The RAND Corporation is a nonprofit research organization providing objective analysis and effective solutions that address the challenges facing the public and private sectors around the world.
This product is part of the RAND Corporation technical report series. Reports may include research findings on a specific topic that is limited in scope; present discussions of the methodology employed in research; provide literature reviews, survey instruments, modeling exercises, guidelines for practitioners and research professionals, and supporting documentation; or deliver preliminary findings. All RAND reports undergo rigorous peer review to ensure that they meet high standards for research quality and objectivity.
Finding Candidate Options for Investment

From Building Blocks to Composite Options and Preliminary Screening

Paul K. Davis, Russell D. Shaver, Gaga Gvineria, Justin Beck

Prepared for the Office of the Secretary of Defense

Approved for public release; distribution unlimited
The research described in this report was prepared for the Office of the Secretary of Defense (OSD) and draws also on research for the Missile Defense Agency (MDA). The research was accomplished in the Acquisition and Technology Policy Center (ATPC) of RAND’s National Defense Research Institute (NDRI), a federally funded research and development center sponsored by the OSD, the Joint Staff, the Unified Combatant Commands, the Department of the Navy, the Marine Corps, the defense agencies, and the defense Intelligence Community under Contract W74V8H-06-C-0002.
This report describes a methodology and prototype tool, the Building Blocks to Composite Options Tool (BCOT), for identifying good candidate options to use in investment analysis. Much of the report is a high-level overview, but parts (particularly the appendices) deal also with mathematics and programming issues. The report is intended primarily as documentation for users of BCOT and those who will extend its functionality in the future—that is, working analysts and modelers. Other interested parties, however, may wish to read the summary and the first two chapters for an overview. The report supplements a broader monograph on analytical methods for capability-area assessments (Davis, Shaver, and Beck, forthcoming), intended for senior officials and analysts in the Office of the Secretary of Defense (OSD), the Joint Staff, and the military services.

Most of the work described here was accomplished in 2006 for the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics (OUD(AT&L)); the report draws also on earlier RAND research for the Missile Defense Agency (MDA). Comments are welcome and should be addressed to the senior author in RAND’s Santa Monica, Calif., office (email: pdavis@rand.org; telephone: 310-451-6912).

The research was performed in the Acquisition and Technology Policy Center (ATPC) of the RAND National Defense Research Institute (NDRI), a federally funded research and development center sponsored by the Office of the Secretary of Defense, the Joint Staff, the Unified Combatant Commands, the Department of the Navy, the Marine Corps, the defense agencies, and the defense Intelligence Community. For more information on the Center, contact its Director, Philip Antón (email: atpc-director@rand.org; telephone: (310-393-0411, ext. 7798).
CONTENTS

Preface ........................................................................................................................ .................... iii
Figures ........................................................................................................................ ................... vii
Tables ..............................................................................................................................................ix
Summary ........................................................................................................................ .................xi
Acronyms, Terms, and Descriptions ..............................................................................................xv
Acknowledgments .........................................................................................................................xvi
1.  Introduction .................................................................................................................................1
2.  BCOT’S Structure and Flow.................................................................................................. ......4
   Getting Started with BCOT ................................................................................................. 4
   High-Level Structure ........................................................................................................... 5
   From Building Blocks to Composite Investment Options..............................................5
   Costs of the Investment Options.....................................................................................6
   The Knotty Problem of Shared Costs.............................................................................7
   Effectiveness .................................................................................................................. 8
   Finding the Best Candidate Options..................................................................................10
   Initial Sorting and Filtering ..........................................................................................10
   Finding Options On or Near the Efficient Frontier .........................................................11
3.  The Centralized Interface: Inputs and Outputs..........................................................................17
4.  A Notional Example......................................................................................................... .........21
   Building Blocks and Composite Options ...........................................................................21
   Force Employment by Scenario Class...............................................................................22
   Estimating Effectiveness ...................................................................................................23
   Quasi-Linear Approximation........................................................................................23
   The “Standard” Calculation of Effectiveness...............................................................25
   Effectiveness vs. Cost Curves ......................................................................................26
   Identifying Points On or Near the Efficient Frontier.........................................................27
   Results by Focus............................................................................................................... .28
   Combining Options for Different Screening Focuses .......................................................32
5.  Conclusions and Next Steps ......................................................................................................35
   Recapitulation................................................................................................................. ...35
   Next Steps ..................................................................................................................... ....38
Appendix
   A.  Effectiveness Calculations.................................................................................................41
      The Quasi-Linear Approximation .................................................................................41
      The Standard Calculation and the Benefits of Decomposition .....................................43
   B.  Subtleties in the Concept of Nearness to the Efficient Frontier............................................45
      Identifying Points On or Near the Efficient Frontier .......................................................45
      Anomalies and How to Deal with Them ........................................................................46
      Anomalies.....................................................................................................................46
FIGURES

S.1 Summary of BCOT’s Logical Flow ................................................................. xii
S.2 Simplified Depiction ....................................................................................... xiv
2.1 BCOT’s Faceplate ......................................................................................... 4
2.2 Top-Level Structure ....................................................................................... 5
2.3 Computing Effectiveness ............................................................................... 9
2.4 Cost-Sorting, Filtering, and Selecting Options ............................................. 11
2.5 Points On, Near, or Away from the Efficient Frontier ............................... 13
3.1 Illustrative Inputs and Outputs of BCOT ...................................................... 18
4.1 Individual Composite Options: Costs vs. Effectiveness ............................... 26
4.2 Individual Options and Dominant Points .................................................. 27
5.1 Simplified Schematic Overview of BCOT Process ..................................... 36
5.2 Summary of BCOT’s Logical Flow .............................................................. 37
B.1 Points On or Near the Efficient Frontier, with Anomalies ....................... 45
C.1 A Schematic Representation of the Genetic Algorithm ............................. 52
TABLES

2.1 Hypothetical Results of Exploratory Analysis with BCOT ..........................................16
3.1 Illustrative Inputs ........................................................................................................19
3.2 Illustrative Outputs......................................................................................................20
4.1 Composite Options for a Simple Notional Case .........................................................22
4.2 Force-Employment Modes for Illustrative Building-Block Options ............................23
4.3 Incremental Effectiveness of Building Blocks for the Mobile Missiles Scenario Class (Quasi-Linear Approximation Only) ........................................................................23
4.4 Sum of Incremental Effectivenesses ...........................................................................24
4.5 Inputs for Standard Calculation of Effectiveness .........................................................25
4.6 Attributes of Options On or Near the Efficient Frontier for the Mobile-Missiles Scenario ..................................................................................................................30
4.7 Attributes of Options On or Near the Efficient Frontier for the Terrorists Scenario ....................................................................................................................30
4.8 Attributes of Options On or Near the Efficient Frontier for the WMD-Facilities Scenario .................................................................................................................31
4.9 Attributes of Options On or Near the Efficient Frontier for the Average of Terrorist and WMD-Facilities Scenario ........................................................................31
4.10 Attributes of Options On or Near the Efficient Frontier for the Average over All Scenarios ...............................................................................................................32
4.11 Union of Options Surviving Screening for at Least One Focus and Set of Parameter Values .............................................................................................................33
A.1 Types of Nonlinear Correction Factor ........................................................................42
A.2 Reducing Inputs by Decomposition ............................................................................44
C.1 Effectiveness and Costs Assumed in Example .............................................................51
C.2 An Initial "Gene Pool": Four of the Possible Composite Options ................................51
G.1 Important Operations in Analytica ............................................................................58
G.2 BCOT-Specific Array Functions ...............................................................................59
G.3 Definition and Functions of UnionNonUnique(A) .....................................................61
G.4 Definition and Functions of UniqueNonUnique(A) ..................................................63
G.5 Definition and Functions of PositiveSubset(A,I) ......................................................64
G.6 Definition and Functions of StringVector(N,I) ......................................................65
G.7 Definition and Functions of String_cats(N,I) ...........................................................66
SUMMARY

This report describes and documents a methodology and a prototype tool, the Building Blocks to Composite Options Tool (BCOT), for identifying investment options suitable for a particular capability area. The methodology assures that a broad range of investment options is considered initially. It then uses a screening technique to narrow the range of options to those deemed worthy of more-extensive assessment in a fuller portfolio-analysis framework, which can be done using RAND’s Portfolio-Analysis Tool (PAT). The methodology draws upon some classic techniques from economics and operations research but extends them significantly and suggests pragmatic approximations in applications, particularly in capabilities-based planning. We document the prototype methodology using an implementation in Analytica®, although we have a version built in Microsoft Excel® as well. We use both versions because each has specific advantages and disadvantages.

BCOT’s basic functioning is summarized in Figure S.1. The steps in the flow are as follows:

1. Identify investment building blocks (e.g., a particular new aircraft or a new weapon). Many of these will be available from pre-existing proposals, but a more comprehensive set can be constructed for a given capability area by defining the mission, working through alternative ways of accomplishing the mission, noting the component capabilities and related systems that would be necessary, and highlighting those that do not presently exist and would therefore have to be developed.

2. Construct all possible composite investment options, i.e., all combinations of the building blocks.

3. Evaluate the composite options by cost and effectiveness and as a function of test scenarios, base-force effectiveness, and assumption sets (sets of values for the parameters used in defining scenarios, performing calculations, and characterizing costs borne by the capability area).

4. Find the set of options that are economically efficient (on or near the efficient frontier, also called the Pareto-optimal frontier) for each of many effectiveness functions, which differ in the relative weight given to scenarios (screening focus) and the assumptions set used for parameter values.

5. Construct the combined set of options that are near the efficient frontier in at least one case of interest (i.e., a particular focus or choice of parameter values), as well as their effectiveness for other choices of focus or assumption sets.
6. Review the results manually, discarding further options, while perhaps adding some options back from the discard pile.

Figure S.1
Summary of BCOT’s Logical Flow

The first three steps are straightforward. Step 4 extends the efficient-frontier methods of economic theory and operations research; it retains not only options that are on the efficient frontier, but also some that are near enough to that frontier so that we hesitate to delete them in an approximate screening procedure. The criterion for retention is how close an option is to the efficient frontier and whether it is redundant with a less costly option, i.e., has the same effective building blocks but also some additional blocks that add costs but not value. BCOT automatically deletes most such redundant options.
Step 5 is also unusual and is a manifestation of our confronting uncertainty seriously. The options that seem economically efficient depend sensitively on numerous assumptions, such as the relative importance ascribed to the different scenario test cases (focus), assumptions affecting the effectiveness calculation, and so on. Rather than making an allegedly “best estimate” of all these matters and then using the options that appear most efficient for that best estimate, we combine the options that are efficient with different choices of focus and assumptions sets. In an analysis based on three test scenarios, this might mean keeping options that are efficient for each one of the scenarios—even if not for an average of the three. Similarly, if results are sensitive to some parameters, such as assumed warning time or the quality of adversary forces, we define different cases corresponding to different combinations of parameter values. Again, it may be that an option appears economically efficient for one case but not for another. We retain an option that is attractive for any of the cases of interest. Although this may seem only logical and straightforward, it is unusual; accomplishing it in a program also proved to be nontrivial.

Step 6, manual review and adjustment, is essential because BCOT is ultimately a mathematical tool that cannot incorporate all of the information known to the analyst. For example, the analyst may recognize that an option that survived screening did so only because the effectiveness calculation ignored a fatal flaw for real-world operations. Conversely, an option may need to be restored because it has strong proponents and would provide extra virtues not included in the BCOT effectiveness calculation. This extra step of bringing to bear human expertise should not be seen as compromising methodology, but rather as something desirable. BCOT will have done its job well if it broadens the scope of considerations beyond what would otherwise have applied and then narrows it to a sufficiently small number of candidate options so that manual review and adjustment is practical. In our prototype work, BCOT generated thousands of options, narrowed them to a set of perhaps three to 100, depending on assumptions, and provided displays that we could use to review and adjust our calculations in a matter of hours or days, rather than weeks or months.

Figure S.2 is a less technical depiction of the same flow, making the point that we may begin with perhaps ten building blocks, construct thousands of possible combinations, evaluate, screen, and end up with perhaps five to 20 composite options meriting further study.
Figure S.2
Simplified Depiction

- Building blocks
- Composite options
- Options with costs and scores
- Screened options

Parameterized test scenarios
Simple modeling of cost and effectiveness
Computer-assisted screening

~10
~10^3 to 10^4
~10^3 to 10^4
~5 to 20
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;L</td>
<td>Acquisition, Technology, and Logistics</td>
<td></td>
</tr>
<tr>
<td>BCOT</td>
<td>Building Block to Composite Options Tool</td>
<td>A computerized tool for generating a wide range of options and then selecting those worthy of more extensive analysis</td>
</tr>
<tr>
<td>DPs</td>
<td>dominant points</td>
<td>Points on the efficient frontier</td>
</tr>
<tr>
<td>EA</td>
<td>exploratory analysis</td>
<td>Analysis that systematically studies how results vary as the inputs are varied simultaneously over their full range of values</td>
</tr>
<tr>
<td>EF</td>
<td>efficient frontier</td>
<td>A line connecting the Pareto-optimal points in an x-y plot of a problem with objectives x and y</td>
</tr>
<tr>
<td>MDA</td>
<td>Missile Defense Agency</td>
<td></td>
</tr>
<tr>
<td>MRM</td>
<td>multiresolution modeling</td>
<td>Modeling that gives the user options about the level of detail with which inputs are specified</td>
</tr>
<tr>
<td>PAT</td>
<td>Portfolio-Analysis Tool</td>
<td>An Excel-based tool (Dreyer and Davis, 2005)</td>
</tr>
<tr>
<td>SOF</td>
<td>Special Operations Forces</td>
<td></td>
</tr>
<tr>
<td>WMD</td>
<td>weapons of mass destruction</td>
<td>Nuclear, chemical, or biological weapons</td>
</tr>
<tr>
<td></td>
<td>assumption set</td>
<td>A set of parameter values used to define scenario test cases and effectiveness calculations</td>
</tr>
<tr>
<td></td>
<td>focus</td>
<td>A “perspective” represented by a set of weights used to construct an effectiveness function from a linear weighted sum over different scenario test cases</td>
</tr>
<tr>
<td></td>
<td>screening focus</td>
<td>The focus used in BCOT to find a set of candidate composite options</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

We would like to acknowledge thorough and useful reviews by RAND consultant Robert Moore and RAND colleague Carl Rhodes. Their suggestions have materially improved the clarity of the report.
CHAPTER ONE
INTRODUCTION

The Department of Defense has considerable interest in examining investment programs by capability area. A common problem in doing so for a given capability area is that many of the options that arise for consideration come from different people and organizations and were developed based on the organizations’ past efforts, knowledge, and interests. The possible options thus reflect diverse assumptions about what capabilities are needed. Only sometimes have the individuals involved thought much about opportunities for synergy, either across Services or across capability areas, except where doing so is natural for their particular interests (e.g., an airplane builder seeing multiple missions). Further, only sometimes do those individuals offer up variants that cost and deliver more or less than what they recommend. As a result, decisionmakers who must allocate limited resources often lack information they need for performing tradeoff analyses, devising combined strategies that exploit synergies and hedge against risks, and making program adjustments wisely (e.g., increasing or decreasing allotments to various programs, relative to what is requested). Thus, there is need for a more comprehensive and systematic approach to option-generation, not merely the evaluation of options being proposed in the usual manner.

