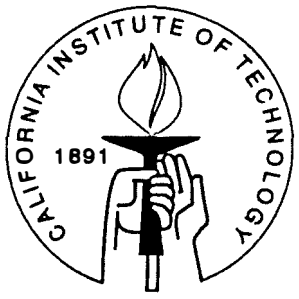
DIVISION OF THE HUMANITIES AND SOCIAL SCIENCES
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA 91125

A BOTTOM-UP EFFICIENT ALGORITHM FOR ALLOCATING PUBLIC PROJECTS WITH POSITIVE COMPLEMENTARITIES

Scott E. Page



SOCIAL SCIENCE WORKING PAPER 885

May 1994

A BOTTOM-UP EFFICIENT ALGORITHM FOR ALLOCATING PUBLIC PROJECTS WITH POSITIVE COMPLEMENTARITIES

Scott E. Page

Abstract

In this paper, we consider the problem of locating an optimal package of public projects from a set of potential projects when the public projects have positive complementarities. We formulate the problem as a discrete nonlinear optimization problem whose domain equals the power set of a finite collection of projects. The main contribution of this paper is the construction of an efficient algorithm, among the set of *bottom–up algorithms*, for projects with *positive* and *positive uniform complementarities*. The restriction to bottom–up algorithms stems from practical considerations discussed in the paper. We also discuss shotcomings of three natural approaches to addressing the problem: exhaustive search over packages, simultaneous evaluation of projects, and sequential evaluation of projects.

A BOTTOM-UP EFFICIENT ALGORITHM FOR ALLOCATING PUBLIC PROJECTS WITH POSITIVE COMPLEMENTARITIES

Scott E. Page*

1 Introduction

Approaches to the problem of allocating public goods and deciding upon public projects take at least three forms. One branch of research concerns the design of mechanisms in which truthful revelation of preferences for a public good or package of public goods is incentive compatible (Groves and Ledyard 1977, Green and Laffont 1977).[1] Recently, Moulin and Jackson (1992) have adopted this approach to the problem of making decisions on public projects. A second branch concerns the construction of political institutions to make decisions on public goods and projects (Inman 1988). A third approach evaluates public projects using cost-benefit analysis (Dreze and Stern 1986).

In the case of public projects, the aforementioned models typically consider an individual project or package of projects. They do not address the problem considered here: *the selection of an optimal package of projects from a set of feasible projects given complementary effects among projects.* And yet, accepting the noncontroversial assumption that the values of many public projects are interdependent: for example, investments in roadways and public transportation increase the values of museums, parks, and stadia; this is the relevant economic problem. Though the aforementioned models can be extended so that all packages of projects are evaluated and the best selected, the enormous number of potential packages which can be created from even a moderate number of projects,

*Division of Humanities and Social Sciences 228-77, California Institute of Technology, Pasadena CA 91125. This paper was a portion of the author's dissertation from Northwestern University. The author's committee members: Mark Satterthwaite, Roger Myerson, and Matthew Jackson and, especially his chairman, Stan Reiter, deserve many thanks for their support, guidance, and advice. Gordon Green and Mike Kirschenheiter commented on and improved several early versions of this paper.

[1]For an overview of solutions to the the problem of allocating public goods see Cornes and Sandler (1986) or Oakland (1987).

precludes preference revelation over the entire set of packages (Bergson 1976).[2] Moreover, as we show in this paper, the assumption of complementary effects among projects implies that applying these mechanisms to evaluate projects either independently or in sequence will not necessarily locate the optimal package.

In this paper, we restrict attention to public projects in which the complementarities are positive. There are three reasons for this restriction. First, to make the selection over packages of projects tractable, we cast the problem as a discrete nonlinear programming problem (Reiter and Sherman 1962). The general problem, in which arbitrary complementaries are allowed, is so large as to require decomposition into cases. Decades of combinatorial optimization theory demonstrates that no single institution, mechanism, or algorithm is efficient for *all* types of complementarities among projects. The relevant question becomes not whether to divide into classes but how to make the division. This leads to our second reason for concentrating on positive complementarities. It seems a logical initial class to consider. Many public projects, such as schools, infrastructure, and public transportation appear to create positive external effects. Finally, even though we assume that the exact value of the complementary effects cannot be known prior to evaluating projects, we might assume that the *class* of complementarities can be known. We might guess (correctly) that a sewer system surrounding an inland lake imposes a positive externality on nearby public parks and beaches, even though we might not know the size of the externality. Provided that the class of functions is known, the appropriate algorithm can be applied.

This paper then can be interpreted as slicing off a class of complementarities and constructing an efficient algorithm for this class. This raises the question of what we mean by "efficient." We rely on the notion of *bottom–up* efficient which has three requirements: First, the algorithm must be weakly minimal in the number of packages evaluated, i.e. there cannot exist another algorithm which never evaluates more packages and sometimes evaluates fewer packages. Second, it must always locate the optimal package. Third, it must be *bottom–up*: in each iteration it only may add projects to the current best package of projects. The restriction to bottom–up algorithms is motivated by practical considerations discussed in section 2 of this paper.

The main result of this paper is the construction of a bottom–up efficient algorithm for projects with positive complementarities and for projects with positive complementarities generated by a single project. We refer to the latter case as the class of *unidirectional* complementarities.[3] The remainder of this paper is organized as follows: In Section 2, we formally define the *multiple public projects* problem as well as value functions defined by *positive* and *positive unidirectional* complementarities. We show that both classes of functions belong to the class of quasi-supermodular functions. In Section 3, we define *sequential* and *simultaneous* evaluation of projects and demonstrate with an example

[2]In the case of a Clarke mechanism (Clarke 1976), agents would be required to report a valuation for each subset of projects, which in the case of thirty projects would number over a *billion*.

[3]Although at this point our results take the form of an algorithm, we expect that insights gained from the algorithm can be incorporated into mechanisms or institutions.

how each may fail to locate the optimal package of projects. In Section 4, we explore the problems of exploding and collapsing complexity. In Section 5, we present a formal characterization of *bottom–up efficiency*. In Section 6, we construct an algorithm, ALGO–BU, which is bottom–up efficient for projects with positive and positive unidirectional complementarities. In the final section, we demonstrate that ALGO–BU is not bottom–up efficient for all subclasses of quasi-supermodular functions and discuss parallellizing the algorithm.

2 The Multiple Public Projects Problem

2.1 Preliminaries

We assume that there are n potential projects and that the decision on each project takes the form of a simple "yes" or "no" (Harris 1978, Boeri 1990). Within this framework, a similar project at a different location or an otherwise identical project of larger size may be treated as a separate project.[4]

The set of **projects**, $N = \{1, 2, ..n\}$

A *package* of projects is a subset of N and represents a decision on each project.

A **package** *of projects,* $X \subseteq N$

We denote the set of all possible packages by S. S is just the power set of N.

The **set of packages**, $S = \{X : X \subseteq N\}$

We further assume that a value can be assigned to each package of projects and that there exists a government or social planner whose objective is to choose the package with the highest value. The value of a package may be interpreted as the aggregate of individual agents' valuations, as the value of a social welfare function defined over S, or as the outcome of cost-benefit analysis. Without loss of generality we assume that the value of providing no projects equals zero.

The **class of value functions**, $G \subseteq V^0 = \{V : V : S \rightarrow \Re, V(\emptyset) = 0\}$

[4]However, it is unlikely that two similar projects confer positive externalities upon one another.

We further assume that there is an *actual value function*, ϑ which belongs to G. We assume that the actual value function is not known a priori. Instead, it is revealed through the evaluation of packages of projects. When a package consists of costly projects, then undertaking these projects may cause a shift in the prices of private goods (Besley and Jewitt 1991, Harris 1978). When this occurs, cost-benefit analysis can becomes problematic (Chipman and More 1978). Nevertheless, we assume that a well-defined value function exists.

The **actual value function**, $\vartheta \in G$

Given our assumption of complementaries among projects, the value of a project depends upon which other projects belong to the package of projects to be undertaken. We distinguish between the value of a project alone, its *isolated value*, and its value in conjunction with other projects, its *marginal value.*

The **isolated value** *of project* $i \in N$, $\vartheta(\{i\})$

The **marginal value** *of project* $i \in N$ *given* $X \subset N$, $MV(i, X) = \vartheta(\{i\} \cup X) - \vartheta(X)$

In the next subsection, we define positive complementarities among projects. Positive complementarities imply that the marginal value of a project is at least as large as its isolated value. For example, the combined value of Yellowstone National Park and the federal highways in Wyoming probably exceeds the sum of the isolated values of Yellowstone and the highways.

In section 4, we define a bottom–up efficient algorithm, ALGO–BU, for functions formed by *positive complementarities* (POS) and functions formed by *positive unidirectional complementarities* (UNI). In order to define these classes of complementarities, we first define the *decomposition basis* (Liepins and Vose 1991).

Given $V : S \rightarrow \Re$, *there exists a unique vector* $\beta_V = (\beta_{V,\emptyset}, \beta_{V,I}, ...\beta_{V,N})$ *indexed over* $I \subseteq N$ *where* $\beta_{V,I} \in \Re$ *and*

$$V(X) = \sum_{I \subseteq X} \beta_{V,I} \quad \forall\, X \subseteq N$$

β_V are the **decomposition basis coefficients** for V,

The decomposition basis is best understood through an example. Consider a community which is deciding whether to build a school, a library, both projects together, or neither project. Suppose that the school's isolated value equals four million dollars, that the library's isolated value equals negative two million dollars, and that the value of the two projects together equals five million dollars.

Example: $N = \{\text{school, library}\}$

$$\vartheta(\emptyset) = 0 \quad \vartheta(\{school\}) = 4 \quad \vartheta(\{library\}) = -2 \quad \vartheta(\{school, library\}) = 5$$

The decomposition basis coefficients are computed as follows: the coefficient of an individual project equals the project's isolated value; and the coefficient for a pair of projects equals the difference between the pair's value and the sum of the coefficients (the isolated values) of the two projects.

$$\beta_{\vartheta,\emptyset} = 0 \quad \beta_{\vartheta,\{school\}} = 4 \quad \beta_{\vartheta,\{library\}} = -2 \quad \beta_{\vartheta\{school,library\}} = 3$$

The decomposition basis can be used to define the class of value functions formed by positive and the class formed by positive unidirectional complementarities.

