approach to aggregate time across event nodes is proposed, for which average time and maximum time are particular cases. This approach is similar to the risk-neutral valuation technique, and it relies on the use of certainty equivalents.

To start with, the decision-maker is asked for a certainty equivalent of a specified uncertain time. Then we calculate the implicit probabilities that equate the expected value of the uncertain time to that certainty equivalent. Those probabilities – time-adjusted probabilities – are used to perform the aggregation of time whenever the same pair of initial probabilities comes up (if the times are in the same order).

Assume that, in an event node, m is the shortest time, M is the longest time, and ρ^- and ρ^+ are their corresponding probabilities, as shown in Figure 1. Also assume that, faced with the uncertain time represented by that node, the decision-maker provides a certainty equivalent time CE. Then the certainty equivalent CE corresponds to an adjustment of probabilities to $\rho_T^- = (M\text{-}CE)/(M\text{-}m)$ and $\rho_T^+ = (CE\text{-}m)/(M\text{-}m)$. These time-adjusted probabilities (ρ_T^-, ρ_T^+) can be used every time the initial probabilities (ρ^-, ρ^+) come up, with the longest time corresponding to ρ^+. Notice that the choice of the longer time as the certainty equivalent leads to the maximum time to completion, and the choice of the average time as the certainty equivalent leads to the average time to completion.

Figure 1 – A generic binomial node with uncertain time

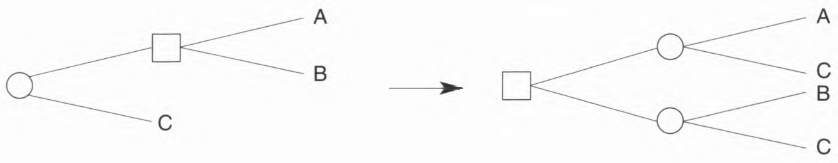


We will now consider the evaluation of decision nodes. We want the tree evaluation process to provide all the non-dominated strategies, and we use three different rules to accomplish this.

The first rule is that two consecutive decision nodes are merged. This means that, when there are consecutive decision nodes, it is the choice among all the alternatives represented by those nodes that is considered, and not consecutive choices among some of those alternatives.

The second 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 shows the use of this rule. Note that if the lower branch were to 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 next rule (the third) in order to prevent the number of alternatives from becoming too large.

Figure 2 – Example of the use of rule 2. A, B and C are different alternatives



The third rule is to eliminate all dominated alternatives in a decision node. This means that one alternative in a decision node can be eliminated 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.

There is no need to explicitly modify the tree in order to apply these rules: they can be applied by considering that sets of alternatives are associated with the tree nodes. With these rules, the decisions involving non-dominated alternatives can be postponed until all event nodes are evaluated. Then, a multicriteria method may be used to choose from the non-dominated strategies.

This section has described the general approach for the use of time and financial value in project decision trees that was proposed by Godinho and Costa (2002). It is an approach that identifies all the non-dominated strategies, allowing the decision maker to use a multicriteria method to choose from them. In the next section we will define a particular model based in this general approach.

3. A model for the use of time and cost in project analysis

This section presents a model based on the above approach. This model assumes that time and cost are the relevant criteria, and that the decision-maker wants to minimise both criteria.

Consider a project that consists of undertaking a homogeneous task, that is, a task that can be split into several subtasks, all of them with identical characteristics. Define x^0 as the amount of work required to complete the task. It is hereafter assumed, without loss of generality, that $x^0 = 1$, and x is used to represent a fraction of the project ($0 \leq x \leq 1$).

Assume that n different processes, P_i, $i = 1,\ldots, n$, may be used to undertake the project. Each process P_i is characterised by:

– a given time, $t_i \geq 0$, representing the duration of each utilisation of the process;

– a given cost per utilisation of the process, $c_i \geq 0$;

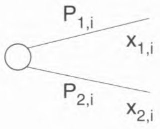
– a set of switching costs, representing the cost of switching to each of the other processes, $c_{i,j} \geq 0$, $j = 1,\ldots, n$, $j \neq i$ ($c_{i,i} = 0$);

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

– an initial set-up time, $t_{0,i} \geq 0$, and an initial cost, $c_{0,i} \geq 0$, representing the time and cost associated with the use of the process in the beginning of the project;

– an additive binomial process to describe project advance under each utilisation of process P_i, which can be represented by the binomial event node shown in figure 3.