This report describes a methodology and a related tool, the Building Blocks to Composite Options Tool (BCOT), for developing candidate options to be given serious consideration. It bears on how to conceive and construct options that might not otherwise be considered and on how to screen huge numbers of possible options so as to narrow down the set of candidates. The report draws on some classic methods of portfolio economics and operations research but extends them significantly with original work and application to defense planning. It illustrates the method with a notional application.

Shaver drew on classic methods to develop a first version of the methodology using Microsoft Excel. Excel has many virtues, including its ubiquity and versatility, built-in graphics capability, and menu-driven operations, such as sorting. It allows arrays to be manipulated easily and sophisticated charts to be constructed readily. Throughout most of our effort, we relied

---
1 The Joint Staff has developed a taxonomy with more than 20 Tier One Joint Capability Areas (JCAs), which decompose into lower-level component capability areas. See www.dtic.mil/futurejointwarfare/cap_areas.htm (as of June 24, 2007). Our methodology can be used with the JCAs, components, or other convenient groupings. In a companion monograph, we use Global Strike and Ballistic Missile Defense as examples of capability areas (Davis, Shaver, and Beek, forthcoming).
primarily upon the Excel version. It is the instrument of choice for some of our work. Most of the
development of BCOT was accomplished in 2006.

Davis generalized the theory and, recognizing some limitations of the Excel version,
designed and built a version of BCOT in the Analytica® modeling system, which has advantages
for some aspects of clarity, extensibility, collaboration, and exploratory analysis. Gvineria then
improved and extended the Analytica model substantially, implementing important but difficult-to-achieve capabilities that greatly extended the capacity for multiparameter exploratory analysis.
A review of the methodology by Beck identified a number of residual problems, including
fundamental difficulties. Subsequently, as the result of a concrete illustrative application (to the
Global Strike problem) and many collaborative sessions, we improved the methodology and both
the Excel and Analytica tools considerably. The result that we describe here is the Analytica-based
version of BCOT, but we continue to use both versions, referring to BCOT and BCOT-Excel
to distinguish between them.

Chapter Two describes BCOT’s higher-level structure and flow, primarily with visual
representations. Chapter Three describes BCOT’s graphical user interface—i.e., the centralized
access point for inputs and outputs. Chapter Four illustrates cryptically a highly simplified
application to the Global Strike capability area. Chapter Five summarizes conclusions and
identifies possible next steps for development of both the methodology and BCOT. Appendices
A through H provide more detail on mathematical issues, including our use of Analytica’s
powerful array-based methods to enable exploratory analysis, and also various programming
subtleties and practical issues for users.

Appendices A and B discuss subtleties of BCOT’s mathematics. Appendix C describes a
genetic-algorithm alternative to BCOT which our colleague Paul Dreyer developed in parallel to
enable us to deal with cases in which huge numbers of BCOT composite options might
overwhelm a personal computer. This method was implemented in Visual Basic. Appendices D,
E, and F provide users with some guidance about how to make common changes in BCOT for
particular applications. Appendix G discusses BCOT’s array mathematics and its
implementation, using built-in and customized Analytica operators. Appendix H advises users on
how to produce graphics by using Excel in combination with Analytica.

BCOT is not only a prototype, it is also a living tool that will be adapted with each
application. With this continuing evolution in mind, we have sought to make BCOT self-
documenting, since built-in documentation can be kept current. This report provides an
overview, which should remain valid, and a discussion of various technical issues that will
probably remain relevant even though details of BCOT itself evolve. The user should begin by
reading this report, but should then rely upon documentation in BCOT itself for up-to-date accuracy.
GETTING STARTED WITH BCOT

When BCOT is opened, a “faceplate” appears on the screen, as shown in Figure 2.1. The Overview node\(^2\) contains a verbal description of the overall tool. The Changes and Notes node (bottom) serves as a simple text-based journal of entries users wish to make. For a given application, it should be used to record changes in BCOT or default assumptions, to note issues for subsequent revisions to address, or to make other comments that might help in maintenance or collaborative analysis. Such commenting is valuable in practice, because it assists in keeping track of model versions and their distinctions.

The Interface module is a centralized collection of inputs and standard outputs. An analyst using BCOT may operate entirely within the interface node, merely changing input assumptions and looking at various displays of results. The Model module contains the model itself, which we

---

\(^2\) Nodes, shown in yellow, are the lowest-level, or atomic, components of BCOT. A node is defined by its name, identifier, definition (e.g., a list of data or equations using the values of other nodes), etc. Modules contain nodes and/or other modules. They are indicated in light blue with dark outlines. The Overview and Changes and Notes nodes are special cases; they are merely placeholders for textual documentation.
shall now discuss from a top-down perspective, returning to the Interface module in Chapter Three.

**HIGH-LEVEL STRUCTURE**

**From Building Blocks to Composite Investment Options**

Double-clicking on Model in Figure 2.1 brings up a window showing BCOT’s contents (Figure 2.2). Working left to right, we first see the Building Blocks node, which contains a simple list of names corresponding to the building-block options. These might be, e.g., programs for development and acquisition of a radar, a defense-suppression package, a missile, or even an airplane. Buying an individual building block may or may not add effectiveness. In some cases, combinations of building blocks are needed.

![Figure 2.2

Top-Level Structure](image)

The point of starting with building blocks is to step backward from the common practice of considering acquisition programs for complete or near-complete systems and to think more in terms of the higher-level ingredients that could go into a system so as to see alternative ways of

---

3 Gray modules such as Cost-Sorted Results contain portions of the BCOT program that accomplish various mathematical manipulations that users will ordinarily take for granted. Other gray modules collect various items that are essential for BCOT operations, but that users will usually ignore. These include the mathematical definitions of functions, lists of allowed parameter values, and so on.
achieving system capabilities—including combinations of ingredients that might not otherwise be suggested. This can enhance the “jointness” of option development, identify synergies for either military utility or economic advantage, and clarify where possibilities exist for adjusting options upward or downward, adjusting performance requirements, changing the number of acquired units, and so on. Thinking about such possibilities can also lead to additional building blocks (e.g., buy only 50 capability-enhancing kits, rather than 300).

The next step (following the arrows in Figure 2.2) is to develop composite investment options, or options, for short. These are investments in combinations of the building blocks. We take a Chinese-menu approach, considering all possible combinations of building blocks. That is, one option might involve buying the second, fourth, and ninth building blocks. Given N building blocks, there are $2^N$ combinations (one less if the baseline of “none” is excluded). Most of these make no sense because the pieces don’t work well together or an option includes some building blocks that add nothing to effectiveness. Such nonsense options are deleted in a filtering process, along with options that make sense but are less distinctly less good than others at a given cost.4

As we shall see, the methodology begins with a moderate number of building blocks (perhaps 10 to 20), generates many thousands of possible options, then leads eventually to a much more modest number of good candidate options.

The Options node of Figure 2.2, then, is just a table (the simplest kind of mathematical array) listing all of the possible options, perhaps thousands of them, and using 0s and 1s to define each option by which building blocks it contains.

Costs of the Investment Options

Users may define costs in various ways when using BCOT, but we have in mind the economically sound approach of using life-cycle costs, or annualized versions thereof, in which case, costs would reflect research and development, procurement (including that to replace peacetime attrition), and operations and maintenance. They would not include unpredictable and exceptional expenses, such as having to replace equipment lost in a future war or in an extended stabilization operation.

BCOT’s Costs module calculates cost for each composite option, assuming that the cost of an option is the sum of the costs of the building blocks that comprise it. Those building-block costs are inputs. This summation approach is an approximation, because it does not explicitly

---

4 As discussed later, some of the options may reappear as BCOT is systematically exercised with diverse assumptions.
include integration costs. Instead, it is assumed that the building-block costs include rough estimates of associated integration costs. We could, of course, treat integration activities as discrete building blocks, but that introduces a certain level of complication with which we did not wish to bother at this time.5

For work within a single capability area, the costs used in BCOT should be only those that are to be borne by the capability area in question. They may be input directly, for each building block, or the user can enter both the full cost of the building block and the fraction of that cost to be charged to the capability area. For example, a new advanced aircraft might have annualized life-cycle costs of $2 billion/year but would be used for a wide range of missions. The Global Strike capability area might be charged only 10 percent, or $200 million/year. The user of BCOT could input either $200 million or both $2 billion and 10 percent. This is a rather trivial flexibility, but one with practical advantages.

The Knotty Problem of Shared Costs

Unfortunately, it is often unclear how much of a building block’s cost “should” be charged to a given capability area. If a capability area needs ten dedicated aircraft, then the marginal cost of extending the size of a large ongoing acquisition by ten aircraft would be the appropriate number to use. Suppose, however, that acquisition of a building block is being considered for multiple purposes and that the building block is expensive (e.g., a new airframe or a constellation of satellites). The corresponding program may go forward only if multiple users sign up for cost-sharing and those users would all like to buy on the margin, leaving the biggest costs to others. The fractional costs are the result of bargaining and may or may not be sensible from a purist perspective. A given user might like to be the last to sign on, with apparent reluctance, affected to strike a better deal (the other users being concerned that the entire program will fall through unless another subscriber can be found).

Another interesting case occurs when multiple users are tentatively interested, but some have little money to pay for the new capabilities. They therefore ask for “new” money from the Department of Defense (DoD). This is fairly common in DoD developments, as when new command-and-control systems are introduced (such as the Global Information Grid (GIG)). Although many potential users recognize the desirability of a new capability, none may wish to pay for it, and all worry about being stuck with an excessive share of its cost if it is mandated. In

5 It would increase the dimensionality of the problem and would require us to adjust the effectiveness functions so as to penalize severely those options that did not include integration.
such cases, the Secretary of Defense may decide whether to go ahead only after broad, cross-cutting consideration of issues and alternatives. If the decision is to go ahead, the Secretary must also specify how costs will be allocated—in some cases, to new user organizations within a Service or a defense agency. In such cases, the cost fractions allocated may reflect a combination of behind-the-scenes bargaining and arbitrary choices, such as which Service or agency should take the lead. “New” money will then flow accordingly, perhaps with cuts in other parts of the DoD budget.

Our point here is that the fractional cost of a building-block capability that is nominally charged to a given capability area may not be predictable ahead of time and may be substantially arbitrary. As discussed in a companion monograph on portfolio analysis for capability areas (Davis, Shaver, and Beck, forthcoming), this causes substantial difficulties for analysis, especially for those charged with making tradeoffs within a capability area for which the cost fraction of a given building block is small and the budget is fixed. The situation can be understood roughly by merely appreciating that the cost of a building block for which a capability area is being charged a small fraction (say, 10 percent) could easily double as the result of downstream negotiations and the eventual inability of other capability areas to pay the fractions they are currently expected to pay. For example, at a given time, planners might anticipate taking the bulk of some new expenditures out of operations and maintenance (O&M). It might transpire, however, that DoD would find itself with a much higher than expected operational tempo, or that management reforms expected to improve O&S efficiencies fail. Without new money entering the system, the shortfalls would be reallocated, and the capability area that was previously getting a bargain might find itself paying dearly for something it would not have chosen to buy at the eventual price.

The bottom line, for this report, is simply that when BCOT is used, it is essential to consider large variations in at least some of the cost fractions assumed, especially those that are nominally small parts of large costs. Users should also record carefully the assumptions going into the cost figures they use, because many of them will prove faulty as the result of the kinds of horse-trading discussed above.

Effectiveness

Alternative Calculations: Quasi-Linear and “Standard.” BCOT itself does not contain an effectiveness function or related data. That information must be provided by the user for the specific problem area. However, BCOT provides placeholders for this purpose, and the default version of BCOT includes an illustrative effectiveness model.
With this in mind, recall that Figure 2.2 shows an Effectiveness module. This module computes an approximate effectiveness for each of the many composite options. If the number of such options were small, the effectiveness might be specified as data for each option. However, because there may be many thousands of options, it is preferable for BCOT to estimate the effectiveness from a much smaller set of inputs, using a simple model.

Double-clicking on Effectiveness reveals an underlying structure, shown in Figure 2.3. The Standard Calculation is ordinarily used; it can be quite complicated, but on an application-specific basis. The Quasi-linear Approximation is a placeholder for an alternative calculation that may require less work to set up in a given application. It must also be tuned to the specific application. It estimates an option’s effectiveness as the sum of effectiveness contributed by its building-block components, modified by some correction terms or factors. This approach works well in some applications (see Appendix A for discussion) but becomes much more cumbersome or is unnatural in others, e.g., as ballistic-missile defense.

**Figure 2.3**
**Computing Effectiveness**

**Determinants of Effectiveness.** The calculated effectiveness of an option depends on the scenario, the force-employment mode,\(^6\) and various parameters of the calculation, such as assumptions about the nature of the enemy’s air defense.\(^7\) A baseline case makes use of existing

---

\(^6\) By “force employment mode” we mean the choice of which forces are to be used, and how, in accomplishing the mission. For example, will the desired effects be achieved with an attack by aircraft, by missiles, or a combination? Effectiveness will typically be different for each case.

\(^7\) We use the term “scenario” for what some might call a “scenario class.” Some of the parameters to which we refer specify details of a scenario (e.g., warning time, level of adversary capability).
capabilities, so that an option is judged by whether additional investment in something actually improves effectiveness relative to what could be done anyway.\textsuperscript{8} The answer again depends on scenario, force employment, and parameters. Buying a sophisticated radar might have substantial value in one scenario but no additional value in others, because the force-employment mode used in them would not involve the radar.

**Best Effectiveness.** Proceeding rightward in Figure 2.3, the next step is to find each option’s effectiveness, by scenario, for the most-effective employment mode possible with the capabilities available under that option (either available in the baseline force or as the result of additional acquisitions made under the option). This is arguably a fair way to assess the option, at least for purposes of screening.\textsuperscript{9} At the time of a real-world operation, of course, force employment might be different and effectiveness might be lower. This has always been an aspect of program building as distinct from “warfighting.”\textsuperscript{10} Using such assumptions in defense planning makes the most sense when program analysts have relatively realistic (although low-resolution) models of force employment and effectiveness.\textsuperscript{11}

Returning now to the level indicated by Figure 2.2, the final top-level module is “Cost sorted results,” described in the next section.

**FINDING THE BEST CANDIDATE OPTIONS**

**Initial Sorting and Filtering**

The Cost-Sorted Results module of Figure 2.2 has the component modules indicated in Figure 2.4. First, it generates a list of composite options, ordered by increasing cost (and then by descending effectiveness). For each, it shows the corresponding effectiveness and cost. It does

---

\textsuperscript{8} A variant approach can have negative building blocks, such as “retire such-and-such a capability,” thereby requiring that some missions be done in different ways than they have been previously. Thus, BCOT can be used to consider minuses as well as pluses. We do not discuss such matters further in this report.

\textsuperscript{9} This assumes that the effectiveness function and force-employment methods used are sufficiently apt. They need not be precise, but they cannot be seriously misleading. Obviously, a building block’s value can be misestimated if, for example, the only force-employment mechanisms used to evaluate it are ineffective.

