OPTIMAL SEARCH ON A STOCHASTIC TREE WITH AN APPLICATION TO MULTI-PHASED R&D SCHEDULING

by

U. G. ROTHBLUM*
and
L. VAN DER HEYDEN**

96/05/TM

* Professor, at Technion, Israel Institute of Technology, Haifa 32000, Israel.

** Professor of Technology Management, at INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

Printed at INSEAD, Fontainebleau, France.

95/12/13

OPTIMAL SEARCH ON A STOCHASTIC TREE

WITH AN APPLICATION TO MULTI-PHASED R&D SCHEDULING

by

Uriel G. Rothblum
Faculty of Industrial Engineering and Management
Technion - Israel Institute of Technology
Haifa 32000, ISRAEL

and

Ludo Van der Heyden
Faculty of Technology Management
INSEAD
Fontainebleau 77305 Cedex, FRANCE

December 1995*

* An early version of this paper was presented at the ORSA/TIMS Conference held in Vancouver in 1989.

OPTIMAL SEARCH ON A STOCHASTIC TREE WITH AN APPLICATION TO MULTI-PHASED R&D SCHEDULING

by

Uriel G. Rothblum and Ludo Van der Heyden

Abstract

We consider the problem of determining a path of feasible edges from the root of a directed tree to any one of its endpoints so as to minimize the expected search cost. Each edge in the tree is characterized by an exploration cost and a feasibility probability. When an edge is explored and found infeasible, all paths using this edge become infeasible too, and other paths must be looked for.

An algorithm for solving the problem is provided. The obtained solution exhibits the dynamic aspect of searches in this setting. In particular, one may encounter repeated jumping between different parts of the tree.

This model was motivated by the problem of determining an optimal schedule for an R&D project presenting several options where each can be decomposed into multiple phases satisfying precedence relationships. The model's solution demonstrates that in such situations the optimal R&D schedule exhibits a parallel, adaptive structure: at any moment in the search, several options might be "at hand" and might in turn be reactivated as infeasibility is revealed amongst certain of these options.

Key words: Stochastic tree
Search theory
Stochastic routing
Stochastic scheduling
R&D scheduling

0. Prologue

Our purpose is to study a probabilistic search problem defined on a rooted directed tree. Every edge of the tree is associated with an *exploration cost* and a *feasibility probability*; the latter represents the probability that the edge is found feasible when explored. The search consists of finding a path of feasible edges from the root of the tree to one of its endpoints so as to minimize the expected cost of the search. A search consists of selecting a sequence of edges and determining, in turn, whether they are feasible of not; in particular, the selected edge at any given stage typically depends on the results of previous explorations. Should an edge be found infeasible, it is pointless to explore any edge in the subtree rooted at the endpoint of this edge. The search-sequences are further constrained by a formal (rather than derived) connectivity requirement, namely, that an edge selected at any particular stage for exploration must be attached to the root of the tree, or be an immediate successor to an edge already explored.

To illustrate the issues to be dealt with, we turn to an example, presented in Figure 1. Here, numbers above edges are used for enumeration and the pair of numbers underneath the edge represents its exploration cost and feasibility probability. The first choice that presents itself

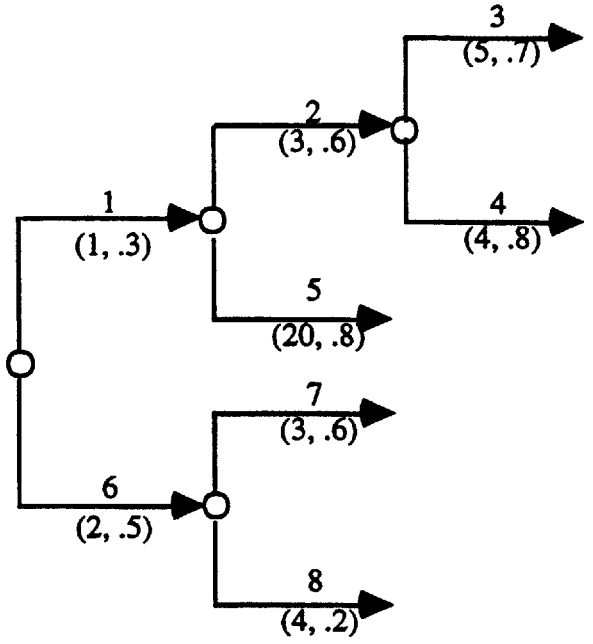

Figure 1

is between edges 1 and 6. On the basis of exploration cost, edge 1 is cheaper than edge 6. But, taking into account risk, the opposite is true as the feasibility probability is higher for edge 6 than for edge 1. The tradeoff between risk and cost is only one of the issues influencing the choice. One also needs to consider the expected cost past successful exploration of these first edges. The successor edges of edge 1 look jointly very promising. The probability of finding a feasible path to an endpoint in the subtree rooted at the endpoint of edge 1 is

$$.8 + .2 \times .6 \times (.8 + .2 \times .7) = .913 .$$

In comparison, the probability of finding a feasible path to an endpoint in the subtree rooted at edge 6 is

$$.2 + .8 \times .6 = .68 .$$

The cost factor mitigates these considerations. For example, it does not look worth it to pay 20 for exploring edge 5 unless one has exhausted the other cheaper options.

More generally, what needs to be addressed for exploring a particular edge is not just its own cost and feasibility characteristics, but also the exploration value of paths from the tail of the given edge to endpoints of the subtree rooted at the tail of that edge. The central question in this paper, indeed, centers on the valuation of an edge in view of the multiplicity of the possible paths from its tail to the endpoints of the tree.

We next motivate the problem that we are about to address by describing several applications, and citing relevant literature. An immediate application of the model concerns the scheduling of multi-phase R&D projects. The edges correspond to possible phases of a project, and edges that are adjacent to the endpoint of a given edge represent options that are available once a phase has successfully been completed. The model assumes that resource constraints are such that the R&D effort proceeds one phase at a time. An important contribution of the paper is the description of a model which exhibits the dynamic and adaptive aspect of R&D behavior

under uncertainty. In such situations, particular options may have been explored for a while, until failure on a particular component redirected the effort. The options are dismissed only when it is clear that feasibility is ruled out. When this is not the case, each of these options may be returned to and examined further when the other "parallel" options in the tree no longer offer a less costly opportunity for determining a feasible path to an endpoint of the tree.

A model similar to the one presented in the current paper, and enriched by time considerations and terminal values at the endpoints of all edges, has been studied by Granot and Zuckerman [1991]. However, their model does not consider the possibility of leaving part of a subtree for subsequent examination, as a function of the exploration of other subtrees. The dynamic structure of their solution is therefore different from the one obtained here.

The R&D interpretation makes it clear that, in operations research parlance, the problem can be labeled a "treasure hunt on a tree," the paths from the root of the tree to the endpoints representing the different ways toward a treasure that is accessed through any endpoint of the tree. The edges might represent different physical areas to cross, and one interesting application might involve time as the cost factor. Another interpretation of the problem involves routing of data, phone calls, vehicles, etc. from the root of the tree, representing the origin, to any endpoint of the tree. The tree structure hints at some form of local routing, where the issue is the optimal access at any of the endpoints to a higher level infrastructure (like a highway). These examples suggest that the problem we consider is, in fact, more characteristic of "exit" rather than "treasure hunting".

The simplest tree to be considered is that having only one set of edges incident to the root as shown in Figure 2 below. This problem was already solved by Smith [1956], who established that the optimal sequence could be found by computing the normalized costs c_i / p_i, $i = 1, ..., n$, and exploring edges in order of nondecreasing normalized cost. This result is interesting in that it shows that the tradeoff between cost and risk is resolved by a simple division of the cost and risk parameters. Our paper generalizes that result to a directed tree.

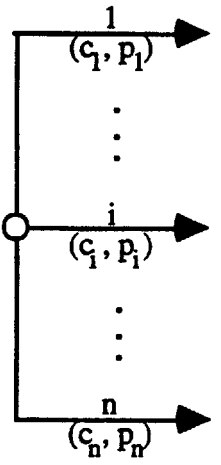

Figure 2

1. Introduction: Model Formulation and Statement of Main Result

A *directed graph* is a pair $\mathcal{G} = (N, E)$ where N is finite set and $E \subseteq N \times N$. The elements of N and E are called, respectively, *nodes* and *edges* (of graph $\mathcal{G}$). For each edge $e = (i, j) \in E$, we refer to i and j, respectively, as the *tail* and the *endpoint* of e. A *directed path* in $\mathcal{G}$ is a finite sequence of edges $e_1, e_2, ..., e_q$ where for each $i = 1, ..., q - 1$, the endpoint of e_i is the tail of e_{i+1}; in this case, we refer to the tail of e_1 and the endpoint of e_q as the *tail* and the *endpoint* of the directed path, respectively, and say that the directed path is *from* its tail *to* its endpoint. Such a directed path is called a *directed cycle* if the endpoint of e_q coincides with the tail of e_1. We say that a directed graph (N, E) is a *rooted directed tree* with *root* r if it does not have a directed cycle and each node $i \in N \setminus \{r\}$ is the endpoint of exactly one edge, while no edge has the root r as endpoint. A directed rooted tree has for each node a unique directed path from the root to that node.

Throughout we assume that a rooted directed tree (N, E) with root r is given. The tree structure defines a precedence relation on the edge set E where an edge e_1 is said to *precede* an edge e_2 if $e_1 \neq e_2$ and if there is a path from the tail of e_1 to the endpoint of e_2. In that case, we also say that e_1 is a *predecessor* of e_2, that e_2 *succeeds* e_1, and that e_2 is a *successor* or e_1. If the endpoint of e_1 is the tail of e_2, we say that e_1 is the (unique) *immediate predecessor* of e_2 and that e_2 is an *immediate successor* of e_1. Of course, being an immediate successor (resp., predecessor) implies being a successor (resp., predecessor). An *endpoint* (of the given tree) is a node in N which is not the tail of any edge, and a *leaf* (of the given tree) is an edge in E whose endpoint is an endpoint of the tree. The set of leaves will be denoted L.

In addition to the given rooted directed tree, the data for our problem includes two real valued functions on the set of edges called the *exploration cost* and *feasibility probability*. The exploration cost assigns to an edge e a positive number that is denoted c_e and to which we refer as the *exploration cost of* e. The feasibility probability assigns to an edge e a number in the open interval $(0, 1)$ that is denoted p_e and to which we refer as the *feasibility probability of*

e . The interpretation of these functions is as follows. Each edge can be in one of two states - *feasible* or *infeasible*, and the state of edge e is a (binary) random variable with probability p_e of being feasible and probability $1 - p_e$ of being infeasible. The random variables associated with the edges are assumed to be stochastically independent. Also, c_e represents the cost of exploring edge e , where exploration determines whether or not edge e is feasible. (Relaxations allowing boundary values $p_e = 1$ and/or $c_e = 0$ are considered in Section 7.)

The decision process consists of consecutive exploration of edges and determining their feasibility. The process is restricted by the requirement that at each stage an edge can be selected for exploration only if all its predecessors (if any) have been explored and found feasible. The goal is to find, with minimal expected cost, a directed path of feasible edges from the root to an endpoint (of the tree) or to determine that no such path exists.

A *policy* for the decision problem described in the above paragraphs is a rule that determines the edge that is to be explored for feasibility at each stage as a function of all the available information at that stage, namely, the sequence of edges that have been explored so far and the outcome of these explorations. The structure of a policy can be complex. By definition, a policy satisfies the requirement that all predecessors of an edge that is to be explored have already been explored and found feasible.