Figure 3 – Binomial event node representing project advance under each utilisation of process P_i



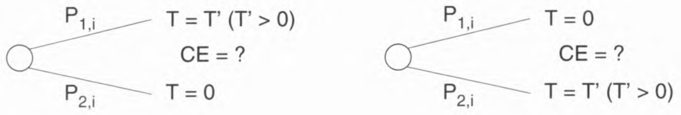
$x_{1,i}$ and $x_{2,i}$ ($x_{1,i} > 0$, $x_{2,i} > 0$) are the possible project advance amounts under process P_i, and $P_{1,i}$ and $P_{2,i}$ are the corresponding probabilities, with $P_{1,i}+P_{2,i}=1$, $P_{1,i}>0$ and $P_{2,i}>0$. Note that this representation assumes the independence of project advances: the probability distribution of project advance under a given process is constant, and therefore independent of previous project advances. It is assumed, without loss of generality, that $x_{1,i} \leq x_{2,i}$. Since it is assumed that $x^0 = 1$, then $x_{1,i}$ and $x_{2,i}$ can be interpreted as fractions of the project. The arc corresponding to an advance $x_{1,i}$ is hereafter identified as arc 1 of the node, and the arc corresponding to an advance $x_{2,i}$ is identified as arc 2. The branches starting immediately after these arcs are identified as branches 1 and 2 of the node, respectively.

This model assumes that the use of the processes is indivisible. This means that the process being used can only be changed at the end of a complete utilisation. If, at the end of the task, the fraction of the project that has not been undertaken is smaller than the project advance in the next use of the process, the time and cost for a complete utilisation of the process will be required anyway.

Following the previously outlined approach, the aggregation of criteria values across event nodes is based on the adjustment of probabilities. For the cost, the binomial model for option valuation is used. Since, in the model being presented, the cost per process utilisation is constant for each process, the cost risk is only related to the project duration risk and to the process choices, and it is thus independent of the price of any traded asset. Therefore, the cost risk is unsystematic, meaning that no risk premium shall be required, and the value-adjusted probabilities are equal to the initial probabilities.

For the calculation of time-adjusted probabilities, the decision-maker is asked to provide two certainty equivalents for each different pair of initial probabilities occurring in the definition of the processes. One certainty equivalent corresponds to an arbitrary time T' > 0 in branch 1 and a time 0 in branch 2, and the other corresponds to a time 0 in branch 1 and a time T' > 0 in branch 2 (see Figure 4). 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 is used when time is longer in branch 1 than in branch 2, and the second pair is used in the opposite situation. Notice that the two different pairs of adjusted probabilities are necessary because the risk preferences of the decision-maker will usually lead her/him to adjust the same pair of probabilities differently, according to the branch that contains the longer time. As an example, if the decision-maker is averse to the time risk, she/he will adjust $P_{1,i}$ to a larger value if branch 1 contains the longer time, and she/he will adjust $P_{1,i}$ to a smaller value if branch 2 contains the longer time.

Figure 4 – Certainty equivalents elicited for the calculation of the 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 do not think it is so. In a real-life problem it may sometimes be possible to define exactly during how much time a process will be used, and there may be continuous statistical distributions for the project advance. Still, models like this one may be used to approximate those problems, by defining processes such that the duration of each utilisation is small. A similar approximation is done in some continuous-time real option problems, particularly when multiple options are involved (see, for example, Trigeorgis, 1993).

An important characteristic of the model is that the decision trees are often very large. These trees may thus require very large amounts of memory space, and the corresponding calculations may take a very long time. Therefore, the approach based on the construction and evaluation of the tree is usually impractical, or even impossible to use. This problem will be considered in Section 4, where an algorithm for an efficient identification of the non-dominated strategies will be presented.

This model is easy to use, and can be applied to the analysis and planning of projects that consist of undertaking homogenous tasks, when time and cost are the relevant criteria. Possible applications of the model include the analysis of some construction projects and production planning. It is also easy to modify the model to make it more general. For example, by making small changes we would be able to drop the assumption of independent project advances, or to use functions of time and cost (for example, utilities) instead of the absolute values of the criteria. However, such modifications may make the development of efficient algorithms for the identification of non-dominated strategies more difficult, or even impossible.

4. An algorithm for the generation of the non-dominated strategies

The model described in the previous section usually leads to very large decision trees, whose construction and evaluation may take a long time and require large amounts of memory space. As an attempt to overcome this problem, an algorithm for an efficient identification of the non-dominated strategies was developed and is now described. The algorithm uses some mathematical properties of the model, which are presented in the appendix.