\textsuperscript{10} See, for example, the classic book on defense economics by Hitch and McKean (1965) or a more recent text on defense planning by Kugler (2006).

\textsuperscript{11} Examples of where this condition might not apply are (1) evaluating effectiveness of hypothetical force-employment missions that in the real world would be vetoed for lack of the necessary search-and-rescue capability; (2) wrongly assuming a willingness to use unmanned systems that could not be recalled en route; or (3) assuming a willingness to use highly secret technology that would be compromised if the systems were captured.
so for each scenario and assumptions set (parameter values) (e.g., the level of air defense or the size of the enemy forces). We refer to a given set of relevant parameters as an “assumptions set.”

BCOT then includes two kinds of filtering. First, an optional filter can be used to delete options that are more expensive than some threshold of plausibility. Although logically unnecessary for what follows, such filtering can be useful in reducing the number of options to be carried along through BCOT's processing. The second kind of filtering is different: If an option is a superset of an earlier option, is no more effective, but is more expensive, then it is a candidate for deletion (see Appendix B).12

The final module is Find Options Near the Efficient Frontier, which we shall discuss in some detail in the next section.

**Figure 2.4**
Cost-Sorting, Filtering, and Selecting Options

![Diagram showing the process of cost, sort, filter, find options near efficient frontier.]

**Finding Options On or Near the Efficient Frontier**

Given that one has many options with varied effectiveness and cost, it is obviously desirable to know which are the “best” as a function of cost. These are the Pareto-optimal options, the options such that no other option with the same cost has higher effectiveness. In a plot of effectiveness versus cost, a line connecting those Pareto-optimal points is often called the “efficient frontier” (Winston, 1994, p. 809). Something similar was discussed in the early work on portfolio theory of Nobel Prize winner Harry Markowitz (Markowitz, 1952), but in such

---

12 In the current version of BCOT, the second type of filtering actually occurs within the Find Options Near Efficient Frontier module. In the future, that functionality will be moved to the Filter for Cost and Effectiveness module.
applications the efficient frontier usually connects Pareto-optimal points in a plot of expected return versus risk for alternative portfolios.\textsuperscript{13} We have extended the efficient-frontier concept to include the concept of being \textit{on or near} the frontier. We have also used a more discretized concept because of the nature of defense-acquisition problems. Figure 2.5 shows the concept schematically. Each dot in the figure represents one of the options, which has an effectiveness (y-axis) and a cost (x-axis). Usually, as cost goes up, effectiveness will go up as well, but this is not always the case. Some options don’t make sense (because, e.g., they combine building blocks that don’t go together), and some are overpriced for the particular capability area. The solid points lie on what we refer to in our work as the “efficient frontier.” It is a staircase function, rather than the nice continuous curve in the original definition of efficient frontier and typical discussions in economics and operations research. We also show the \textit{convex hull}, which is often mentioned in the literature; it is the lightly dashed line defining a convex shape. At any given level of investment (x-axis), the best effectiveness that can be obtained from any of the options is the effectiveness of the efficient-frontier curve. There may be options that have the same cost but lower effectiveness, or that have the same effectiveness but higher cost. Those are below the efficient frontier. We call the dark points of Figure 2.5 “dominant points.” These are points on the efficient frontier that are either the least costly for a given effectiveness or the most effective for a given cost. Thus, in Figure 2.5, the leftmost gray point is not a dominant point even though it sits on the efficient frontier. Similarly, the open-circle point that sits on the vertical part of the efficient frontier is not a dominant point.\textsuperscript{14, 15}

In a schoolbook problem, students would identify the options on the efficient frontier and discard the others as inferior. Many tools used in finance do this, as well. However, we wish to go well beyond that. After all, BCOT is for preliminary screening. It will ordinarily be used with a highly simplified effectiveness function. An option that looks good with that function may look less good when considered in a more comprehensive analysis examining risks of various kinds, or

\textsuperscript{13} Many modern references discuss classic efficient-frontier calculations and related risk-reward issues. One that is both solid and readily available is an online textbook by Stanford’s William Sharpe, a 1990 winner of the Nobel Prize (Sharpe, 2006).

\textsuperscript{14} Strictly speaking, it does not really make sense to refer to the two points in question as “on” the efficient frontier, because the efficient frontier in question is discontinuous and therefore has no defined value between edges of a horizontal or vertical stretch.

\textsuperscript{15} The usual definition of efficient frontier refers to the smooth curve created by connecting each of the dominant points. That is sensible in theoretical economics, where continuity can be assumed in the nature of options and effectiveness. In our problem, however, there are no composite options between the points shown.
in parametric excursions on both effectiveness measures and costs.\textsuperscript{16} An option in Figure 2.5 lying below the efficient frontier may end up looking relatively more attractive than a point on the frontier. Since we do not want our screening approach to lose potentially good options, BCOT keeps many composite options at or near the efficient frontier. However, BCOT drops many composite options that are more expensive but no more effective than previous dominant points (“previous” refers to thinking in left-right terms). This is accomplished using set theory, as discussed in Appendix B.\textsuperscript{17} It is important also to realize that at a later point in the methodology, we combine options that survive screening with each of several choices of effectiveness function (e.g., choices that weigh scenarios differently).

---

\textsuperscript{16} We have in mind portfolio analysis as described in Davis, Shaver, and Beck (2007).

\textsuperscript{17} It could be argued the open circle in the very middle, which sits along the vertical line indicating a discontinuity, should be considered to be on the efficient frontier. However, it is substantially less effective than the higher point, for the same cost. Thus, it should not be included. It will not be included as long as the program processes options left-to-right in order of ascending costs \textit{and}, within any group of equal-cost options, in descending order of effectiveness. As a fine point, note that the staircase function we use in defining the efficient frontier is not actually defined mathematically for ranges in which it is horizontal or vertical. Thus, one could argue that the points are not strictly “on” the efficient frontier.
Given the desire to retain options near the efficient frontier, the question now becomes "How near?" That is an application-specific question. However, if the effectiveness function measures something necessary (whether it is sufficient or not), “near” might be something like 5 or 10 percent. For example, if effectiveness in ideal circumstances were 0.8 at best (e.g., 80 percent likelihood of success), something more than 10 percent inferior by this measure might reasonably be regarded as unacceptable. That would be a heuristic judgment, but heuristics are powerful devices in decisionmaking and planning (e.g., requirements for 0.95 levels of readiness, or 0.95 levels of reliability). In our prototype applications, we have treated “near” as being between 0 and about 10 percent in the vertical (effectiveness) direction. “Near” may be set differently for the horizontal direction (cost), which makes particular sense when cost estimates are unreliable. In our prototype work, we have often set “near” to mean within 5 or 10 percent in effectiveness and 10 or 20 percent in cost. As the criteria are weakened, more options pass through the screening. Using a 5/10 percent criterion, the options corresponding to black and gray points in Figure 2.5 would be retained. More-complicated cases are illustrated in the Appendix B.18

The mathematics inside the Find Options Near Efficient Frontier module accomplishes what is indicated schematically in Figure 2.5. The result is that the ultimate output of BCOT consists of

- A list of good candidate composite options, options worthy of more-extensive examination in a full portfolio-analysis framework such as that obtained using RAND's Portfolio-Analysis Tool (PAT).19
- Tables and charts showing first-order cost and effectiveness values for the various options.

**Manual Checking.** Although it is technically attractive to think in terms of a purely mathematical approach, we concluded that a computer-assisted approach that includes checking by the user is better. As discussed further in Appendix B, identifying nonredundant points near the efficient frontier is challenging in detail, both conceptually and mathematically.

---

18 Ultimately, it will be essential to use BCOT in connection with exploratory analysis (Davis, 2002) that systematically tests the robustness of BCOT’s outputs (the list of screened options) against changes of assumptions about scenario weightings, parameter values, and costs. Without this, the BCOT methodology will sometimes exclude options that it would be wise to retain.

19 A first version of PAT, specialized for missile defense, is documented in Dreyer and Davis (2005), but PAT has subsequently been enriched and revised substantially (Davis, Shaver, and Beck, forthcoming).
Pragmatically, the analyst needs to review the results of BCOT’s default algorithms and make occasional adjustments, sometimes dropping options and occasionally restoring one that had been screened out. The manual check allows the analyst to apply application-specific subtleties that cannot be encoded easily. An automated approach would retain many more points, the vast majority of which would not really be of interest.

**The Importance of Exploratory Analysis.** These results depend upon numerous inputs, such as assumptions and data about the building-block options, the scenarios, and parameterized aspects of scenario-weighting and force-employment effectiveness. Thus, BCOT can be used to explore how results change across the assumption space. Such exploratory analysis is essential, because the uncertainties are large and interrelated. Exploratory analysis is intended to be a fundamental part of capabilities-based planning (Davis, 2002), much as broad parametric analysis is essential to the work of good design engineers. Although such use is still relatively unusual (see, however, Davis, Bigelow, and McEver (2000) and Davis, McEver, and Wilson (2002)), exploratory analysis is increasingly recognized as important to good analysis of modern defense problems (National Research Council, 2006), as well as problems in other areas.

For work with BCOT, exploratory analysis will unquestionably demonstrate that even identification of options that should be considered as candidates for further study will depend heavily on the various assumptions. It is not yet clear what form exploratory analysis would best take. In prior applications, we have often varied assumptions parametrically and merely observed the combinations of assumptions under which various conclusions were valid. In using BCOT, however, it will probably be desirable to attach subjective probability distributions to the various assumption sets, and subjective importance weightings to test scenarios, so that the output might be like that shown in Table 2.1. In this purely notional listing, N options survive screening under exploratory analysis if we require that an option show up near the efficient frontier with at least a probability of 0.5. Note that “probability” in such exploratory analysis has a methodological meaning, rather than implying something about the real-world likelihood of applicability.

As of this time, we cannot speculate further, because substantial additional research is needed. Nonetheless, we include this speculation to emphasize that work of the kind we propose with BCOT will not be valid if it is based simply on nominal assumptions. Exploratory analysis is essential.
<table>
<thead>
<tr>
<th>Option</th>
<th>Effectiveness (one-sigma range)</th>
<th>Cost (one-sigma range)</th>
<th>“Probability” of Being on Efficient Frontier</th>
<th>“Probability” of Being Near Efficient Frontier</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8 (0.5–0.85)</td>
<td>1,000 ($900-1,200)</td>
<td>0.8</td>
<td>0.95</td>
</tr>
<tr>
<td>2</td>
<td>0.85 (0.5–0.9)</td>
<td>1,250 ($900-1250)</td>
<td>0.55</td>
<td>0.65</td>
</tr>
<tr>
<td>3</td>
<td>0.85 (0.75–0.9)</td>
<td>1,300 ($800-1,300)</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0.9 (0.5–0.95)</td>
<td>2,500 ($1,900-3,800)</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0.95 (0.4–0.98)</td>
<td>4,500 ($3,500-$10,000)</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

*This “probability” is the weighted fraction of the case space for which the option in question is on or near the dominant points of the efficient frontier.
CHAPTER THREE
THE CENTRALIZED INTERFACE: INPUTS AND OUTPUTS

Because of BCOT’s interactive nature, almost all of its aspects can be considered inputs. For example, equations exist in some of the modules. Where necessary, many of the equations can be readily changed—in much the same way that users can change simple equations in spreadsheet programs. More typically, the intention is for BCOT’s structure to be constant within a given application, so that the inputs can be reduced to a simple list and collected in an interface. Similarly, although it is easy to generate displays for any variable in the entire computational process within BCOT, a small subset usually suffices, and those can be collected in the same interface. Returning to the faceplate level (Figure 2.1), double-clicking on Interface brings up the graphical user interface shown in Figure 3.1. The interface shown is specific to the application that we were using at that particular time and need not be discussed further here. It suffices to note that there is a single-page interface with about 30 input nodes. The lower part of the figure shows the contents of a lower-level interface, that for Other Inputs. Using BCOT, then, is not trivial, but it is a reasonably sized tool.

BCOT’s inputs are shown along the left side of Figure 3.1; a smattering of the many potential outputs is shown in the right column. Tables 3.1 and 3.2 explain the terminology and briefly describe the illustrative inputs and outputs. Note that this interface applies only to the example used for this documentation (the example described in Chapter 4), not to what would be seen in a fresh copy of the evolving BCOT or another problem.

Ultimately, then, BCOT is rather simple conceptually, although the underlying mathematics and programming proved to be distinctly nontrivial.

---

20 Although changing equations is usually rather easy, we have in mind analysts able to understand the key features of the Analytica model and its high-level programming language. Further, some changes—although easy to make—require an understanding of array-based mathematics and programming. Our intention has been to hide those aspects of BCOT so that they need not be encountered by most users.

21 Changes in the scenarios or force-employment modes used, or the parameters used to characterize effectiveness, require going beyond this interface, as discussed in Appendices D, E, and F. The primary reason is that changes in those elements require providing additional data and additional input nodes.
Figure 3.1
Illustrative Inputs and Outputs of BCOT
Table 3.1
Illustrative Inputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Building Blocks and Their Costs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Building blocks</td>
<td>Names of building-block options</td>
<td>Building blocks are primitive options; combinations create “composite options”</td>
</tr>
<tr>
<td>Cost switch</td>
<td>Specifies if building-block cost is calculated or input</td>
<td></td>
</tr>
<tr>
<td>Building-block cost inputs</td>
<td>Costs for direct input</td>
<td>Annualized 20-year costs</td>
</tr>
<tr>
<td>Building-block base costs</td>
<td>Total cost of the building blocks</td>
<td>Annualized 20-year costs</td>
</tr>
<tr>
<td>Fractional cost allocated</td>
<td>Fraction of total cost allocated to capability area</td>
<td>See discussion in Chapter 2.</td>
</tr>
<tr>
<td><strong>Effectiveness Calculations</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effectiveness switch</td>
<td>Specifies whether effectiveness is calculated using quasi-linear or the “standard” calculation</td>
<td>Effectiveness functions must be provided by the user; BCOT merely provides placeholder</td>
</tr>
<tr>
<td>Baseline effectiveness&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Effectiveness (0 to 1) with baseline forces only</td>
<td></td>
</tr>
<tr>
<td>Building-block effectiveness&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Increment of effectiveness added by each building block</td>
<td>These can be inferred from more-detailed calculations</td>
</tr>
<tr>
<td>Building-block usage&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Specifies which building blocks are used in employment modes</td>
<td>A given building block may be used in several modes</td>
</tr>
<tr>
<td>Weights (by Focus)</td>
<td>Used to calculate net effectiveness across scenarios</td>
<td></td>
</tr>
<tr>
<td>Cost Ceiling</td>
<td>Maximum option cost considered</td>
<td></td>
</tr>
<tr>
<td><strong>Efficient-Frontier Parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient-Frontier Parameter</td>
<td>Defines “close” as a fraction of frontier’s value in effectiveness</td>
<td>Example: “near” might mean 0.95 for effectiveness and 0.9 for cost; parameter values would then be 0.95 and 2</td>
</tr>
<tr>
<td>Cost-Over-Effectiveness Parameter</td>
<td>As above, but in cost dimension; expressed as a ratio</td>
<td></td>
</tr>
<tr>
<td>Cases Included in the Union</td>
<td>A table specifying which parameters are to be varied across assumption sets</td>
<td>If P1 and P2 are specified, with two values each, the union will combine options from four assumptions sets</td>
</tr>
<tr>
<td>Focus Components for the Union</td>
<td>A table specifying which values of focus are to be varied in developing alternative sets of candidate options before taking the union; three options are currently available without programming changes</td>
<td>Allows calculation of union of options retained under individual scenarios (rather than an average over multiple scenarios); Effective if Focus is selected in the table of Cases Included in the Union</td>
</tr>
</tbody>
</table>