We will consider a class of polices referred to as *order policies*. A policy in this class is determined by an ordered list of the edges. The order is consistent with the precedence relation, that is, each edge appears in the list after all of its predecessors. Each such list determines a policy with the following implementation. At the first stage, the first edge in the given list is explored and eliminated from the list. If the edge is found feasible, the next edge in the list is explored; alternatively, if the edge is found infeasible all of its successors are eliminated as well. The process is continued iteratively. At each stage one explores the first uneliminated edge in the list and then eliminates that edge; further, if the exploration finds the edge infeasible, all successors of that edge are eliminated as well, while in the other case the exploration continues

with the next edge in the list. The process stops when a leaf is explored and is found feasible, or when all edges have been eliminated and no feasible leaf has been found.

The contribution of this paper is the construction of an order policy which is optimal in the class of all policies. Our construction is based on first computing a scalar, called index, for each edge of the tree. These indices are then used to order the edges in the following way. At the beginning the available edges are those without predecessors. At each stage, the policy we construct selects the edge with the lowest index among the edges available for selection at that time. The selected edge is then dropped from the set of available edges. If the selected edge is found feasible, all of its immediate successors become available; if the selected edge is found infeasible, none of its successors will ever become available. It is easily seen that the set of available edges at each stage are those that have not yet been explored and, in addition, all of their immediate predecessors, if any, have been explored and were found feasible. Policies which rely on indices have been shown to be optimal for an important class of scheduling problems, called bandit problems (see Gittins and Jones [1974], Whittle [1982] and Katehakis and Veinott [1987]).

The rest of the paper is organized as follows. In Section 2 and 3 we develop some preliminaries and study characteristics of sequences of edges. These are used in Section 4 to describe and analyze the Index Computation Algorithm. These edge-indices are applied in Section 5 to the construction of an order policy for the scheduling problem; this policy is shown to be optimal. The analysis in Sections 4 to 6 is conducted in the context of a family of problems defined on subsets of edges called upsets. This embedding facilitates the application of dynamic programming arguments. A nondegeneracy assumption which simplifies the analysis is initially imposed; it is removed in Section 6. Finally, the example of the current section is solved in Section 7 and our concluding remarks are presented in Section 8.

2. *Expected Cost and Feasibility Probability of Edge-Sequences*

In order to prepare for applying dynamic programming to the analysis of our problem we consider edge-sequences subject to two requirements - that the edges in the sequence are distinct and that no edge in the sequence precedes any one of its predecessors. Let Γ denote the set of all such edge-sequences. We shall use script capital letters to denote edge-sequences, e.g., $\mathcal{U}, \mathcal{V}, \mathcal{W}$. The empty list, denoted $\emptyset$, is in Γ. In this section we establish some preliminaries on cost and probability computations on edge-sequences in Γ.

Given $\mathcal{U} = (e_1, e_2, ..., e_t) \in \Gamma$, we denote the set of edges appearing in $\mathcal{U}$ by $\underline{\mathcal{U}}$, that is, $\underline{\mathcal{U}} = \{e_1, e_2, ..., e_t\}$. So, underlined script capital letters - as well as regular type capital letters - are used to denote (unordered) sets of edges.

Given $\mathcal{U} \in \Gamma$, consider the iterative search process where at each stage the first uneliminated edge in $\mathcal{U}$ is explored and then eliminated; further, if the explored edge is found to be infeasible, all its successors in $\underline{\mathcal{U}}$ are eliminated as well. The process stops when either a leaf is explored and found feasible, or when all edges in $\underline{\mathcal{U}}$ have been eliminated. This search process is referred to as *the search process associated with* $\mathcal{U}$. When $\underline{\mathcal{U}} = E$ this exploration process is that of an order policy to which we refer as the *order policy associated with* $\mathcal{U}$.

For $\mathcal{U} \in \Gamma$ and $e \in \underline{\mathcal{U}}$, let $P^{\mathcal{U}}(e)$ be the probability that edge e will be explored under the search process associated with $\mathcal{U}$. Of course, these probabilities depend not only on the set $\underline{\mathcal{U}}$ of edges listed in $\mathcal{U}$, but also on the order $\mathcal{U}$ imposes on these edges.

The *expected cost associated with* $\mathcal{U} \in \Gamma$, denoted $C(\mathcal{U})$, is the expected cost of the search process associated with $\mathcal{U}$ and is given by

$$C(\mathcal{U}) = \sum_{e \in \underline{\mathcal{U}}} c_e P^{\mathcal{U}}(e) \tag{2.1}$$

(where the empty sum is defined to be 0). If $\underline{\mathcal{U}} = E$, $C(\mathcal{U})$ is the expected cost of the order policy associated with $\mathcal{U}$.

The *success probability associated with* $\mathcal{U} \in \Gamma$, denoted $P(\mathcal{U})$, is the probability that the search process associated with $\mathcal{U}$ will stop after exploring a feasible leaf (rather than after eliminating all edges); it is given by

$$P(\mathcal{U}) = \sum_{e \in L \cap \underline{\mathcal{U}}} p_e P^{\mathcal{U}}(e), \tag{2.2}$$

with L as the set of leaves of the tree. We observe that $P(\mathcal{U})$ equals the probability that $\underline{\mathcal{U}}$ contains a feasible leaf all of whose predecessors in $\underline{\mathcal{U}}$ are feasible; hence, $P(\mathcal{U})$ depends on $\mathcal{U}$ only through the set $\underline{\mathcal{U}}$ of edges that are listed in $\mathcal{U}$, that is,

$$\mathcal{U}, \mathcal{U}' \in \Gamma \text{ and } \underline{\mathcal{U}} = \underline{\mathcal{U}}' \Rightarrow P(\mathcal{U}) = P(\mathcal{U}'). \tag{2.3}$$

As $0 < p_e < 1$ for each edge e, $P(\mathcal{U}) < 1$ for each $\mathcal{U} \in \Gamma$; also, $P(\mathcal{U}) > 0$ if and only if $\underline{\mathcal{U}}$ contains a leaf. When $\mathcal{U}$ consists of a single edge e, that is $\mathcal{U} = (e)$, we use the notation $P(e)$ and $C(e)$ for $P(\mathcal{U})$ and $C(\mathcal{U})$, respectively. Of course, $P(e) = p_e$ if $e \in L$ and $P(e) = 0$ if $e \in E \setminus L$; also, $C(e) = c_e$ for all $e \in E$.

Given two edge-sequences $\mathcal{U} = (e_1, e_2, ..., e_t)$ and $\mathcal{V} = (f_1, f_2, ..., f_s)$, we denote by $(\mathcal{U}, \mathcal{V})$ the merged sequence $(e_1, e_2, ..., e_t, f_1, f_2, ..., f_s)$. We note that $(\mathcal{U}, \mathcal{V}) \in \Gamma$ if and only if: (i) $\underline{\mathcal{U}} \cap \underline{\mathcal{V}} = \emptyset$, (ii) both $\mathcal{U}$ and $\mathcal{V}$ are in Γ, and (iii) no edge listed in $\mathcal{U}$ is a predecessor of an edge listed in $\mathcal{V}$; in this case

$$P^{(\mathcal{U},\mathcal{V})}(e) = P^{\mathcal{U}}(e) \text{ for each } e \in \underline{\mathcal{U}}. \tag{2.4}$$

As the operation of merging edge-sequences is associative, we omit parentheses when merging multiple edge-sequences, e.g., we write $(\mathcal{U},\mathcal{V},\mathcal{W})$ for $((\mathcal{U},\mathcal{V}), \mathcal{W})$. Also, for an edge $e \in E$, we omit parentheses when we merge (e) with other edge-sequences, e.g., we write $(\mathcal{U}, e, \mathcal{V})$ for $(\mathcal{U}, (e), \mathcal{V})$.

The next three lemmas explore the expected cost and success probabilities of edge-sequences which result from the merging operation.

Lemma 2.1.

Let $\mathcal{U}$, $\mathcal{U}'$ and $\mathcal{V}$ be edge-sequences with $(\mathcal{U}, \mathcal{V}) \in \Gamma$, $\mathcal{U}' \in \Gamma$ and $\underline{\mathcal{U}} = \underline{\mathcal{U}'}$. Then $(\mathcal{U}', \mathcal{V}) \in \Gamma$ and

$$C(\mathcal{U}, \mathcal{V}) - C(\mathcal{U}', \mathcal{V}) = C(\mathcal{U}) - C(\mathcal{U}') . \tag{2.5}$$

Proof.

The fact that $(\mathcal{U}', \mathcal{V}) \in \Gamma$ is trite. We first show that

$$P^{(\mathcal{U},\mathcal{V})}(e) = P^{(\mathcal{U}',\mathcal{V})}(e) \text{ for every } e \in \underline{\mathcal{V}}. \tag{2.6}$$

Indeed, let $e \in \underline{\mathcal{V}}$. The search associated with $(\mathcal{U}, \mathcal{V})$ will explore edge e if and only if e has no infeasible predecessor in the union of the edges in $\underline{\mathcal{U}}$ and the edges in $\underline{\mathcal{V}}$ that are listed ahead of e. The latter event is invariant under the exchange of $\mathcal{U}$ and $\mathcal{U}'$, hence, so is the former; we conclude equality of the corresponding probabilities, verifying (2.6). Next, (2.4) and (2.6) imply that

$$C(\mathcal{U}, \mathcal{V}) - C(\mathcal{U}', \mathcal{V}) = \sum_{e\in \underline{\mathcal{U}}} c_e P^{(\mathcal{U},\mathcal{V})}(e) + \sum_{e\in \underline{\mathcal{V}}} c_e P^{(\mathcal{U},\mathcal{V})}(e) - \sum_{e\in \underline{\mathcal{U}}} c_e P^{(\mathcal{U}',\mathcal{V})}(e) - \sum_{e\in \underline{\mathcal{V}}} c_e P^{(\mathcal{U}',\mathcal{V})}(e)$$

$$= \sum_{e\in \underline{\mathcal{U}}} c_e P^{\mathcal{U}}(e) - \sum_{e\in \underline{\mathcal{U}'}} c_e P^{\mathcal{U}}(e) = C(\mathcal{U}) - C(\mathcal{U}') . \quad \|$$

Two edges-sequences $\mathcal{U}$ and $\mathcal{V}$ are called *interchangeable* if $\underline{\mathcal{U}} \cap \underline{\mathcal{V}} = \emptyset$ and both $(\mathcal{U}, \mathcal{V})$ and $(\mathcal{V}, \mathcal{U})$ are in Γ, that is, no edge of $\underline{\mathcal{U}}$ is a predecessor of an edge of $\underline{\mathcal{V}}$ or vice versa.

Lemma 2.2.

Let $\mathcal{U}$ and $\mathcal{V}$ be two interchangeable edge-sequences. Then

$$C(\mathcal{U}, \mathcal{V}) = C(\mathcal{U}) + [1 - P(\mathcal{U})]C(\mathcal{V}) \tag{2.7}$$

and

$$P(\mathcal{U}, \mathcal{V}) = P(\mathcal{U}) + [1 - P(\mathcal{U})]P(\mathcal{V}) . \tag{2.8}$$

Proof.