4.1. Main ideas of the algorithm

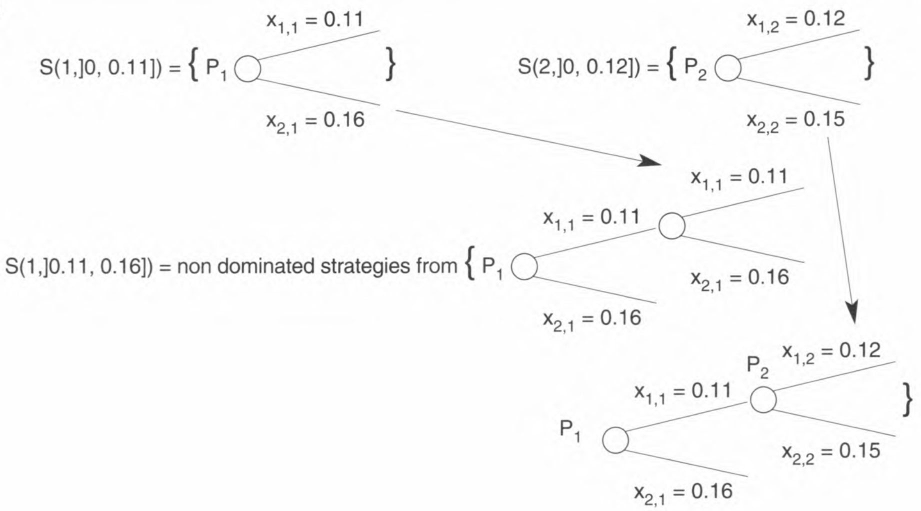
Each branch of the tree corresponds to a fraction of the project: the fraction of the project that was not undertaken in the previous arcs of the tree. Different branches of the tree, occurring in different places in the tree, may correspond to the same fraction of the project, thus leading to identical sets of non-dominated strategies. Moreover, all the branches of the tree that correspond to fractions x of the project belonging to certain intervals I lead to the same set of non-dominated strategies (see property 2 in the Appendix).

Let S(i,I) be the common set of non-dominated strategies for undertaking fractions $x \in I$ of the project starting with the use of process P_i. The algorithm calculates the sets of non-dominated strategies S(i,I) for intervals I of consecutively larger values of x, until the sets of non-dominated strategies for $x = x^0 = 1$ and for all processes are calculated.

As we explained before, the sets of non-dominated strategies that allow the completion of a given fraction x of the project by starting with a given process are the same for a given interval of values of x. To exemplify this, consider the existence of two processes, P_1 and P_2, with $x_{1,1} = 0.11$ and $x_{2,1} = 0.16$, for P_1, and $x_{1,2} = 0.12$ and $x_{2,2} = 0.15$, for P_2. If a fraction of the project that is larger or equal than 0.89 is already executed, and process P_1 is used after that, then, after using P_1 once, the project will be completed. Since this is true if the fraction of the project not previously undertaken is positive and smaller than or equal to 0.11, then S(1,]0, 0.11]) is the set that only contains the strategy of using P_1 once. So, when the fraction of the project that was not previously undertaken is $x \in]0, 0.11]$ and P_1 is used next, the only non-dominated strategy is the strategy belonging to S(1,]0, 0.11]). Similarly, for P_2, S(2,]0, 0.12]) is the set that only contains the strategy of using P_2 once. For larger values of x, the sets of non-dominated strategies are also identical for all the values of x belonging to some intervals, allowing us to avoid repeating the calculation of branches that lead to the same non-dominated strategies.

The algorithm begins with the identification of the non-dominated strategies for small project fractions. In our previous example, it would start with the calculation of S(1,]0, 0.11]) and S(2,]0, 0.12]). Then, the sets of non-dominated strategies that were previously calculated are taken into account in the identification of the non-dominated strategies for consecutively larger project fractions. In the 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 branch 1. Since the non-dominated strategy to be used in branch 1 could be from either S(1,]0, 0.11]) or S(2,]0, 0.12]), two different strategies would be generated. Notice that a strategy generated in such a manner is valid for undertaking a project fraction smaller than or equal to 0.16, because the use of another strategy in branch 2 is not being considered. So, the next set to be considered would be S(1,]0.11, 0.16]), which is the set of those two strategies, if none of them is dominated, or a set containing one of those strategies, if the other one is dominated. Figure 5 represents these sets. The algorithm then proceeds until the complete project ($x = x^0 = 1$) is considered.

Figure 5 – Initial sets calculated by the algorithm for the example presented in this subsection



So, the identification of non-dominated strategies for larger fractions of the project uses the non-dominated strategies generated for smaller fractions, allowing the algorithm to discard dominated strategies as soon as they are identified. This way, it is also possible to avoid the repetition of calculations for branches with similar characteristics occurring in different places in the tree; otherwise such repetition of calculations would happen several times. Property 1 (presented in the Appendix) is also used by the algorithm, to avoid the calculation of some sets of non-dominated strategies, and to allow a more efficient calculation of other sets.

In short, we can say that the algorithm starts with tree branches corresponding to small fractions of the project (initially the fractions undertaken in the tree leaves), and then considers consecutively larger fractions until all the non-dominated strategies for the complete tree are identified. For a complete description of the algorithm, see Godinho (2003)

