M-ARY TREES FOR COMBINATORIAL ASSET MANAGEMENT DECISION PROBLEMS

***Chakraborty S [a], Fidge C [a], Ma L [b] and Sun Y [b]**

[a] Computer Science, Faculty of Science and Technology, Queensland University of Technology, QLD 4101, Australia.

[b] Engineering Systems, Faculty of Built Environment and Eng, Queensland University of Technology, QLD 4101, Australia.

*Corresponding Author - Email: subrata.chakraborty@qut.edu.au, Tel: +61 7 31382442, Fax: +61 7 31381469

Abstract

A novel *m*-ary tree based approach is presented to solve asset management decisions which are combinatorial in nature. The approach introduces a new dynamic constraint based control mechanism which is capable of excluding infeasible solutions from the solution space. The approach also provides a solution to the challenges with ordering of assets decisions.

Keywords: Combinatorial decision problem, *m*-ary tree, brute force technique, asset management.

1. Introduction

Asset management is about finding the right balance between cost, performance and risks to any asset. Asset management helps us to align corporate goals with maintenance spending and to draw up future asset plans (Brown and Spare, 2004). In large organizations with a large number of assets, the task of the asset manager is a challenging one. In this paper we focus on a combinatorial decision support problem relevant to asset maintenance. The problem concerns a group of assets, each with multiple maintenance options, for each of which the decision maker needs to select one option. While making these option selections the decision maker must find the combination that satisfies all technical and business constraints best. The most obvious way to handle this problem is to generate all candidate solutions (combinations of options) and then compare them with respect to the decision constraints. In practice, however, this simple, brute-force algorithm is too computationally intensive for large numbers of assets or maintenance options. The number of candidate solutions and required comparisons increases exponentially with the number of assets. In order to solve such an intractable computational problem we need to significantly reduce the number of potential solutions to be compared by limiting the generation of infeasible solutions.

One of the most popular tree-based approaches in management and decision analysis is the decision tree (Quinlan, 1986, 1999). Decision trees have been successfully used in asset management (Sun et al., 2010; Emerson et al., 2011). The decision tree approach is very efficient for decision problems with small numbers of assets. However decision trees are designed for sequentially guiding choices in order to identify a single solution, not for generating large numbers of equally-acceptable solutions.

In the area of financial optimisation and asset-liability assessment stochastic programming models are also popular. The stochastic programming models generally require developing a scenario tree (Kusy & Ziemba, 1986; Carino et al., 1994; Consigli & Dempster, 1998; Kouwenberg, 2001; Yu et al., 2003). The stochastic scenario tree can be considered as similar to the *m*-ary tree (Drmota, 2009).

Binary trees (Bayer and McCreight, 1972) and their variants have been an integral part of computing since the early days. They are widely used in algorithms, programming, data structures, searching and sorting (Wirth, 1986; Binstock and Rex, 1995; Adamson, 1996; Baldwin and Scragg, 2004). Sorted binary trees have a natural ability to work efficiently with a large number of variables (decisions) due to their ability to rapidly partition the search space. This capacity makes the binary tree based approach suitable to deal with combinatorial decision problems. With a binary tree each parent node has two branches. In the case of asset management decision problems a parent node can be considered to denote an asset and each of its branches as alternative maintenance options for that asset. In practice, however, an asset may have more than just two maintenance options. To handle this we need to use an *m*-ary tree (Drmota, 2009), which is a generalization of binary trees where a parent node has m branches.

In the following sections we first explain the decision problem we are concerned with in mathematical terms followed by the details of an *m*-ary tree suitable for solving the problem. We then present a novel *m*-ary tree based approach to solve the combinatorial asset management decision problem followed by an industry-based case study to demonstrate the applicability of the new approach.