<sup>a</sup> Applies to quasi-linear approximation.
## Table 3.2
### Illustrative Outputs

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost-Sorted Efficient-Frontier Option Attributes by Focus</td>
<td>A juxtaposition of cost-sorted options near the efficient frontier, along with their cost and effectiveness scores under different focuses</td>
<td>Allows viewing the set of options near the efficient frontier and their effectiveness scores for different cases (under different assumptions)</td>
</tr>
<tr>
<td>Attributes of Combined Options: Union over All Parameters</td>
<td>Cost-sorted union of options near the efficient frontier for the complete set of assumptions, along with their costs and effectiveness scores under different focuses</td>
<td>Allows viewing in one list all options (with their attributes) that are near the efficient frontier for at least one effectiveness function; would not scale well with numerous parameters</td>
</tr>
<tr>
<td>Attributes of Combined Options: Union over Selected Parameters</td>
<td>As above, except the union is taken over results varying only selected parameters, rather than all parameters</td>
<td>For example, allows seeing the list of options that are near the efficient frontier under for at least one effectiveness function</td>
</tr>
<tr>
<td>Individual Outcomes and Dominant-Points Options vs. Costs</td>
<td>Plots cost vs. effectiveness for three sets of options: all filtered options; options that were dropped because some building blocks did not add effectiveness, only costs, to dominant options; options that were retained for being at or near the efficient frontier.</td>
<td>Visualizes the effects of the two-step algorithm for dropping cost-ineffective options</td>
</tr>
<tr>
<td>Cost Ranges where Individual Building Blocks Become Part of the Solution</td>
<td>For every building block, plots cost ranges corresponding to the retained options of which the building block is a part</td>
<td>Shows which building blocks will be acquired within the cost ranges indicated by bars</td>
</tr>
</tbody>
</table>

NOTE: “Near the efficient frontier” actually means near the dominant points of the efficient frontier.
CHAPTER FOUR
A NOTIONAL EXAMPLE

This chapter employs BCOT schematically in an application to a notional, highly simplified version of the Global Strike problem described in Davis, Shaver, and Beck (2007). It assumes that the building-block investment options involve buying an aircraft or an upgrade thereof, buying a missile capability, buying some special weapon, and buying specialized capabilities to improve attacks by special operations forces (SOF). We refer to these briefly as the aircraft, missile, weapon, and SOF building blocks. The example also assumes four employment modes: air attack, missile attack, SOF attack, and a joint air-missile-SOF attack.\footnote{The reader may find the distinction between option and force-employment mode to be confusing. It is one thing to have a variety of force capabilities, either as part of the baseline or as the result of the particular investment option; it is another thing to decide how to conduct the mission of a particular scenario. The capability provided by a particular option may be less suited to the scenario than a capability present in the baseline. Or it may be that the mission would best be performed with a combination of capabilities (e.g., aircraft and SOF, or aircraft and missiles). A fully specified force-employment mode would also have a well-defined concept of operations (CONOPS), but we do not discuss CONOPS here.} In a real application, of course, there would be much more specificity, identifying particular aircraft such as an advanced long-range bomber, a new conventional ballistic missile (whether for ground launch or launch from submarines), etc. Further, the building blocks might include options enhancing surveillance or targeting. The employment modes would also be more elaborately defined. What follows is intended merely for explanatory purposes.

BUILDING BLOCKS AND COMPOSITE OPTIONS

Let us assume the four building blocks mentioned above and denote them as A, M, W, and S, respectively. These building blocks can be purchased in different combinations, each constituting a “composite option,” or simply “option,” for short. This implies a total of \(2^4-1 = 15\) composite options, which does not count the option of buying nothing. Table 4.1 generates that set of options (called the "power set" mathematically). The 0s and 1s indicate whether a given option contains a particular building block. Option 5, for example, is buying the aircraft building block and the weapon building block.

BCOT generates the equivalent of Table 4.1 automatically, given only the list of building blocks. The result may be large numbers of composite options rather than the 15 shown in the example. That is, 10, 20, and 30 building blocks imply roughly \(10^3\), \(10^6\), and \(10^9\) composite options, respectively. Given the simplicity of the calculations and the speed of modern
computers, large numbers are not necessarily a problem, but personal computers still have memory limitations, so we have had in mind many thousands—but not millions or billions—of cases.

Table 4.1
Composite Options for a Simple Notional Case

<table>
<thead>
<tr>
<th>Option N</th>
<th>Option Name</th>
<th>Aircraft</th>
<th>Missile</th>
<th>Weapon</th>
<th>SOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>AM</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>W</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>AW</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>MW</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>AMW</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>AS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>MS</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>AMS</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>WS</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>AWS</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>MWS</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>AMWS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Option names indicate which building-blocks capabilities are included, using A, M, W, and S to denote the aircraft, missile, weapon, and SOF building blocks. For example, AM means that both the aircraft and missile building blocks are present.

FORCE EMPLOYMENT BY SCENARIO CLASS

The effectiveness of a given option will depend on the scenario class (including the mission to be accomplished) and the force-employment mode, i.e., the way in which forces are employed to accomplish the mission. For our notional example of Global Strike, we assume three scenario classes, denoted briefly by the targets they are to attack: mobile missiles, terrorist groups, and WMD facilities. The employment modes considered are air attack, missile attack, SOF attack, and joint air-missile-SOF attack. A given building block may or may not be used in a given force-employment mode in a particular scenario class. The user must specify whether each building block is used in each force-employment mode. One BCOT input, then, is the equivalent of Table 4.2. As an example, this indicates that in the joint employment mode (last column), all of the building blocks are used, if available for the given option.
Table 4.2
Force-Employment Modes for Illustrative Building-Block Options

<table>
<thead>
<tr>
<th>Building Block</th>
<th>Force-Employment Mode</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Air Attack</td>
<td>Missile Attack</td>
<td>SOF Attack</td>
</tr>
<tr>
<td>Aircraft</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Missile</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Weapon</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>SOF</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**ESTIMATING EFFECTIVENESS**

As discussed earlier, BCOT uses two different methods for estimating effectiveness of composite options. In either case, the user must specify the algorithmic details and data.

**Quasi-Linear Approximation**

The quasi-linear approximation of effectiveness for a given option is the sum of the baseline effectiveness applicable if none of the building blocks are purchased, the incremental improvements due to the individual building blocks available in the option, and possible nonlinear correction factors. All of these depend on the scenario class and force-employment mode. Thus, the user must specify quite a number of related inputs, as illustrated by Table 4.3. This example shows only the inputs for the scenario class involving attack of mobile missiles; similar tables would apply for the other scenario classes. For the example in Table 4.3, missiles are deemed ineffective because they are not able to detect, find, and accurately strike mobile missiles. The advanced weapon is assumed to contribute increments of 0.2, 0.0, 0.1, and 0.2 in air attacks, missile attacks, SOF attacks, and joint air-missile-SOF attacks, respectively.

Table 4.3
Incremental Effectiveness of Building Blocks for the Mobile Missiles Scenario Class (Quasi-Linear Approximation Only)

<table>
<thead>
<tr>
<th>Building Block</th>
<th>Force-Employment Mode</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Air Attack</td>
<td>Missile Attack</td>
<td>SOF Attack</td>
<td>Joint Air-Missile-SOF Attack</td>
</tr>
<tr>
<td>Aircraft</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>Missile</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Weapon</td>
<td>0.2</td>
<td>0</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>SOF</td>
<td>0</td>
<td>0</td>
<td>0.088</td>
<td>0</td>
</tr>
</tbody>
</table>

The linear part of the quasi-linear effectiveness calculation adds up the increments of effectiveness for each of the building blocks present in a given composite option. The result is a BCOT output, the equivalent of Table 4.4. In this output, for example, option 13, which involves
buying the aircraft, weapon, and SOF building blocks, contributes 0.3 to effectiveness in the joint air-missile-SOF mode of attack (last column). Note again that Table 4.4 is for the mobile-missile scenario class only.

### Table 4.4
Sum of Incremental Effectivenesses

<table>
<thead>
<tr>
<th>Scenario Class: Mobile Missiles</th>
<th>Force-Employment Mode</th>
<th>Joint Air-Missile-SOF Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Option Name</td>
<td>Air Attack</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>AM</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>W</td>
<td>0.2</td>
</tr>
<tr>
<td>5</td>
<td>AW</td>
<td>0.2</td>
</tr>
<tr>
<td>6</td>
<td>MW</td>
<td>0.2</td>
</tr>
<tr>
<td>7</td>
<td>AMW</td>
<td>0.2</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>AS</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>MS</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>AMS</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>WS</td>
<td>0.2</td>
</tr>
<tr>
<td>13</td>
<td>AWS</td>
<td>0.2</td>
</tr>
<tr>
<td>14</td>
<td>MWS</td>
<td>0.2</td>
</tr>
<tr>
<td>15</td>
<td>AMWS</td>
<td>0.2</td>
</tr>
</tbody>
</table>

The next step is to provide nonlinear correction factors for cases in which the user knows they are important. Appendix A describes the related mathematics, but—simply as an example—suppose that effectiveness in a joint operation involving the new aircraft, missile, and SOF capability would be extremely good, much more than just the sum of incremental improvements added by the component building blocks. Suppose that we believed effectiveness in the last option of Table 4.4 (row 15) for the mobile-missile scenario should be 0.6. Perhaps the missile component of a joint attack, while not very effective in itself for attacking mobile missiles, would be able to so disrupt air defenses as to leave aircraft free to loiter, search, and destroy. In such a speculative situation, the user might specify an input that would either add to, multiply, or substitute for the linear approximation. Suppose that the force-multiplier approach were taken. The input would be 2, corresponding to a doubling of effectiveness that would take effectiveness from 0.3 to 0.6. BCOT’s mathematical machinery would assure that this multiplier were applied only in the AMWS case—the case in which all of the building blocks were present and used in the joint employment mode.
As another example, a nonlinear correction could zero out any option that failed to contain all the critical building-block components of a given employment concept. Such checks are easy to make with Analytica’s array mathematics.

The “Standard” Calculation of Effectiveness

In the standard calculation, more inputs are needed, whether they are made directly or generated from side calculations or an embedded model. In the extreme, one might have input tables such as Table 4.5, one for each scenario class. For simplicity, this table assumes that almost all the options are totally ineffective. For the notional example, buying a new aircraft and weapon (option 5) buys some effectiveness, with an air attack being as effective as a joint attack that includes SOF. Buying a new aircraft, weapon, and specialized SOF capability would have more value. However, by adding a missile option as well (row 15, AMWS), the data assert that a fairly high probability of success can be achieved—but only in a coordinated joint operation (the last column). Note that if we regard small effectiveness values as essentially zero, then the inputs for Table 4.5 will be reasonably well approximated by the quasi-linear calculation coupled with the factor-of-two synergy-related multiplier mentioned in the previous section.

Table 4.5
Inputs for Standard Calculation of Effectiveness

<table>
<thead>
<tr>
<th>Option</th>
<th>Option Name</th>
<th>Air Attack</th>
<th>Missile Attack</th>
<th>SOF Attack</th>
<th>Joint Air-Missile-SOF Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>AM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>W</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>AW</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>6</td>
<td>MW</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>AMW</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>AS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>MS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>AMS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>WS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>AWS</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>14</td>
<td>MWS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>AMWS</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Abbreviations are: A, M, W, and S for attacks by air, missile, weapon, and SOF.
When the number of options is large or the effectiveness values are functions of parameters such as the quality of enemy air defenses and the degree of stealthiness in the aircraft option, such direct data entry can be overwhelming. Fortunately, it is possible to decompose the effort, as described in Appendix A. This merely requires using some theory and specifying parameters of the theory as inputs. For example, effectiveness may be calculated as the product of probabilities determining the chances for success of an option: those of acquiring the target, penetrating defenses, and killing the target. In this notional example, the values vary with the type of mission, force-employment mode, and quality of air defenses. Furthermore, instead of specifying all parameter values for each composite option, we exploit the fact that only some building blocks are relevant for any given employment mode.

Both approaches define final effectiveness scores of composite options as the effectiveness using the best force-employment option for the scenario test case available for a given investment option.

**Effectiveness vs. Cost Curves**

We can visualize the results of the calculations by plotting composite options with costs on the x-axis and effectiveness on y-axis. Each option, then, appears as a point. Figure 4.1 shows such a plot for the notional example and the case in which the scenario class is mobile missiles.

![Individual Composite Options: Costs vs. Effectiveness](image-url)
IDENTIFYING POINTS ON OR NEAR THE EFFICIENT FRONTIER

Figure 4.2 shows how BCOT narrows the options of interest, using the two-staged procedure described earlier: (1) find points near the efficient frontier,\(^{23}\) then (2) discard redundant points. It also shows the efficient frontier itself. In Figure 4.2 the three options shown with black diamonds are on the frontier. There are no options near but not on it. The dark-circle points are to be dropped because they are too far from the frontier. Options shown in Figure 4.2 as gray points are dropped because they are redundant, i.e., they contain building blocks that add to the option’s cost but do not contribute further effectiveness for the mission in question. A somewhat confusing aspect of the figure is that some points (e.g., AWS and MWS, costing about $90 million) seem to lie on the efficient frontier in the sense of lying on the horizontal line extending to the right from WS. They have the same effectiveness as WS, but higher cost. As a result, they are not dominant points. To the contrary, WS is superior by virtue of its lower cost.

\(^{23}\) More precisely, near the dominant points of the efficient frontier.
RESULTS BY FOCUS

The effectiveness of a composite option varies with the scenario. What effectiveness, then, should be used? BCOT has a concept of net effectiveness across scenarios, which is a linear weighted sum of effectivenesses for the various scenarios. (Depending on the perspective, or focus, with which we view the various scenarios, we assign different weighting factors).

We use the term “focus” to correspond to a particular set of weighting factors. BCOT will then find different options to be attractive (i.e., to be on or near the efficient frontier, and not redundant), depending on focus. This is quite important, because there is no meaningful way to define the “right” single focus. We want the methodology to find candidate options that are effective across a range of focuses.

The net effectiveness is also a function of the assumptions set—i.e., the various input parameters used in the calculations, parameters reflecting assumptions about, e.g., the probability that a new aircraft would penetrate advanced air defenses or the probability that a new missile would destroy a target upon impact.

Suppose that we use BCOT with a particular focus to find a set of candidate options. We refer to that as the “screening focus.” Once we have identified the options that survive, using that screening focus, we want to see how the options fare for the individual scenarios and for various averages of effectiveness over scenarios. That is, we want to display the effectiveness of options found with a particular screening focus as a function of focus. The syntax is a bit awkward, but it makes sense with a bit of thought. We also want to know how the screened options fare with different assumption sets (parameter values), but we do not illustrate that here.