4.2. The performance of the algorithm

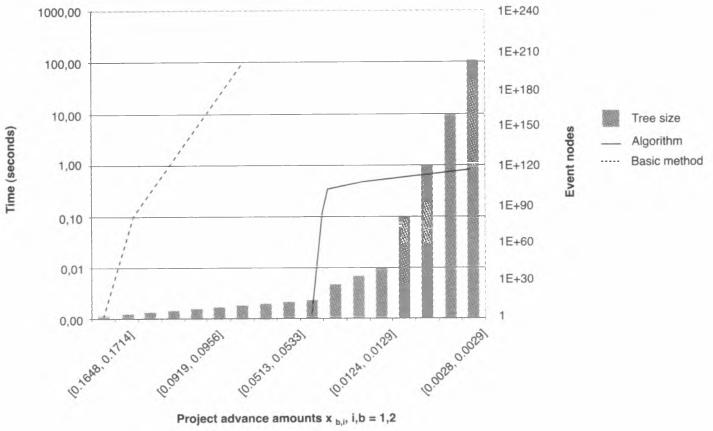
Some computational tests were conducted in order to assess the performance of the algorithm. Those tests aimed to examine the performance of the algorithm in several different situations, and to compare the performance of the algorithm with the performance of an alternative method. This alternative method, the "basic method", consists of building and evaluating the tree according to the approach defined in section 2. We now present some computational results – other results, and other details about the tests, can be found in Godinho (2003).

These tests used sequences of files. The first element of each sequence was generated by defining some parameters as constants, and the remaining parameters as samples of given uniform distributions. The other elements of each sequence were defined through sequential changes in given parameters. Twenty sequences were generated for each set of parameter distributions, and average running times were then calculated for each set. Only non-dominated strategies with different times and costs were generated (alternative strategies with identical cost and time were not generated).

The sets presented in this paper consider two processes, the first one with $t_1 = 1$ and $c_1 = 2$, and the second one with $t_2 = 2$ and $c_2 = 1$. These sets were designed to analyse the performance of the algorithm and the basic method when the size of the tree increases. In the first set, the number of non-dominated strategies was kept very small (equal to 2) and, in the second set, the number of non-dominated strategies was allowed to increase quickly with the size of the tree. In order to achieve this, we started all sequences of each set with large advance amounts ($x_{b,i}$ initially belonged to [0.25, 0.26], i = 1, 2, b = 1, 2), and decreased these amounts along each sequence. In order to keep the number of non-dominated strategies small, we defined large switching costs and set-up times in the first set ($c_{i,j} = t_{i,j} = 10000$). In the second set, we used null switching costs and set-up times, thus allowing the number of non-dominated strategies to increase quickly. The following distributions were used for the probabilities: the basic probabilities for both branches belong to [0.49, 0.51], the probability for the branch with the longest time belongs to [0.55, 0.6], and the probability for the branch with the shortest time belongs to [0.4, 0.45].

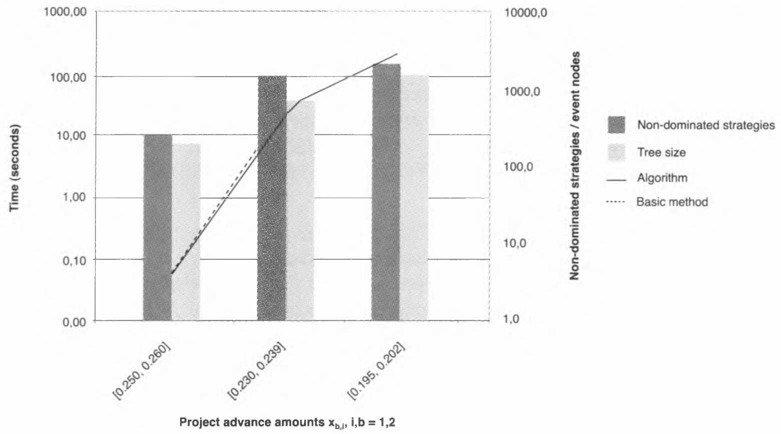
For the first set of sequences (see Chart 1), the algorithm performed much better than the basic method. In fact, it was only possible to run the basic method until the tree size reached some hundred million nodes ($x_{b,i} \in [0.0716, 0.0744]$), because disk and memory space were insufficient for larger trees. However, it was still possible to run the algorithm, with very small execution times (never much more than 1 second), for all the parameter values that were used (the tree size is estimated to be about 10^{31} event nodes for the smallest advance amounts that were considered). This shows that, when the number of non-dominated strategies is small (because of large set-up times and switching costs), the algorithm is clearly faster and much more efficient in memory usage terms than the basic method.

Chart 1 – Calculation time (scale on the left) and tree size (scale on the right) for different project advance amounts, for the tests on the first set of sequences. Logarithmic scale is used in the Y axis



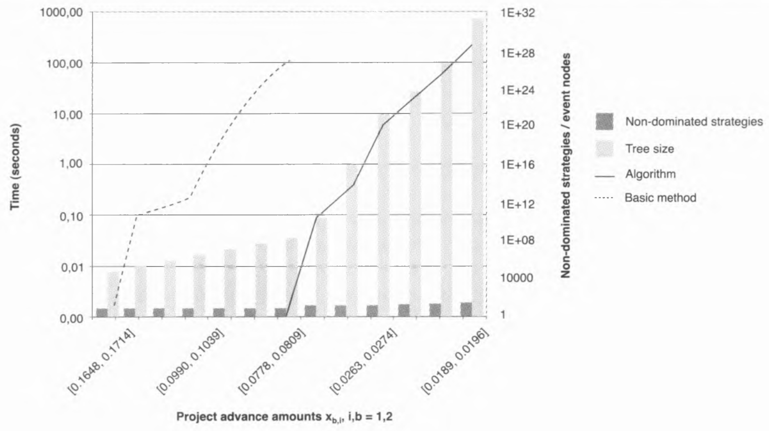
For the second set of sequences (see Chart 2), the results were quite different. In fact, the number of non-dominated strategies was very large, and the calculation times were similar for both the algorithm and the basic method. Notice that the large number of non-dominated strategies soon made their identification impracticable, both for the algorithm and for the basic method.