2.2 Positive and Positive Unidirectional Complementarities

The class of value functions *formed by positive complementarities*, which we denote by POS, is characterized by two features. First, the isolated value of each project is bounded from below. Second, all nonlinear effects, as measured by the decomposition basis, are positive. An implication of the second assumption is that any project with a positive isolated value also has a positive marginal value given any package of projects.

POS $= \{V \in V^0 : \exists\, \theta \in \Re^+$ *such that* $\beta_{V,I}$ *satisfy (i) and (ii)*$\}$

$$\begin{array}{lll} (i) & -\theta \leq \beta_{V,I} & \text{for } I \text{ s.t. } \mid I \mid = 1 \\ (ii) & \beta_{V,I} \geq 0 & \text{for } I \text{ s.t. } \mid I \mid > 1 \end{array}$$

We next define the class of value functions *formed by positive unidirectional complementarities*, which we denote by UNI. This class of value functions characterizes public

projects in which a single project (project 1) confers a positive complementary effect on the other $N-1$ projects, whose values are otherwise independent. For example, a dam upstream from a series of public parks separated by great distances may increase the parks' values by regulating flows, while the values of the parks may be otherwise independent.

$$\mathbf{UNI} = \{V \in V^0 : \exists\, \theta \in \Re^+ \textit{ such that } \beta_{V,I} \textit{ satisfy (i) - (iii)}\}$$

$$\begin{array}{lll} \textit{(i)} & -\theta \leq \beta_{V,I} & \textit{for } I \textit{ s.t. } \mid I \mid = 1 \\ \textit{(ii)} & \beta_{V,I} = 0 & \textit{if } \mid I \mid \geq 2 \textit{ and } 1 \notin I \\ \textit{(iii)} & \beta_{V,I} \geq 0 & \textit{if } \mid I \mid \geq 2 \textit{ and } 1 \in I \end{array}$$

Observe that if a value function V belongs to UNI then it also belongs to POS but that the converse does not hold, so UNI is a strict subset of POS. To simplify the analysis, we first show that both POS and UNI belong to the class of quasi-supermodular functions, QSM, which is defined as follows:

$$\mathbf{QSM} = \{V : V \in V^0, V \textit{ satisfies (i)-(ii)}\}$$

$$\begin{array}{lllll} \textit{(i)} & V(X) \geq V(X \cap Y) & \Rightarrow & V(X \cup Y) \geq V(Y) & \forall\, X, Y \subseteq N \\ \textit{(ii)} & V(X) > V(X \cap Y) & \Rightarrow & V(X \cup Y) > V(Y) & \forall\, X, Y \subseteq N \end{array}$$

In fact, both classes of functions are *supermodular*, an even stronger condition (Topkis 1976, Fisher, etal. 1978). Supermodularity is a discrete analog to a positive second derivative. The class of *supermodular functions*, SM, is defined as follows:

$$\mathbf{SM} = \{V : V \in V^0,\, V(X) - V(X \cap Y) \leq V(X \cup Y) - V(Y) \forall\, X, Y \subseteq N\}$$

We rely only on quasi-supermodularity in our analysis.

3 Simultaneous and Sequential Evaluation

A natural approach to the multiple public projects problem is to evaluate the projects simultaneously and to undertake those projects with positive isolated values. We will refer to this as *simultaneous evaluation of projects.* Alternatively, the projects might be

evaluated in sequence with those projects with positive marginal values given previous projects being undertaken. We will refer to this as *sequential evaluation of projects.* In this section, we formalize both of these approaches and demonstrate through an example that both may result in suboptimal packages of projects being undertaken.

The two approaches fail for different reasons. Simultaneous evaluation considers only the isolated values of projects. Therefore, a project with a negative isolated value but which confers massive positive complementarities will not be undertaken. Sequential evaluation, on the other hand, does take into account some, though not all, complementarities between projects. For instance, if a project with a negative isolated value confers substantial positive complementarities on projects not yet considered, then it will not be undertaken.

The following example serves three purposes. First, it formalizes what we mean by both sequential and simultaneous evaluation of projects. Second, it formalizes the intuition described above as to how both approaches may fail to locate the optimal package. Third, it provides an example of a value function which belongs to the class UNI.

Example: $N = \{1, 2, 3\}$

$$\begin{array}{llll} V(\emptyset) = 0 & V(\{1\}) = 2 & V(\{2\}) = -3 & V(\{3\}) = -1 \\ V(\{1,2\}) = 1 & V(\{1,3\}) = 4 & V(\{2,3\}) = -4 & V(\{1,2,3\}) = 6 \end{array}$$

First, we consider simultaneous evaluation of projects. Each project is evaluated in isolation. The only project with a positive isolated value is project 1. Therefore, simultaneous evaluation leads to the provision of only the first project. It failed to locate the optimal package because it could not measure the complementarities between projects. Turning now to sequential evaluation of projects, we first place the projects in an order. For convenience, we assume that the projects are ordered 1,2,3. Project 1 is undertaken because it has a positive isolated value. Project 2 is not undertaken because its *marginal value* given project 1 equals -1. Project 3, however, has a marginal value of 2 given project 1, so it is undertaken. Therefore, sequential search leads to the undertaking of the package {1,3}. The optimal package, {1,2,3}, is not located because project 2 only has a positive marginal value when both projects 1 and 3 are being undertaken.

4 Complexity

The value functions discussed so far are all nonlinear. In this section, we discuss the measurement of nonlinearity for the multiple public projects problem. Our most important

conclusion is that measures which rely upon counting the number of variables comprising the nonlinear effects may be inaccurate. Suppose that we restrict complementarities to pairs of projects. In other words, there do not exist external effects between larger subsets of projects. Two would be a natural choice for the *size* of the resulting value function. We show that this measurement may be misleading. On the one hand, simple nonlinear effects may combine to form complex problems. To demonstrate this point, we construct what we refer to as a set of *collective public projects*, in which no strict subset of projects has a positive value, but in which the set of all projects merits undertaking. On the other hand, large nonlinear effects between many variables may be irrelevant for optimization. Our analysis consists of three parts. First, we define the *decomposition size* of a value function, which formalizes the notion of size just discussed. Second, we demonstrate through two examples and a claim how this measure may confuse potential and relevant complexity. And finally, we mention an alternative notion of complexity, *cover size* which relates to the number of project decisions which must be coordinated in order to locate the optimal package of projects (Page 1993).

Decomposition size is calculated using the decomposition basis coefficients defined in the previous section. The decomposition size of a value function V equals the cardinality of the largest subset which has a nonzero decomposition basis coefficient.

The **decomposition size** *of* V, $size_d(V) = max\ \{\mid I \mid :\ I \subseteq N,\ \beta_{V,I} \neq 0\}$

In the example of Section 2 which included the school and the library, the decomposition size of the value function would equal two because the subset consisting of both the school and the library had a positive decomposition basis coefficient. Decomposition size measures nonlinearity by the size of the *encoded* nonlinear effects. The two examples which follow help to clarify this notion of complexity and demonstrate weaknesses of decomposition size as a measure.

Example: $N = \{1,2,3\}$ Let $\beta_{V,I} = \mid I \mid \ \forall\, I \subseteq N$. Therefore,

$$\begin{array}{llll} V(\emptyset) = 0 & V(\{1\}) = 1 & V(\{2\}) = 1 & V(\{3\}) = 1 \\ V(\{1,2\}) = 4 & V(\{1,3\}) = 4 & V(\{2,3\}) = 4 & V(\{1,2,3\}) = 12 \end{array}$$

Though this function has a decomposition size of three, it is not difficult to solve. Either simultaneous or sequential evaluation of projects locates the optimal package. The complementarities between projects are irrelevant in that the isolated values of the projects are sufficient information for optimization.

The example below has a smaller decomposition size, yet we argue that it is more complex. In this example, each project has a negative isolated value, but there are positive complementarities between pairs of projects.

Example: $N = \{1, 2, 3, 4\}$ Define $\beta_{V,I}$ as follows:

$$\begin{array}{lll} \beta_{V,I} & = -4 & \text{if } \mid I \mid \; = 1 \\ \beta_{V,I} & = 3 & \text{if } \mid I \mid \; = 2 \\ \beta_{V,I} & = 0 & \text{else} \end{array}$$

Therefore,

$$\begin{array}{lll} V(\emptyset) = 0 & & \\ V(X) = -4 & & \text{if } \mid X \mid \; = 1 \\ V(X) = -5 & & \text{if } \mid X \mid \; = 2 \\ V(X) = -3 & & \text{if } \mid X \mid \; = 3 \\ V(N) = 2 & & \end{array}$$

In this example, the decomposition size of the function equals two. However, neither simultaneous nor sequential evaluation of projects locates the optimal package of projects. In fact, each will lead to *no* projects being undertaken. This function is clearly more difficult to optimize than the function in the previous example, yet it has a smaller decomposition size. This example serves a second purpose as well. When every proper subset of projects has a negative value yet the set of all projects has a positive value, we say that the value function forms a set of *collective public projects.*

A value function, V forms a set of **collective public projects** *if the following two strict inequalities hold:*

$$V(X) < 0 \qquad \forall\, X \subset N,\, X \neq \emptyset$$

$$V(N) > V(\emptyset) = 0$$

We now show that this contrived example can be extended to allow for any number of projects. Claim **??** states that for any positive integer n, there exists a value function defined over n projects with a decomposition size of two which forms a set of *collective public projects.*

Claim 4.1 *For any positive integer n and set of projects N, such that* $\mid N \mid = n$, $\exists\, V : S \rightarrow \Re$ *s.t.* $size_d(V) = 2$ *and V forms a set of collective public projects of size n.*

pf: If $n = 2$, the proof is trivial. Therefore, assume that $n \geq 3$. Define the decomposition basis coefficients of V as follows:

$$\begin{array}{lll} \beta_{V,I} & = -1 & \text{if } \mid I \mid = 1 \\ \beta_{V,I} & = \frac{4}{2n-3} & \text{if } \mid I \mid = 2 \\ \beta_{V,I} & = 0 & \text{else} \end{array}$$

Therefore,

$$\begin{array}{ll} V(\emptyset) = 0 & \\ V(X) = -1 & \text{if } \mid X \mid = 1 \\ V(X) = \dfrac{2 \mid X \mid (\mid X \mid -1)}{2n-3} - \mid X \mid & \text{if } 1 < \mid X \mid < n \\ V(N) = \dfrac{n}{2n-3} & \end{array}$$

It is straightforward to show that $V(X) < 0$ for $0 \subset X \subset N$ and that $V(N) > 0$.