For our example, we do not vary assumptions (i.e., the parameter values referred to above). Instead, we consider only five focuses, which correspond to calculating net effectiveness using only the mobile-missile scenario, the terrorist scenario, the WMD-facilities scenario, an equally weighted average of effectiveness over all three scenarios, and an average of effectivenesses over just the terrorist and WMD-facility scenario.

Tables 4.6 through 4.10 show results for our notional example for different choices of screening focus and focus. These, unlike the earlier tables, are screenshots of actual BCOT outputs. The last five columns represent the effectiveness scores under different focus values, while the value in the rotation box called Screening Focus indicates the focus that was used to select the set of options on or near efficient frontier – those shown in the first column. As a comparison of the tables reveals, the set of screened options is, in general, different for different values of screening focus. In Table 4.6, only three candidate options emerge for a screening focus that looks only at the mobile-missiles scenario. As it happens, these three options also do as well as this or better when evaluated by the other focuses. For example, the last option
achieves an effectiveness of 0.495 for the mobile-missiles scenario, but effectivenesses of 0.85 and 0.7083, respectively, for the terrorist and WMD-facilities scenarios.

Tables 4.7 and 4.8 show the candidate options generated with screening focuses of terrorist-scenario only and WMD-facilities only. Note that the sets of candidate options are different from that of Table 4.6. In Table 4.8 we also see that some of the candidate options found by using the WMD-facilities focus do much more poorly when assessed for the other focuses. Table 4.9 shows options screened with an effectiveness function that averages over effectiveness for the WMD-facilities and terrorist scenarios. Table 4.10 shows results for a focus that averages over all three scenario classes with equal weights. In this case, there are only six rather than seven options, they are a bit different than those for the other focuses, and all of them do rather well for all scenarios.

The candidate options found by screening would vary significantly with different assumption sets (not shown here).
Table 4.6
Attributes of Options On or Near the Efficient Frontier for the Mobile-Missiles Scenario

<table>
<thead>
<tr>
<th>EF Options</th>
<th>Cost ($Myr)</th>
<th>Eff Mobiles</th>
<th>Eff Terrorists</th>
<th>Eff Facilities</th>
<th>Avg Eff</th>
<th>Avg Terr. Facil</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOF</td>
<td>21.5</td>
<td>0.2</td>
<td>0.35</td>
<td>0.25</td>
<td>0.2607</td>
<td>0.3</td>
</tr>
<tr>
<td>WeaponSOF</td>
<td>48.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
<td>0.4333</td>
<td>0.45</td>
</tr>
<tr>
<td>Aircraft/MissleWeaponSOF</td>
<td>132.5</td>
<td>0.495</td>
<td>0.85</td>
<td>0.78</td>
<td>0.7083</td>
<td>0.815</td>
</tr>
</tbody>
</table>

Table 4.7
Attributes of Options On or Near the Efficient Frontier for the Terrorists Scenario

<table>
<thead>
<tr>
<th>EF Options</th>
<th>Cost ($Myr)</th>
<th>Eff Mobiles</th>
<th>Eff Terrorists</th>
<th>Eff Facilities</th>
<th>Avg Eff</th>
<th>Avg Terr. Facil</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOF</td>
<td>21.5</td>
<td>0.2</td>
<td>0.35</td>
<td>0.25</td>
<td>0.2607</td>
<td>0.3</td>
</tr>
<tr>
<td>WeaponSOF</td>
<td>48.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
<td>0.4333</td>
<td>0.45</td>
</tr>
<tr>
<td>Aircraft/MissleWeaponSOF</td>
<td>132.5</td>
<td>0.495</td>
<td>0.85</td>
<td>0.78</td>
<td>0.7083</td>
<td>0.815</td>
</tr>
</tbody>
</table>
Table 4.8
Attributes of Options On or Near the Efficient Frontier for the WMD-Facilities Scenario

<table>
<thead>
<tr>
<th>EF Options</th>
<th>Cost (M$yr)</th>
<th>Eff-Mobiles</th>
<th>Eff-Terrorists</th>
<th>Eff-Facilities</th>
<th>Avg Eff</th>
<th>Avg-Terr, Facil</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SOF</td>
<td>21.5</td>
<td>0.2</td>
<td>0.35</td>
<td>0.25</td>
<td>0.2667</td>
<td>0.3</td>
</tr>
<tr>
<td>3 Aircraft</td>
<td>37.5</td>
<td>0.2165</td>
<td>0.01</td>
<td>0.25</td>
<td>0.1422</td>
<td>0.205</td>
</tr>
<tr>
<td>5 Aircraft/SOF</td>
<td>59</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
<td>0.4633</td>
<td>0.5</td>
</tr>
<tr>
<td>6 Aircraft/Weapon</td>
<td>64.5</td>
<td>0.15</td>
<td>0.05</td>
<td>0.5</td>
<td>0.2333</td>
<td>0.275</td>
</tr>
<tr>
<td>8 Aircraft/Missile/SOF</td>
<td>86</td>
<td>0.4</td>
<td>0.5</td>
<td>0.55</td>
<td>0.5167</td>
<td>0.575</td>
</tr>
<tr>
<td>9 Aircraft/Missile weapon/SOF</td>
<td>105.5</td>
<td>0.385</td>
<td>0.7</td>
<td>0.5</td>
<td>0.575</td>
<td>0.67</td>
</tr>
<tr>
<td>7 Aircraft/Missile weapon/SOF</td>
<td>132.5</td>
<td>0.485</td>
<td>0.85</td>
<td>0.78</td>
<td>0.7083</td>
<td>0.815</td>
</tr>
</tbody>
</table>

Table 4.9
Attributes of Options On or Near the Efficient Frontier for the Average of Terrorist and WMD-Facilities Scenario

<table>
<thead>
<tr>
<th>EF Options</th>
<th>Cost (M$yr)</th>
<th>Eff-Mobiles</th>
<th>Eff-Terrorists</th>
<th>Eff-Facilities</th>
<th>Avg Eff</th>
<th>Avg-Terr, Facil</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SOF</td>
<td>21.5</td>
<td>0.2</td>
<td>0.35</td>
<td>0.25</td>
<td>0.2667</td>
<td>0.3</td>
</tr>
<tr>
<td>2 Weapon/SOF</td>
<td>45.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
<td>0.4333</td>
<td>0.45</td>
</tr>
<tr>
<td>4 Aircraft/SOF</td>
<td>60</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
<td>0.4433</td>
<td>0.5</td>
</tr>
<tr>
<td>5 Aircraft/Weapon</td>
<td>80</td>
<td>0.4</td>
<td>0.6</td>
<td>0.55</td>
<td>0.5107</td>
<td>0.575</td>
</tr>
<tr>
<td>8 Aircraft/Missile/SOF</td>
<td>105.5</td>
<td>0.385</td>
<td>0.7</td>
<td>0.64</td>
<td>0.575</td>
<td>0.67</td>
</tr>
<tr>
<td>7 Aircraft/Missile weapon/SOF</td>
<td>132.5</td>
<td>0.485</td>
<td>0.85</td>
<td>0.78</td>
<td>0.7083</td>
<td>0.815</td>
</tr>
</tbody>
</table>
Although the issue is not very significant for the highly simplified example in this chapter, the fact that different options survive screening, depending on the focus and assumption set used, means that it is dangerous to take the “optimization” aspects of the efficient-frontier technique too seriously. Consequently, the proper methodology, consistent with the desire to hedge, is to consider the union of options that look good under at least one screening focus (i.e., at least one weighting of scenarios and one plausible set of input parameters). Table 4.11 shows the result for our simple example. In this case, the set of options is only slightly larger (eight instead of seven) than the set we would have obtained if we had used only the screening focus that averages across scenarios. If we also varied the values of parameters, however, and if we had alternative kinds of aircraft, weapons, and so on, then the expansion of options surviving screening would be much larger because, typically, the relative goodness of options depends significantly on those parameter values. For example, an option that presupposes substantial strategic warning may be the best if such warning exists, but entirely inadequate if it does not.
Table 4.11
Union of Options Surviving Screening for at Least One Focus and Set of Parameter Values

<table>
<thead>
<tr>
<th>EF Options</th>
<th>Cost (M$yr)</th>
<th>Eff Mobiles</th>
<th>Eff-Terrorists</th>
<th>Eff-Facilities</th>
<th>Avg Eff</th>
<th>Avg-Terr, Facil</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SOF</td>
<td>21.5</td>
<td>0.2</td>
<td>0.35</td>
<td>0.25</td>
<td>0.2687</td>
</tr>
<tr>
<td>2</td>
<td>Aircraft</td>
<td>37.5</td>
<td>0.0165</td>
<td>0.01</td>
<td>0.4</td>
<td>0.1422</td>
</tr>
<tr>
<td>3</td>
<td>WeaponSOF</td>
<td>46.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.4</td>
<td>0.4230</td>
</tr>
<tr>
<td>4</td>
<td>Aircraft</td>
<td>57.5</td>
<td>0.33</td>
<td>0.5</td>
<td>0.5</td>
<td>0.4633</td>
</tr>
<tr>
<td>5</td>
<td>Aircraft</td>
<td>64.5</td>
<td>0.15</td>
<td>0.05</td>
<td>0.5</td>
<td>0.2330</td>
</tr>
<tr>
<td>6</td>
<td>Aircraft</td>
<td>98</td>
<td>0.4</td>
<td>0.6</td>
<td>0.65</td>
<td>0.6167</td>
</tr>
<tr>
<td>7</td>
<td>Aircraft</td>
<td>105.5</td>
<td>0.388</td>
<td>0.7</td>
<td>0.64</td>
<td>0.575</td>
</tr>
<tr>
<td>8</td>
<td>Aircraft</td>
<td>132.5</td>
<td>0.485</td>
<td>0.05</td>
<td>0.78</td>
<td>0.7603</td>
</tr>
</tbody>
</table>
CHAPTER FIVE
CONCLUSIONS AND NEXT STEPS

RECAPITULATION

Chapter 4 illustrated the overall concept of BCOT, which is summarized in Figures 5.1 and Figure 5.2. Figure 5.1 is a simplified depiction, whereas Figure 5.2 is more detailed, showing inputs and outputs. The flow is as follows:

1. Identify investment building blocks (e.g., a particular new aircraft or a new weapon).
   Many of these will be available from pre-existing proposals, but a more comprehensive set can be constructed for a given capability area by defining the mission, working through alternative ways of accomplishing the mission, noting the component capabilities and related systems that would be necessary, and highlighting those that do not presently exist and would therefore have to be developed.

2. Construct all possible composite investment options, i.e., all combinations of the building blocks.

3. Evaluate the composite options by cost and effectiveness and as a function of test scenarios, baseline effectiveness, and assumption sets (sets of values for the parameters used in defining scenarios and performing calculations).

4. Find the set of options that are economically efficient (on or near the efficient frontier, also called the Pareto-optimal frontier) for each of many effectiveness functions differing in the relative weight given to scenarios (screening focus) and the assumptions set used for parameter values.

5. Construct the combined set of options that are near the efficient frontier in at least one case of interest (i.e., a particular focus or choice of parameter values), and calculate their effectiveness for other choices of focus or assumption set.

6. Review the results manually, discarding some options, while perhaps adding some options back from the discard pile.
Figure 5.1
Simplified Schematic Overview of BCOT Process
Figure 5.2
Summary of BCOT's Logical Flow

1. Identify building-block options
2. Create composite options
3. Calculate costs and effectiveness of composite options by scenario and assumption set
4. Find options near efficient frontier
5. Find options near at least one efficient frontier (costs and effectiveness by scenario and assumption set)
6. Review manually and adjust

Final options: good candidates for detailed evaluation in portfolio framework
NEXT STEPS

BCOT is a prototype, not a finished product. Moreover, the BCOT methodology is still quite new, and much work is needed to fully develop the exploratory-analysis techniques needed to assure robustness of results.

The most important next step is to hone the exploratory-analysis concepts with a relatively complex application that includes quite a number of uncertainties about relative emphasis of scenarios and assumptions to be made in estimating effectiveness. This could be done within the Global Strike domain or any of many other domains. Our current image of the results includes something like Table 2.1, which hypothesized a summary of exploratory analysis that would characterize how frequently a given option survives screening as one considers a broad “case space” of assumptions about scenario weightings and parameter values. Such a summary of results would be appropriate only if the alternative focuses used had been very well chosen. That is a matter of strategic planning, not mathematics. For example, if planners were particularly concerned about capability for an unlikely scenario (e.g., capability for assured nuclear retaliation, so as to assure deterrence), the test set should not overly “dilute” that scenario by having large numbers of alternative focuses in which it is given minimal weight. Conversely, the test set should not put so much emphasis on the unlikely scenario as to suppress options that would be very useful for much more likely scenarios.

Many improvements are needed in the Analytica and Excel versions of BCOT. These include

- Reprogramming for further streamlining and clarity
- Improved internal documentation
- Creating functions and possibly adjunct tools in Microsoft Excel to make it easier to set up and execute complex experimental designs for exploratory analysis over alternative “focuses” and alternative assumptions about many parameter values
- Improving the algorithm for discarding inappropriately redundant composite options to catch those that are supersets of less-expensive options other than the previous dominant point
- Careful verification testing
- Productizing, so as to make it easier to use for individuals outside of RAND
- Treating integration costs separately rather than embedding them in the costs of R&D for the separate building blocks.

The above comments refer to the Analytica version of BCOT. It is also desirable to improve the Excel version. Some of those improvements might include

- Standard Excel reprogramming to replace cell references with named references
• Attachment of “macros” to execute operations that currently are accomplished manually
• Use of pivot-table mechanisms to improve the interface for at least limited exploratory analysis
Appendix A

EFFECTIVENESS CALCULATIONS

This appendix elaborates on mathematical issues involved in the two methods described in the main text for calculating the effectiveness of composite options. The first is fast and convenient; the second is for more-careful calculations.

THE QUASI-LINEAR APPROXIMATION

As described in the text, the effectiveness of a composite option can sometimes be approximated to first order as a sum of incremental effectivenesses resulting from its constituent building blocks. That approximation is not generally valid. In particular, many options consisting of only a single building block would have no real-world effectiveness, because other critical components of capability would be needed. In a simple linear approach, however, effectiveness would be assumed to be that of the building block.

Despite such issues, this BCOT approximation is often adequate because of details. In particular:

- BCOT permits the user to define exceptions to linearity, most of which are obvious. Because the biggest concern is inadvertently dropping good options, the nonlinear corrections are most important for those instances of positive synergy where having building blocks A and B is worth more than the sum of the separate worths of building blocks A and B. However, the nonlinear corrections can also assure that if neither building block A nor B has value in the real world unless both are present, then the quasi-linear approximation will enforce that.

- The options for which the effectiveness calculation are most seriously wrong are filtered out by the efficient-frontier methodology. If an option would “really” have effectiveness 0.2 but is approximated as having value 0.1 because of an error in the quasi-linear formula, no problem is created, because the option would lose out to far more-effective options anyway, especially if options effectiveness below a threshold of, say, 0.5 are deleted.