Chart 2 – Calculation time (scale on the left), number of non-dominated strategies and tree size (scale on the right) for different project advance amounts, for the tests on the second set of sequences. Logarithmic scale is used in the Y axis



It is also interesting to examine the results we achieved when we used extreme time-adjusted probabilities: a 100% probability for the branch with the longest time and a 0% probability for the branch with the shortest time (notice that using these probabilities is the same as using maximum time to completion to aggregate time). When these time-adjusted probabilities were used, there was no noticeable change in the results for the first set of sequences (because the number of non-dominated strategies was already very small for this set), but the results for the second set were significantly changed. In the second set of sequences (Chart 3), the number of non-dominated strategies was lower, and the algorithm performance became much better than the basic method performance.

Chart 3 – Calculation time (scale on the left), number of non-dominated strategies and tree size (scale on the right) for different project advance amounts, for the tests on the second set of sequences when extreme time adjusted probabilities were used. Logarithmic scale is used in the Y axis



The results of the tests led us to some important conclusions:

- as would be expected, the performance of both methods worsens as the number of non-dominated strategies increases and as the tree size increases;
- the performance of the algorithm is less sensitive to the increase in tree size, and more sensitive to the increase in the number of non-dominated strategies;
- the performance of the algorithm is not clearly better than the performance of the basic method when the number of non-dominated strategies is large;
- the algorithm performs much better than the basic method, both in terms of running time and in terms of memory usage, when the number of non-dominated strategies is small;
- when the maximum time to completion is used (and, in fact, whenever extreme probabilities of 0% or 100% are used), the number of non-dominated strategies is small, and the algorithm performs very well.

So, although the algorithm does not appear to be able to efficiently generate a large number of non-dominated strategies, its performance is very good when the number of non-dominated strategies is small. The use of the algorithm to identify some representative strategies, instead of all the non-dominated strategies, may thus be an efficient way to employ the model.

4.3. The use of maximum error parameters

For an effective use of the model, it is important to define a way to improve the performance of the algorithm when the number of non-dominated strategies is large. We developed a new approach, based on the use of maximum error parameters, which allows the algorithm to identify some representative strategies instead of generating all the non-dominated strategies. This way, we are able to perform a faster identification of the strategies, allowing us to use the model in some situations that we would otherwise be unable to analyse.

To understand the meaning of these parameters, let us define ϵ_t as the time error parameter and ϵ_c as the cost error parameter. These parameters mean that, for a given non-dominated strategy with time t and cost c that ceases to be generated, it can be guaranteed that there is at least one strategy that is generated with a time $t' \leq t + \epsilon_t$ and a cost $c' \leq c + \epsilon_c$. This way, it is possible to generate only one "representative" strategy for each neighbourhood of size (ϵ_t, ϵ_c). Thus, a smaller number of strategies is generated by the algorithm, and its performance is improved. The improvement in the algorithm performance depends on the values of ϵ_t and ϵ_c: the larger these values are, the better the algorithm performs, and the furthest may the generated strategies be from the true non-dominated strategies.

The practical implementation of these parameters in the algorithm required some adaptations. In fact, instead of the global time and cost error parameters considered above, we defined time and cost error parameters that are used by the algorithm every time that an elimination of dominated strategies occurs. Instead of only eliminating dominated strategies, the algorithm also eliminates some non-dominated ones, if another one exists in the neighbourhood defined by the error parameters. From these parameters, and from the maximum number of tree levels, it is possible to calculate upper bounds for the maximum errors considered before (ϵ_t and ϵ_c).

The practical use of these maximum error parameters proved to be very useful, allowing us to run the model for some parameter sets that we would otherwise be unable to handle. The application we present in the next section is one such example.

5. An application of the model

This section presents an application of the model, in order to show that it may be useful in real-life situations.

Consider the case of a textile company that received an order for 10 000 shirts from a soccer team. The company managers want to produce the shirts as quickly as possible, but they also want to have as low a production cost as possible. The company has three different sets of equipment, let us call them A, B and C, that can be used. The number and skills of the available workers, as well as other company commitments, do not allow the simultaneous use of more than one of these sets in the production of shirts. The decision about which equipment to use may be made on a daily basis.