We first prove that

(2.9) $$P^{(\mathcal{U},\mathcal{V})}(e) = [1 - P(\mathcal{U})]P^{\mathcal{V}}(e) \text{ for every edge } e \in \underline{\mathcal{V}}.$$

Let $e \in \underline{\mathcal{V}}$. As $\mathcal{U}$ and $\mathcal{V}$ are interchangeable, $(\mathcal{U}, \mathcal{V}) \in \Gamma$ and no edge in $\underline{\mathcal{U}}$ is a predecessor of an edge in $\underline{\mathcal{V}}$, implying that no edge of $\underline{\mathcal{V}}$ is eliminated while exploring edges of $\underline{\mathcal{U}}$. Considering the exploration process corresponding to $(\mathcal{U}, \mathcal{V})$ and conditioning on the event that $\underline{\mathcal{U}}$ has no feasible leaf all of whose predecessors are feasible we get that $P^{(\mathcal{U},\mathcal{V})}(e) = [1 - P(\mathcal{U})]P^{\mathcal{V}}(e)$, establishing (2.9). Next, from (2.4) and (2.9)

$$C(\mathcal{U}, \mathcal{V}) = \sum_{e\in \underline{\mathcal{U}}} c_e P^{(\mathcal{U},\mathcal{V})}(e) + \sum_{e\in \underline{\mathcal{V}}} c_e P^{(\mathcal{U},\mathcal{V})}(e)$$

$$= \sum_{e\in \underline{\mathcal{U}}} c_e P^{\mathcal{U}}(e) + \sum_{e\in \underline{\mathcal{V}}} c_e [1 - P(\mathcal{U})]P^{\mathcal{V}}(e)$$

$$= C(\mathcal{U}) + [1 - P(\mathcal{U})]\Big[\sum_{e\in \underline{\mathcal{V}}} c_e P^{(\mathcal{V})}(e)\Big] = C(\mathcal{U}) + [1 - P(\mathcal{U})]C(\mathcal{V}),$$

and

$$P(\mathcal{U}, \mathcal{V}) = \sum_{e\in L\cap \underline{\mathcal{U}}} p_e P^{(\mathcal{U},\mathcal{V})}(e) + \sum_{e\in L\cap \underline{\mathcal{V}}} p_e P^{(\mathcal{U},\mathcal{V})}(e)$$

$$= \sum_{e\in L\cap \underline{\mathcal{U}}} p_e P^{\mathcal{U}}(e) + \sum_{e\in L\cap(\{b\}\cup \underline{\mathcal{V}})} p_e [1 - P(\mathcal{U})]P^{\mathcal{V}}(e)$$

$$= P(\mathcal{U}) + [1 - P(\mathcal{U})]P(\mathcal{V}),$$

establishing (2.7) and (2.8). ||

Lemma 2.3.

Let $\mathcal{U}$ and $\mathcal{V}$ be two edges-sequences and let b be an edge which is a predecessor of all edges in $\underline{\mathcal{V}}$ with $(\mathcal{U}, b, \mathcal{V}) \in \Gamma$. Then

(2.10) $$C(\mathcal{U}, b, \mathcal{V}) = C(\mathcal{U}) + P^{(\mathcal{U},b)}(b)C(b, \mathcal{V}),$$

(2.11) $$P(\mathcal{U}, b, \mathcal{V}) = P(\mathcal{U}) + P^{(\mathcal{U},b)}(b)P(b, \mathcal{V}),$$

(2.12) $$C(b, \mathcal{V}) = c_b + p_b C(\mathcal{V})$$

and, if $\mathcal{V} \neq \emptyset$,

(2.13) $$P(b, \mathcal{V}) = p_b P(\mathcal{V}).$$

Proof.

We first prove that

(2.14) $$P^{(\mathcal{U},b,\mathcal{V})}(e) = P^{(\mathcal{U},b)}(b)P^{(b,\mathcal{V})}(e) \text{ for every } e \in \underline{\mathcal{V}}$$

and

(2.15) $$P^{(b,\mathcal{V})}(e) = p_b P^{\mathcal{V}}(e) \text{ for every } e \in \underline{\mathcal{V}}.$$

Consider the search process defined by $(\mathcal{U}, b, \mathcal{V})$. As b is a predecessor of all edges in $\underline{\mathcal{V}}$, the set of predecessors in $\underline{\mathcal{U}}$ of b and of any edge in $\underline{\mathcal{V}}$ coincide; hence, during the exploration of the edges of $\underline{\mathcal{U}}$, all edges in $\{b\} \cup \underline{\mathcal{V}}$ are eliminated if and only if this occurs to any one of these edges. Now, let e be an edge in $\underline{\mathcal{V}}$. It follows that if b is not explored neither will be e, and alternatively, if b is explored then no edge of $\underline{\mathcal{V}}$ is eliminated during the exploration of the edges of $\underline{\mathcal{U}}$ and the probability that e is explored conditioned on the event that b is explored is $P^{(b,\mathcal{V})}(e)$. So, the standard conditioning formula implies that $P^{(\mathcal{U},b,\mathcal{V})}(e) = P^{(\mathcal{U},b)}(b)P^{(b,\mathcal{V})}(e)$, establishing (2.14). Also, (2.15) is immediate by calculating the probability that e is explored under the search process defined by $(b, \mathcal{V})$ on the events that b is feasible and infeasible.

Next, by (2.4) and by (2.14)

$$C(\mathcal{U}, b, \mathcal{V}) = \sum_{e \in \underline{\mathcal{U}}} c_e P^{(\mathcal{U},b,\mathcal{V})}(e) + \sum_{e \in \{b\} \cup \underline{\mathcal{V}}} c_e P^{(\mathcal{U},b,\mathcal{V})}(e)$$

$$= \sum_{e \in \underline{\mathcal{U}}} c_e P^{\mathcal{U}}(e) + \sum_{e \in \{b\} \cup \underline{\mathcal{V}}} c_e P^{(\mathcal{U},b)}(b)P^{(b,\mathcal{V})}(e)$$

$$= C(\mathcal{U}) + P^{(\mathcal{U},b)}(b)\Big[\sum_{e\in \{b\}\cup\mathcal{V}} c_e P^{(b,\mathcal{V})}(e)\Big] = C(\mathcal{U}) + P^{(\mathcal{U},b)}(b)C(b,\mathcal{V}),$$

establishing (2.10). Similar arguments show that

$$P(\mathcal{U}, b, \mathcal{V}) = \sum_{e\in L\cap\mathcal{U}} p_e P^{(\mathcal{U},b,\mathcal{V})} + \sum_{e\in L\cap(\{b\}\cup\mathcal{V})} p_e P^{(\mathcal{U},b,\mathcal{V})}(e)$$

$$= \sum_{e\in L\cap\mathcal{U}} p_e P^{\mathcal{U}}(e) + \sum_{e\in L\cap(\{b\}\cup\mathcal{V})} p_e P^{(\mathcal{U},b)}(b) P^{(b,\mathcal{V})}(e)$$

$$= P(\mathcal{U}) + P^{(\mathcal{U},b)}(b)P(b,\mathcal{V}).$$

Next, as the equality in (2.12) is trite when $\mathcal{V} = \emptyset$, assume that $\mathcal{V} \neq \emptyset$. As b is a predecessor of all edges in $\mathcal{V} \neq \emptyset$, b is not a leaf. Also, $P^{(b,\mathcal{V})}(b) = 1$. Hence, by (2.15),

$$C(b,\mathcal{V}) = \sum_{e\in\{b\}\cup\mathcal{V}} c_e P^{(b,\mathcal{V})}(e) = c_b P^{(b,\mathcal{V})}(b) + \sum_{e\in\mathcal{V}} c_e p_b P^{\mathcal{V}}(e) = c_b + p_b C(\mathcal{V}),$$

and

$$P(b,\mathcal{V}) = \sum_{e\in L\cap(\{b\}\cup\mathcal{V})} p_e P^{(b,\mathcal{V})}(e) = \sum_{e\in L\cap\mathcal{V}} p_e p_b P^{\mathcal{V}}(e) = p_b P(\mathcal{V}). \quad \|$$

3. *Indices of Edge-Sequences*

In the current section, we use the expected cost and feasibility probability of edge-sequences in Γ to define indices, which are scalars essential to our development. We then establish some elementary inequalities for these indices.

The *index associated with* an edge-sequence $\mathcal{U} \in \Gamma$ that satisfies $P(\underline{\mathcal{U}}) \neq 0$, denoted $I(\mathcal{U})$, is defined as $I(\mathcal{U}) \equiv C(\mathcal{U}) / P(\mathcal{U})$. We recall that $P(\mathcal{U}) \neq 0$ if and only if $\mathcal{U}$ contains a leaf. For $\mathcal{U}, \mathcal{U}' \in \Gamma$ with $\underline{\mathcal{U}} = \underline{\mathcal{U}'}$, we have from (2.3) that $P(\underline{\mathcal{U}}) = P(\underline{\mathcal{U}'})$; in particular, if $P(\underline{\mathcal{U}}) = P(\underline{\mathcal{U}'}) \neq 0$, then

(3.1) $\quad I(\mathcal{U}) \leq I(\mathcal{U}')$ if and only if $C(\mathcal{U}) \leq C(\mathcal{U}')$;

further, this equivalence also holds with strict inequalities replacing the weak inequalities.

The next two lemmas concern inequalities for the indices of interchangeable and merged edge-sequences. Their proof uses the standard inequality

(3.2) $\quad \min\{\frac{a}{b}, \frac{c}{d}\} \leq \frac{a + \alpha c}{b + \alpha d} \leq \max\{\frac{a}{b}, \frac{c}{d}\}$ for real numbers a, $b > 0$, c, $d > 0$ and $\alpha \geq 0$,

which is known to hold with strict inequalities whenever $\alpha > 0$ and $a / b \neq c / d$.

Lemma 3.1.

Let $\mathcal{U}$ and $\mathcal{V}$ be interchangeable edge-sequences satisfying $P(\mathcal{U}) \neq 0$ and $P(\mathcal{V}) \neq 0$. Then the following are equivalent:

a) $I(\mathcal{U}) \leq I(\mathcal{V})$,

b) $I(\mathcal{U}) \leq I(\mathcal{U}, \mathcal{V})$,

c) $I(\mathcal{V}, \mathcal{U}) \leq I(\mathcal{V})$,

d) $I(\mathcal{U}, \mathcal{V}) \leq I(\mathcal{V}, \mathcal{U})$, and

e) $C(\mathcal{U}, \mathcal{V}) \leq C(\mathcal{V}, \mathcal{U})$.

Further, the above equivalences holds with strict inequalities replacing the weak inequalities.

Proof.

(a) $\Leftrightarrow$ (b) and (a) $\Leftrightarrow$ (c) : By Lemma 2.2,

$$I(\mathcal{U}, \mathcal{V}) = \frac{C(\mathcal{U}) + [1 - P(\mathcal{U})]C(b, \mathcal{V})}{P(\mathcal{U}) + [1 - P(\mathcal{U})]P(b, \mathcal{V})} ; \tag{3.3}$$

hence, from (3.2) we have that

$$\min \{I(\mathcal{U}), I(\mathcal{V})\} \leq I(\mathcal{U}, \mathcal{V}) \leq \max \{I(\mathcal{U}), I(\mathcal{V})\} . \tag{3.4}$$

Further, as $P(\mathcal{U}) < 1$, the strict version of (3.2) implies that (3.4) holds with strict inequalities when $I(\mathcal{U}) \neq I(\mathcal{V})$. It follows from (3.4) that if $I(\mathcal{U}) \leq I(\mathcal{V})$, then $I(\mathcal{U}) = \min \{I(\mathcal{U}), I(\mathcal{V})\} \leq I(\mathcal{U}, \mathcal{V})$ and $I(\mathcal{V}, \mathcal{U}) \leq \max \{I(\mathcal{V}), I(\mathcal{U}\} = I(\mathcal{V})$, establishing the implications (a) $\Rightarrow$ (b) and (a) $\Rightarrow$ (c) . Further, the strict version of (3.4) implies that if $I(\mathcal{U}) > I(\mathcal{V})$, then $I(\mathcal{V}) = \min \{I(\mathcal{U}), I(\mathcal{V})\} < I(\mathcal{V}, \mathcal{U})$ and $I(\mathcal{U}, \mathcal{V}) < \max \{I(\mathcal{U}), I(\mathcal{V})\} = I(\mathcal{U})$, establishing the implications (c) $\Rightarrow$ (a) and (b) $\Rightarrow$ (a) .