Table A.1 summarizes the set of nonlinear correction-factor types that we developed for use in BCOT. As indicated in the third column, the different corrections apply in different cases. The first corrects for positive synergy, the second corrects when one of the building blocks magnifies the value of another building block, the third is used when the linear approximation is poor, and the last is used to assure that all critical components are present.
<table>
<thead>
<tr>
<th>Type</th>
<th>Definition of Joint Effectiveness</th>
<th>Purpose of the Correction</th>
<th>Notional Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>$E(A) \delta(A) + E(B) \delta(B) + C \delta(A) \delta(B)$ [a]</td>
<td>Synergy (positive or negative)</td>
<td>Aircraft and SOF have effectiveness 0.25 and 0.2, respectively, against mobile missiles; when used together effectiveness is 0.9; C equals 0.45</td>
</tr>
<tr>
<td>Multiplication</td>
<td>${E(A) \delta(A) + E(B) \delta(B)} \times {1 + C \delta(A) \delta(B)}$</td>
<td>Force multiplier</td>
<td>Aircraft and SOF have effectiveness 0.1 and 0.3, respectively, against terrorist targets; when used together, effectiveness is 0.6; C equals 0.5; correction factor is 1.5</td>
</tr>
<tr>
<td>Substitution</td>
<td>${E(A) \delta(A) + E(B) \delta(B)} \times {1 + C \delta(A) \delta(B)}/(E(A) + E(B) + \epsilon)$</td>
<td>Linear form may be bad even as first approximation, in which case the correction is to replace the result with something better</td>
<td>Some effectiveness calculations should be products of components: for example, a new aircraft might increase penetration probability, and a new weapon might improve kill probability; what matters is the product of probabilities</td>
</tr>
<tr>
<td>Critical component check</td>
<td>$\delta(a) \delta(b)$</td>
<td>An option may have zero effectiveness unless a combination of building blocks is present.</td>
<td>A new aircraft or a new weapon might add no capability unless both were present, providing both penetration capability and kill capability</td>
</tr>
</tbody>
</table>

[a] The variable $\delta(A)$ (referred to in physics as the Kronecker delta) is 1 or 0, depending on whether the building block A is present. C is a constant. $E(A)$ is effectiveness with only building block A; $E(B)$ is that with only building block B; and the entire expression gives effectiveness if both A and B are present.
Although decidedly imperfect, the quasi-linear approximation can be very useful for a quick first cut. Subsequently, a more careful approach is needed, as discussed below. In other cases, the quasi-linear approximation is not even a good first cut.24

THE STANDARD CALCULATION AND THE BENEFITS OF DECOMPOSITION

In BCOT’s standard calculation of effectiveness, a much more complicated capability-area-specific algorithm must be specified. Correctly specifying effectiveness for the many composite options generated by BCOT is nontrivial because of the sheer number of options, which may be thousands or tens of thousands. No one wants to manually enter effectiveness for each of those options separately, even if it could be done without making errors. Fortunately, in any given problem, it will be possible to decompose the burden, the result of which is to require many fewer inputs. Further, the effectiveness values can usually be calculated from one or another theory, which means that the inputs required are only those for the parameters of that theory. Suppose, for example, that effectiveness can be expressed as the product of three probabilities, one each for penetration of defenses (both ingress and egress), finding the target, and destroying the target. The individual probabilities might depend on the building blocks possessed by a given option, the scenario, and the “case” (a combination of other parameter values for, e.g., level of air defense, size of the area in which targets are to be sought, and so on). If the case were determined by the values of five parameters, each of which could have two values, there would be 32 cases.

To illustrate the power of decomposition, suppose that there are N building blocks that may or may not be present in a given option. That implies $2^N$ options. If we needed to specify P items (e.g., probabilities) for each of the options in each of C cases in each of S scenarios, we would have a huge number of inputs $(2N)(C)(S)$. However, if the options fell into two groups G1 and G2 (perhaps aircraft- and missile-delivery options), each with N/2 relevant building blocks (i.e. for a missile option, it is irrelevant whether one has also bought aids for penetrating air defenses). The number of inputs required in this case is $2(2N/2)(C)(S)$. Table A.2 illustrates the significance of the difference.

24 An example is in the capability area of Ballistic Missile Defense. Effectiveness of the overall defense system is a highly nonlinear function of the effectiveness of its components. The capability of a layered system may, for example, be measured by the probability of intercepting an attacker. That is $1 - (\text{probability of leakage from layer 1})(\text{probability of leakage from layer 2})$. The individual probabilities may also involve products representing the probabilities that each of the critical components of the system (e.g., detection and tracking system, interceptor, and kill vehicle) works. A capabilities model for missile defense has recently been developed at RAND (Willis, Bonomo, Davis, and Hillestad, 2006).
In our prototype experiments using BCOT for the Global Strike problem, we used both the quasi-linear approximation and a better method of calculating effectivenesses that exploited the kind of decomposition indicated in Table A.2. Although we still had to input a lot of low-level data (thousands of items), the magnitude of work was tolerable (because so much copy-and-paste was possible), and the data could be reviewed visually.

Table A.2
Reducing Inputs by Decomposition

<table>
<thead>
<tr>
<th>Approach</th>
<th>Building Blocks for each of Two Groups</th>
<th>Scenarios</th>
<th>Cases</th>
<th>Items to Specify</th>
<th>Number of Inputs Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic</td>
<td>10</td>
<td>3</td>
<td>32</td>
<td>3</td>
<td>294,912</td>
</tr>
<tr>
<td>Decomposed</td>
<td>5, 5</td>
<td>3</td>
<td>32</td>
<td>3</td>
<td>18,432</td>
</tr>
</tbody>
</table>
Appendix B

SUBTLETIES IN THE CONCEPT OF NEARNESS TO THE EFFICIENT FRONTIER

IDENTIFYING POINTS ON OR NEAR THE EFFICIENT FRONTIER

Finding points on the efficient frontier is relatively straightforward. The concept of points near the efficient frontier is, however, ambiguous and problem-laden. The intention is to find points that are not much inferior to dominant points (those on the frontier itself), as measured by effectiveness or cost. Thus an option that has the same effectiveness as a dominant point and is only slightly more expensive would be considered near the frontier, as would an option with the same cost as a dominant point but slightly less effectiveness. This seems straightforward to implement with a simple logic comparing each successive option in a cost-ordered sequence with the previous dominant point. Geometrically, it corresponds to drawing a rectangle to the right and below that dominant point, with the dimensions of the rectangle corresponding to the nearness criteria (perhaps expressed as fractions of the dominant point’s values of effectiveness and cost). Figure B.1 shows this schematically. Dark points are dominant points; gray points are near the efficient frontier (or, more precisely, near the dominant points); and white points are neither on nor near and are therefore discarded.

Figure B.1
Points On or Near the Efficient Frontier, with Anomalies
ANOMALIES AND HOW TO DEAL WITH THEM

Anomalies

In Figure B.1, X and Y are dominant points—i.e., points on the efficient frontier itself. Consider, however, the other points, a, b,…g. The first, point a, has only slightly less effectiveness and cost than X and is therefore considered near the efficient frontier. Points b through f are actually as effective as X, but they are excluded (i.e., are not considered near the efficient frontier) because they are too much more costly than X, lying outside the rectangle. This, however, reflects a left-to-right perspective, not a global perspective. Point g is considered near the efficient frontier, but it is actually less effective and more costly than points b through f. Arguably, that seems anomalous and illogical.

Mathematical Avoidance of Anomalies

Mathematically, the way to avoid this problem is to regard a point as a next dominant point only if its effectiveness is significantly greater than that of the previous dominant point. Referring again to Figure B.1, point Y would no longer be a dominant point, and both Y and g would be discarded. However, if Y and g were a good deal more effective than shown in the figure, then both would also be a good deal more effective than points b through f. That is, the anomalies would no longer exist.

We have not yet implemented this feature in BCOT, but we plan to do so.

AVOIDING REDUNDANCIES

Redundancies

Another troublesome problem is that in some applications that include inexpensive building blocks, there may be redundant options that are near the efficient frontier but not worthy of being retained. Suppose that X is on or near the efficient frontier, has a certain effectiveness, and comprises building blocks 1, 4, and 8. Consider now another option, X’, that consists of building blocks 1, 4, 8, and 15, where building block 15 provides no benefit but adds a modest cost. Because building block 15 is inexpensive, X’ will also be near the efficient frontier, but we would prefer to have BCOT eliminate it automatically.

Algorithm for Deleting Redundant Options

The algorithm for eliminating redundancies is relatively straightforward using set-theory concepts. In simple English, the algorithm is
• Given a set of points (options), order the points first by ascending cost and second by descending effectiveness.
• Tag each point by whether it is or is not a dominant point (DP). The criterion can include a minimum increase in effectiveness from the previous dominant point.
• Note that if there are N DPs, then there are N – 1 ordered sets containing a DP and all points with costs equal to or more than the cost of that DP, but less than the cost of the next DP. Each point belongs uniquely to one of these N – 1 sets.
• Each point, then, is uniquely associated with a single DP (which may be itself) and an interval set.
• For each of the N – 1 interval sets, tag each point as being the interval’s DP, a superset of that DP or of an earlier non-DP point within the interval, or “other.”
• Define the set of non-redundant points near the efficient frontier as the subset of original points that meet the following criteria: The point must be a DP or must (1) have effectiveness within a specified multiple less than 1 of its DP’s effectiveness; (2) cost no more than a specified multiple greater than 1 of the DP’s cost; and (3) not be a superset of another point in the interval set.

The current version of BCOT eliminates points in an interval that are supersets of the interval’s dominant point. Other redundant points can be eliminated after manual inspection. In the Excel version of BCOT, eliminating these “other” redundancies is straightforward but tedious. For both versions of BCOT, enhancements will be necessary to fully automate the elimination of redundancies.
Appendix C

A GENETIC ALGORITHM APPROACH FOR IDENTIFYING GOOD CANDIDATE OPTIONS

Paul Dreyer

INTRODUCTION

The methods described in the main text of this report can be used in many, and perhaps most, cases of interest. However, combinatorial explosion could make them difficult or impractical if, e.g., the number of scenarios increased markedly. This appendix discusses a methodological hedge, one in which candidate options for in-depth assessment are identified by using a genetic-algorithm technique. Initial experiments with the technique have proved quite promising, so the technique could be pursued in more depth if the methods of the main text appear inadequate in a particular application. The discussion here, although brief, is intended to provide the essential concepts to enable follow-up, if needed, to have a good start. The discussion is very different in character from the remainder of the report. The first part is abstract, using the lingo common in genetic-algorithm work. The latter part uses the same example as that in Chapter 4.

EXPLAINING GENETIC ALGORITHMS

Genetic algorithms (GA) is a technique used to find optimal (or near-optimal) solutions to search and optimization problems. It mirrors the biological concept of natural selection, where the fittest organisms survive into future generations and reproduce, with the hope that the offspring of fit organisms will be as strong as or stronger than their parents. In addition, mutations that improve the fitness of organisms should be allowed to propagate, while those that weaken the organism should not survive in the population.

For GA, “genes” correspond to strings of bits of a fixed length, used to represent potential options. A fitness function defined on these bit strings determines the quality of each option. To begin a GA, the “gene pool” is seeded with a collection of random genes that span the entire search space. The fitness of each gene is computed, and a fraction of the fittest genes is retained for the next generation. To fill out the remainder of the gene pool, additional genes are created by “mating” fit genes. There are several different ways to implement mating between genes, including exchanging segments (e.g., if AA is a gene consisting of two substrings A and A’, and BB’ is another gene where the lengths of A and B and A’ and B’ are equal, the two offspring would be AB’ and BA’). Mating may also incorporate mutations into the offspring (flipping single bits in the string). The fitness of these offspring is determined, and the fittest genes survive into the next generation. Then the mating process is repeated. Usually, there is a stopping
criterion to end the process, either when the average or maximum fitness of the surviving gene pool reaches some threshold or the process is terminated after a fixed number of generations. Note that only a fraction of the entire set of options is searched, so there is no guarantee that the optimal solution will always be found.

IMPLEMENTATION OF GA FOR THE GLOBAL STRIKE PROBLEM

To simplify explaining how the genetic algorithm was implemented for the Global Strike problem, we introduce some notation: \( I_1, \ldots, I_n \) denotes the set of \( n \) investments under consideration, each with cost \( C_j \). An investment option corresponds to a bit string \( b_1 b_2 \ldots b_n \) of length \( n \), where \( b_j = 1 \) if and only if program \( I_j \) is a part of the option, and \( b_j = 0 \) otherwise. Here the \( b \)'s are the building-block options from which composite options are created.

Accordingly, the cost of the option corresponding to the bit string \( b_1 b_2 \ldots b_n \) is \( \sum_{j=1}^{n} C_j b_j \).

The fitness function for each bit string was determined by the performance of the associated investment option in three missions: \( M_1, M_2, \) and \( M_3 \). Let \( E_{ij} \) denote the effectiveness of program \( I_i \) in performing mission \( M_j \) (for our purposes, effectiveness was scaled to between 0 and 1). The effectiveness of an investment option \( b_1 b_2 \ldots b_n \) in performing mission \( M_j \) is \( E_j(b_1 \ldots b_n) = \max \{ b_j E_i \} \). Although this assumes that only one item may be used to perform a given mission and does not allow for combining items to increase effectiveness, that is not a general constraint: Other calculations of effectiveness for each mission may be used. The fitness function assumed for the bit string \( b_1 b_2 \ldots b_n \) is effectiveness over cost:

\[
F(b_1 \ldots b_n) = \frac{a_0 + a_1 E_1 + a_2 E_2 + a_3 E_3 + a_4 E_1 E_2 + a_5 E_1 E_3 + a_6 E_2 E_3 + a_7 E_1 E_2 E_3}{\sum_{j=1}^{n} C_j b_j}
\]

where the \( a_i \) are defined by the user, and \( E_i \) is shorthand for \( E_i(b_1 b_2 \ldots b_n) \). Note that, depending on the choice of constants, the fitness function can emphasize one mission above another or deem options that perform well in multiple missions as fitter than others. In addition, the genetic algorithm could be run several times with different weights on the effectiveness values to find more-robust investment options that score well over several fitness functions.

The process begins by populating the “gene pool” with a set of random bit strings, each representing an investment option. For each generation, the fitness function \( F(b_1 b_2 \ldots b_n) \) is computed for all of the bit strings representing new investment options, and all of the options are sorted by their fitness. Optionally, an option can be deemed unfit (and removed from consideration) if its total cost exceeds some threshold. The top half of the options (ranked by fitness) survive into the next generation. To produce offspring (which are also added to the next generation), the genetic algorithm typically involves operations such as crossover and mutation.
generation), the surviving bit strings are randomly paired off. A duplicate of each parent is created, then for each bit, these duplicates exchange bits with each other with probability 0.5. So, for example, if the parents were 11111 and 00000, and their offspring exchange their second, third, and fifth bits, the resulting offspring would be 10010 and 01101. Finally, each bit of each offspring is subject to a mutation (changing a zero to a one, or vice versa), with some probability. This mutation probability is usually set to be fairly low so as to not drift the characteristics of the offspring too far from those of their fit parents, but enough to get some genetic variation. The offspring are also checked to make sure they do not already appear in the gene pool and again, optionally checked to see if their cost exceeds some threshold. Where some of the offspring have not been retained in the gene pool due to cost constraints, additional random bit strings are added into the pool to return it to its initial size.