Equipment A produces an average μ = 3000 shirts in a 5-day week, with a standard deviation of σ = 250 shirts. Its use costs €500 per day, and its initial set-up time and cost are 2 days and €1000 euros, respectively. Equipment B produces an average μ = 4800 shirts in a 5-day week, with a standard deviation of σ =500 shirts. Its use costs €1000 per day, and its initial set-up time and cost are 2 days and €2000, respectively. Equipment C produces an average μ = 2400 shirts in a 5-day week, with a standard deviation of σ = 250 shirts. Its use costs €400 per day, and its initial set-up time and cost are 1 day and €400, respectively. These characteristics of the equipment are summarized in Table 1.

Table 1 – Characteristics of the equipment

	Equipment		
	A	B	C
Average production	3000 shirts/week	4800 shirts/week	2400 shirts/week
Standard deviation of the production	250 shirts	500 shirts	250 shirts
Cost	€500/day	€1000/day	€400/day
Initial set-up time	2 days	2 days	1 days
Initial set-up cost	€1000	€2000	€400

The set-up times and switching costs for changing the equipment being used are shown in Table 2. If it can be assumed both that the number of shirts produced in a given day is independent of the number of shirts produced in the previous day and that the binomial project advance distribution provides a good enough approximation for the production distribution, then the model presented in Section 3 may be used.

Table 2 – Set-up times and switching costs for switching from one set of equipment to another

From/To		Equipment		
		A	B	C
Equipment	A		2 days / €1500	1 day / €500
	B	2 days / €1500		1 day / €1000
	C	1 day / €800	1 day / €1700	

We define three different processes, P_1, P_2 and P_3. P_1 consists of using equipment A for one day, P_2 consists of using equipment B for one day and P_3 consists of using equipment C for one day. For each of these processes, t_i is 1 day, c_i is the cost of using the equipment for one day and $t_{0,i}$ and $c_{0,i}$ are the initial cost and initial set-up time for the equipment, so $t_1 = t_2 = t_3 = 1$ day, c_1 = €500 euros, c_2 = €1000, c_3 = €400, $t_{0,1} = t_{0,2} = 2$ days, $t_{0,3} = 1$ day, $c_{0,1}$ = €1000, $c_{0,2}$ = €2000 and $c_{0,3}$ = €400. The set-up times and switching costs are $t_{1,2} = t_{2,1} = 2$ days, $t_{1,3} = t_{2,3} = t_{3,1} = t_{3,2}$ = 1 day, $c_{1,2} = c_{2,1}$ = €1500, $c_{1,3}$ = €500, $c_{2,3}$ = €1000, $c_{3,1}$ = €800 and $c_{3,2}$ = €1700.

We now define additive binomial processes whose project advance amounts are consistent with the average and standard deviation of the shirt production. For each process, the average and standard deviation of shirt production are fitted to the average and standard deviation of a binomial distribution by using the following equations:

$$\frac{\mu}{5} = P_{1,i}x_{1,i} + P_{2,i}x_{2,i}$$

$$\frac{\sigma}{\sqrt{5}} = (x_{2,i} - x_{1,i})\sqrt{P_{1,i}P_{2,i}}$$

Since $P_{2,i} = 1-P_{1,i}$, there are two equations and three unknowns. So it is possible to let one of the unknowns assume a given value, to solve the system and then to assess the validity of the values of the parameters (to assess whether the probabilities belong to]0,1[and whether the project advance amounts are positive).

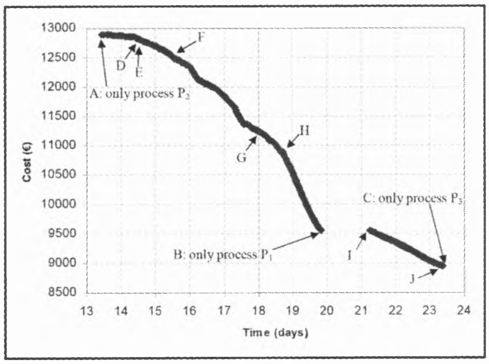
We define initial probabilities of 50% for the three processes. Solving the equation systems for all processes, we get $x_{1,1} = 488.2$ shirts and $x_{2,1} = 711.8$ shirts for process P_1, $x_{1,2} = 736.4$ shirts and $x_{2,2} = 1183.6$ shirts for process P_2 and $x_{1,3} = 368.2$ shirts and $x_{2,3} = 591.8$ shirts for process P_3. In order to keep $x^0 = 1$, the project advance amounts are normalised by dividing them by the total number of shirts to be produced (10000).

Since, for both processes, 50%/50% probabilities are used, the managers are only asked to provide one certainty equivalent. Assuming that a 50% probability of 0 days and a 50% probability of 10 days is considered equivalent to a certain time of 6 days, the time-adjusted probability for the longest time is 60%, and the time-adjusted probability for the shortest time is 40% This means that $P^{T,1}_{1,1} = P^{T,2}_{2,1} = P^{T,1}_{1,2} = P^{T,2}_{2,2} = 0.6$ and $P^{T,1}_{2,1} = P^{T,2}_{1,1} = P^{T,1}_{2,2} = P^{T,2}_{1,2} = 0.4$.