(d) $\Leftrightarrow$ (e) : This equivalence is immediate from (3.1) and the observation that $\underline{(\mathcal{U}, \mathcal{V})} = \underline{(\mathcal{V}, \mathcal{U})}$.

(d) $\Leftrightarrow$ (a) : By Lemma 2.2, $I(\mathcal{U}, \mathcal{V}) \leq I(\mathcal{V}, \mathcal{U})$ if and only if $C(\mathcal{U}) + [1 - P(\mathcal{U})]C(\mathcal{V}) \leq C(\mathcal{V}) + [1 - P(\mathcal{V})]C(\mathcal{U})$, and the last inequality is obviously equivalent to the inequality $I(\mathcal{U}) = C(\mathcal{U}) / P(\mathcal{U}) \leq C(\mathcal{V}) / P(\mathcal{V}) = I(\mathcal{V})$.

The strict inequality version of the lemma follows from the above arguments, modified by the exchange of weak and strict inequalities. ||

Lemma 3.2.

Let $\mathcal{U}$ and $\mathcal{V}$ be edge-sequences and let b be an edge such that $(\mathcal{U}, b, \mathcal{V}) \in \Gamma$, with b as a predecessor of all edges in $\underline{\mathcal{V}}$, $P(\mathcal{U}) \neq 0$ and $P(b, \mathcal{V}) \neq 0$. Then

$$\min \{I(\mathcal{U}), I(b, \mathcal{V})\} \leq I(\mathcal{U}, b, \mathcal{V}) \leq \max \{I(\mathcal{U}), I(b, \mathcal{V})\} ; \tag{3.5}$$

further, strict inequalities hold in (3.5) whenever $I(\mathcal{U}) \neq I(b, \mathcal{V})$. Also, if $P(\mathcal{V}) \neq 0$, then

(3.6) $I(\mathcal{V}) < I(b, \mathcal{V})$.

Proof.

As $P(\mathcal{U}) \neq 0$, $\underline{\mathcal{U}}$ contains a leaf, implying that so does $\underline{\mathcal{U}} \cup \{b\} \cup \underline{\mathcal{V}}$; so, $P(\mathcal{U}, b, \mathcal{V}) \neq 0$. Now, by Lemma 2.3,

$$(3.7) \qquad I(\mathcal{U}, b, \mathcal{V}) = \frac{C(\mathcal{U}) + P^{(\mathcal{U},b)}(b)C(b, \mathcal{V})}{P(\mathcal{U}) + P^{(\mathcal{U},b)}(b)P(b, \mathcal{V})},$$

and inequality (3.5) follows immediately from (3.2). Further, as $0 < p_e < 1$ for each edge e, $P^{(\mathcal{U},b)}(b) \neq 0$ and the strict version of (3.5) when $I(\mathcal{U}) \neq I(b, \mathcal{V})$ follows from the strict version of (3.2). Finally, if $P(\mathcal{V}) \neq 0$, also from Lemma 2.3,

$$I(b, \mathcal{V}) = \frac{c_b + p_b C(\mathcal{V})}{p_b P(\mathcal{V})} > \frac{C(\mathcal{V})}{P(\mathcal{V})} = I(\mathcal{V}).$$

establishing (3.6). ||

4. The Index Computation Algorithm

In the current section we describe an algorithm which computes a scalar for each edge, referred to as the *index* of that edge . These indices are then used in the next section to construct an order policy that will be shown to be optimal.

We need one additional definition. The *available set for* a subset U of E , denoted A(U) , is the set of edges not in U that have immediate predecessors in U . For an edge e , $A(e) \equiv A(\{e\})$ is the set of immediate successors of e .

The Index Computation Algorithm

Step 1: Computation of indices of leaves:

Set:

$E^* \leftarrow L$.

For each leaf e , set:

$C_e \leftarrow c_e$

$P_e \leftarrow p_e$

$I_e \leftarrow C_e / P_e$

$A_e \leftarrow A(e)$

$S_e \leftarrow \emptyset$.

Step 2: Computation of indices of edges which are not leaves:

Step 2A: Edge selection

If $E^* = E$, STOP.

If $E \setminus E^* \neq \emptyset$, select an edge $e \in E \setminus E^*$ whose immediate successors are all in E^* and set

$E^* \leftarrow E^* \cup \{e\}$.

Step 2B: Initialization of index computation of the edge

Let g be an index-minimizing edge in A(e) . Set

$C_e \leftarrow c_e + p_e C_g$

$P_e \leftarrow p_e P_g$

$I_e \leftarrow C_e / P_e$

$A_e \leftarrow [A(e) \setminus \{g\}] \cup A_g$

$S_e \leftarrow (g, S_g)$

If $A_e = \emptyset$, go to step 2A

If $A_e \neq \emptyset$, go to step 2C.

Step 2C: Recursive update of the index of the edge

Let g be an index-minimizing edge in A_e.

If $I_g \geq I_e$ go to step 2A.

If $I_g < I_e$ set

$$C_e \leftarrow C_e + P^{(e,S_e,g)}(g)C_g$$

$$P_e \leftarrow P_e + P^{(e,S_e,g)}(g)P_g$$

$$I_e \leftarrow C_e / P_e$$

$$A_e \leftarrow (A_e \setminus \{g\}) \cup A_g$$

$$S_e \leftarrow (S_e, g, S_g).$$

If $A_e = \emptyset$, go to step 2A.

If $A_e \neq \emptyset$, go to step 2C.

When $E \setminus E^* \neq \emptyset$ in step 2A, any maximal element in $E \setminus E^*$ with respect to the relation of being a successor has the property that all its immediate successors are in E^*; thus, the algorithm will stop only on command (in step 2A). Also, in each execution of step 2 with a particular selection of an edge e in step 2A, an edge that is deleted from A_e can never be reintroduced into A_e, so the number of executions of step 2C is finite. As E^* is increasing in each execution of step 2, one will eventually enter step 2A with $E \setminus E^* = \emptyset$, at which point the algorithm stops. So, termination on command will always occur.

We say that the underlying scheduling problem is *nondegenerate* if for every pair of distinct edges e and e', no final value of I_e coincides with any temporary or final value of $I_{e'}$. In general, it is possible to encounter ties in steps 2B and 2C of an execution of the Index Computation Algorithm, but this will not occur when the problem is nondegenerate. In that case, one will not encounter equality in any of the test inequalities $I_g \geq I_e$ in step 2C. Throughout the current section and the following one we assume that the problem is nondegenerate. This assumption is relaxed in Section 7.

For a leaf e, the final quintuple $(C_e, P_e, I_e, A_e, S_e)$ is determined during the execution of step 1 which is deterministic. For an edge e which is not a leaf, its final quintuple is computed during the single execution of step 2 in which e is the edge that is selected in step 2A, and under the nondegeneracy assumption, the execution of the corresponding steps 2B and 2C is

deterministic. It follows that for each edge e the final values C_e, P_e, I_e, A_e and $\mathcal{S}_e$ are uniquely determined. Henceforth, unless stated otherwise, we refer to the collection $\{(C_e, P_e, I_e, A_e, \mathcal{S}_e)\}_{e \in E}$ of uniquely determined final values as *the output of the algorithm*. In particular, we refer to the output values of (the number) I_e, (the set) A_e and (the edge-sequence) $\mathcal{S}_e$, respectively, as the *index*, the *available set* and the *continuation-sequence* of edge e. Also, we refer to the *execution of step 2 that determines the parameters of* edge e. We note that the nondegeneracy assumption assures that the indices of the edges are distinct.

The next two lemmas establish properties of the (temporary) values of the edge-parameters as generated and updated through the execution of the Index Computation Algorithm; their technical proofs are deferred to the Appendix.

Lemma 4.1.

Consider an execution of the Index Computation Algorithm. At the end of step 1 or any substep of step 2, for each edge $e \in E$:

a) $(e, \mathcal{S}_e) \in \Gamma$,

b) $A_e = A(\{e\} \cup \mathcal{S}_e)$

c) e is a predecessor to each edge in $A_e \cup \mathcal{S}_e$, and

d) $C_e = C(e, \mathcal{S}_e)$, $P_e = P(e, \mathcal{S}_e)$ and $I_e = I(e, \mathcal{S}_e)$.

Lemma 4.2.

Consider the execution of step 2 of the Index Computation Algorithm, and let $e \in E \setminus L$ be the edge whose parameters are determined during that step. Then at the end of step 2B and of each step 2C, the (temporary) values of $(I_e, A_e, \mathcal{S}_e)$ and the (final) values of $(I_w, A_w, \mathcal{S}_w)_{w \in \mathcal{S}_e \cup A_e}$ satisfy

(4.1) $\max_{u \in \mathcal{S}_e} I_u < I_e$

and

(4.2) $\max_{u \in S_e} I_u < \min_{v \in A_e} I_v$.

The next two lemmas concern the output of the Index Computation Algorithm.

Lemma 4.3.

Consider the output $\{(C_e, P_e, I_e, A_e, S_e)\}_{e \in E}$ of the Index Computation Algorithm. Then

(4.3) $\max_{u \in S_e} I_u < I_e < \min_{v \in A_e} I_v$ for each $e \in E \setminus L$.

Proof.

Let $e \in E \setminus L$ and consider the execution of step 2 that determines the parameters of e. As the edges in $S_e \cup A_e$ are successors of e (part (c) of Lemma 4.1) the parameters of these edges have already been determined by that time; thus, the left inequality of (4.3) follows from (4.1) applied when the considered execution of step 2 is completed. The right inequality of (4.3) follows from the termination condition at that time. ||

Lemma 4.4.

Consider the output $\{(C_e, P_e, I_e, A_e, S_e)\}_{e \in E}$ of the Index Computation Algorithm. Then for each edge e, S_e has the representation $S_e = (g_1, S_{g_1}, g_2, S_{g_2}, ..., g_q, S_{g_q})$ where q is a nonnegative integer and $g_1, g_2, ..., g_q$ are edges satisfying

(4.4) $I_{g_1} < I_{g_2} < ... < I_{g_q} < I_e$

and

(4.5) $I(e, g_1, S_{g_1}, ..., g_{i+1}, S_{g_{i+1}}) < I(e, g_1, S_{g_1}, ..., g_i, S_{g_i})$ for $i = 1, ..., q - 1$.

Proof.

If e is a leaf, $S_e = \emptyset$ and the conclusions of the lemma are trite. So, assume that e is not a leaf. The representation of S_e then follows directly from the substeps of step 2 of the Index Computation Algorithm, in fact, $g_1, g_2, ..., g_q$ are the edges selected, respectively, in step 2B

and steps 2C in the execution of step 2 that determines the parameters of e. Moreover, for $i = 1, ..., q - 1$, if g_{i+1} was in A_e when (g_i, S_{g_i}) is added to S_e, then the minimal selection of g_i assures that $I_{g_i} < I_{g_{i+1}}$, and alternatively, if g_{i+1} was not in A_e at that point, then $g_{i+1} \in A(g_i)$ and Lemma 4.3 (with $e = g_i$) assures that $I_{g_i} < I_{g_{i+1}}$. Also, $(g_{i+1}, S_{g_{i+1}})$ is added to S_e because the test in the corresponding execution of step 2C yields $I_{g_{i+1}} < I_e$, that is, $I(g_{i+1}, S_{g_{i+1}}) = I_{g_{i+1}} < I_e = I(e, S_e) = I(e, g_1, S_{g_1}, ..., g_i, S_{g_i})$; so, Lemma 3.1 implies the corresponding inequality of (4.5). Finally, the conclusion $I_{g_q} < I_e$ follows from the fact that $g_q \in S_e$ and Lemma 4.3. ||

5. Construction of Optimal Order Policies

The edge-indices are used in the current section to sequence the edges and construct an order policy which is then shown to be optimal. At each stage, an index-minimizing edge is selected from among the edges that are available. As dynamic programming is used to establish optimality, it becomes necessary to consider sequencing problems over subsets of edges, a procedure which is referred to as embedding (see Denardo [1982]). The reader is reminded that the nondegeneracy assumption is in force throughout this section.