This process repeats for a fixed number of generations (determined by the user), and the top \(k\) options are displayed (where \(k\) is determined by the user). To relate this methodology to the efficient-frontier methodology used in the rest of this report, the user could take the top \(k\) options and generate an efficient frontier restricted to this set of good investment options and then subject the options at or near the efficient frontier to a deeper analysis across multiple measures and display the results using PAT.

### A Simple Example of GA for the Global Strike Problem

Following a simple example through a generation as is done in Chapter 4 may make the GA methodology clearer. We will have the same four building blocks: aircraft (A), missile (M), SOF (S), and weapon (W) enhancements. Instead of using 0s and 1s to denote our bit strings, we will let upper- and lower-case letters denote whether a building block is or is not contained in the investment option, respectively. So, AMSW corresponds to an investment option that includes all four building blocks, while amsw is the option that includes none of the building blocks. This is slightly different notation from that used in Chapter 4, only because the bit switching is easier to see when all four characters are included (so, for example, AMSW and amsw could mate to produce AMsw and amSW). Note that in this case, there are only 16 possible investment options, which is well within the ability of the BCOT methodology (or the back of an envelope, for that matter).

We will assume the cost and mission-effectiveness values in Table C.1 for our example, with the effectiveness of an investment option being the sum of the maximum values of mission effectiveness for each mission of the building blocks included in the option divided by the option’s cost.
Table C.1
Effectiveness and Costs Assumed in Example

<table>
<thead>
<tr>
<th>System</th>
<th>Mission A</th>
<th>Mission B</th>
<th>Mission C</th>
<th>Cost ($ millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft (A)</td>
<td>0.3</td>
<td>0.6</td>
<td>0.2</td>
<td>30</td>
</tr>
<tr>
<td>Missile (M)</td>
<td>0.4</td>
<td>0.1</td>
<td>0.6</td>
<td>40</td>
</tr>
<tr>
<td>SOF (S)</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>15</td>
</tr>
<tr>
<td>Weapon (W)</td>
<td>0.1</td>
<td>0.4</td>
<td>0.5</td>
<td>20</td>
</tr>
</tbody>
</table>

Since our potential set of genes is so small, we will populate the initial gene pool with only four random genes: AMsw, aMsW, AmsW, and aMSw. The first of these (AMsw) corresponds to an investment option that has only two of the building block options (A and M). Many of the possible combinations of building blocks are not present in this initial gene pool.

Table C.2
An Initial “Gene Pool”: Four of the Possible Composite Options

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AMsw</td>
<td>0.4</td>
<td>0.6</td>
<td>0.6</td>
<td>1.6</td>
<td>70</td>
<td>0.0228</td>
</tr>
<tr>
<td>aMsW</td>
<td>0.4</td>
<td>0.4</td>
<td>0.6</td>
<td>1.4</td>
<td>60</td>
<td>0.0233</td>
</tr>
<tr>
<td>AmsW</td>
<td>0.3</td>
<td>0.6</td>
<td>0.5</td>
<td>1.4</td>
<td>50</td>
<td>0.0280</td>
</tr>
<tr>
<td>aMSw</td>
<td>0.4</td>
<td>0.3</td>
<td>0.6</td>
<td>1.3</td>
<td>55</td>
<td>0.0236</td>
</tr>
</tbody>
</table>

Table C.2 based on data in Table C.1, shows that the bottom two investment options have the highest fitness. Thus, they progress to the next generation. However, the GA methodology also generates some variants of those options—i.e., it considers some composite options with different building-block combinations. In GA lingo, the two options “produce a pair of offspring which are also added to the gene pool, with a mutation possibly taking place as well.” In our example, the initial offspring of AmsW and aMSw might include Am+Sw, or AmSw, and aMsW. If a mutation occurred in the second offspring, a SOF building block might appear (the s would become an S). Thus, the offspring would be AmSw and aMSW, which mutate to AmSw and aMSW. As a result, new options not in the original set would be considered. Over generations, a great many options would be considered, sampling the space of the possible combination options. As has often been observed about GA methods, it is not intuitively obvious that this sampling will be “good,” i.e., that it will eventually generate the best results. However, in practice—over quite a wide range of problems—GA methods have been found to produce remarkably good results.

Figure C.1 describes the GA method graphically. Step 1 has a starter set of genes (e.g., investment options), with 1s and 0s indicating the presence and nonpresence, respectively, of a...
given trait—in this case, a specific building-block option. Step 2 illustrates for the leftmost option that each option’s effectiveness is measured. In step 3, the worst half are discarded. In step 4, the various initial genes (investment options) are mated, as illustrated for the leftmost two of step 1. In step 4’s top center, we see an offspring that has half of its building-block options (1, 3, 4, and 7) from its red parent and half (2, 5, 6, and 8) from its blue parent. Another offspring (bottom center) is formed similarly, but with the choices reversed. That is, it inherits the presence or nonexistence of traits (building blocks) 1, 3, 4, and 7 from its blue parent and of 2, 5, 6, and 8 from its red parent. In the last part of step 4, mutation may (or may not) occur; this is illustrated in this case by the second building block in the top offspring and the fifth building block of the lower offspring being turned off (light red and light blue, respectively).

Figure C.1
A Schematic Representation of the Genetic Algorithm

Goal: Given a set of $N$ components and a means of evaluating the (cost-) effectiveness of a subset of those components, create a set of investment options to investigate further in PAT.

PAT GA Screener uses a genetic algorithm to search through the set of possibilities to find candidates for further consideration.

**Step 1:** Generate a random pool of potential investment options.

```
1 0 1 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0
1 1 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 0
```

**Step 2:** Determine the effectiveness (fitness) of each investment option.

```
1 1 0 1 0 1 1 0
Fitness evaluator
Fitness score
```

**Step 3:** Toss out the lowest half scoring options, pair off and mate the upper half, producing two new offspring options.

**Step 4:** “Mutate” the offspring options, and return to Step 2.

```
1 0 1 1 0 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0
1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 0 0 1 1 1
```
Appendix D

CHANGING BUILDING BLOCKS OR SCENARIOS

The Building Blocks to Composite Options Tool (BCOT) is very flexible, but care must be taken when changes are made. Many changes such as changes to a node’s name or identifier, will ripple through any Analytica program automatically, but some will not. Fortunately, Analytica provides systematic ways to identify all of those side effects and make changes accordingly. The program need not get out of control, and taking some care on these matters can avoid troublesome, costly errors.

ADDING OR CHANGING BUILDING BLOCKS

If more building blocks are added, then BCOT will run, but the answers will be incorrect unless appropriate data is added, notably:

- Cost and effectiveness data
- Data on which employment options can use which building blocks.

The precise nodes requiring adjustment will change as BCOT evolves. Fortunately, it is easy to identify all those nodes. By opening the Building Blocks node, one finds an interface window with not only fields for title, identifier, definition, and description (self-documentation), but also a list of inputs and outputs. That list is guaranteed to be comprehensive. Changes need to be made in some or all of the output nodes.

Merely changing the names of building blocks (e.g., for clarity) requires no further adjustments. Deleting a building block requires no further changes. Substituting one building block for another will require the same changes of data as those required when adding a new one. Changing the order of building blocks requires changing the data accordingly.

ADDING SCENARIOS

In BCOT, an option is measured by its effectiveness in each scenario or by a combined effectiveness across scenarios. Changes in the scenarios used may or may not require corresponding changes elsewhere in BCOT. By and large, the rules are the same as those for adding, changing, or deleting building blocks, although the list of scenarios is an “index variable” (in Analytica-speak) and is found in the top-level module, Indices and Functions. Nonetheless, there are important special issues.

One subtlety involves connections with the index “Focus” and various output nodes. If a new scenario is added, it is necessary also to respecify the weights to be used for different focus cases. That should come naturally if the user follows the rules about checking each output node
of scenarios as described above for building blocks, but the point is worth highlighting. Similarly, if the user changes scenarios, changes will likely be needed in the nodes that specify what is to be varied before the union of options that are near the efficient frontier in at least one case of interest can be generated. The user should find the Non-Unique Union of Options module and determine there what needs to be changed. For example, if a new scenario has been added, that may affect what focus cases are needed, not just how they are weighted. Further, it may then affect the combination of focuses that should be considered when generating the union of relevant options.

The second subtlety is that when a new scenario is introduced, it is likely that it should involve one or several new parameters, such as strategic warning, detection capability for a new class of target given a particular building block, etc. If so, then new input parameters will need to be defined, and effectiveness data in the interface will need to be updated.
Appendix E

CHANGING LIST NAMES (SCENARIOS, FOCUS, ETC.)

As noted in Appendix D, if the user changes any of a number of nodes, e.g., edits the name of a scenario or force-employment mode, it is necessary to check all of the output nodes affected and to make the required adjustments. Building Blocks, Employment Modes, Scenarios, and Focus are the principal examples; they are defined by lists (e.g., Scenario A, Scenario B, Scenario C). Some calculations within BCOT refer to specific elements of those variables. For example, a calculation might include the expression Effectiveness [Scenario='Scenario A'], where the item in quotes within the bracketed phrase identifies an item within a particular list. Thus, the user is looking at a particular element in a particular dimension of an array. An equivalent expression might use the operator Subscript. If the name of such a list item is changed in the definition of the list variable, e.g., by changing “Scenario A” to “Scenario New A,” the expression Effectiveness [Scenario='Scenario A'] will fail, causing a so-called null error. Thus, everywhere “Scenario A” appears, it must be changed to “Scenario New A”.

Such changes can be made by simply knowing the program well or by systematically looking in all of the nodes that refer to the list variable (its output nodes, as specified in its definitions pane) and making corrections as necessary. This can be tedious, but it is important.

An alternative approach can be regarded as a trick, but it is useful: Analytica files can be opened and edited by word-processing programs. An easy way to change names appearing in lists, then, is to open such a program, use the search-and-replace functionality carefully, save, and reopen in Analytica. Global replaces are always dangerous, so the user should search item by item and assure that the replacement is appropriate.

---

25 To avoid unintended changes being made by the word-processing program (especially with Microsoft Word, which notoriously makes many automatic “corrections” unless these options are turned off), we recommend using a plain-text program. Notable examples are TextPad or NotePad for Windows and TextEdit and SimpleText for the Macintosh (although we use the Windows version of Analytica, much of our work has been done on Macintosh platforms, where we keep the files on the Macintosh side rather than within the Virtual PC® or Parallels® volume.
Appendix F

CHANGING PARAMETERS

Adding more parameters as inputs to those already in the model will require providing data input for different values of these parameters; updating the list of parameters whenever a new parameter is used as an input variable (e.g., the list of parameters over which the union of efficient options is taken), and making major programming changes when increasing resolution in certain parts of the model, such as the effectiveness module. For example, we may want to calculate effectiveness values for various combinations of force elements for different employment modes bottom-up, rather than inputting them, as is currently done. Examples of additional parameters might be the quality of opponents’ air defenses, how well the targets are protected, and so on. In addition to specifying the inputs for different values of these parameters, the user must define new intermediate variables and update the values of existing variables to implement such bottom-up calculation.

A word of caution is in order here. Although Analytica’s functionality allows the modeler to easily track how changes propagate down the stretch, certain features of the current version of the Effectiveness module require extra attention. Namely, the different combinations of contributing force elements in a given employment mode are calculated in Analytica—i.e., as an Analytica output—rather than input in advance. This is done to accommodate future possible additions of building blocks to the model. Effectiveness scores are then input for each of these combinations, making the respective variable an output, not an input node, in Analytica parlance. The computational mechanics of that in Analytica are somewhat complicated, so the higher-resolution modification should take note of it and make necessary adjustments in its code to ensure compatibility. Alternatively, should a user implementing changes opt for a simpler code with force-element combinations provided as inputs, the current low-resolution definitions should be changed accordingly. Thus, we recommend that to ensure that the existing capabilities are not lost, the person implementing such changes be very comfortable with the model and save the existing version before introducing changes to it.

---

26 As to the output variables, however, Analytica’s array-abstraction capability should handle those nicely and ensure that the model runs properly. In addition, the user-built functions (such as PositiveSubset) are specified so that it can handle additional parameters easily.
Unfortunately, even programs intended to be relatively simple and high-level tend to become more complicated with time as they are generalized and refined. As this happens, it is often necessary to use more-sophisticated features of the programming language (or even to shift to a general-purpose language such as C++). Analytica has a rather powerful programming language, which makes it possible to do some things with astonishing simplicity and clarity, but we pushed the boundaries of what can be done easily in Analytica as we developed BCOT, so some elaboration is appropriate on items that can cause trouble for a user who is not an expert programmer.

ARRAY OPERATIONS

Many Analytica operations can be readily understood by anyone with knowledge of elementary linear algebra. For example, one may be dealing with a concept represented by $C$, which is just a product of $A$ and $B$. If $A$ and $B$ are linear arrays (vectors), $C$ is as well and can be defined simply as $A \times B$. The scalar product or “dot product” of linear algebra is then denoted in Analytica as $\text{Sum}(A \times B, \text{index}_\text{name})$ if the arrays $A$ and $B$ are “indexed” by something called $\text{index}_\text{name}$.

In more-advanced modeling, it may be necessary to do relatively complex array manipulations that are unfamiliar to users who are rusty at linear algebra. These include resorting the elements of an array, shifting rows or columns around in an array, “reducing” an array so that the result shows only some of the rows or columns of the original, and so on. In Excel, many such operations can be done with simple menu commands, cut-and-paste, or insertions and deletions. However, these operations leave no history: One starts with a block of rows and columns, manipulates them manually in various ways, and ends up with something new. There is, however, no record of what was done. The procedure can be written down carefully, so that the results are reproducible, but this is notoriously tricky. Further, one cannot “see” the underlying mathematics or meaning in a long laundry list of procedures. In an Analytica model, the prescription for the sequence of operations can be expressed mathematically, which is satisfying and clear—but only if one understands the syntax of the array-manipulation language. It is not our purpose here to provide a tutorial on programming in Analytica, but we emphasize to technically educated readers the feasibility of brushing up on a few functionalities by reading the users manual, rather than imagining that only professional computer programmers can understand. The functionalities that we used extensively are listed in Table G.1. The middle
column refers to built-in Analytica operators or functions that are defined in the *Analytica User Guide*. The last column mentions some custom functions that we developed specifically for BCOT. They are described in the next section.