The constructive proof of Claim ?? shows how simple complementary effects may combine to form complex value functions. The size of the encoded effects therefore may not be an appropriate measure of the relevant complexity for optimization. Page (1993) has proposed an alternative measure of the amount of nonlinearity, or complexity, called *cover size*. Loosely speaking, a cover for a function determines the extent to which a problem can be decomposed into subproblems which can be solved in parallel. The *size* of a cover equals the number of decisions in the largest subproblem. For example, the function in the first example in this section can be decomposed into three subproblems, each consisting of one project. Its cover size equals one. In contrast, the function in the second example in this section cannot be decomposed at all, all four projects must be considered together.[5] Therefore, this function has a cover size of four.

5 Bottom–Up Efficiency

Algorithmic performance may be measured by the number of searches required to locate the optimum package, or the number of searches required to locate and verify the optimum package. We rely on the latter criterion because it is more reasonable. If, for example, the optimal package consisted of no projects being undertaken, we would like to know how many projects would be evaluated before this fact was recognized. Our notion

[5]Evaluation of any proper subset of projects would lead to that package of projects not being undertaken.

of efficiency also includes a criterion pertaining to the method of search: we require that our algorithm be *bottom–up*. An algorithm is bottom–up if any package of projects to be evaluated is a superset of the current best package of projects. In the example with the school and the library, a bottom–up algorithm which first evaluates the school must then evaluate the school and the library as a package. It cannot evaluate the library alone. This follows from the fact that the school has a positive value.

The restriction to bottom–up algorithms is motivated by four practical considerations specific to the multiple public projects problem. First, decisions on some public projects may be required before the entire set of potential projects is known to the decision maker. For example, in the United States, decisions on where to build federal highways were made long before projects such as the supercollider once under construction in Texas were even part of the choice set. Second, evaluating the costs and benefits of an addition to an existing package of projects at a given level of accuracy should be more precise and less costly than evaluating the costs and benefits of an addition to a *hypothetical* package of projects.[6] Third, decisions on public projects are often made by different government sectors. A bottom–up algorithm can be "parallellized" to a some degree as we discuss in the conclusion. Finally, once a project has been undertaken, the political costs of abandoning the project may be significant. The supercollider notwithstanding, projects may persist even though their costs outweigh their benefits.

Bottom–up efficiency differs from standard notions of algorithmic efficiency in that it emphasizes more than just the number, or the order of the number of searches. ALGO–BU is not, in general, *efficient* in the traditional sense. For problems with positive complementarities, the Ford-Fulkerson algorithm, which is not bottom–up, evaluates a number of packages which is polynomial in the number of projects (Ford and Fulkerson 1962). In comparison, for some functions ALGO–BU searches the entire space of packages, which is exponential in the number of projects.

In order to define an algorithm, we need to be explicit about what information is known prior to and during search. We assume prior knowledge of the class of functions from which the actual value function is drawn, but no prior knowledge of the likelihood of any particular function being drawn. During an algorithm's search, the evaluated packages provide additional information about the actual value function. We assume that an algorithm maintains a *record* of the packages which have been evaluated and their function values which it uses to determine which package to evaluate next.

A **record** *of* $\vartheta, \Lambda(\vartheta) = \{(X_1, \vartheta(X_1)), ..., (X_{2^n}, \vartheta(X_{2^n}))\}$ *where* $X_i \subseteq N$

Note that the definition of a record does not include the restriction that the X_i's are

[6]This assumption can be motivated with an example. Suppose that the first project is a public transit system. Total usage of a hypothetical transit system is a random variable with substantial variation. In contrast, once the transit system has been built, total usage is hard data. Less uncertainty should imply less costly and more precise predictions of the effects of future projects.

unique. Therefore, two distinct value functions can have identical records provided that some package is represented more than once in each record. In fact, for the algorithms we construct, prior to any package evaluations, all of the X_i's in a record will equal the empty set. To simplify the discussion, each package in a record and its value will be referred to as an *entry.*

An **entry**, $\Lambda_i(\vartheta) = (X_i, \vartheta(X_i))$ *where* $X_i \subseteq N$

An algorithm's choice of the next package to be evaluated is determined by a *search rule.* A search rule maps a record into the set of packages. If the search rule maps into the empty set, the search stops. The search rule takes into account the class of functions from which ϑ was drawn.

The set of **search rules**, $R = \{\tau : \tau : \Lambda(\vartheta) \times G \rightarrow S,\ \vartheta \in G\}$

An *algorithm* applied to $\vartheta \in G$ consists of an initial package T_1 and a search rule τ_A.

An **algorithm** *applied to* $\vartheta \in G$, $A = (T_1, \tau_A)$ *with* $T_1 \subseteq N$ *and* $\tau_A \in R$. *Search is defined as follows:*

Step 0:	$\Lambda(\vartheta)_i = (\emptyset, \vartheta(\emptyset))$	*for* $i = 1$ *to* 2^n
	$i = 1$	{*iteration number*}
	$T_i \subseteq N$	
Step 1:	*Evaluate* $T_i \subseteq N$	
Step 2:	$\Lambda_i(V) = (T_i, \vartheta(T_i))$	
Step 3:	*if* $\tau_A(\Lambda(\vartheta), G) = \emptyset$	*goto Step 5*
	if $\tau_A(\Lambda(\vartheta), G) = X$	*goto Step 4*
Step 4:	$i = i + 1$	
	$T_i = X$	
	goto Step 1	
Step 5:	$num(A, \vartheta, G) = i$	

$T_i(A, \vartheta, G)$ denotes the ith package evaluated by A when A is applied to ϑ in G.

The **ith package evaluated by** A *given* ϑ *in* G, $T_i(A, \vartheta, G)$

The *number of searches* by A for ϑ in G equals the number of packages which are evaluated when the algorithm is applied to ϑ in G.

The **number of searches by** A *for* ϑ *in* G, $num(A, \vartheta, G)$

Two algorithms are said to be *equivalent* for G if they create identical records for all ϑ in G.

A *and* $\acute{A}$ *are* **equivalent algorithms**, $A \equiv \acute{A}$, *iff* $T_i(A, \vartheta, G) = T_i(\acute{A}, \vartheta, G)$ *for all* $\vartheta \in G$ *and all* $i \leq max\{num(A, \vartheta, G), num(\acute{A}, \vartheta, G)\}$

The *best package* up to m iterations of A is the package among the first $m-1$ packages evaluated for which ϑ has the highest value. If more than one package has the highest value, then the last package evaluated from among those with the highest value is considered best.

The **best package** *up to* m *iterations of* A, $B_m(A, \vartheta, G)$ *satisfies* $(i) - (iii)$

$$\begin{array}{lll} (i) & B_m(A, \vartheta, G) = T_i(A, \vartheta, G) & \text{for some } i < m \\ (ii) & \vartheta(B_m(A, \vartheta, G)) \geq \vartheta(T_j(A, \vartheta, G)) & \forall\, j < i \\ (iii) & \vartheta(B_m(A, \vartheta, G)) > \vartheta(T_k(A, \vartheta, G)) & \forall\, k \text{ s.t. } i < k < m \end{array}$$

Bottom–up efficiency consists of three conditions. The first condition is that the algorithm be *optimizing* over G.

An algorithm A *is* **optimizing** *over* G *iff* $\forall\, \vartheta \in G$, $\vartheta(B_{n^*}(A, \vartheta, G)) > \vartheta(X)\ \ \forall\, X \subseteq N$, *where* $n^* = num(A, \vartheta, G)$

The second condition is that the algorithm be *bottom–up* which requires that each T_i contain the best package up to i. The bottom–up condition can be interpreted as not allowing the abandonment of a project which belongs to the current best package.

A *is* **bottom–up** *in* G *iff* $B_m(A, \vartheta, G) \subseteq T_m(A, \vartheta, G)\ \ \forall\, m \leq num(A, \vartheta, G)\ \ \forall\, \vartheta \in G$

Hereafter, $B_m(A, \vartheta, G)$ and $T_i(A, \vartheta, G)$ are denoted by B_m and T_i respectively. The third condition for bottom–up efficiency is that the algorithm be *weakly minimal in the*

number of projects evaluated, i.e. if G is the class of functions under consideration, given any other bottom–up, optimizing algorithm, $\acute{A}$, there exists a function V in G such that A relies on strictly fewer searches than $\acute{A}$.

An algorithm A is **bottom–up efficient** *for a class of functions G if it satisfies (i)-(iii):*

(i) A *is optimizing over* G
(ii) A *is bottom–up in* G
(iii) if $\acute{A} \not\equiv A$ *and* $\acute{A}$ *satisfies (i) and (ii), then* $\exists\, V \in G$ *such that* $num(A, V, G) < num(\acute{A}, V, G)$

In the next section we define an algorithm ALGO–BU which we later show to be bottom–up efficient for POS and UNI.

6 ALGO–BU

In this section, we construct a bottom–up algorithm, ALGO–BU, which is optimizing for QSM and bottom–up efficient for both POS and UNI. ALGO–BU can be described as follows: First, the packages of projects are ordered using the standard map from S into the integers. Second, the packages are "considered" for evaluation according to that order. ALGO–BU evaluates a considered package if and only if the *strong dominating set* of the package is empty given the values of previously evaluated packages.[7]

6.1 Dominated Projects

Before defining ALGO–BU, we define *undominated* packages of projects. We will show in an example that undominated is a weaker restriction than having a nonempty strong dominating set. Let G be a class of value functions. A package of projects, X, is *undominated* in G if there exists a value function V which belongs to G such that X is the unique maximizer of V.

A package $X \subseteq N$ is **undominated** *in G if* $\exists\, V \in G$ *s.t.* $V(X) > V(Y)\ \forall\, Y \subseteq N$, $Y \neq X$.

[7]To determine whether or not the strong dominating set of a package is empty entails solving a linear programming problem. For fewer than a hundred projects, these computational costs will be inconsequential compared to the costs of evaluating the benefits and costs of the package of projects.

Alternatively a package X is dominated in G if for any value function, V, which belongs to G, there exists a package which has at least as high a value under V.