2. Problem Statement

The assumption is that each decision must contain one and only one maintenance option for each asset. Given a set of assets N, each with a set of $a_n (n = 1, 2, ..., N)$ maintenance options, the number of possible decisions D can be expressed as the product

$$D = \prod_{n=1}^{N} a_n \quad (1)$$

The assumption is that the maintenance options for any asset are independent of the maintenance options of other assets, i.e., the choice of one maintenance option for an asset does not constrain the options for another asset. The number of possible decisions D includes all potential solutions including infeasible ones, i.e., options that are invalid because they individually violate some overall constraint or because they are incompatible with other options. With a large number of assets and maintenance actions the value of D can be astronomical. For example, consider 10 assets with 4 maintenance actions for each and the number of possible maintenance decisions will be 4^{10} (= 1,048,576). Generally, large organizations make maintenance decisions for hundreds or even thousands of assets. The number of possible maintenance decisions is extremely large by nature and it is a classic combinatorial problem. In order to find the best possible decision among all the possible decisions we can use a simple brute force technique as follows:

Step 1: Create all possible maintenance decisions.
Step 2: Compare the candidate solutions based on some constraint like cost, time, importance, etc.
Step 3: Exclude non feasible solutions and choose one of the remaining acceptable solutions.

In theory the above technique seems easily achievable. But it is impractical to use it to solve real life problems due to its very high computational requirements. Furthermore, in practice decision makers are not always after the best solution, rather they are satisfied with any solution that meets their criteria. Limited amounts of optimization testing are usually conducted to check the validity of the chosen decision. The algorithm presented in this paper provides a solution to this combinatorial decision problem which can:

a) Reduce the number of maintenance decisions generated.
b) Solve the issue of maintenance decision conflicts dynamically.
c) Apply various decision constraints effectively.

3. M-ary Trees

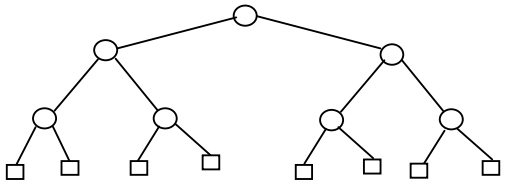

Figure 1: Binary Tree

A binary tree as shown in Figure 1 contains parent nodes (nodes with successors) and child nodes (nodes without any successor). Binary trees are a rooted tree and recursive in nature. In a fully balanced binary tree (Baldwin and Scragg, 2004) with tree height h, the number of leaves L can be calculated as

$$L = 2^h \quad (2)$$

Binary trees can be generalized as *m*-ary rooted trees where $m \geq 2$ is a fixed integer (Drmota, 2009). The number of leaves in an *m*-ary tree can be obtained by modifying Equation 2 as

$$L = m^h \quad (3)$$

Equation 3 can be generalised as

$$L = \prod_{i=1}^{h} m_i \tag{4}$$

In an *m*-ary tree all the parent nodes in each level of the tree have a fixed number of child nodes m. But Equation 4 is capable of calculating the number of leaves in a tree even if the number of child nodes is different for the parent nodes of any particular level of the tree (m may be different for different levels of the tree).

Let us consider an asset management scenario for Equation 4. The set of assets N represents the height of the tree h and the set of maintenance options $a_n (n = 1, 2, ..., N)$ represents the number of child nodes m_i at each level of the tree. The number of decisions D which consists of one maintenance option for each asset can be compared with the number of leaves L. Based on this analogy we can establish that Equation 1 represents the *m*-ary tree where m is variable across assets.

4. Our Approach

Step 1: Get asset list
Step 2: Get maintenance options for each asset
Step 3: Get attributes for each maintenance option
Step 4: Get filtering criteria
Step 5: Get decision constraints
Step 6: Add default maintenance option
Step 7: Filter assets
Step 8: Control solution space
Step 9: Compare decisions

The details of the approach are discussed as follows:

Step 1 - 5: Inputs

The approach requires the following inputs:

a) An asset inventory listing all assets A.
b) A set of maintenance options for each asset $a_n (n = 1, 2, ..., A)$.
c) Attributes of each maintenance option that influence the decision outcome. Examples of attributes can be time, cost, safety, operational significance, etc.
d) A set of asset filtering criteria F. Examples of filtering criteria can be geographical location, known operational constraints, time of the year, etc.
e) Decision constraints C such as
   a. Incompatible maintenance options.
   b. Limit constraints (time, cost, resources, safety, etc.)