A set of edges F is called an *upset* if all successors of edges in F are also in F. Given an upset of edges F, we consider the problem of sequencing edges for the (restricted) problem in which the graph $\mathcal{G}$ is replaced by $\mathcal{G}^F \equiv (N, F)$ while feasibility probabilities and exploration costs of the edges of F remain unchanged, and we refer to this (restricted) problem as the *sequencing problem with input* F. An order policy for the sequencing problem with input F is determined by an edge-sequence $\mathcal{V} \in \Gamma$ with $\underline{\mathcal{V}} = F$. If $\mathcal{U}$ is an edge-sequence with $\underline{\mathcal{U}} \supseteq F$, the *restriction of* $\mathcal{U}$ *to* F, denoted $\mathcal{U}^F$, is the subsequence obtained from $\mathcal{U}$ by deleting the arcs in $\underline{\mathcal{U}} \setminus F$ while maintaining the order prescribed by the original sequence; in particular, $\mathcal{U}^F \in \Gamma$ whenever $\mathcal{U} \in \Gamma$. Finally, the set of edges in F having no predecessor in F will be denoted $I^*(F)$.

We next describe an algorithm which uses the edge-indices to sequence the edges of upsets. The resulting edge-sequence will be used to construct an optimal (order) policy for the corresponding (restricted) sequencing problem.

The Sequencing Algorithm

The input for the algorithm is an upset of edges F.

Step 1: Initialization

Set:

$$A \leftarrow I^*(F)$$
$$\mathcal{T} \leftarrow \emptyset.$$

Step 2: Sequence construction:

If $A = \emptyset$ STOP.

If $A \neq \emptyset$, select edge e as an index-minimizing edge in A and set:

$A \leftarrow (A \setminus \{e\}) \cup A(e)$

$\mathcal{T} \leftarrow (\mathcal{T}, e)$.

Go back to step 2.

The next lemma explores states useful properties of the parameters that are constructed within the Sequencing Algorithm. The formal inductive proof is deferred to the Appendix.

Lemma 5.1.

Let F be an upset of edges. Then, after each execution of Step 2 of the Sequencing Algorithm with input F we have that:

(a) $\mathcal{T}$, A are subsets of F,

(b) $A = [\Gamma^*(F) \setminus \mathcal{T}] \cup A(\mathcal{T})$

(c) all predecessors in F of an edge in $\mathcal{T}$ are also in $\mathcal{T}$,

(d) $\mathcal{T} \in \Gamma$, and

(e) $A = \emptyset$ if and only if $\mathcal{T} = F$.

In particular, the algorithm stops on command after exactly |F| iterations of the sequence construction step, and at termination $\mathcal{T} = F$.

As the nondegeneracy assumption is in force, the Sequencing Algorithm is deterministic (no selections) and, with any input F, the final value of $\mathcal{T}$ is uniquely determined; we denote this final edge-sequence by $\mathcal{T}^F$ and refer to it as the *output-sequence* of the Sequencing Algorithm with input F.

The next lemma establishes important properties of this sequence. Here, again, the technical proof is deferred to the Appendix.

Lemma 5.2.

Let F be an upset of edges let $\mathcal{T}^F$ be the output-sequence of the Sequencing Algorithm

with input F. Then in $\mathcal{T}^F$, each edge e is immediately followed by the edges of its continuation-sequence $\mathcal{S}_e$, in order.

Lemma 5.2 implies that the execution of the Sequencing Algorithm can be accelerated by modifying the updates of Step 2 to:

$$A \leftarrow [A \setminus (\{e\} \cup \mathcal{S}_e)] \cup A(\mathcal{S}_e)$$
$$\mathcal{T} \leftarrow (\mathcal{T}, e, \mathcal{S}_e)$$

We will then refer to the resulting algorithm as the *Accelerated Sequencing Algorithm.*

Also, Lemma 5.2 implies that if F is an upset of edges, the output-sequence of the Sequencing Algorithm with input F has the form $(e_1, \mathcal{S}_{e_1}, ..., e_q, \mathcal{S}_{e_q})$ for corresponding edges $e_1, ..., e_q$ in F.

The next lemma identifies conditions for the output of the Sequencing Algorithm with one input to determine the output of the Sequencing Algorithm with another input. The straightforward proof is left to the reader.

Lemma 5.3.

If F, F' and $F \setminus F'$ are upsets of edges, then $\mathcal{T}^{F \setminus F'} = (\mathcal{T}^F)_{F \setminus F'}$. ||

We are now ready to state our main result asserting that the order policy associated with the output list of the Sequencing Algorithm is the only optimal policy.

Theorem 5.4.

Let F be an upset of edges and let $\mathcal{T}^F$ be the output-sequence of the Sequencing Algorithm with input F. Then the order policy associated with $\mathcal{T}^F$ is the unique optimal policy for the sequencing problem with input F.

Proof.

Our proof follows by induction on the size of the set F. The induction assumption is trivial for the empty set (which is an upset of edges). Suppose it holds for every upset of edges having less than m edges, and consider an upset of edges with exactly m edges. The finiteness of the set of all policies assures that the sequencing problem with input F has an optimal policy.

Consider such an optimal policy and let b be the first edge that the policy selects, in particular, $b \in I^*(F)$. Let H be the set of all of successors of b. The definition of a policy assures that b has no predecessor in F, implying that H, $F \setminus \{b\}$ and $F \setminus (\{b\} \cup H)$ are all upsets of edges. As $0 < p_b < 1$, we conclude from the optimality principle of dynamic programming and the induction assumption that once b is explored, the continuation of the (optimal) policy either follows the order policy associated with $\mathcal{T}^{F\setminus\{b\}}$ if b is found feasible, or the order policy associated with $\mathcal{T}^{F\setminus(\{b\}\cup H)} = \mathcal{T}^{(F\setminus\{b\})\setminus H}$ if b is found infeasible. As $F \setminus \{b\}$, H and $(F \setminus \{b\}) \setminus H$ are upsets of edges, we have from Lemma 5.3 that $\mathcal{T}^{(F\setminus\{b\})\setminus H} = [\mathcal{T}^{F\setminus\{b\}}]_{(F\setminus\{b\})\setminus H}$. Hence, the given optimal policy is the order policy associated with the edge-sequence $(b, \mathcal{T}^{F\setminus\{b\}})$.

Let a be the first edge in $\mathcal{T}^F$. The (iterative) description of the Sequencing Algorithm implies that $a \in I^*(F)$ and $\mathcal{T}^F = (a, \mathcal{T}^{F\setminus\{a\}})$. It follows that in order to prove that the given optimal policy is the order policy associated with $\mathcal{T}^F$, it suffices to show that $a = b$. Suppose, by way of contradiction, that $a \neq b$. As a and b are in $I^*(F)$, the selection of a over b in the first iteration of the sequence construction step of the Sequencing Algorithm with input F assures that

(5.1) $\qquad I_a < I_b$.

By Lemma 4.4, $\mathcal{S}_b$ has the form $\mathcal{S}_b = (b_1, \mathcal{S}_{b_1}, b_2, \mathcal{S}_{b_2}, ..., b_t, \mathcal{S}_{b_t})$ where t is a nonnegative integer, $b_1, b_2, ..., b_t$ are edges in the upset H,

(5.2) $\qquad I_{b_1} < I_{b_2} < ... < I_{b_t} < I_b$;

and

(5.3) $\qquad I(b, b_1, \mathcal{S}_{b_1}, ..., b_{i+1}, \mathcal{S}_{b_{i+1}}) < I(b, b_1, \mathcal{S}_{b_1}, ..., b_i, \mathcal{S}_{b_i})$ for $i = 1, ..., t - 1$.

With $b_{t+1} \equiv b$, (5.1) implies that $I_a < I_{b_{t+1}}$; hence, we can select the smallest integer $s \in \{1, 2, ..., t\}$ satisfying $I_a < I_{b_{s+1}}$. It then follows from the restatement of the (accelerated) Sequencing Algorithm introduced in the paragraph following Lemma 5.2 that

(5.4) $\mathcal{T}^{F\setminus\{b\}} = (b_1, \mathcal{S}_{b_1}, b_2, \mathcal{S}_{b_2}, ..., b_s, \mathcal{S}_{b_s}, a, \mathcal{S}_a, c_1, \mathcal{S}_{c_1}, ..., c_m, \mathcal{S}_{c_m})$

for corresponding edges $c_1, ..., c_m \in F \setminus \{b\}$. Let $\mathcal{U} \equiv (b, b_1, \mathcal{S}_{b_1}, ..., b_s, \mathcal{S}_{b_s})$ and $\mathcal{W} \equiv (c_1, \mathcal{S}_{c_1}, ..., c_m, \mathcal{S}_{c_m})$; in particular, $(b, \mathcal{T}^{F\setminus\{b\}}) = (\mathcal{U}, a, \mathcal{S}_a, \mathcal{W})$. We next observe that (5.3) implies that $I(b, \mathcal{S}_b) \le I(\mathcal{U})$; this inequality combines with (5.1) and two applications of part (d) of Lemma 4.1 to show that

(5.5) $I(a, \mathcal{S}_a) = I_a < I_b = I(b, \mathcal{S}_b) \le I(\mathcal{U})$.

As F is an upset and $a, b \in I^*(F)$, the sets of successors of a and b are disjoint. Consequently, neither a nor any one of its successors is a successor of b and, vice versa, neither b nor any one of its successors is a successor of a; these conclusions imply that $(a, \mathcal{S}_a)$ and $\mathcal{U}$ are interchangeable. We conclude from (5.5) and (the strict version of) Lemma 2.2 that $I(a, \mathcal{S}_a, \mathcal{U}) < I(\mathcal{U}, a, \mathcal{S}_a)$; consequently, by Lemma 2.1,

(5.6) $I(a, \mathcal{S}_a, \mathcal{U}, \mathcal{W}) - I(\mathcal{U}, a, \mathcal{S}_a, \mathcal{W}) = I(a, \mathcal{S}_a, \mathcal{U}) - I(\mathcal{U}, a, \mathcal{S}_a) < 0$.

As the success probability of a list depends only on the listed elements, (5.6) implies that

(5.7) $C(a, \mathcal{S}_a, \mathcal{U}, \mathcal{W}) < C(\mathcal{U}, a, \mathcal{S}_a, \mathcal{W}) = C(b, \mathcal{T}^{F\setminus\{b\}})$,

that is, the order policy associated with the edge-sequence $(a, \mathcal{S}_a, \mathcal{U}, \mathcal{W})$ has lower expected cost than the one associated with the edge-sequence $(b, \mathcal{T}^{F\setminus\{b\}})$. As the latter policy is assumed to be optimal, we have obtained a contradiction which proves that the assertion $a \neq b$ is false. ||

6. Extensions

The nondegeneracy assumption is relaxed in the current section. We follow a standard approach of perturbing the data of a problem that does not necessarily satisfy the nondegeneracy assumption so that the perturbed problem does satisfy it. It is then observed that in running the Index Computation Algorithm and the Sequencing Algorithm one can avoid consideration of specific values of the perturbation parameter by using lexicographic rules for comparison.