A package $X \subseteq N$ *is* **dominated** *in* G *if* $\forall\, V \in G\ \exists\, Y \neq X$ *s.t.* $V(Y) \geq V(X)$

As mentioned above, ALGO–BU evaluates a package only if has an empty strong dominating set. The strong dominating set is defined relative to a class of feasible value functions G. The strong dominating set of a package X in G consists of all packages Y which satisfy three conditions: first, for any function V which belongs G, $V(Y)$ must be at least as large as $V(X)$; second, Y must strictly contain X; and third, for any package W containing X and contained in Y, $V(W)$ cannot be larger than $V(Y)$.

The **strong dominating set** *of* X *in* G, $SD(X,G) = \{\, Y : Y \subseteq N :$ Y *satisfies* $(i) - (iii)\,\}$

(i)	$V(X) \leq V(Y)$	$\forall : V \in G$
(ii)	$X \subset Y$	
(iii)	$X \subset W \subset Y \ \Rightarrow\ V(W) \leq V(Y)$	$\forall\, V \in G$

Corollary **??** shows that if the strong dominating set of X in G is not empty then X is dominated.

Corollary 6.1 $SD(X,G) \neq \emptyset \Rightarrow X$ *dominated in* $G\ \forall\, G \subseteq V^0$.

pf: $Y \in SD(X,G) \Rightarrow V(Y) \geq V(X)\, \forall\, V \in G$.

The implication does not hold in the other direction as is shown by the following example:

Example: Let $N = \{1,2,3\}$ and let $\acute{P} = \{V : V \in POS$, and $V(23) = 10\}$. It is easy to show $\acute{P} \subset$ QSM. Therefore, $V(\{1\}) < V(\{1,2,3\})\, \forall\, V \in \acute{P}$. By definition, $\{1\}$ is dominated in $\acute{P}$. However, $SD(\{1\}, \acute{P}) = \emptyset$.

Claim **??** states that if both Y and Z belong to $SD(X,G)$, and if Y is a subset of Z, then Z belongs to the strong dominating set of Y in G.

Claim 6.1 $Y, Z \in SD(X,G)$ *and* $Y \subseteq Z \Rightarrow Z \in SD(Y,G)\ \forall\, G \subseteq V^0$

pf: $Y \in SD(X,G)$ implies $X \subseteq Y$. By assumption, $Y \subseteq Z$. Therefore, $Z \in SD(X,G)$ implies $V(Y) \leq V(Z)$. Finally, given that $X \subseteq Y$, it follows immediately that $\{W : Y \subset W \subset Z\} \subseteq \{W : X \subset W \subset Z\}$, which completes the proof.

Hereafter, we assume that G is contained in QSM. Claim **??** states that if Y belongs to the strong dominating set of X and Z belongs to the strong dominating set of Y, then Z belongs to the strong dominating set of X. In other words, belonging to the strong dominating set is a transitive relation.

Claim 6.2 $Y \in SD(X,G), Z \in SD(Y,G) \Rightarrow Z \in SD(X,G)\ \forall G \subseteq QSM$

pf: It suffices to show that Z satisfies conditions (i)-(iii) in the definition of $SD(X,G)$. The proof consists of three parts.

Part 1: $V(X) \leq V(Z)\ \forall V \in G$

pf of part 1: $Y \in SD(X,G), Z \in SD(Y,G) \Rightarrow V(X) \leq V(Y) \leq V(Z)\ \forall V \in G$

Part 2: $X \subset Z$

pf of part 2: $Y \in SD(X,G), Z \in SD(Y,G) \Rightarrow X \subset Y \subset Z$

Part 3: $X \subset W \subset Z \Rightarrow V(W) \leq V(Z)\ \forall V \in G$

pf of part 3: Let $W \subseteq N$, s.t. $X \subset W \subset Z$. It follows that $X \subseteq Y \cap W$. Therefore, $Y \in SD(X,G)$ implies $V(Y) \geq V(Y \cap W)\ \forall V \in G$. By assumption, $G \subseteq$ QSM. Therefore, $V(W \cup Y) \geq V(W)\ \forall V \in G$. Finally, $Z \in SD(Y,G)$ and $W \subseteq Z$ which together imply $V(Z) \geq V(W \cup Y)\ \forall V \in G$. Therefore, $V(Z) \geq V(W)$.

Claim **??** states that if Y belongs to the strong dominating set of X in G and if X is contained in $\acute{Y}$, then either Y is contained in $\acute{Y}$ or the union of Y and $\acute{Y}$ belongs to the strong dominating set of $\acute{Y}$ in G.

Claim 6.3 $\forall G \subseteq QSM$. *If* $Y \in SD(X,G)$ *and* $X \subseteq \acute{Y}$, *then one of the following must hold:*

(i) $Y \subseteq \acute{Y}$
(ii) $Y \cup \acute{Y} \in SD(\acute{Y}, G)$

pf: Suppose that (i) does not hold. It suffices to show $Y \cup \acute{Y}$ satisfies conditions (i)-(iii) in the definition of $SD(X, G)$. The proof consists of three parts.

Part 1: $V(Y) \leq V(Y \cup \acute{Y}) \ \forall V \in G$

pf of part 1: $Y \in SD(X, QSM) \Rightarrow X \subset Y$. By assumption, $X \subseteq \acute{Y}$. Therefore, $X \subseteq Y \cap \acute{Y} \subseteq Y$ and $V(Y \cap \acute{Y}) \leq V(Y)$. $G \subseteq QSM \Rightarrow V(\acute{Y}) \leq V(Y \cup \acute{Y}) \ \forall V \in G$.

Part 2: $\acute{Y} \subseteq Y \cup \acute{Y}$

pf of part 2: follows directly.

Part 3: $\acute{Y} \subset W \subset Y \cup \acute{Y} \Rightarrow V(W) \leq V(Y \cup \acute{Y}) \ \forall V \in G$

pf of part 3: Choose $W \subseteq N$ s.t. $\acute{Y} \subset W \subset Y \cup \acute{Y}$. It follows that $X \subseteq Y \cap W \subseteq Y$. By assumption, $Y \in SD(X, G)$, which implies $V(Y) \geq V(Y \cap W) \ \forall V \in G$. Furthermore, $G \subseteq QSM$ implies $V(Y \cup W) \geq V(W) \ \forall V \in G$. Finally, $\acute{Y} \subseteq W$ and $W \subset Y \cup \acute{Y}$ which together imply $Y \cup W = Y \cup \acute{Y}$, which completes the proof.

A corollary of Claim **??** and Claim **??** is that if both Y and $\acute{Y}$ belong to $SD(X, G)$, then their union also belongs to $SD(X, G)$.

Corollary 6.2 : $\forall G \subseteq QSM$, Y, $\acute{Y} \in SD(X, G) \Rightarrow Y \cup \acute{Y} \in SD(X, G)$

pf: $Y \in SD(X, G)$ and $X \subseteq \acute{Y}$. Therefore, by Claim **??**, one of the following must hold:

(i) $Y \subseteq \acute{Y}$
(ii) $Y \cup \acute{Y} \in SD(\acute{Y}, G)$.

If (i) holds, the proof is complete. Therefore, (ii) holds. By assumption, $\acute{Y} \in SD(X, G)$ and by Claim **??**, $Y \cup \acute{Y} \in SD(X, G)$.

We refer to the union of all of the packages in $SD(X,G)$ as the *strong dominating package of* X *in* G.

The **strong dominating package** *of* X *in* G, $D(X,G) \subseteq N$, *equals the union of the packages in* $SD(X,G)$.

$$D(X,G) = \bigcup_{Y \in SD(X,G)} Y$$

Corollary **??** states that the strong dominating package of X in G belongs to the strong dominating set of X in G.

Corollary 6.3 $D(X,G) \in SD(X,G) \; \forall G \subseteq QSM$

pf: Follows from Corollary **??**.

Claim **??** states that the strong dominating set of X in G is empty if and only if the dominating package of X in G equals the empty set.

Claim 6.4 $SD(X,G) = \emptyset \Leftrightarrow D(X,G) = \emptyset \;\; \forall G \subseteq QSM$

pf: follows from the definition of $D(X,G)$ and Corollary **??**.

Corollary **??** states that if the dominating package of X in G is the empty package, and if Y is a subset of X, then X contains the dominating package of Y in G.

Corollary 6.4 $D(X,G) = \emptyset \Rightarrow \forall Y \subseteq X,\; D(Y,G) \subseteq X \;\; \forall G \subseteq QSM$

pf: If $D(Y,G) = \emptyset$, the proof is complete. Suppose $D(Y,G) \neq \emptyset$. By assumption, $Y \subseteq X$. By Corollary **??**, $D(Y,G) \in SD(Y,G)$. Therefore, by Claim **??**, one of the following must hold:

(i) $D(Y,G) \subseteq X$
(ii) $D(Y,G) \cup X \in SD(X,G)$

By Claim **??**, $D(X,G) = \emptyset$ implies that $SD(X,G) = \emptyset$. If $X \neq \emptyset$, then (i) holds. Therefore, we need only consider the case that $X = \emptyset$. However, $X = \emptyset$ implies that $X = Y$, which completes the proof.

Corollary **??** states that for any package X, the strong dominating set of $D(X,G)$ in G equals the empty set provided that G is contained in QSM.

Corollary 6.5 $SD(D(X,G),G) = D(D(X,G)) = \emptyset \ \ \forall\, G \subseteq QSM$

pf: Suppose $Y \neq \emptyset$ and $Y \in SD(D(X,G),G)$. It follows from Claim **??** that $Y \in SD(X,G)$. Therefore, $Y \subseteq D(X,G)$, a contradiction. By Claim **??**, $D(D(X,G) = \emptyset$.

6.2 ALGO–BU

We now define ALGO–BU(G) for $G \subseteq$ QSM. ALGO–BU first orders the packages and then considers the packages in that order. ALGO–BU evaluates a package only if its strong dominating set is empty. To determine whether this is the case requires solving a linear programming problem. To describe ALGO–BU, we first define the *index function*, which is the standard mapping from S into the integers.