Step 6: Add default maintenance option

In this stage a default maintenance option is added to each asset. The default option is considered as 'No Action' for the asset. It has no cost associated with it and it does not require any time or any other resources. The key benefit is that it allows reaching all possible solutions without creating multiple tree structures for making asset decisions in different order.

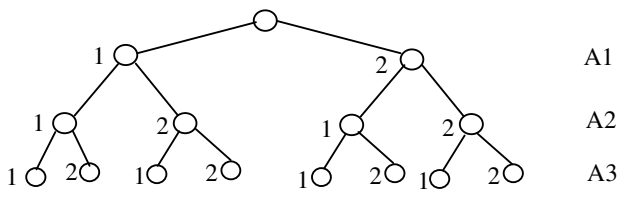

Figure 2: Three assets with two options each

Considering Figure 2, which shows the potential search space generated when sequentially making decisions for an inventory containing 3 assets. If we want to make a decision consisting of options from assets A2 and A3, we must make a decision for asset A1 as well. The only way to make any decision with options from A2 and A3 only and not with A1 is to

restructure the tree with starting node being A2 or A3. Thus to reach all possible decision scenarios we may need to develop multiple tree structures. The addition of the default option to each asset provides access paths to all possible solutions without the need to create several tree structures for each possible ordering of the assets. Figure 3 shows the improvement from Figure 2 that is capable of providing access to all possible decisions. X represents the default option. When selected, it indicates that no initially given option will be selected for that asset and yet it will provide access to other asset options to the lower levels of the tree.

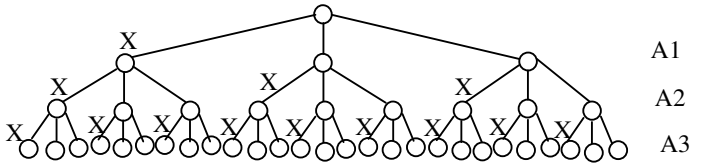

Figure 3: Three assets with two options and added default option

Step 7: Filter Assets

From the asset inventory A, the set of assets N relevant to current decision making can be filtered based on given filtering criteria F as the elements *x* for which the filtering criteria are true, i.e.,

$$N = \{x \mid x \in A \wedge F(x)\} \tag{5}$$

Step 8: Control Solution Space

The complete solution space contains all the possible decisions (combinations of maintenance options for each asset) D, i.e., all the paths from the root of the state space tree to its leaves. The control stage is a directed search through the complete solution space D excluding branches consisting of infeasible decisions. This process dynamically reduces the solution space by not generating infeasible decisions and their offspring. The set of feasible decisions d can be obtained as the set of elements *x* for which the decision criteria hold, i.e.

$$d = \{x \mid x \in D \wedge C(x)\} \tag{6}$$

Step 9: Compare Decisions

The comparison is essentially a sorting operation for the set of feasible decisions d based on comparison criteria which may include control criteria thresholds. Due to the dynamic control applied in Step 2 the number of comparisons required can reduce significantly depending on the decision problem.

5. Case Analysis

In this section we present a small example to demonstrate the potential gains provided by our approach.

5.1 Case Description

A large Australian power generating corporation currently maintains thousands of assets as part of their regular operation. They often face the challenge of deciding which assets to maintain, when to maintain them, and how several assets can be grouped together for maintenance. The major constraints and issues they must consider include available maintenance time, costs, and remaining time for safe operation, resource availability, and operational criticality.