Consider a problem which does not necessarily satisfy the nondegeneracy assumption. If the Index Computation Algorithm (or the Sequencing Algorithm) are run under this relaxation, ties may be encountered; thus, it is necessary to determine how such potential ties are to be resolved. Consider any (enumeration) function $n:E \rightarrow \{1, ..., |E|\}$ having the property that $n(e) < n(e')$ for every pair of edges e and e' where e is a predecessor of e'. We next consider perturbations of the exploration costs with c_e being replaced by $c_e + \varepsilon^{n(e)}$ where ε is a small positive number whose specific value is not yet specified. To express dependence on ε, we index the (temporary and final) values of the edge parameters generated by the Index Computation Algorithm by ε, e.g., we write $C_e(\varepsilon)$, $P_e(\varepsilon)$ and $I_e(\varepsilon)$.

For sufficiently small positive ε, executions of the Index Computation Algorithm yield uniform (independent of ε) comparisons of generated indices; further, (temporary and final) values of $P_e(\varepsilon)$ are independent of ε and (temporary and final) values of $C_e(\varepsilon)$ are polynomials in ε. Observing that (temporary and final) values of $C_e(\varepsilon)$ for edge e consist of the exploration cost of e (possibly) modified by weighted exploration costs of its predecessors, the consistency of the enumeration function with the relation of being a predecessor assures that the leading two terms of $C_e(\varepsilon)$ (as a polynomial in ε) have the form $C_e + \varepsilon^{n(e)}$. We conclude that (temporary and final) values of $I_e(\varepsilon)$ are polynomials in ε whose leading two terms for sufficiently small positive ε have the form $I_e + \varepsilon^{n(e)} / P_e$ where P_e is the common value of $P_e(\varepsilon)$. It follows that the polynomials describing the (temporary and final) values of the indices for distinct edges cannot coincide; further, they are comparable for sufficiently small positive ε

by comparing their values with $\varepsilon = 0$ and using $n(\varepsilon)$ to break ties; so, lexicographic comparisons can be used make uniform index comparisons for sufficiently small positive ε. We conclude that for sufficiently small positive ε the perturbed problem satisfies the nondegeneracy assumption and the Index Computation Algorithm and the Sequencing Algorithm can be run uniformly for sufficiently small positive ε by using the above lexicographic comparisons. It then follows from Theorem 5.4 that the resulting order policy is uniformly optimal for sufficiently small positive ε, and by continuity arguments, such a policy will be optimal for the original (unperturbed) problem.

The above perturbation technique allows one to consider zero exploration costs for the edges; so, positivity of exploration costs can be relaxed to nonnegativity.

We finally consider problems where the feasibility probabilities are allowed to take the boundary value $p_e = 1$. (The boundary value $p_e = 0$ is uninteresting as it concerns the introduction of edges whose exploration will never "open a path" to a leaf.) Here the analysis is simple. If one replaces p_e by $p_e - \delta$, the (possibly lexicographic) comparisons during executions of the (possibly modified version of) the Index Computation Algorithm and the Sequencing Algorithm will be independent of sufficiently small positive δ. These algorithms will produce an order policy which is optimal for sufficiently small positive δ; in particular, by continuity, this policy will be optimal for $\delta = 0$, that is for the unperturbed problem.

7. An Example

We illustrate the solution method developed in the earlier sections by solving the problem introduced in the prologue and exhibited in Figure 1. We start with the Index Computation Algorithm.

The Index Computation Algorithm

Step 1: Computation of indices of leaves:

$E^* \leftarrow \{3, 4, 5, 7, 8\}$

$C_3 \leftarrow 5$, $C_4 \leftarrow 4$, $C_5 \leftarrow 20$, $C_7 \leftarrow 3$, $C_8 \leftarrow 4$

$P_3 \leftarrow .7$, $P_4 \leftarrow .8$, $P_5 \leftarrow .8$, $P_7 \leftarrow 6$, $P_8 \leftarrow .2$

$I_3 \leftarrow 5/.7 = 7.14$, $I_4 \leftarrow 4/.8 = 5$, $I_5 \leftarrow 20/.8 = 25$,

$I_7 \leftarrow 3/.6 = 5$, $P_8 \leftarrow 4/.2 = 20$

$A_3, A_4, A_5, A_7, A_8 \leftarrow \emptyset$

$S_3, S_4, S_5, S_7, S_8 \leftarrow \emptyset$

Step 2: Computation of indices of edges which are not leaves:

Step 2A: Edge selection

$e \leftarrow 2$

$E^* \leftarrow \{2, 3, 4, 5, 7, 8\}$

$E \setminus E^* = \{1, 2, 6\}$

$A(2) \leftarrow \{3, 4\}$

Step 2B: Initialization of index computation of edge 2

$g \leftarrow \text{argmin}_{f \in \{3,4\}} I_f = 4$ and $I_g = 5$ (as $I_3 = 7.14$ and $I_4 = 5$)

$C_2 \leftarrow c_2 + p_2 C_4 = 3 + (.6)(4) = 5.4$

$P_2 \leftarrow p_2 P_4 = (.6)(.8) = .48$

$I_2 \leftarrow C_2 / P_2 = 5.4 / .48 = 11.25$

$A_2 \leftarrow [A(2) \setminus \{4\}] \cup A_4 = \{3, 4\} \setminus \{4\} = \{3\}$

$S_2 \leftarrow (4, S_4) = (4)$

Step 2C: Recursive update of the index of edge 2

$g \leftarrow \text{argmin}_{f \in \{3\}} I_f = 3$ and $I_g = 7.14 < 11.25 = I_2$ (as $I_3 = 7.14$)

$P^{(e,S_e,g)}(g) = P^{(2,4,3)}(3) = (.6)(.2) = .12$

$C_2 \leftarrow C_2 + P^{(2,4,3)}(3) C_3 = 5.4 + (.12)5 = 6.0$

$P_2 \leftarrow P_2 + P^{(2,4,3)}(3) P_3 = .48 + (.12).7 = .56$

$I_2 \leftarrow C_2 / P_2 = 6.0 / .56 = 10.71$

$A_2 \leftarrow [A(2) \setminus \{3\}] \cup A_3 = \{3\} \setminus \{3\} = \emptyset$

$\mathcal{S}_2 \leftarrow (\mathcal{S}_2, 3, \mathcal{S}_3) = (4, 3)$

Step 2A: Edge selection

$E \setminus E^* = \{1, 6\}$

$e \leftarrow 1$

$E^* \leftarrow \{1, 2, 3, 4, 5, 7, 8\}$

$A(1) \leftarrow \{2, 5\}$

Step 2B: Initialization of index computation of edge 1

$g \leftarrow \text{argmin}_{f \in \{2,5\}} I_f = 2$ and $I_g = 10.71$ (as $I_2 = 10.71$ and $I_5 = 25$)

$C_1 \leftarrow c_1 + p_1 C_2 = 1 + (.3)(6) = 2.8$

$P_1 \leftarrow p_1 P_2 = (.3)(.56) = .17$

$I_1 \leftarrow C_1 / P_1 = 2.8 / .17 = 16.47$

$A_1 \leftarrow [A(1) \setminus \{2\}] \cup A_2 = \{2, 5\} \setminus \{2\} = \{5\}$

$\mathcal{S}_1 \leftarrow (2, \mathcal{S}_2) = (2, 4, 3)$

Step 2C: Recursive update of the index of edge 1

$g \leftarrow \text{argmin}_{f \in \{5\}} I_f = 5$ and $I_g = 25 > 16.47 = I_1$ (as $I_5 = 25$)

Step 2A: Edge selection

$E \setminus E^* = \{6\}$

$e \leftarrow 6$

$E^* \leftarrow \{1, 2, 3, 4, 5, 6, 7, 8\} = E$

$A(6) \leftarrow \{7, 8\}$

Step 2B: Initialization of index computation of edge 6

$g \leftarrow \text{argmin}_{f \in \{7,8\}} I_f = 7$ and $I_g = 5$ (as $I_7 = 5$ and $I_8 = 20$)

$C_6 \leftarrow c_6 + p_6 C_7 = 2 + (.5)(3) = 3.5$

$P_6 \leftarrow p_6 P_7 = (.5)(.6) = .3$

$I_6 \leftarrow C_6 / P_6 = 3.5 / .3 = 11.67$

$A_6 \leftarrow [A(6) \setminus \{7\}] \cup A_7 = \{7, 8\} \setminus \{7\} = \{8\}$

$\mathcal{S}_6 \leftarrow (7, \mathcal{S}_7) = (7)$

Step 2C: Recursive update of the index of edge 6

$g \leftarrow \text{argmin}_{f \in \{8\}} I_f = 8$ and $I_g = 25 > 11.67 = I_6$ (as $I_5 = 25$)

Step 2A: Edge selection

$E \setminus E^* = \emptyset$, STOP.

The results of the Index Computation Algorithm are next illustrated in Figure 3. The computed indices are shown below each edge, whereas the number of the edge is indicated above.

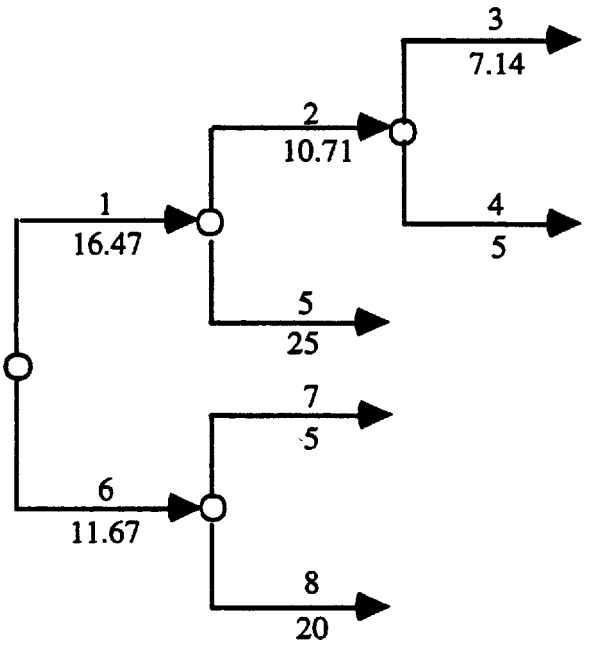

Figure 3

We now apply the Sequencing Algorithm with input E

The Sequencing Algorithm

Step 1: Initialization

$A \leftarrow I^*(E) = \{1, 6\}$

$\mathcal{T} \leftarrow \emptyset$

$\mathcal{I} = \emptyset$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{1,6\}} I_f = 6$ (as $I_1 = 16.47$ and $I_6 = 11.67$)

$A \leftarrow (A \setminus \{6\}) \cup A(6) = (\{1, 6\} \setminus \{6\}) \cup \{7, 8\} = \{1, 7, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 6) = (6)$

$\mathcal{I} = \{6\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{1,7,8\}} I_f = 7$ (as $I_1 = 16.47$, $I_7 = 5$ and $I_8 = 20$)

$A \leftarrow (A \setminus \{7\}) \cup A(7) = (\{1, 7, 8\} \setminus \{7\}) \cup \emptyset = \{1, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 7) = (6, 7)$

$\underline{\mathcal{T}} = \{6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{1,8\}} I_f = 1$ (as $I_1 = 16.47$ and $I_8 = 20$)

$A \leftarrow (A \setminus \{1\}) \cup A(1) = (\{1, 8\} \setminus \{1\}) \cup \{2, 5\} = \{2, 5, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 1) = (6, 7, 1)$

$\underline{\mathcal{T}} = \{1, 6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{2,5,8\}} I_f = 2$ (as $I_2 = 10.71$, $I_5 = 25$ and $I_8 = 20$)

$A \leftarrow (A \setminus \{2\}) \cup A(2) = (\{2, 5, 8\} \setminus \{2\}) \cup \{3, 4\} = \{3, 4, 5, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 2) = (6, 7, 1, 2)$

$\underline{\mathcal{T}} = \{1, 2, 6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{3,4,5,8\}} I_f = 4$ (as $I_3 = 7.14$, $I_4 = 5$, $I_5 = 25$ and $I_8 = 20$)

$A \leftarrow (A \setminus \{4\}) \cup A(4) = (\{3, 4, 5, 8\} \setminus \{4\}) \cup \emptyset = \{3, 5, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 4) = (6, 7, 1, 2, 4)$

$\underline{\mathcal{T}} = \{1, 2, 4, 6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{3,5,8\}} I_f = 3$ (as $I_3 = 7.14$, $I_5 = 25$ and $I_8 = 20$)

$A \leftarrow (A \setminus \{3\}) \cup A(3) = (\{3, 5, 8\} \setminus \{3\}) \cup \emptyset = \{5, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 3) = (6, 7, 1, 2, 4, 3)$

$\underline{\mathcal{T}} = \{1, 2, 3, 4, 6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{5,8\}} I_f = 8$ (as $I_5 = 25$ and $I_8 = 20$)

$A \leftarrow (A \setminus \{8\}) \cup A(8) = (\{5, 8\} \setminus \{8\}) \cup \emptyset = \{5\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 8) = (6, 7, 1, 2, 4, 3, 8)$

$\underline{\mathcal{T}} = \{1, 2, 3, 4, 6, 7, 8\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{5\}} I_f = 8$ (as $I_5 = 25$)

$A \leftarrow (A \setminus \{5\}) \cup A(5) = (\{5\} \setminus \{5\}) \cup \emptyset = \emptyset$

$\mathcal{T} \leftarrow (\mathcal{T}, 5) = (6, 7, 1, 2, 4, 3, 8, 5)$, STOP.