Now the model parameters are completely defined, and it seems that the model can be used to assess the efficient time/cost combinations. However, the trees that correspond to this application are very large, and the number of non-dominated strategies is also very large. This made it impossible to use either the algorithm or the basic method to generate all the non-dominated strategies. Instead, we ran the algorithm using the maximum error parameters, as described in subsection 4.3. The parameter values that we used considered a maximum time error of 0.011 days and a maximum cost error of €1 in each tree level. Since there are at most 28 levels in one such strategy tree, this leads to maximum global errors of 0.31 days and €28. As a percentage of the absolute criteria values, maximum time error is about 1.7% and maximum cost error is about 0.3%.

The model was run, allowing us to obtain 645 time/cost combinations, which are represented in Figure 6.

Figure 6 – Time/cost combinations for the representative strategies generated by the algorithm for the shirt production example



Two separated lines are clearly defined in the plot shown in Figure 6. The shortest line is on the bottom right of the plot and includes the strategies with the smallest costs. On the bottom right of this line is the strategy that consists of using only process P_3 (strategy C), which is the strategy that allows us to complete the project with the smallest cost. As we go left along this line, we get strategies with increasing costs and decreasing times. All the strategies in this line start with the use of process P_3 and, apart from strategy C, they all change to processes P_1 and P_2 in different situations. As we move left along the line, the changes from process P_3 to another process occur earlier in the tree. In strategy J, which is near strategy C, the first possible change occurs when about 33% of the project is complete, while in strategy I, at the other end of the line, the first possible change occurs when about 7% of the project is completed.

It is clear that the strategies near the left side of this shortest line (near I) do not seem attractive, since for a small additional cost it is possible to have a significant decrease in time by choosing strategy B.

The longest line is in the top left side of the plot and includes the strategies with the shortest times. This line does not look straight and, in fact, we may say that its form seems quite "odd". Notice that its form is clearly due to the non-linear effect of switching costs and set-up times in the model.

The strategy that consists of using only process P_2 (strategy A, on the top left of the line) is found at one extreme end of this line, and this is the strategy that allows us to complete the project in the shortest time. At the other extreme end of this line, we find the strategy that consists of using only process P_1 (strategy B, on the bottom right of the line). Near strategy A, we have strategies that start by using process P_2, then switch to P_3 in some situations, and sometimes switch again from P_3 to P_2 or P_1. For example, strategy D consists of starting with process P_2 and switching to P_3 in a particular situation. Near the other end of this line, the time/cost combinations correspond to strategies that start by using process P_1 and then change to process P_2. For example, strategy H starts by using P_1 and then changes to P_2 in some situations in which the project is advancing slowly.

Near the middle of this longest line, we have strategies that start by using P_3 and then switch to P_1 or P_2. It may seem odd that some strategies that start by using P_3 appear in the middle of this line, since the strategy that consists of only using P_3, and other strategies that start with P_3, are in the shortest line, and have quite different times and costs from the strategies in the middle of this longest line. There is a reason for this, though. If we want to use P_1 or P_2, depending on whether the project is advancing rapidly or slowly, it may pay to start by using P_3 and switch to one of the other processes after it is possible to know whether the project is advancing quickly or not. This is because P_3 has small switching costs and set-up times. Also, we may want to start by using P_3 and then switch to P_2 in order to speed up the project: combining the two processes gives us the intermediate times and costs. Both these situations occur in the middle of this longest line: P_3 is used at the beginning and then a change to another process occurs (eventually, when it is possible to know how quickly the project is advancing). On the other hand, in the shortest, bottom right line, we have strategies with low costs and, to achieve that, we use P_3 during most of the project. The reason for using P_3 is thus different for the two areas of the plot.

Some examples of strategies in the middle of the longest line that start by using P_3 are E, F and G. Strategy E starts with P_3 and then switches to P_2. Strategies F and G both start by using P_3 and then switch to P_2 (particularly when the project is advancing slowly) or to P_1 (particularly when the project is advancing quickly); in strategy F, some additional changes from P_1 to P_2 occur when the project advances slowly after switching to P_1.

After the strategies are generated, the decision-maker can use a multicriteria method to choose the preferred time/cost combination, and embark on the corresponding strategy.

Let us assume, as an example, that strategy G is chosen. This strategy has a cost of €11256 and a time of 17.95 days. The first process to be used in the strategy is process P_3. If the project advances quickly on the first two days, or slowly on the first day and quickly on the second, then a change to process P_1 occurs on the third day. If the project advances slowly for the first two days, then a change to process P_2 occurs on the third day. If the project advances quickly on the first day and slowly on the second, then a change to P_2 also occurs, but later. This change occurs on the fourth day if the project advances slowly on the third day, it occurs on the fifth day if the project advances quickly on the third day and slowly on the fourth, and it occurs on the sixth day if the project advances quickly on both the third and the fourth day.

6. Conclusions

In this paper we have presented a useful bicriteria model for project analysis that is based on decision trees, and described an application of the model to a production planning problem.