The **index function**, $index : S \rightarrow \{0, 1, 2^n - 1\}$ *according to the following rule:*

$$index(X) = \sum_{i \in X} 2^{(i-1)}$$

Example: If $n = 3$ the ordering created by the index function is:

index({∅}) = 0 index({1}) = 1 index({2}) = 2 index({1,2}) = 3

index({3}) = 4 index({1,3}) = 5 index({2,3}) = 6 index({1,2,3}) = 7

In the first iteration ALGO–BU evaluates the first nonempty package in the ordering created by *index*, which is {1}. To determine the second package evaluated by ALGO–BU, we need to define the *set of functions in G consistent with ϑ up to j.*

The set of functions in G **consistent** *with ϑ up to j, $\Gamma^j(\vartheta, G)$ is defined as follows:*

$$\Gamma^1(\vartheta, G) = G$$

$$\Gamma^j(\vartheta, G) = \{V \in G : \Lambda(V)_i = \Lambda(\vartheta)_i \text{ for } i = 1 \text{ to } j-1 \} \quad \text{for } j > 1$$

In the second iteration, ALGO–BU considers the next package in the ordering created by index, which is {2}. The package {2} is evaluated only if $SD(\{2\}, \Gamma^2(\vartheta, G))$ is empty. If, for example, {1,2} belongs to $SD(\{2\}, \Gamma^2(\vartheta, G))$, then {2} cannot be the optimal package if G is contained in QSM. Therefore, ALGO–BU moves to the next package in the ordering created by index, which is {1,2}. The package {1,2} is evaluated only if $SD(\{1,2\}, \Gamma^2(\vartheta, G))$ is empty. This process continues until ALGO–BU considers a package X such that $SD(X, \Gamma^2(\vartheta, G))$ equals the empty set. X is the second package evaluated by ALGO–BU, which we denote by T_2.

ALGO–BU applied to ϑ *in* G: $ALGO\text{–}BU(\vartheta, G) = (\{1\}, \tau)$

Step 0:	$\Lambda(V)_i = (\emptyset, V(0))$	*for* $i = 1$ *to* 2^n
	$i = 1$	{*iteration number*}
	dex = *1*	{*index number*}
Step 1:	*Evaluate* $T_i \subseteq N$	
Step 2:	$\Lambda_i(\vartheta) = (T_i, \vartheta(T_i))$	
Step 3A:	dex = dex + *1*	
	if dex $> 2^n$ *goto Step 5*	
	if dex $\leq 2^n$ *choose* X *s.t.* $index(X) =$ dex	
Step 3B:	*If* $SD(X, \Gamma^i(\vartheta, G)) \neq \emptyset$	*goto Step 3A*
	else $\tau(\Lambda(\vartheta), G) = X$	*goto Step 4*
Step 4:	$i = i + 1$	
	$T_i = X$	
	goto Step 1	
Step 5:	$num(ALGO\text{–}BU, \vartheta, G) = i$	
	end	

By the construction of ALGO–BU, each package is considered only once. We let $t(X)$ denote the iteration in which ALGO–BU considers X.

The **iteration** *of* X *in ALGO–BU,* $t(X) = min\{j : index(T_j) \geq index(X)\}$

The following series of claims are used to show that ALGO–BU is bottom–up and optimizing for QSM. Claim **??** states that as more packages are evaluated by ALGO–BU, the set of functions in G consistent with ϑ up to j cannot increase.

Claim 6.5 $\acute{j} > j \Rightarrow \Gamma^{\acute{j}}(\vartheta, G) \subseteq \Gamma^j(\vartheta, G)\ \forall\, G \subseteq V^0\ \forall\, \vartheta \in G$

pf: Let $V \in \Gamma^{\acute{j}}(\vartheta, G)$. By construction, $V(T_i) = \vartheta(T_i)$ for $i < \acute{j}$. Given that $\acute{j} > j$, it follows that $V(T_i) = \vartheta(T_i)$ for $i < j$. Therefore, $V \in \Gamma^j(\vartheta, G)$.—

A corollary of Claim **??** is that the set $SD(X, \Gamma^j)$ does not decrease as j increases.

Corollary 6.6 $\acute{j} > j \Rightarrow SD(X, \Gamma^{\acute{j}}(\vartheta, G)) \subseteq SD(X, \Gamma^j(\vartheta, G))\ \forall\, G \subseteq V^0\ \forall\, \vartheta \in G$

pf: Follows from Claim **??**.

Claim ?? states that if the strong dominating set of X is not empty at $t(X)$, and further that if X is contained in a package which is later evaluated, then there exists a package T_j which belongs to the strong dominating set of X in $\Gamma^j(\vartheta, G)$. Note that this and future claims require that G is contained in QSM. In the proof of Claim ?? and in all subsequent proofs, $SD(X, \Gamma^j(\vartheta, \cdot))$ is abbreviated as $SD(X, j)$. Similarly $D(X, \Gamma^j(\vartheta, \cdot))$ is abbreviated as $D(X, j)$.

Claim 6.6 *Let $G \subseteq QSM$ and $\vartheta \in G$. Suppose that $X \subset T_j$ for some j and that $SD(X, \Gamma^{t(X)}(\vartheta, G)) \neq \emptyset$. It follows that $\exists\, T_{\hat{j}} \subseteq T_j$ s.t. $T_{\hat{j}} \in SD(X, \Gamma^{\hat{j}}(\vartheta, G))$*

pf: $SD(X, t(X)) \neq \emptyset \Rightarrow D(X, t(X)) \neq \emptyset$. By Corollary ??, $D(X, \hat{j}) \neq \emptyset\ \forall\, \hat{j} > t(X)$. There are two possible cases:

Case 1: $\exists\, \hat{j} \leq j$ s.t. $index(D(X, \hat{j})) \leq index(T_{\hat{j}})$

pf of case 1: By Corollary ??, $SD(D(X, \hat{j}), \hat{j}) = \emptyset$. It follows that $T_{\hat{j}} = D(X, \hat{j})$. By Corollary ??, $SD(T_j, j) = \emptyset \Rightarrow SD(T_j, \hat{j}) = \emptyset$. Therefore, by Claim ??, $T_{\hat{j}} \subseteq T_j$.

Case 2: $\forall\, \hat{j} < j$, $index(D(X, \hat{j})) > index(T_{\hat{j}})$

pf of Case 2: $index(D(X, j)) > index(T_j)$ which implies $D(X, j) \not\subseteq T_j$. However, by Claim ??, $D(X, j) \cup T_j \in SD(T_j, j)$, a contradiction.

Corollary ?? states that the intersection of any two packages evaluated by ALGO–BU was also evaluated.

Corollary 6.7 *Let $G \subseteq QSM$ and $\vartheta \in G$. Suppose $X = T_i \cap T_j$ for some i and j. It follows that $SD(X, \Gamma^{t(X)}(\vartheta, G)) = \emptyset$*

pf: Suppose $SD(X, t(X)) \neq \emptyset$. By Claim ??, $X \subseteq T_j$ implies $\exists\, T_{\hat{j}} \subseteq T_j$ s.t. $T_{\hat{j}} \in SD(X, \hat{j})$. Similarly, $\exists\, T_{\hat{i}} \subseteq T_i$ s.t. $T_{\hat{i}} \in SD(X, \hat{i})$. Without loss of generality, assume $\hat{i} < \hat{j}$. By Corollary ??, $T_{\hat{i}} \cup T_{\hat{j}} \in SD(X, \hat{i})$. By Claim ?? it follows that $T_{\hat{i}} \subseteq T_j$ and $T_{\hat{j}} \subseteq T_i$. Which together imply that $T_{\hat{i}} = T_{\hat{j}} = X$.

Corollary ?? is used in the proof that ALGO–BU is bottom–up and in subsequent proofs. It states that the best package up to m has at least as large a value as any package which it contains.

Corollary 6.8 *Let $G \subseteq QSM$ and $\vartheta \in G$. If $V \in \Gamma^m(\vartheta, G)$, then if $X \subseteq B_m$, $V(B_m) \geq V(X)$.*

pf: By construction $B_m = T_j$ for some $j < m$. If $SD(X, t(x)) = \emptyset$ then the proof is complete by the definition of B_m. Suppose $SD(X, t(X)) \neq \emptyset$. By Claim **??**, it follows that $\exists\, T_{j'}$ s.t. $T_{j'} \in SD(X, j')$ such that $T_{j'} \subseteq B_m$ and $V(B_m) \geq V(T_{j'})$ which completes the proof.

Claim **??** and Claim **??** state that if G is contained in QSM, then for any ϑ belonging to G, ALGO–BU applied to ϑ is bottom–up and optimizing.

Claim 6.7 *Let $G \subseteq QSM$ and $\vartheta \in G$. ALGO–BU applied to ϑ is bottom–up.*

pf: Suppose $B_j \not\subset T_j$. By Corollary **??**, $B_j \cap T_j = T_i$ for some $i < j$. By Corollary **??**, $V(B_j) \geq V(T_i)$. By assumption, $\Gamma^j(\vartheta, G) \subseteq$ QSM. Therefore, given that $V(B_j) \geq V(T_i) \forall V \in \Gamma^j$, it follows that $V(T_j \cup B_j) \geq V(T_j) \forall V \in \Gamma^j$. This implies $SD(T_j, j) \neq \emptyset$, a contradiction.

Claim 6.8 *Let $G \subseteq QSM$ and $\vartheta \in G$. ALGO–BU is optimizing for ϑ.*

pf: Choose $X \subseteq N$ s.t. $\vartheta(X) \geq \vartheta(Y)\ \forall Y \subseteq N$ and $\vartheta(X) > \vartheta(Y) \forall Y \subseteq N$ s.t. $X \subset Y$. It is straightforward to show that $\vartheta \in$ QSM implies that X is unique. It follows that $SD(X, j) = \emptyset$ for all j. Therefore, $\exists\, j$ s.t. $T_j = X$, which completes the proof.

Claim **??** and Claim **??** state that ALGO–BU is bottom–up efficient for POS and UNI respectively. The proofs to these claims rely on the construction of *minimal reductions* of value functions and are contained in an appendix.

Claim 6.9 *ALGO–BU is bottom–up efficient for POS.*

pf: See appendix.

Claim 6.10 *ALGO–BU is bottom–up efficient for UNI.*

pf: See appendix.