$\underline{\mathcal{T}} = \{1, 2, 3, 4, 5, 6, 7, 8\}$

Our results and the above computation determine that the order policy associated with the edge-sequence (6, 7, 1, 2, 4, 3, 8, 5) is optimal. We next apply the Accelerated Sequencing Algorithm to demonstrate its improved performance.

The Accelerated Sequencing Algorithm

Step 1: Initialization

$A \leftarrow I^*(E) = \{1, 6\}$

$\mathcal{T} \leftarrow \emptyset$

$\mathcal{I} = \emptyset$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{1,6\}} I_f = 6$ (as $I_1 = 16.47$ and $I_6 = 11.67$) and $S_6 = (7)$

$A \leftarrow (A \setminus \{6, 7\}) \cup A(6, 7) = (\{1, 6\} \setminus \{6, 7\}) \cup \{8\} = \{1, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 6, S_6) = (6, 7)$

$\mathcal{I} = \{6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{1,8\}} I_f = 1$ (as $I_1 = 16.47$ and $I_8 = 20$) and $S_1 = (2, 4, 3)$

$A \leftarrow (A \setminus \{1, 2, 3, 4\}) \cup A(1, 2, 3, 4) = (\{1, 8\} \setminus \{1, 2, 3, 4\}) \cup \{5\} = \{5, 8\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 1, S_1) = (6, 7, 1, 2, 4, 3)$

$\mathcal{I} = \{1, 2, 3, 4, 6, 7\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{5,8\}} I_f = 8$ (as $I_5 = 25$ and $I_8 = 20$) and $S_8 = \emptyset$

$A \leftarrow (A \setminus \{8\}) \cup A(8) = (\{5, 8\} \setminus \{8\}) \cup \emptyset = \{5\}$

$\mathcal{T} \leftarrow (\mathcal{T}, 8, S_8) = (6, 7, 1, 2, 4, 3, 8)$

$\mathcal{I} = \{1, 2, 3, 4, 6, 7, 8\}$

Step 2: Sequence construction:

$e \leftarrow \text{argmin}_{f \in \{5\}} I_f = 5$ (as $I_5 = 25$) and $S_5 = \emptyset$

$A \leftarrow (A \setminus \{5\}) \cup A(5) = (\{5\} \setminus \{5\}) \cup \emptyset = \emptyset$

$\mathcal{T} \leftarrow (\mathcal{T}, 5) = (6, 7, 1, 2, 4, 3, 8, 5)$, STOP.

$\mathcal{I} = \{1, 2, 3, 4, 5, 6, 7, 8\}$

8. Conclusions and Discussion

We have solved a probabilistic search problem on a rooted directed tree, thereby generalizing a well-known result by Smith [1956]. This problem presents a decision maker with choices concerning the tradeoff between risk and cost that are inherent in many choice situations, such as in multi-component R&D scheduling.

We have provided an algorithmic solution and analyzed the structure of the resulting solution. The most important contribution of the paper might be the demonstration of the dynamic aspect of optimal searches in this setting. Indeed, the solution exhibits local searches which at times are discontinuously interrupted; when a local search is interrupted the search continues, again in local fashion, in another part of the tree. In the R&D context, the optimal solution keeps several options "on hand" at any given time; these options might become reactivated late in the search as a function of the information generated on alternative options. Over time, an optimal search can revisit particular areas of the tree several times. These conclusions run counter to the common wisdom of "freezing" design options as soon as possible. In fact, in this model nothing gets frozen until the end of the search.

One open question is to go beyond the algorithm and to identify, as a function of edge parameters and the topology of the tree, the valuation of different positions in the tree.

The paper is the result of two separate motivations. The first one is to improve our understanding of the Gittins index result for multi-armed bandit problems, already referred to earlier. We expected the problem to have a representation as a simple bandit problem and the "index policy" we construct to be a particular case of a "Gittins index policy." We further hoped that the simple tree structure would add insight into important generalizations. But, we were not able to obtain a representation of our problem in a standard framework and, despite a potential representation of our indices through stopping times, we were not able to derive the optimality of our "index policies" as a straightforward application of existing results in the Gittins literature, let alone, provide insight into more complicated results. The main problem is the fact that edges

are not always explorable and that their availability for exploration at a particular point of time depends on the outcome of the search at that time. It follows that the model may fit into a framework of "bandit generating problem."

A second motivation lies in the solution of a directed stochastic shortest path problem. The problem we solve can be seen as a very special case of a stochastic shortest path problem obtained by connecting all endpoints of the leaves of the directed tree to a single terminal node. The general shortest path problem is a question we intend to turn to next.

Finally, further generalizations are relatively straightforward to consider. For example, consider the introduction of additional factors in the model than cost. One obvious factor is time, other factors might be any resource that can be brought to bear on the resolution of a particular uncertainty by influencing the outcome, costwise and probabilitywise.

Appendix

Proof of Lemma 4.1.

We use induction to prove the conclusions of the lemma after each update of edge-parameters. First, the conclusions of the lemma are trite after the execution of step 1. Next, for the inductive step, assume that the conclusions of the lemma hold at the end of the k-th update of edge-parameters and consider the (k+1)-st update. Suppose that the parameters of edge $e \in E \setminus L$ are updated at that stage. This update takes place through the execution of either step 2B or step 2C, and we consider the two cases separately. For convenience, we index all variables of e by the superscripts ^{o} and ^{n} to indicate old and new values.

First, assume that the (k+1)-st update takes place during an execution of step 2B. Let g be the selected edge in that step. Then g is an immediate successor of e, $\mathcal{S}_e^n = (g, \mathcal{S}_g)$, and $A_e^n = [A(e) \setminus \{g\}] \cup A_g$. Also, by the induction hypothesis, $(g, \mathcal{S}_g) \in \Gamma$, $A_g = A(\{g\} \cup \mathcal{S}_g)$, every edge in $A_g \cup \mathcal{S}_g$ is a successor of g, $C_g = C(g, \mathcal{S}_g)$ and $P_g = P(g, \mathcal{S}_g)$. Now, first, as g is a successor of e and the edges of $\mathcal{S}_g$ are successors of g, $(e, \mathcal{S}_e^n) = (e, g, \mathcal{S}_g) \in \Gamma$. Second,

$$A_e^n = [A(e) \setminus \{g\}] \cup A_g = [A(e) \cup A(\{g\} \cup \mathcal{S}_g)] \setminus \{g\} = A(\{e, g\} \cup \mathcal{S}_g) = A(\{e\} \cup \mathcal{S}_e^n).$$

Third, every edge in $A_g \cup \mathcal{S}_g$ is a successor of the immediate predecessor of g, namely of e, consequently, every edge in $A_e^n \cup \mathcal{S}_e^n = [(A(e) \setminus \{g\}) \cup A_g] \cup [\{g\} \cup \mathcal{S}_g]$ is a successor of e. Finally, from the induction assumption, $C_g = C(g, \mathcal{S}_g)$ and $P_g = P(g, \mathcal{S}_g)$, and therefore as $P^{(e,g)}(g) = p_e$, Lemma 2.3 (with $U = (e)$, $b = g$ and $\mathcal{V} = \mathcal{S}_g$) implies that

$$C(e, \mathcal{S}_e^n) = C(e, g, \mathcal{S}_g) = C(e) + P^{(e,g)} C(g, \mathcal{S}_g) = c_e + p_e C_g = C_e^n$$

and (as e is not a leaf assuring that $P(e) = 0$)

$$P(e, \mathcal{S}_e^n) = P(e, g, \mathcal{S}_g) = P(e) + P^{(e,g)} P(g, \mathcal{S}_g) = p_e P_g = P_e^n;$$

consequently, $I_e^n = C_e^n / P_e^n = C(e, \mathcal{S}_e^n) / P(e, \mathcal{S}_e^n) = I(e, \mathcal{S}_e^n)$.

Alternatively, assume that the (k+1)-st update takes place during an execution of step 2C. Let g be the selected edge in that step. Then $g \in A_e^o$, $\mathcal{S}_e^n = (\mathcal{S}_e^o, g, \mathcal{S}_g)$ and $A_e^n = [A_e^o \setminus \{g\}] \cup A_g$. Also, we observe that the induction hypothesis applies not just to the variables of e indexed by ^{o}, but also to the (final values of the) variables of g. Now, first, by the induction assumption $(e, \mathcal{S}_e^o) \in \Gamma$, $(g, \mathcal{S}_g) \in \Gamma$ and all elements in $\mathcal{S}_g$ are successors of g; these facts combine with the selection of g is as an immediate successor of e to prove that $(e, \mathcal{S}_e^n) = (e, \mathcal{S}_e^o, g, \mathcal{S}_g) \in \Gamma$. Second,

$$A_e^n = (A_e^o \setminus \{g\}) \cup A_g = [A(\{e\} \cup \mathcal{S}_e^o) \cup A(\{g\} \cup \mathcal{S}_g)] \setminus \{g\}$$

$$= A(\{e\} \cup \mathcal{S}_e^o \cup \{g\} \cup \mathcal{S}_g) = A(\{e\} \cup \mathcal{S}_e^n).$$

Third, again from the induction assumption, every edge in $A_g \cup \mathcal{S}_g$ is a successor of the immediate predecessor of g, namely of e, and every edge in $A_e^o \cup \mathcal{S}_e^o$ is a successor of e; consequently, every edge in $A_e^n \cup \mathcal{S}_e^n = [(A^o \setminus \{g\} \cup A_g] \cup [(\mathcal{S}_e^o \cup \{g\} \cup \mathcal{S}_g)]$ is a successor of e. Finally, from the induction assumption, $C_g = C(g, \mathcal{S}_g)$, $P_g = P(g, \mathcal{S}_g)$, $C(e, \mathcal{S}_e^o) = C_e^o$ and $P(e, \mathcal{S}_e^o) = P_e^o$, and therefore by Lemma 2.3 (with $U = (e, \mathcal{S}_e)$, $b = g$ and $\mathcal{V} = \mathcal{S}_g$)

$$C(e, \mathcal{S}_e^n) = C(e, \mathcal{S}_e^o, g, \mathcal{S}_g) = C(e, \mathcal{S}_e^o) + P^{(e, \mathcal{S}_e^o, g)}(g) C(g, \mathcal{S}_g) = C_e^n$$

and

$$P(e, \mathcal{S}_e^n) = P(e, \mathcal{S}_e^o, g, \mathcal{S}_g) = P(e, \mathcal{S}_e^o) + P^{(e, \mathcal{S}_e^o, g)}(g) P(g, \mathcal{S}_g) = P_e^n;$$

consequently, $I_e^n = C_e^n / P_e^n = C(e, \mathcal{S}_e^n) / P(e, \mathcal{S}_e^n) = I(e, \mathcal{S}_e^n)$. ||

Proof of Lemma 4.2.