In order to illustrate the approach, we will solve a much downsized version of the actual decision problem. The solution will be obtained by applying a single decision constraint. The inputs to the algorithm are as follows:

a) The asset inventory listing all available assets A which contains several thousand assets.
b) The set of available maintenance options for each asset $a_n (n = 1, 2, ..., A)$ whose number varies for different assets. The operational criticality of each asset is also available.
c) The available attributes of each maintenance option that influence the decision outcome. The most significant attributes considered are cost, safety and time.
d) The set of asset filtering criteria F. Major filtering is done based on operational area segments in the plant and available major downtime.
e) Decision constraints C such as
    1) Incompatible maintenance constraints: Not considered currently.
    2) Limiting constraints: Cost and time limits are available.

5.2 Solution using the algorithm

Steps 1-6 were applied to get all the required inputs and to add the default option for each asset. The default option is denoted as option 1 for each asset.

Step 7: Filter Assets

Based on the power plant segmentation we were able to filter the asset inventory. The filtered asset set N contains a few hundred assets. For the sake of this example, let us assume that we were able to subdivide one of the plant segments into a much smaller area consisting of just 4 assets. Each asset has 3 maintenance options including the default option added using Step 6. The related cost attributes of each option are presented in Table 1.

Table 1: Assets, maintenance options and cost

Asset	*Maintenance Option (O)*	*Cost ($ 100k)*
A1	1	0
	2	10
	3	40
A2	1	0
	2	5
	3	15
A3	1	0
	2	18
	3	30
A4	1	0
	2	20
	3	35

Step 8: Control

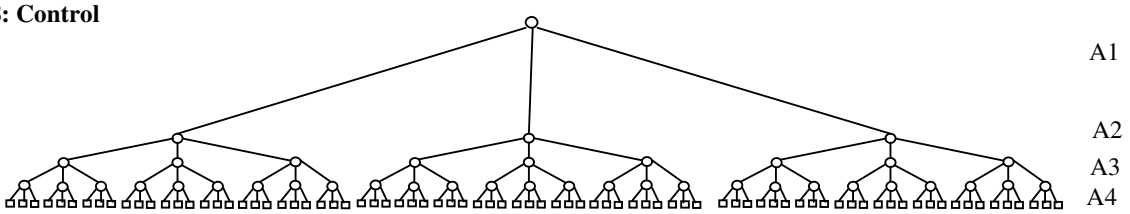

Figure 4: Tree for complete solution space

With 4 assets having 3 maintenance options each, if we apply the brute force technique we will be generating 81 decisions as shown in Figure 4. The decisions and associated costs are listed in Table 2 which is our complete solution space for this decision problem.

Table 2: Decisions and cost

Decision	*Asset Options*	*Cost ($ 100k)*
D1	A11 A21 A31 A41	0
D2	A11 A21 A32 A41	18
D3	A11 A21 A33 A41	30
D4	A11 A22 A31 A41	5
D5	A11 A22 A32 A41	23
D6	A11 A22 A33 A41	35
D7	A11 A23 A31 A41	15
D8	A11 A23 A32 A41	33
D9	A11 A23 A33 A41	45
D10	A12 A21 A31 A41	10
D11	A12 A21 A32 A41	28
D12	A12 A21 A33 A41	40
D13	A12 A22 A31 A41	15
D14	A12 A22 A32 A41	33
D15	A12 A22 A33 A41	45
D16	A12 A23 A31 A41	25
D17	A12 A23 A32 A41	43

Decision	*Asset Options*	*Cost ($ 100k)*
D18	A12 A23 A33 A41	55
D19	A13 A21 A31 A41	40
D20	A13 A21 A32 A41	58
D21	A13 A21 A33 A41	70
D22	A13 A22 A31 A41	45
D23	A13 A22 A32 A41	63
D24	A13 A22 A33 A41	75
D25	A13 A23 A31 A41	55
D26	A13 A23 A32 A41	73
D27	A13 A23 A33 A41	85
D28	A11 A21 A31 A42	20
D29	A11 A21 A32 A42	38
D30	A11 A21 A33 A42	50
D31	A11 A22 A31 A42	25
D32	A11 A22 A32 A42	43
D33	A11 A22 A33 A42	55
D34	A11 A23 A31 A42	35