### Table G.1
**Important Operations in Analytica**

<table>
<thead>
<tr>
<th>Functional Need</th>
<th>Analytica Operation</th>
<th>Special BCOT Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete rows with values of 0</td>
<td><em>Subset</em></td>
<td><em>PositiveSubset</em></td>
</tr>
<tr>
<td>Select only rows that meet some criterion</td>
<td><em>Subscript</em>: extracts only rows such as the row with name “alpha”</td>
<td></td>
</tr>
<tr>
<td>Select only rows in certain positions in a list</td>
<td><em>Slice</em></td>
<td></td>
</tr>
<tr>
<td>Select only rows that contain specific values</td>
<td><em>Subindex</em>: extracts only rows whose cells equal specified values (limitation: selects only the last one of two identical rows)</td>
<td></td>
</tr>
<tr>
<td>Sort rows in the list in ascending (descending) order of number values.</td>
<td><em>SortIndex</em>: allows reshuffling rows so that the numbers in the respective cells are in ascending order</td>
<td></td>
</tr>
<tr>
<td>Select only unique rows when two or more rows are identical</td>
<td><em>Unique</em>: extracts only those rows that are different from each other</td>
<td></td>
</tr>
<tr>
<td>Create some union of cell values in a multidimensional array</td>
<td><em>MDArrayToTable</em>: converts a multidimensional array into two-way table with each row containing the original cell values in the last column and different combinations of values of other parameters in all other columns</td>
<td><em>UnionNonUnique</em></td>
</tr>
<tr>
<td>Marking items</td>
<td><em>If A = &lt;Some condition&gt; Then Marker_value</em></td>
<td></td>
</tr>
<tr>
<td>Numbering Rows in an array A</td>
<td><em>Cumulate (1, A)</em></td>
<td></td>
</tr>
<tr>
<td>Counting entries in the list</td>
<td><em>Sum</em>: values in the list can be counted by dichotomizing them, using some criteria and taking the sum of all entries</td>
<td><em>Size</em>: counts all cells in a multidimensional array; if the count is needed for only some portion of an array, some array-reducing function above can be used first</td>
</tr>
</tbody>
</table>

### SPECIAL BCOT ARRAY-MANIPULATION FUNCTIONS

The user-defined functions that we defined for BCOT are summarized in Table G.2 and then discussed one by one.
Table G.2  
BCOT-Specific Array Functions

<table>
<thead>
<tr>
<th>User-Defined Function</th>
<th>Operation</th>
<th>Functional Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnionNonUnique(A)</td>
<td>Collects all values in a multidimensional array in one list</td>
<td>Intermediate step for calculating the union of all options that are near the efficient frontier in at least one case</td>
</tr>
<tr>
<td>ArrayMaximum(A)</td>
<td>Returns the maximum of all cell values in an array of any dimension (i.e., maximum over all indices); allows application of Analytica function Max() to a multidimensional array</td>
<td>Used as a part of PositiveSubset() to insure a multidimensional array of a filtered subset of the complete set of options is rectangular and contains as few cells as possible.</td>
</tr>
<tr>
<td>PositiveSubset(A,I)</td>
<td>Selects a subset of positive values from values in each of the one-dimensional slices of a multidimensional array A defined along an index I</td>
<td>Applies Analytica function Subset() to multidimensional arrays</td>
</tr>
<tr>
<td>Stringvector(N,I)</td>
<td>Generates an array consisting of strings of bits for building blocks (elements of I) corresponding to the power set of options when N is specified as a list of position numbers of options in the power set</td>
<td>Intermediate to generate a power set of options; also generates all possible combinations of force elements relevant to a given force-employment mode</td>
</tr>
<tr>
<td>String_cats(N,I)</td>
<td>Replaces in the array generated by Stringvector function 1s by the names of corresponding building blocks and 0s by empty cells and for each row joins these text values across building-blocks columns.</td>
<td>Generates the list of names of options in the power set of an option; also all possible combinations of force elements relevant to a given force-employment mode</td>
</tr>
</tbody>
</table>

*a This definition for the user-specified function was suggested by Lumina.

**UNIONNONUNIQUE(A)**

**Operation**

Collects all values in a multidimensional array in one list. Example: The values in a multidimensional array (A) are the names of investment options that were pre-screened for nearness to the efficient frontier under all different cases (scenarios, combinations of values of multiple parameters). We need to put all these options in one list with a view to deriving the union of all options that are near the efficient frontier in at least one case. Since the same options can be identified as being near the efficient frontier under various assumptions, the same option can be listed multiple times—hence the name NonUniqueUnion(A).
**Function in BCOT**

Intermediate step for calculating the union of all options that are near the efficient frontier in at least one case.

**Parameters**

A = A multidimensional array of values in Analytica. Function NonUniqueUnion specifies it as ArrayType to tell Analytica that it must be an array.

**Definition and Functions**

The definition and functions of UniqueNonUnique are given in Table G.3
Table G.3  
Definition and Functions of UnionNonUnique(A)

<table>
<thead>
<tr>
<th>Steps</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Rows:=1.Size(A);</td>
<td>Creates new rows that contain integers from 1 to the number of cells in an array A</td>
<td>Creates as many rows as there are cells in an array: when a multidimensional array is transformed into a two-way table, values in each cell of an array must be placed in one column of every row</td>
</tr>
<tr>
<td>Index IndNames:=Indexnames(A);</td>
<td>Generates all the index names of a multidimensional table</td>
<td>Creates all the column names except the one for the last column of a two-way table a multidimensional table is to be transformed into</td>
</tr>
<tr>
<td>Index Cols:= Concat(IndNames, [“OPTIONS”]);</td>
<td>Generates a new index Cols by appending the name “OPTIONS” to the left of the list of index names of A</td>
<td>Creates the names of the columns for a two-way table where the last column “OPTIONS” must include the names of options—all the values in the cells of the array A</td>
</tr>
<tr>
<td>Var x:= MDArrayToTable(A,Rows, Cols);</td>
<td>Creates a two-way table rows vs. cols x from the array A in which all the values of A are placed in the last column and different combinations of index values are placed in other columns</td>
<td>Collects all the cell values of A–policy options—in one column</td>
</tr>
<tr>
<td>Var y:=x[Cols=“OPTIONS”];</td>
<td>Create a list y that selects only the column of x named “OPTIONS” and discards all other columns</td>
<td>Has a list of all entries of A</td>
</tr>
<tr>
<td>Index SubsetIndex:=Subset(y&lt;&gt;“”);</td>
<td>Creates an index SubsetIndex that sets aside those rows of y that do not have empty cells</td>
<td>Discards the empty cells in the list of all entries of A</td>
</tr>
<tr>
<td>Var ReturnUnion:= Subscript(y,y.Rows,Subset Index);</td>
<td>Defines variable ReturnUnion that is a list of non-empty cells of y</td>
<td>Discards the empty cells in the list of all entries of A</td>
</tr>
<tr>
<td>Index UnionOfOptions:= 1.Size(ReturnUnion);</td>
<td>Defines a new index UnionOfOptions that contains integers ranging from 1 to as many as there are entries in ReturnUnion</td>
<td>Marks the retain values–option names—of A</td>
</tr>
<tr>
<td>Array(UnionOfOptions,Return Union)</td>
<td>Reindexes a list ReturnUnion by the index UnionOfOptions</td>
<td>Creates a numbered list of non-empty values of A–option names</td>
</tr>
</tbody>
</table>
ARRAYMAXIMUM(A)

Operation

Returns the maximum of all cell values in an array (A) of any dimension (i.e., maximum over all indices). Allows application of Analytica function Max() to a multidimensional array,

Example:

The values in a multidimensional array (A) are the effectiveness scores for various investment options for all cases (scenarios). We ultimately want to select the data for only a subset of these options for which effectiveness scores are greater than some prespecified level. But for every case, the number of options in the subset will be different and less than the original number. Analytica’s array-abstraction capability works only for rectangular arrays. The ArrayMaximum() function defines the maximum number of options across all cases in a subset of options retained after some sort of filtering operation. We set the number of “rows” for options in the resulting array to the maximum number of options in the subset to ensure that the resulting array is rectangular and contain as few cells as possible; the cases that have fewer options than the maximum will have empty cells.

Function in BCOT

Ensures a multidimensional array of options constituting a filtered subset of the complete set of options that is rectangular and contains as few cells as possible when a function PositiveSubset() is applied to it.

Parameters

A: A multidimensional array of values in Analytica. Function ArrayMaximum specifies it as ArrayType to tell Analytica that it must be an array.

Definition and Functions

The definition and functions of ArrayMaximum(A) are given in Table G.4.
Table G.4  
Definition and Functions of UniqueNonUnique(A)

<table>
<thead>
<tr>
<th>Steps</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var x:=UnionNonUnique(A);</td>
<td>Defines variable x that collects all cell values of an array A into one list</td>
<td>Reduces a multidimensional array to a 1-D array thus making possible the application of function Max()</td>
</tr>
<tr>
<td>Max(x,x.UnionOfOptions)</td>
<td>Selects the maximum value from the list x</td>
<td></td>
</tr>
</tbody>
</table>

POSITIVESUBSET(A,I)

Operation
Selects a subset of positive values from values in each of the one-dimensional slices (lists) of a multidimensional array A defined along an index I.\(^{27}\) Example:
The values in a multidimensional array (A) are the effectiveness scores for various investment options as specified in index I for all cases (combinations of values of multiple parameters). We need to select the data for only a subset of these options for which effectiveness scores are greater than some prespecified level.

Function in BCOT
Applies Analytica function Subset to more than 1-D arrays. Subset can be applied only to lists.

Parameters
A: A multidimensional array of values in Analytica. Function PositiveSubset specifies it as ArrayType to tell Analytica that it must be an array.
I: A list of text or number values in Analytica. Function PositiveSubset specifies it as IndexType to tell Analytica that it must be an index.

Definition and Functions
The definition and functions of PositiveSubset(A,I) are given in Table G.5.

---

\(^{27}\) Generalization: Select a subset of values of a specific type (not only positive) in A. In this general case, the selection criterion expressed as an (in)equality should also be specified (the example of this syntax is PositiveSubset(A>=k)). The criterion converts the original array into 1s and 0s, depending on whether the criteria is fulfilled, and then applies PositiveSubset() to select positive values.
Table G.5
Definition and Functions of PositiveSubset(A,I)

<table>
<thead>
<tr>
<th>Steps</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var NumScreenedOptions:= ArrayMaximum(Sum(If A&gt;0 Then 1 Else 0,i));</td>
<td>Expression “If A&gt;0 Then 1 Else 0” converts the positive values in the array k into 1s and the negative values into 0s. Sumation sums the 1s across I</td>
<td>The ArrayMaximum() function takes the maximum of the resulting sums across all cases</td>
</tr>
<tr>
<td>Derive the maximum possible length of a positive subset for all cases 1.NumScreenedOptions;</td>
<td>Derive the maximum possible length of a positive subset for all cases</td>
<td>Index ScreenedOptions:=</td>
</tr>
</tbody>
</table>

Stringvector(N,I)

Operation
Generates an array consisting of strings of bits for building blocks (elements of I) corresponding to the power set of options when N is specified as a list of position numbers of options in the power set. Example:

If there are n building blocks, there must be $2^n$ possible combinations of building blocks in composite options, i.e., those consisting of combinations of building blocks in the power set of options, or $2^{n-1}$ if we exclude the combination with no building blocks. There will be $2^{n-1}$ strings of bits of length equal to the number of building blocks for each combination of building blocks. If N is specified as a list of integers to mark rows for each combination of building blocks and I is the list of names of the building blocks, then the Stringvector(N,I) function generates a table in which each row has different combinations of 1s and 0s in columns corresponding to building blocks.

Function in BCOT
Serves as an intermediate step to generate a power set of options. Also used to generate a list of all possible combinations of force elements relevant to a given employment mode.

Parameters
N: A list of numbers. Function StringVector specifies the values in the list as Numeric to tell Analytica that only numbers are accepted as values in the list. In BCOT, the numbers are typically the position numbers of all options in the power set of options.
I: A list of numeric or textual values in Analytica. Function StringVector specifies it as IndexType to tell Analytica that it must be an index. In BCOT, the I index is a list of building blocks.

**Definition and Functions**

The definition and functions of StringVector(N,I) are given in Table G.6.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mod(Floor(N/2^(cumulate(1,I)-1)),2)</td>
<td>A trick to generate strings of 1s and 0s for all combinations of the values of I. The result is index by I.</td>
<td>Generates an array consisting of strings of 1s and 0s for all combination of building blocks.</td>
</tr>
</tbody>
</table>

**STRING_CATS(N,I)**

**Operation**

In the array generated by StringVector function replaces 1s by the names of corresponding building blocks and 0s by empty cells and for each row joins these text values across building blocks (columns).

Example: If N is specified as a list of integers to mark rows for each combination of building blocks and I is the list of names of the building blocks, then the previous StringVector (N,I) function generates a table where each row has different combinations of 1s and 0s in columns corresponding to building blocks. Then the String_cats function replaces 1s by the names of corresponding building blocks and 0s by empty cells and joins these text values across building blocks to generate the names of options in the power set.

**Function in BCOT**

Generates the list of names of options in the power set of option as well as all possible combinations of force elements relevant to a given force-employment mode.

**Parameters**

N: A list of numbers. Function String_cats specifies the values in the list as Numeric to tell Analytica that only numbers are accepted as values. In BCOT, the numbers are typically the position numbers of all options in the power set of options.

I: A list of numeric or textual values in Analytica. Function String_cats specifies it as IndexType to tell Analytica that it must be an index. In BCOT, the I index is a list of building blocks.
**Definition and Functions**

The definition and functions of String_cats(N,I) are given in Table G.7.

**Table G.7**

<table>
<thead>
<tr>
<th>Steps</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>If StringVector(N, I) Then I Else &quot;&quot;&quot;, Labels</td>
<td>Replaces 1s and 0s in the array produced by the StringVector function by the names of building blocks and empty cells, respectively</td>
<td>Joins the text values across index I (e.g., building blocks)</td>
</tr>
<tr>
<td>Join( If StringVector(N, I) Then I Else &quot;&quot;&quot;, I)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Join( If StringVector(N, I) Then Labels Else &quot;&quot;&quot;, Labels)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix H

EXCEL-BASED GRAPHICS FOR BCOT

The current version of Analytica (Analytica 3) has much more-limited graphics capability than Excel, the programming system in which we first developed the ideas for BCOT. Analytica 3 provides the option, on a node-by-node basis, of displaying results in Excel rather than Analytica.\(^{28}\) It does so using Microsoft’s OLE mechanism. For a few cases (i.e., for a few desired output nodes), we used this option, because we were unable to construct visually useful graphics in Analytica itself. For example, this was necessary for the chart plotting cost ranges where individual building blocks become part of the solution.

A few cautions are necessary for users of BCOT, however. First, the Excel graphic generated automatically from Analytica will often not have the format desired, or even the chart type desired. It can take some time to manipulate the graphic accordingly. Second, if changes are subsequently made to some of BCOT’s inputs, the Excel output will change format again, necessitating more tedious manipulations.

Because of these issues, we recommend that users avoid the mechanism described above and that they instead use the approach\(^{29}\) of (1) exporting the relevant Analytica table of data, (2) importing that data into Excel, (3) formatting the Excel graphic as desired, and (4) establishing a link that will update the Excel graphic automatically if the Analytica input changes. The resulting graphic will then be stable. The disadvantage of this approach is that users must be careful to keep the relevant Analytica and Excel files in the same folder or otherwise avoid losing linkages—a classic problem understood by long-time programmers.

---

\(^{28}\) We are told by Lumina Corp. that Analytica 4, now in beta testing, will have a much improved graphics package and will no longer include the linkage to Excel.

\(^{29}\) Instructions are provided in the Analytica 3 user manual under “Linking Analytica Results to Other Applications.”
BIBLIOGRAPHY