We prove the lemma by induction of the number of times step 2 has been executed. Assume that the indices of all edges, whose parameters have been determined during the first $k \geq 0$ executions of step 2, satisfy the conclusions of the lemma. Consider the (k+1)-st execution of step 2, and assume that $e \in E \setminus L$ is the edge whose parameters are determined at that step.

We first observe that the final values of the parameters of all successors of e have already been determined, in particular, those of $A_e \cup \mathcal{S}_e$ (part (c) of Lemma 4.1). As in the proof of Lemma 4.1 we index all variables of e by the supescripts ^{o} and ^{n} to indicate old and new values.

Consider the execution of step 2B and let g be the selected edge in that step. By part (d) of Lemma 4.1, $I_e^n = I(e, \mathcal{S}_e^n) = I(e, g, \mathcal{S}_g)$. Now, part (c) of Lemma 4.1 assures that g is a predecessor of each edge in $\mathcal{S}_g$; hence, so is its immediate predecessor e. We conclude from this fact and inequality (3.6) of Lemma 3.2 (with $b = e$ and $\mathcal{V} = (g, \mathcal{S}_g) \neq \emptyset$) that $I_e^n = I(e, g, \mathcal{S}_g) > I(g, \mathcal{S}_g) = I_g$. Also, by the induction assumption $I_g > \max_{u \in \mathcal{S}_g} I_u$. So, $I_e^n > I_g = \max_{u \in \{g\} \cup \mathcal{S}_g} I_u = \max_{u \in \mathcal{S}_e^n} I_u$. Next, the selection of g as the minimizer of the index over A(e), together with the nondegeneracy assumption, imply that $I_g < \min_{v \in A(e) \setminus \{g\}} I_v$; also, the induction assumption assures that $I_g < \min_{v \in A_g} I_v$. Hence, $\max_{u \in \mathcal{S}_e^n} I_u = I_g < \min_{v \in [A(e) \setminus \{g\}] \cup A_g} I_v = \min_{v \in A_e^n} I_v$.

A secondary induction is next used to establish the conclusions of the lemma for executions of step 2C. Suppose the conclusions of the lemma hold at the beginning of step 2C and let g be the selected edge at that step. By part (d) of Lemma 4.1, $I_e^o = I(e, \mathcal{S}_e^o)$, $I_e^n = I(e, \mathcal{S}_e^n) = I(e, \mathcal{S}_e^o, g, \mathcal{S}_g)$ and $I_g = I(g, \mathcal{S}_g)$. Now, part (a) of Lemma 4.1 assures that $(e, \mathcal{S}_e^o, g, \mathcal{S}_g) \in \Gamma$, and part (c) of that lemma assures that g is a predecessor of each edge in $\mathcal{S}_g$; hence, Lemma 3.2 applies with $\mathcal{U} = (e, \mathcal{S}_e^o)$, $b = g$ and $\mathcal{V} = \mathcal{S}_g$. As the selection of g assures that $I(g, \mathcal{S}_g) = I_g < I_e^o = I(e, \mathcal{S}_e^o)$, we conclude from inequality (3.5) of Lemma 3.2 that $I_g = I(g, \mathcal{S}_g) < I(e, \mathcal{S}_e^n, \mathcal{S}_e^o, g, \mathcal{S}_g) < I(e, \mathcal{S}_e^o)$, proving that, $I_g < I_e^n < I_e^o$ (equality will not occur by the nondegeneracy assumption). Also, by the first induction assumption, $I_g > \max_{u \in \mathcal{S}_g} I_u$; by the secondary induction assumption, $\max_{u \in \mathcal{S}_e^o} I_u < \min_{v \in A_e^o} I_v = I_g$. So, $\max_{u \in \mathcal{S}_e^n} I_u = \max_{u \in \mathcal{S}_e^o \cup \{g\} \cup \mathcal{S}_g} I_u = I_g < I_e^n$. Also, the selection of g as the minimizer of the index over A_e^o (combined with the nondegeneracy assumption) and the induction assumption assure, respectively, that $I_g < \min_{v \in A_e^o \setminus \{g\}} I_v$ and that $I_g < \min_{v \in A_g} I_v$. Hence, $\max_{u \in \mathcal{S}_e^n} I_u = I_g < \min_{v \in (A_e^o \setminus \{g\}) \cup A_g} I_v = \min_{v \in A_e^n} I_v$. ||

Proof of Lemma 5.1.

The proof proceeds by induction. At initiation, $A = I^*(F)$ and $\mathcal{T} = \emptyset$ and the assertions (a) - (d) are trite. Assume that the parameters A^0 and $\mathcal{T}^0$ satisfy assertions (a) - (d) upon entering Step 2, and assume further that the algorithm does not stop in this iteration. Consider the updated values, say A^n and $\mathcal{T}^n$ of these parameters after the execution of Step 2. It is straightforward from the induction hypothesis that A^n and $\mathcal{T}^n$ satisfy (a) - (b). To verify (c) it suffices to show that if edge e is the selected edge in the considered execution of step 2, then all predecessors of e that are in F are in $\mathcal{T}^n = \mathcal{T}^0 \cup \{e\}$. Indeed, from part (b) of the inductive hypothesis, $e \in A^0 = [I^*(F) \setminus \mathcal{T}^0] \cup A(\mathcal{T}^0)$, hence, either e has no predecessor in F or has an immediate predecessor which is in $\mathcal{T}^0$, say e'; in the latter case, part (c) of the induction hypothesis assures that all predecessors of e' that are in F are in $\mathcal{T}^0$, implying that all predecessors of e that are in F are contained in $\mathcal{T}^n = \mathcal{T}^0 \cup \{e\}$. The established assertion (c) and the induction hypothesis asserting that $\mathcal{T}^0 \in \Gamma$ immediately implies that $\mathcal{T}^n = (\mathcal{T}^0, e) \in \Gamma$, verifying (d). Finally, (b) implies that A^n is empty if and only if $I^*(F) \subseteq \mathcal{T}^n$ and $A(\mathcal{T}^n) = \emptyset$. This situation occurs if and only if $\mathcal{T}^n$ contains $I^*(F)$ and all of the successors of edges in $I^*(F)$. As F is an upset of edges, the latter is equivalent to having $F \subseteq \mathcal{T}^n$. As (a) assures that $\mathcal{T}^n \subseteq F$, (e) follows.

The conclusion that the algorithm stops on command after exactly $|F|$ iterations, and at termination $\mathcal{T} = F$ is immediate from (e) coupled with the observation that the set $\mathcal{T}$ is initially empty and that each update of Step 2 augments it with a single element. ||

Proof of Lemma 5.2.

Our proof proceeds by backward induction on the elements of $\mathcal{T}^F$. To establish the base of the induction, let e be the last edge of $\mathcal{T}^F$. We then have that in the execution of Step 2 in which e is selected, the updating of A results in an empty set, implying that $A(e) = \emptyset$. So, e has no immediate successors, that is, e is a leaf. We conclude that $\mathcal{S}_e$ is empty and the assertion that in $\mathcal{T}^F$, e is immediately followed by $\mathcal{S}_e$ is trite.

Now, consider an edge $e \in \mathcal{T}^F$ where each edge that follows it in $\mathcal{T}^F$ is immediately followed by its continuation-sequence. Now, if e is a leaf, $S_e = \emptyset$ and the conclusion of the lemma is trivial. Alternatively, assume that e is not a leaf. Consider the execution of step 2 of the Index Computation Algorithm in which the parameters of e are computed and let $g_0, g_1, g_2, ..., g_q$ be, respectively, the edges selected in the executions of step 2B and the following steps 2C, ordered consecutively. Then $S_e = (g_0, S_{g_0}, g_1, S_{g_1}, ..., g_q, S_{g_q})$, and as F is an upset of edges, $g_0, g_1, g_2, ..., g_q$ as successors of e (part (c) of Lemma 4.1) are in F. We will prove by a secondary induction that for each $r = 0, 1, ..., q$, e is followed in $\mathcal{T}^F$ by $(g_0, S_{g_0}, g_1, S_{g_1}, ..., g_r, S_{g_r})$.

Consider first the execution of step 2 of the Sequencing Algorithm with input F in which edge e is added to the list $\mathcal{T}$ and let A^o and A^n denote the value of A at the beginning and at the end of that step. Then $e \in A^o$, $I_e = \min_{h \in A^o} I_h$ and $g_0 \in A(e) \subseteq A^n$. Now, as g_0 is the minimizer of the index among the immediate successors of e, we have that $I_{g_0} = \min_{h \in A(e)} I_h$, and as $g_0 \in S_e$, Lemma 4.3 implies that $I_{g_0} < I_e = \min_{h \in A^o} I_h$. We conclude that $I_{g_0} = \min_{h \in (A^o \setminus \{e\}) \cup A(e)} I_h = \min_{h \in A^n} I_h$, implying that g_0 is the selected edge in the following execution of step 2 of the Sequencing Algorithm (with input F), that is, g_0 immediately follows e in $\mathcal{T}^F$. We further have from the induction assumption that g_0 is immediately followed in $\mathcal{T}^F$ by the edges of S_{g_0}, ordered consecutively.

To continue with the secondary induction, suppose that for some integer $0 \le r < q$, we know that e is immediately followed in $\mathcal{T}^F$ by $\mathcal{U} \equiv (g_0, S_{g_0}, g_1, S_{g_1}, ..., g_{r-1}, S_{g_{r-1}})$; in particular, $\underline{\mathcal{U}} \subseteq \mathcal{T}^F = F$ (Lemma 5.1). Let $W \equiv \{e\} \cup \underline{\mathcal{U}}$. Consider the execution of step 2 of the Index Computation Algorithm in which the parameters of e are determined - the value of S_e at the beginning of step 2C in which edge g_r is the selected edge is $\mathcal{U}$, hence, by part (b) of Lemma 4.1, the value of A_e at that point is $A(\{e\} \cup \underline{\mathcal{U}}) = A(W)$ and the selection of g_r assures that $g_r \in A(W)$ and $I_{g_r} = \min_{h \in A(W)} I_h$. Also, as $g_r \in S_e$, Lemma 4.3 implies that $I_{g_r} < I_e = \min_{h \in A^o} I_h$. So, we have that $g_r \in A(W)$ and $I_{g_r} = \min_{h \in (A^o \setminus W) \cup A(W)} I_h$. Now, consider the beginning of the execution of step 2 of the Sequencing Algorithm with input F that

follows the one in which the last edge in $\mathcal{U}$ is added to $\mathcal{T}$. Observing that the value of A at that point is $(A^0 \setminus W) \cup A(W)$, we have that $g_r \in A$ and $I_{g_r} = \min_{h \in A} I_h$. This implies that g_r is the selected edge, that is, g_r immediately follows the edges of $\mathcal{U}$ in $\mathcal{T}^F$. We further have from the induction assumption that g_r is immediately followed in $\mathcal{T}^F$ by the edges of $\mathcal{S}_{g_r}$, ordered consecutively. Thus we established the secondary inductive assumption, and thereby completed our proof. ||