7 Discussion

Claim ?? and Claim ?? state that ALGO–BU is bottom–up and optimizing for QSM. The example below demonstrates that ALGO–BU is not bottom–up efficient for all subclasses of QSM. However, it remains an open question as to whether ALGO–BU is bottom–up efficient for the entire class QSM.[8]

Example: $N = \{1, 2, ...9\}$. $G = \{V : V \in$ QSM, and satisfies (i)-(iv) $\}$

$$
\begin{array}{ll}
(i) & V(\{1\}) < 0 \\
(ii) & V(\{2\}) > 0 \\
(iii) & V(\{1\}) = -0.x_1x_2x_3x_4 \qquad x_i \neq 2 \text{ and } x_i \neq x_j \text{ for } i \neq j \\
(iv) & \{2, x_1, x_2, x_3, x_4\} \text{ maximizes } V
\end{array}
$$

Condition (iii) implies that if $V(\{1\}) = 0.6374$, then the optimal package is {2,3,4,6,7}. It is straightforward to show that an algorithm which evaluates the package {1} first and then immediately evaluates the optimal package is bottom–up efficient. ALGO–BU does not evaluate package {1} because {1,2} belongs to $SD(\{1\}, G)$. In this example, ALGO–BU is not bottom–up efficient because the value of the package {1} identifies which package is optimal. Obviously, this example is contrived, but it is suggestive of the difficulties in constructing an algorithm which is bottom–up efficient for subclasses of QSM.

An advantage of bottom–up algorithms is the degree to which they can be parallelized. If the set of projects N is decomposed into subsets $\{N_1, N_2, ..N_k\}$ so that each project lies in at least one subset, but possibly more, i.e. the decomposition need *not* be a partition and a bottom–up algorithm is applied to each subset of projects, then the union of the optima for the subproblems necessarily is contained in, but not necessarily equal to the optimum for the larger problem. An implication of this result is that if distinct, and possibly overlapping decision making authority is assigned to subsets of projects, the union of the resulting decisions will be *contained in* the optimal package of projects for the larger problem.

[8]The difficulty of proving that ALGO–BU is not bottom–up efficient for QSM is that the decomposition basis coefficients for subsets of three or more projects can be negative for functions in QSM, which is not allowed in our constructive proof.

Appendix

A Proofs of Bottom–Up Efficiency

The appendix consists of two sections. In the first subsection, we provide a complete proof that ALGO–BU is bottom–up efficient for POS. In the second section section, we provide a sketch of the proof that ALGO–BU is bottom–up efficient for UNI.

A.1 Positive Complementarities

For any value function V which belongs to $\Gamma^j(\vartheta, POS)$, let $\beta_{V,I}$ denote its decomposition basis coefficients. Our proof that ALGO–BU is bottom–up efficient relies on the construction of two functions, the *reduction* of V in $\Gamma^j(\vartheta, POS)$ and the *minimal reduction* of V in $\Gamma^j(\vartheta, POS)$. Each is defined by its decomposition basis coefficients.

The decomposition basis coefficients of the **reduction** *of V in $\Gamma^j(\vartheta, POS)$, $r_{Vj} : S \rightarrow \Re$, are defined as follows:*

$$\begin{array}{lll} \beta_{r_{Vj},I} & = \beta_{V,I} & \text{if } \exists\, i < j \text{ s.t. } I \subseteq T_i \\ \beta_{r_{Vj},I} & = -\delta & \text{if } \mid I \mid = 1 \text{ and } \nexists\, i < j \text{ s.t } I \subseteq T_i \\ \beta_{r_{Vj},I} & = 0 & \text{if } \mid I \mid \geq 2 \text{ and } \nexists\, i < j \text{ s.t } I \subseteq T_i \end{array}$$

Claim **??** states that r_{Vj}, the reduction of V in $\Gamma^j(\vartheta, POS)$, belongs to $\Gamma^j(\vartheta, POS)$.

Claim A.1 $V \in \Gamma^j(\vartheta, POS) \Rightarrow r_{Vj} \in \Gamma^j(\vartheta, POS)\ \forall\, \vartheta \in POS$

pf: It suffices to show that $\forall\, \acute{j} < j, V(T_{\acute{j}}) = r_{Vj}(T_{\acute{j}})$. Note that $\beta_{r_{Vj},I} = \beta_{V,I}\ \forall\, I \subseteq T_{\acute{j}}$ Therefore,

$$V(T_{\acute{j}}) = \sum_{I \subseteq T_{\acute{j}}} \beta_{V,I} = \sum_{I \subseteq T_{\acute{j}}} \beta_{r_{Vj},I} = r_{Vj}(T_{\acute{j}})$$

which completes the proof.

In order to define the minimal reduction of V in $\Gamma^j(\vartheta, POS)$, we first define the *tested package dominating I* for any set of projects $I \subset N$.

The **tested package dominating** I **up to** j, $DT(I,j) = \bigcap_{\{T_i:\, i<j,\, I\subseteq T_i\}} T_i$

Claim **??** states that is the strong dominating set of X up to j is empty then the tested package dominating I up to j is contained in X for any I contained in X.

Claim A.2 $SD(X,\Gamma^j(\vartheta,POS)) = \emptyset \Rightarrow DT(I,j) \subseteq X \;\; \forall\, I \subseteq X.$

pf: By Claim **??**, if $\exists\, i \leq j$, s.t. $I \subseteq T_i$, then $\exists\, \acute{j}$ such that $T_{\acute{j}} \subseteq T_i$ and $I \subseteq T_{\acute{j}}$ and $T_{\acute{j}} \in SD(I,\acute{j})$. By Corollary **??**, $SD(I,\acute{j}) \subseteq SD(I,j)$ and by Claim **??**, $I \subseteq X$ and $SD(X,j) = \emptyset$ implies $T_{\acute{j}} \subseteq X$.

The proof that ALGO–BU is bottom–up uses the *minimal reduction* of the value function, which places positive value only on those packages which have been evaluated. We defined the minimal reduction of V in $\Gamma(\vartheta,POS)$ in terms of its decomposition basis coefficients.

The decomposition basis coefficients of the **minimal reduction** *of* V *in* $\Gamma^j(\vartheta,POS)$, m_{Vj} *are defined as follows:*

$$\begin{array}{lll}
\beta_{m_{Vj},I} = 0 & & \text{if } |\, I \,| > 1 \text{ and } \nexists\, i<j \text{ s.t. } I = T_i \\
\beta_{m_{Vj},I} = 0 & & \text{if } |\, I \,| = 1 \text{ and } \exists\, i<j \text{ s.t. } I \subset T_i \\
\beta_{m_{Vj},I} = -\theta & & \text{if } |\, I \,| = 1 \text{ and } \nexists\, i<j \text{ s.t } I \subseteq T_i \\
\beta_{m_{Vj},I} = \sum_{\{K:DT(K,j)=I\}} \beta_{V,K} & & \text{if } \exists\, i<j \text{ s.t. } I = T_i
\end{array}$$

Claim A.3 *Given* $V, \acute{V} \in \Gamma^j(\vartheta,POS)$, $m_{Vj}(X) = m_{\acute{V}j}(X) \;\; \forall\, X \subseteq N.$

pf: By inspection, $\beta_{m_{Vj},I} = \beta_{m_{\acute{V}j},I}$ for all $I \subseteq N$.

Given that the minimal reduction of V in $\Gamma^j(\vartheta,POS)$ is independent of V, hereafter, we denote it by ϑ_j. We now show that the value of the reduction on V in $\Gamma^j(\vartheta,POS)$ equals the value of the minimal reduction on V in $\Gamma^j(\vartheta,POS)$ for any package whose strong dominating set in $\Gamma^j(\vartheta,POS)$ is empty.

Claim A.4 $SD(X,\Gamma^j(\vartheta,POS)) = \emptyset \Rightarrow r_{Vj}(X) = \vartheta_j(X) \;\; \forall\, V \in \Gamma^j(\vartheta,POS)$

pf: $SD(X,j) = \emptyset$. By Claim ??, $SD(X,j) = \emptyset \Rightarrow DT(I,j) \subseteq X \ \forall\, I \subseteq X$. Let $k = \mid \{x : x \in X,\ x \notin T_i \text{ for all } i < j\} \mid$. It follows that

$$r_{Vj} = -k \cdot \theta + \sum_{I \subseteq T_i} \beta_{V,I}$$

Rearranging terms obtains

$$r_{Vj} = -k \cdot \theta + \sum_{T_i \subseteq X} \sum_{\{I : DT(I,j) = T_i\}} \beta_{V,I}$$

which reduces to

$$r_{Vj} = -k \cdot \theta + \sum_{T_i \subseteq X} \beta_{m_{Vj},I}$$

Therefore, $r_{Vj}(X) = \vartheta_j(X)$.

We can now state the following corollary

Corollary A.1 : $\forall\, W, X \subseteq N$, *satisfying (i) - (iii):*

$$\begin{array}{ll} (i) & D(X, \Gamma^j(\vartheta, POS)) = \emptyset \\ (ii) & D(W, \Gamma^j(\vartheta, POS)) = \emptyset \\ (iii) & W \subseteq X \end{array}$$

$\vartheta_j(W) \leq \vartheta_j(X) \Rightarrow V(W) \leq V(X)\ \forall\, V \in \Gamma^j(\vartheta, POS)$

pf: By Claim ??, it suffices to show that for any $V \in \Gamma^j(\vartheta, POS)$, $m_{Vj}(W) \leq m_{Vj}(X)$ implies $V(W) \leq V(X)$. Let $B(W,j) = \{I : I \subseteq W, DT(I,j) = \emptyset\}$. If $W \subseteq X$, then $B(W,j) \subseteq B(X,j)$. Further, if $I \subseteq W$ and $\beta_{V,I} \neq \beta_{r_\vartheta,I}$ then $I \in B(W)$. By construction, $\beta_{V,I} \neq \beta_{r_{Vj},I}$. Therefore, $\beta_{V,I} \geq \beta_{r_{Vj},I}$. Define δ as follows:

$$\delta = \sum_{I \in B(X,j)} (\beta_{V,I} - \beta_{r_{Vj},I}) - \sum_{I \in B(W,j)} (\beta_{V,I} - \beta_{r_{Vj},I})$$

By construction, $V(X) - V(W) = r_{Vj}(X) - r_{Vj}(W) + \delta$. and $B(W) \subseteq B(X)$ implies $\delta \geq 0$. Finally, by Claim ??, $r_{Vj}(X) - r_{Vj}(W) = m_{\vartheta j}(X) - m_{\vartheta j}(W)$, which completes the proof.

Corollary A.2 *If* $D(X,\Gamma^j) = \emptyset$ *and* $W \subseteq X$, $\vartheta_j(D(W,\Gamma^j(\vartheta,POS))) \leq \vartheta_j(X) \Rightarrow V(W) \leq V(X)\ \forall\, V \in \Gamma^j(\vartheta,POS)$

pf: By Corollary **??**, if $D(W,j) = \emptyset$ the proof is complete. If $D(W,j) \neq \emptyset$, then by Corollary **??**, $D(W,j) \subseteq X$. By Claim **??**, $V(D(W,j)) \leq V(X)\ \forall\, V \in \Gamma^j(\vartheta,POS)$. And, from the definition of $D(W,j), V(W) \leq V(D(W,j))\ \forall\, V \in \Gamma^j(\vartheta,POS)$, which completes the proof.