Following a general approach that uses bicriteria decision trees to represent investment projects, a model for the use of time and cost in project analysis was developed. This model assumes that some different processes may be used to undertake a homogeneous task. After the processes are defined, the complete decision tree can be automatically built, and the non-dominated strategies can be identified.

This model usually leads to very large decision trees, whose construction and evaluation may take a long time and require very large amounts of memory space. Therefore, the approach based on the construction and evaluation of the tree is usually impractical, or even impossible. In an attempt to overcome this problem, we have proposed an algorithm for the identification of the non-dominated strategies. The algorithm does not require the construction of the complete decision tree, since it is able to disregard non-interesting branches as soon as they come up, and to avoid the repetition of calculations for similar branches of the tree. The algorithm also provides a more compact representation of the tree, since identical branches located in different points of the tree are represented only once.

The computational tests have shown that the algorithm performs well when the number of non-dominated strategies is small, but it does not perform so well when that number is large. In order to overcome the problem of model usage with a large number of non-dominated strategies, we introduced an error parameter that allows us to avoid generating all the non-dominated strategies that are very close to each other. Using this approximation, we are able to reduce the number of strategies that are generated. This way the performance of the algorithm is improved, and we are able to apply the model to some cases that we would otherwise be unable to handle.

Finally, we applied the model to a production planning problem. We defined the problem, calculated the model parameters, and generated the strategies. We then plotted and analysed the different time/cost combinations that we obtained, and examined which types of strategies led to different combinations of criteria values. This application allowed us to conclude that the model can be easily applied to some problems, providing useful insights about the types of strategy that should be used in order to achieve different combinations of criteria values.

References

Brealey, R.; Myers, S. (2000), *Principles of Corporate Finance*, Boston, McGraw-Hill.

Godinho, P. C. (2003), *Árvores de Decisão Bicritério em Análise de Projectos (PhD Thesis)*, Coimbra, Faculty of Economics of the University of Coimbra.

Godinho, P. C.; Costa, J. P. (2002), «A Note on the Use of Bicriteria Decision Trees in Capital Budgeting», *Global Business & Economics Review*, 4, 1, 147-158.

Hertz, D. B.; Thomas, H. (1983), *Risk Analysis and its Applications*, Chichester, John Wiley and Sons.

Magee, S. (1964), «How to Use Decision Trees in Capital Investment», *Harvard Business Review*, 42, 79-96.

Smith, J. E.; Mccardle, K. F. (1998), «Valuing Oil Properties: Integrating Option Pricing and Decision Analysis Approaches», *Operations Research*, 46, 2, 198-217.

Smith, J. E.; Nau, R. F. (1995), «Valuing Risky Projects: Option Pricing Theory and Decision Analysis», *Management Science*, 41, 5, 795-816.

Trigeorgis, L. (1993), «The Nature of Option Interactions and the Valuation of Investments with Multiple Real Options», *Journal of Financial and Quantitative Analysis*, 28, 1, 1-20.

Appendix – some mathematical properties of the model

This appendix presents two mathematical properties of the model that were used in the development of the algorithm. For the proof of these properties, see Godinho (2003).

Property 1: Let S(i,x), i = 1,…, n, be the set of all strategies that begin with the use of process P_i, can be used to undertake a fraction x of the project, and are not dominated by any other strategy belonging to this same set. Then:

a) if x' > x, s ∈ S(i, x), and s can be used to undertake a fraction x' of the project, then s ∈ S(i, x');

b) if x' > x and ∀ s ∈ S(i, x), s can be used to undertake a fraction x' of the project, then S(i, x') = S(i, x).

Property 2: Let S(i, x), i = 1,…, n, be the set of all strategies that begin with the use of process P_i, can be used to undertake a fraction x of the project, and are not dominated by any other strategy belonging to this same set. Consider a process P_i and a fraction x of the project such that x ∈ $]x_{1,i}, 1[$. Let x' be a fraction of the project such that x' > x, and let E_j, j = 1,…, n, be the set of all values of x that are smaller than $x'-x_{1,i}$ and that cause a modification in S(j, x), that is:

$$E_j = \{e \in \rfloor 0, x' - x_{1,i} \lfloor : \forall \delta > 0, S(j, e) \neq S(j, e + \delta)\}$$

Define the set A in the following way:

$$A = \begin{cases} \bigcup_{j=1}^{n} \{\alpha > 0: \alpha = e + x_{b,i}, e \in E_j, b \in \{1, 2\}\}, & \text{if } x > x_{2,i} \\ \{x_{2,i}\} \cup \bigcup_{j=1}^{n} \{\alpha > 0: \alpha = e + x_{1,i}, e \in E_j\}, & \text{if } x \leq x_{2,i} \end{cases}$$

If x ∈ A and there are no elements of A with values between x and x', then S(i, x) = S(i, x'). That is,

$$A \cap [x, x'[= \varnothing \Rightarrow S(i, x) = S(i, x').$$