D35	A11 A23 A32 A42	63
D36	A11 A23 A33 A42	65
D37	A12 A21 A31 A42	30
D38	A12 A21 A32 A42	58
D39	A12 A21 A33 A42	60
D40	A12 A22 A31 A42	35
D41	A12 A22 A32 A42	53
D42	A12 A22 A33 A42	65
D43	A12 A23 A31 A42	45
D44	A12 A23 A32 A42	63
D45	A12 A23 A33 A42	75
D46	A13 A21 A31 A42	60
D47	A13 A21 A32 A42	78
D48	A13 A21 A33 A42	90
D49	A13 A22 A31 A42	65
D50	A13 A22 A32 A42	83
D51	A13 A22 A33 A42	95
D52	A13 A23 A31 A42	65
D53	A13 A23 A32 A42	93
D54	A13 A23 A33 A42	105
D55	A11 A21 A31 A43	35
D56	A11 A21 A32 A43	53
D57	A11 A21 A33 A43	65
D58	A11 A22 A31 A43	40

D59	A11 A22 A32 A43	68
D60	A11 A22 A33 A43	70
D61	A11 A23 A31 A43	50
D62	A11 A23 A32 A43	68
D63	A11 A23 A33 A43	80
D64	A12 A21 A31 A43	45
D65	A12 A21 A32 A43	63
D66	A12 A21 A33 A43	75
D67	A12 A22 A31 A43	50
D68	A12 A22 A32 A43	68
D69	A12 A22 A33 A43	80
D70	A12 A23 A31 A43	70
D71	A12 A23 A32 A43	78
D72	A12 A23 A33 A43	90
D73	A13 A21 A31 A43	75
D74	A13 A21 A32 A43	93
D75	A13 A21 A33 A43	105
D76	A13 A22 A31 A43	80
D77	A13 A22 A32 A43	98
D78	A13 A22 A33 A43	110
D79	A13 A23 A31 A43	90
D80	A13 A23 A32 A43	108
D81	A13 A23 A33 A43	120

To apply our new approach let us assume we have a budget limit of $3,000,000. Our constraint here is the cost constraint for control purposes. Based on this constraint if we start generating the tree we will only generate the non circled region as shown in Figure 5. The decisions that are excluded from this control phase without even generating them are circled in red. The feasible decisions generated are listed in Table 3.

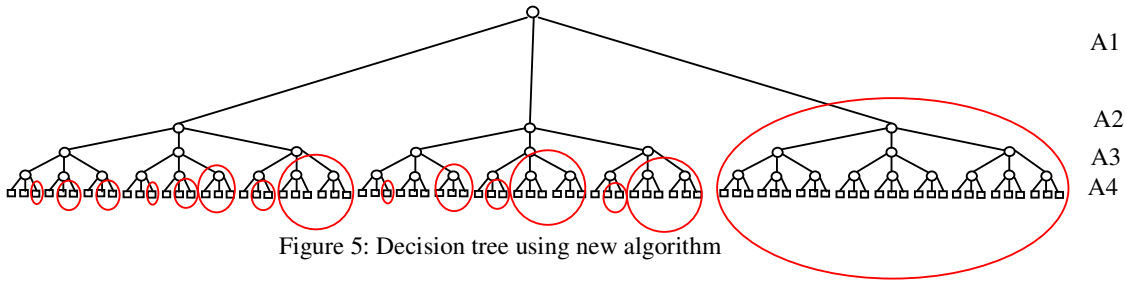

Figure 5: Decision tree using new algorithm

Table 3: Feasible decisions and cost

Decision	*Asset Options*	*Cost (AUD in 100,000)*
D1	A11 A21 A31 A41	0
D2	A11 A21 A32 A41	18
D3	A11 A21 A33 A41	30
D4	A11 A22 A31 A41	5
D5	A11 A22 A32 A41	23
D7	A11 A23 A31 A41	15
D10	A12 A21 A31 A41	10
D11	A12 A21 A32 A41	28
D13	A12 A22 A31 A41	15
D16	A12 A23 A31 A41	25
D28	A11 A21 A31 A42	20
D31	A11 A22 A31 A42	25
D37	A12 A21 A31 A42	30