Claim **??** states that if the strong dominating set of X in $\Gamma^j(\vartheta,POS)$ is empty, then there cannot exist a Y which contains X with a higher value under ϑ_j.

Claim A.5 $SD(X,\Gamma^j(\vartheta,POS)) = \emptyset \Rightarrow \nexists\, Y$ *s.t.* $X \subset Y$ *and* $\vartheta_j(X) \leq \vartheta_j(Y)$

pf: The proof is by contradiction. Suppose $\exists\, Y$ s.t. $X \subset Y$ and $\vartheta_j(X) \leq \vartheta_j(Y)$. Let $LD(X,j) = \{Y : X \subset Y, \vartheta_j(X) \leq \vartheta_j(Y), D(Y,j) = \emptyset\}$. It follows that $LD(X,j) \neq \emptyset$ and that $\exists\, Y \in LD(X,j)$ s.t. $\nexists\, \acute{Y} \in LD(X,j)$ s.t. $\acute{Y} \subset Y$.

It suffices to show that if $LD(X,j) \neq \emptyset$, then Y belongs to $SD(X,j)$, a contradiction. The proof proceeds in three parts.

Part 1: $V(X) \leq V(Y)\ \forall\, V \in G$

pf of part 1: By assumption, $D(Y,j) = \emptyset$ and $\vartheta_j(Y) \geq \vartheta_j(X)$. The result follows by Corollary **??**.

Part 2: $X \subseteq Y$

pf of part 2: By assumption, $X \subset Y$.

Part 3: $X \subset W \subset Y \Rightarrow V(W) \leq V(Y)\ \forall\, V \in G$

pf of part 3: Let $W \subseteq N$ s.t. $X \subset W \subset Y$. There are two possibilities:

$$\begin{array}{ll} (i) & D(W,j) = \emptyset \\ (ii) & D(W,j) \neq \emptyset \end{array}$$

Suppose first that (i) holds. By assumption, $\vartheta_j(W) \leq \vartheta_j(Y)$, and by Claim **??**, $V(W) \leq V(Y) \forall V \in \Gamma^j(\vartheta, POS)$. Suppose instead that (ii) holds. By Corollary **??**, $D(W,j) \subseteq Y$, and by assumption, $\vartheta_j(D(W,j)) \leq \vartheta_j(Y)$. By Claim **??**, $V(D(W,j)) \leq V(Y)\ \forall V \in \Gamma^j(\vartheta, POS)$. From the definition of $D(W,j), V(W) \leq V(D(W,j)\ \forall V \in \Gamma^j(\vartheta, POS)$, which completes the proof.

Claim **??** states that X is dominated in $\Gamma^j(\vartheta, POS)$ if and only if the strong dominating set of X in $\Gamma^j(\vartheta, POS)$ is nonempty.

Claim A.6 *X dominated in* $\Gamma^j(\vartheta, POS) \Leftrightarrow SD(X, \Gamma^j(\vartheta, POS)) \neq \emptyset$

pf: ($\Leftarrow$) Follows from Claim **??**.

($\Rightarrow$) The proof is by contradiction. Suppose $SD(X,j) = \emptyset$. $D(X,j) = \emptyset$ by Claim **??**. It follows that $\forall Y$ s.t. $X \subseteq Y$, $\vartheta_j(X) > \vartheta_j(D(Y,j)) \geq \vartheta_j(Y)$. It suffices to show that $\exists V \in \Gamma^j(\vartheta, POS)$ such that $V(X) > V(Y)\ \forall Y \subseteq N$. Define the decomposition basis coefficients for V as follows:

$$\begin{array}{ll} \beta_{V,I} & = \beta_{\vartheta_j,I} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad I \neq X \\ \beta_{V,X} & = max\{\vartheta_j(I) : I \subseteq N, X \not\subseteq I\} - \vartheta(X) + 1 \end{array}$$

There are two possible cases. If $X \subseteq Y$ then it follows that

$$V(Y) - V(X) = \vartheta(Y) - \vartheta_j(X) < 0$$

Alternatively, if $X \not\subseteq Y$, then it follows that

$$V(Y) - V(X) = \vartheta_j(Y) - \vartheta_j(X) - \beta_{V,X} < 0$$

, which completes the proof.

Corollary **??** states that X is dominated if and only if its strong dominating package is not empty.

Corollary A.3 *X dominated in* $\Gamma^j(\vartheta, POS) \Leftrightarrow D(X, \Gamma^j(\vartheta, POS)) \neq \emptyset$

pf: Follows from Claim **??** and Claim **??**.

Claim A.7 *If A bottom–up and optimizing for POS then $T_{n^*} = N$, where $n^* = num(A, \vartheta, POS)$*

pf: The proof proceeds in two parts:

Part 1: $\forall \vartheta \in POS$ and $\forall j < num(A, \vartheta, POS)$, $\exists V \in \Gamma^j(\vartheta, POS)$ s.t. N optimizes V

pf of part 1: Let X optimize ϑ. Define the decomposition basis coefficients of V as follows:

$$\begin{aligned} \beta_{V,I} &= \beta_{\vartheta_j,I} && \text{if } I \neq N \\ \beta_{V,X} &= \beta_{\vartheta,N} + \vartheta(X) - \vartheta(N) + 1 \end{aligned}$$

It follows that $V \in \Gamma^j(\vartheta, POS)$ and N optimizes V.

Part 2: If $T_i = N$, $\forall X \subset N$, s.t. $X \neq T_j$ for $j < i, X$ is dominated.

pf of part 2: Suppose not. Let X maximize ϑ. Let B_i be the best package located up to iteration i. Define the decomposition basis coefficients of $V \in \Gamma^i(\vartheta, POS)$ as follows:

$$\begin{aligned} \beta_{V,I} &= \beta_{\vartheta_j,I} && \text{if } I \neq N \\ \beta_{V,N} &= \beta_{\vartheta,N} + (\vartheta(B_i) - \vartheta(N)) + 0.5(\vartheta(X) - \vartheta(B_i)) \end{aligned}$$

It follows that $B_{i+1} = N$. However, X optimizes ϑ, so there exists a $j > i+1$ s.t. $X = T_j$, but this contradicts the assumption that A is bottom–up.

Claim A.8 *ALGO–BU is bottom–up efficient for POS.*

pf. Claims **??** and Claim **??** state that ALGO–BU is optimizing and bottom–up for $G \subseteq$ QSM. Therefore, it suffices to show that for any A which is bottom–up and optimizing for POS, $\exists V^* \in POS$ s.t. $num(\text{ALGO–BU}, V^*) < num(A, V^*)$. Let Z_i denote the ith package tested by A. $A \not\equiv$ ALGO–BU implies $\exists V \in POS$ s.t. $T_i \neq Z_i$ for some i. Let $j = min\{i : T_i \neq Z_i\}$ For $i < j$, $T_i = Z_i$. Let $\Gamma^i_A(V,\text{POS})$ = the class of functions in POS consistent with V up to iteration i of the algorithm A. It follows that $\Gamma^j_A(\vartheta,\text{POS}) = \Gamma^j(\vartheta,\text{POS})$. Let B^A_i denotes the best package up to iteration i of the algorithm A. It follows that $B^A_j = B_j$.

The proof proceeds in two parts: first, we create the function V^*; second, we show that $num(\text{ALGO-BU}, V^*) < num(A, V^*)$.

Let ϑ_j denote the minimal reduction of V in $\Gamma^j(V,\text{POS})$. By Claim ??, $\vartheta_j \in \Gamma^j(V,\text{POS})$. Define the decomposition basis coefficients for $V_0 \in \Gamma^j(V,\text{POS})$ as follows:

$$\begin{array}{ll} \beta_{V_0,I} & = \beta_{\vartheta_j,I} \qquad \text{for } I \neq T_j \\ \beta_{V_0,T_j} & = \vartheta_j(B_j) - \vartheta_j(T_j) + 1 \end{array}$$

By construction, $B_{j+1} = T_j$. More importantly, T_j optimizes V_0. We now recursively define V_{i+1} given V_i. Let T_{j+i} be the $(j+i)$st package evaluated given V_i. Define the decomposition basis coefficients of V_{i+1} as follows:

$$\begin{array}{ll} \beta_{V_{i+1},I} & = \beta_{V_i,I} \qquad \text{for } I \neq T_{j+i} \\ \beta_{V_{i+1},T_{j+i}} & = V_i(B_{j+i}) - V_i(T_{j+i}) + 1 \end{array}$$

It follows from Claim ?? that T_{j+i} optimizes V_i. Moreover, given that $B_{j+i} \subset B_{j+i+1}$, it follows that there exists an i^* such that $T_{j+i*} = N$. Let $V^* = V_{i*}$. It is straightforward to show that for all $i < i*$, $\Gamma^k(V_i,\text{POS}) = \Gamma^k(V^*,\text{POS})$ for $k \leq i$. Moreover, for $0 \leq i \leq i^*$, $T_{j+i} = B_{j+i+1}$, and $num(\text{ALGO-BU}, V^*) = j + i^*$.

It suffices to show $(j + i^*) < num(A, V^*)$. By above $V^* \in \Gamma^j(V,\text{POS}) = \Gamma^j_A(V,\text{POS})$. Given $\acute{V}$, let $Z_k(\acute{V})$ denote the kth package evaluated by A given the function $\acute{V}$, and let Z_k denote the kth package evaluated by A given V^*. We first show that there exists an $a(0) > 0$ such that $Z_{j+a(0)} = T_j$. Choose $b \geq 0$ such that $T_j \subseteq Z_{j+b}$ and $T_j \not\subseteq Z_{j+c}$ for $c < b$. We know that b exists by Claim ??. By construction, $V_0(Z_{j+c}) = V^*(Z_{j+c})$ for $c < b$.

Suppose that $Z_{j+b} \neq T_j$. Define the decomposition basis coefficients for $\acute{V}_0$ as follows:

$$\begin{array}{ll} \beta_{\acute{V}_0,I} & = \beta_{V_0,I} \qquad \text{for } I \neq Z_{j+b} \\ \beta_{\acute{V}_0,Z_{j+b}} & = V_0(T_j) - V_0(Z_{j+b}) - \frac{1}{2} \end{array}$$

By assumption, $Z_{j+b} \not\subset Z_{j+c}$ for $0 \leq c < b$. Therefore, $Z_i(\acute{V}_0) = Z_i$ for $i \leq j + b$. By construction $B^A_{j+b+1} = Z_{j+b}$, however, $T_j \subset Z_{j+b}$ optimizes $\acute{V}_0$, which contradicts, A bottom-up and optimizing. Therefore, $Z_{j+b} = T_j$. Let $a(0) = b$. By assumption, $Z_j \neq T_j$, which implies $a(0) > 0$.

Given $i \in \{1,..,i^*\}$, we now prove by induction that there exists an $a(i) > i$ such that $Z_{j+a(i)} = T_{j+i}$. We suppose it holds for $(i-1)$, and show that it must hold for i. By the inductive assumption, there exists an $a(i-1) > (i-1)$ such that $Z_{j+a(i-1)} = T_{j+i-1}$ and $Z_{j+a(i-1)} \not\subseteq Z_k$ for any $k < j + a(i-1)$. Choose $b \geq 0$ such that $T_{j+i} \subseteq Z_{j+b}$ and $T_{j+i} \not\subseteq Z_{j+c}$ for $c < b$. Again, we know that b exists by Claim ??. By construction, $V_i(Z_{j+c}) = V^*(Z_{j+c})$ for $c < b$. Suppose that $Z_{j+b} \neq T_{j+i}$.

Define the decomposition basis coefficients for $\acute{V}_i$ as follows:

$$\begin{array}{ll} \beta_{\acute{V}_i,I} & = \beta_{V_i,I} \qquad\qquad \text{for } I \neq Z_{j+b} \\ \beta_{\acute{V}_i,Z_{j+b}} & = V_i(T_{j+i}) - V_i(Z_{j+b}) - \frac{1}{2} \end{array}$$

By assumption, $Z_{j+b} \not\subset Z_{j+c}$ for $0 \leq c < b$. Therefore, $Z_i(\acute{V}_i) = Z_i$ for $i \leq j + b$. By construction $B^A_{j+b+1} = Z_{j+b}$, however, $T_{j+i} \subset Z_{j+b}$ optimizes $\acute{V}_0$, which contradicts, A bottom–up and optimizing. Therefore, $Z_{j+b} = T_{j+i}$. Let $a(i) = b$. By the inductive assumption, $Z_{j+a(i-1)} \not\subseteq Z_{j+k}$ for $0 \leq k < a(i-1)$. Therefore, $a(i) \geq [a(i-1)+1] > (i+1)$, which completes the proof that $(j + i^*) < num(A, V^*)$.

A.2 Unidirectional Complementarities

For any value function V which belongs to $\Gamma^j(\vartheta, UNI)$, let $\beta_{V,I}$ denote its decomposition basis coefficients. The proof that ALGO–BU is bottom–up efficient for UNI is similar to the proof for POS. The only difference is in the construction of the *minimal reduction* of V in $\Gamma^j(\vartheta, UNI)$.

The *minimal reduction* of ϑ in $\Gamma(\vartheta, UNI)$ is defined as follows:

The decomposition basis coefficients of the **minimal reduction** *of V in $\Gamma^j(\vartheta, UNI)$, m_{Vj} are defined as follows:*

$$\begin{array}{lll} \beta_{m_{Vj},I} & = -\theta & \text{if } \mid I \mid = 1 \text{ and } i < j \text{ s.t } I \subseteq T_i \\ \beta_{m_{Vj},I} & = -\beta_{V,I} & \text{if } \mid I \mid = 1 \text{ and } \exists\, i < j \text{ s.t } I \subseteq T_i \text{ and } 1 \notin T_i \\ \beta_{m_{Vj},I} & = \sum_{\{K:DT(K,j)=I\}} \beta_{V,K} & \text{if } \mid I \mid > 1 \text{ and } \exists\, i \leq j \text{ s.t. } I = T_i \text{ and } 1 \in I \\ \beta_{m_{Vj},I} & = 0 & \text{else} \end{array}$$

The analogues of Claim ?? through Claim ?? follow with UNI substituted for POS. The construction of the V_1 in the proof of bottom–up efficiency differs slightly, so the beginning of the proof is presented below:

Claim A.9 *ALGO-BU is bottom-up efficient for UNI.*

pf. Claims ?? and Claim ?? state that ALGO-BU is optimizing and bottom-up for $G \subseteq$ QSM. Therefore, it suffices to show that for any A which is bottom-up and optimizing for UNI, $\exists V^* \in$ UNI s.t. $num(\text{ALGO-BU}, V^*) < num(A, V^*)$. Let Z_i denote the ith package tested by A and $\Gamma_A^j(\vartheta,\text{UNI})$ be the set of functions which are consistent with A up to j. $\acute{A} \not\equiv$ ALGO-BU implies $\exists\, V \in$ UNI s.t. $T_i \neq Z_i$ for some i. Let $j = min\{i : T_i \neq Z_i\}$ If $1 \in T_j$, then the proof is identical to the proof for POS. Therefore, we must only consider the case where, $1 \notin$ POS. By the definition of ALGO-BU, if $1 \notin T_j$, then there exists an $i \in T_j$ such that $i \notin T_{\acute{j}}\ \forall \acute{j} < j$. Let $M^* = max_{\{X \subset N\}}\{\mid \vartheta_j(B_j) - \vartheta_j(X) \mid\}$ Define the decomposition basis coefficients for $V_0 \in \Gamma^j(\vartheta,\text{UNI})$ as follows:

$$\begin{aligned} &\beta_{V_0,I} = \beta_{\vartheta_j,I} && \text{for } I \neq T_j \\ &\beta_{V_0,i} = \vartheta(B_j) - \vartheta(T_j) + M^* + 1 \end{aligned}$$

By the construction of ALGO-BU, $1 \in T_{j+1}$, so define V_i for $i \geq 1$ as in the proof for $V \in$ POS. As before, there exists an i^*, such that $T_{j+i^*} = N$. Let $V^* = V_{i^*}$. It suffices to show that there exists a $a(0) > 0$ such that $Z_{j+a(0)} = T_j$, thereafter the proof is identical to the proof for POS. Choose $b \geq 0$ such that $T_j \subseteq Z_{j+b}$ and $T_j \not\subset Z_{j+c}$ for $c < b$. We know that b exists by Claim ??. By construction, $V_0(Z_{j+c}) = V^*(Z_{j+c})$ for $c < b$. Suppose that $Z_{j+b} \neq T_j$. If $1 \notin Z_{j+b}$ then let $\acute{V}_0 = V_0$. If $1 \in Z_{j+b}$ then define the decomposition basis coefficients for $\acute{V}_0$ as follows:

$$\begin{aligned} &\beta_{\acute{V}_0,I} &&= \beta_{V_0,Ivi} && \text{for } I \neq Z_{j+b} \\ &\beta_{\acute{V}_0,Z_{j+b}} &&= V_0(T_j) - V_0(Z_{j+b}) - \tfrac{1}{2} \end{aligned}$$

By assumption, $Z_{j+b} \not\subseteq Z_{j+c}$ for $0 \leq c < b$. Therefore, in either case, $Z_i(\acute{V}_0) = Z_i$ for $i \leq j + b$. By construction $B_{j+b+1}^A = Z_{j+b}$, however, $T_j \subset Z_{j+b}$ optimizes $\acute{V}_0$, which contradicts, A bottom-up and optimizing. Therefore, $Z_{j+b} = T_j$. Let $a(0) = b$. By assumption, $Z_j \neq T_j$, which implies $a(0) > 0$. Hereafter, the proof is identical to the proof for POS. —

References

Bergson, A. (1976) "Social Choice and Welfare Economics Under Representative Government" *Journal of Public Economics* Vol 6: 171-190.

Besley, T. and I. Jewitt (1991) "Decentralized Public Good Supply," *Econometrica*, 59(6):1769-1778.

Boeri, T. (1990) *Beyond the Rule of Thumb: Methods for Evaluating Public Investment Projects*, Westview, Boulder.

Chipman, J. and J. More (1978) "The New Welfare Economics 1939-1974," *International Economic Review* 19:3-34.

Clarke, E. (1976) "Multipart Pricing of Public Goods," *Public Choice* 11:17-33.

Cornes, R. and T. Sandler (1986) *The Theory of Externalities, Public Goods, and Club Goods*, Cambridge University Press, Cambridge.

Dreze, J. and N Stern (1987) "The Theory of Cost-Benefit Analysis," in *The Handbook of Public Economics*, edited by A. Auerbach and M. Feldstein, North Holland, New York.

Fisher, M., G. Nemhauser, and L. Wolsey (1978) "An Analysis of Approximations for Maximizing Submodular Set Functions I," *Mathematical Programming* 14: 265-294.

Ford, L. and D. Fulkerson (1962) *Flows in Networks*, Princeton University Press, Princeton.

Green, J. and J.J. Laffont (1977) "Characterization of Satisfactory Mechanisms for the Revelation of Preferences for Public Goods", *Econometrica* 45:427-438.

Groves, T. and J. Ledyard (1977) "Optimal Allocation of Public Goods: A Solution to the Free Rider Problem," *Econometrica* 45:783-809.

Harris, R. (1978) "On the Choice of Large Projects," *Canadian Journal of Economics* 11:404-423.

Inman, R. (1987) "Markets, Governments, and the "New" Political Economy," in *The Handbook of Public Economics* edited by A. Auerbach and M. Feldstein, North Holland, New York.

Jackson, M. and H. Moulin (1992) "Implementing a Public Project and Distributing its Costs," *Journal of Economic Theory* 57:1: 125-141.

Liepins, G. and M. Vose (1991) "Polynomials, Basis Sets, and Deceptiveness in Genetic Search," *Complex Systems* 5:45-61.

Oakland, W. (1987) "Theory of Public Goods," in *The Handbook of Public Economics* edited by A. Auerbach and M. Feldstein, North Holland, New York.

Page, S. 1993 "Covers" *Social Science Working Paper # 872*, California Institute of Technology, Pasadena CA.

Reiter, S. and G. Sherman (1962) "Allocating Indivisible Resources Affording External Economies or Diseconomies," *International Economic Review* 3(1):108-135.

Topkis, D. (1978) "Minimizing a Submodular Function Over a Lattice," *Operations Research* 26:305-321.