Step 9: Compare decisions

An appropriate sorting technique can be used to find the best decision among the feasible solutions. Comparing Tables 2 and 3 we can see that if we use the brute force technique we will end up comparing 27 decisions to each other. On the other hand with the new algorithm we need to compare only the 10 feasible decisions. This certainly reduces the computational burden.

6. Conclusion:

The new approach proposed in this paper can be of practical use in the case of asset management decision problems where a large number of assets are involved. The dynamic control does require some additional computations but the benefits far outweigh this extra effort by eliminating large numbers of infeasible decisions without even assessing them. Performance of the algorithm is dependent on the shape of the tree and the proportion of infeasible solutions. Decision problems with higher numbers of infeasible solutions will achieve better efficiency with this algorithm. Further comparative studies are being carried out to understand the actual computational efficiency of the algorithm.

References:

1. Adamson, I T. (1996) Data structures and algorithms: a first course. London, New York: Springer.
2. Baldwin, D and Scragg, G. (2004) Algorithms and Data Structures: The Science of Computing. Hingham, MA, USA: Charles River Media.
3. Bayer, R and McCreight, E. (1972) Organization and maintenance of large ordered indexes. *Acta Informatica*, 1 (3), 173–189.
4. Binstock, A and Rex J. (1995) Practical algorithms for programmers. Reading, Mass: Addison-Wesley.
5. Brown, RE Spare, JH. (2004) Asset management, risk, and distribution system planning. *Power Systems Conference and Exposition, 2004*. IEEE PES, 3, 1681- 1686.
6. Carino, DR, Kent, T, Myers, DH, Stacey, C, Watanabe, K & Ziemba, WT. (1994) Russell-Yasuda Kasai model: an asset-liability model for a Japanese insurance company using multi-stage stochastic programming. *Interfaces*, 24 (1), 29-.
7. Consigli, G & Dempster, M A H. (1998) Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, 81(), 131-.
8. Drmota, M (2009). Random Trees. Dordrecht: Springer Vienna.
9. Emerson, D., Nayak, R., Weligamage, J. and Piyatrapoomi, N. (2011). Identifying differences in wet and dry road crashes using data mining. In J Mathew et al. (Eds.) Engineering Asset Management and Infrastructure Sustainability: *Proceedings of the Fifth World Congress on Engineering Asset Management (WCEAM 2010, Brisbane)*. London: Springer-Verlag London Ltd.
10. Kouwenberg, R. (2001) Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research*, 134 (2), 279-
11. Kusy, M & Ziemba, W. (1986) A bank asset and liability management model. *Operations Research*, 34(3), 356-.
12. Quinlan, J R. (1986) Induction of decision trees, *Machine Learning*, (1), 81-106
13. Quinlan, J R (1999). Simplifying decision trees. *International Journal of Human-Computer Studies*, 51 (2), 497-.
14. Sun, Y, Ma, L, & Fidge, C. (2010) Using decision trees in economiser repair decision making. *Proceedings of 2010 Prognostics & System Health Management Conference. Macau, China: IEEE*, p.MU3037, 2010.
15. Wirth, N. (1986) Algorithms and data structures. Englewood Cliffs, N.J.: Prentice-Hall.
16. Yu, L-Y, Ji, X-D & Wang, S-Y. (2003) Stochastic programming models in financial optimization: A survey, *Advanced Modeling and Optimization*, **5** (1), 1–26.