USING TRACKING COLUMNS TO IMPROVE OPTIMIZATION WITH A GENETIC ALGORITHM

Gavin T. Byrnes
Decision Lens, Inc.
Arlington, VA USA
E-mail: gbyrnes@decisionlens.com

Abstract

It has been established that a genetic algorithm can be used in linear programming to optimize within an AHP framework. This optimization process frequently reveals a bias towards cheap projects (even if they are poor-performing), which hurts its value in practice. Tracking columns and multiple pools are two methods of combating this bias. Tracking columns allow for more control over optimization by adding or removing constraints on the number of alternatives funded overall, from different groups, or to require certain additional thresholds for more realism. Multiple pools can be used to earmark resources for specific types of projects, allowing for a more balanced portfolio.

Keywords: genetic algorithm, linear programming, tracking columns, optimization

1. Introduction

AHP models can be used in budget allocation to construct value scores which are then optimized into a portfolio constrained by the resources available using a genetic algorithm based on concepts of linear programming. The alternatives are assigned rating scores developed through a typical AHP process involving prioritization of relevant criteria and creating rating scales for these criteria on which the alternatives are evaluated. The genetic algorithm then maximizes the total value attainable using the given funding structure. This approach is hampered in real-world value by a frequent bias towards cheaper and less effective projects; this paper presents an example of this bias, along with analysis of a few potential solutions.

The model used was created for use in water management by a department of public utilities. Names of projects and some other identifying details have been changed. The purpose of the model was to first evaluate projects in an AHP model and then maximize portfolio score based on the costs and values of the projects and the available money.

2. Literature Review

The AHP model and genetic algorithm in this paper come from the software of Decision Lens, Inc., which implements Thomas Saaty's AHP in a web and desktop-based platform which includes a genetic algorithm for resource allocation.

The optimization problem at hand is a linear programming problem; given alternatives with value scores and costs, and a budget of resources, maximize the total sum of value that can be achieved using the resources to fund the alternatives. This type of multivariate linear programming problem is common in literature in the past several decades.

The genetic algorithm used is described in papers by William Adams; it constructs seeds of potential portfolios and navigates through potential viable solutions to a linear programming problem, "turning" alternatives on and off by determining whether or not they are funded, and gradually constructing an optimal or near-optimal portfolio.

Most optimization processes built using this type of genetic algorithm have a bias toward cheap projects during optimization. This occurs because the linear construction of portfolio value means that portfolio value is additive. Since value in the Decision Lens framework of AHP is constrained to a 0 to 1 scale, there is usually a wider spread of costs than value scores, so a cost-benefit analysis favors projects with lower costs even if they have lower scores.

3. Hypotheses/Objectives

This paper aims to describe the process of linear optimization using genetic algorithms, allowing for tracking column constraints, logical constraints, and pool constraints. It will demonstrate the hypothesis that pure linear optimization favors cheap projects over expensive ones (even if the expensive projects are better) and present and analyze some methods of addressing this bias.

4. Research Design/Methodology

The model used is an AHP model with 56 alternatives and two or more pools, depending on the optimization scenario. It is based on a real-world model used by a department of public utilities; the alternative names have been changed and some of the criteria have been modified to protect the security of the original data. The costs of the alternatives range from very cheap projects in the \$100K range to expensive ones \$50M and higher. The basic model is broken into two pools, referred to as C Pool and D Pool (not their real names). C Pool contains 50% of the sum of the costs of projects in C Pool; D Pool contains the same percentage for its projects. In some of the scenarios created to combat bias toward cheap projects, the pools are broken down further by cost (0-\$1M, \$1M-\$10M, and \$10+M), in all cases with the pool containing 50% of the funds needed.

Some of the scenarios also contain tracking columns, defined as a metric that is neither an AHP criterion nor a cost but nevertheless affects the optimization. In this case, we track alternatives based on three categories (high cost, medium cost, and low cost) and may have upper or lower bounds on how many in each category can be funded.

The following table shows the projects funded by cost and value by regular linear optimization using a genetic algorithm, where low-cost refers to projects costing $0-3M, medium is $3-15M, high cost is $15-100M, and projects are grouped by value into approximately the top third, middle third, and lower third.

# Of Projects Funded by Cost and Value

	Total (Cost)	Total (Value)	Funded (Cost)	Funded (Value)
Low (Cost/Value)	17	18	17	17
Med (Cost/Value)	23	19	23	18
High (Cost/Value)	16	19	8	13

Regular optimization results for this model do not adequately serve to inform a real life situation. Only eight projects are not funded; six of these are in the upper third by value! All the cheap projects are funded, including some with very low scores. In the real world, funding a vast number of bad projects is rarely preferable to funding a smaller number of good projects. To combat this issue we turn to analysis of tracking columns and pool breakdowns.

5. Data/Model Analysis

There are several ways to use tracking columns to constrain the results to better match reality.

I. Limit the number of projects funded in total
II. Require quotas of projects by value or cost
III. Create different funding pools for large/medium/small projects

I. Limit the number of projects funded in total

In addition to overvaluing cheap projects, the original method of optimization also does not take into account questions of the efficiency of pursuing so many projects. One way to use tracking columns is quite simple; assign a value of "1 project" to each project and use that tracking column to limit the number of projects. This is also likely to give a more realistic and diverse portfolio, with a smaller number of projects and a better balance between larger and smaller projects. Running the same optimization but with a requirement that no more than 25 projects be funded gives the following breakdown by value/cost:

# Of Projects Funded by Cost and Value

	Total (Cost)	Total (Value)	Funded (Cost)	Funded (Value)
Low (Cost/Value)	17	18	6	1

Med (Cost/Value)	23	19	11	9
High (Cost/Value)	16	19	8	15

The portfolio value is superficially lower, but two more high-value projects can be funded than before and there are significantly fewer real-life resources and overhead being used on low value projects. The requirement for number of projects funded is not set in stone, of course; if using the tracking column cuts the portfolio too far down the other way, we could allow a higher number of projects to be funded, in this case 30 or 35, keeping some of the high-value projects we've gained while adding a few low or medium cost projects back into the portfolio.

II. Require quotas of projects by value or cost

Another way to use tracking columns is to make the low/medium/high distinction explicit in the model and demand that the portfolio meet certain requirements with respect to these (e.g., fund no more than five low cost projects, or must fund the five highest-scoring projects.) This is often a better simulation of real life scenarios in which diversity across portfolio is highly valued or some projects are required to be funded in all cases. The following table is the funding breakdown in the scenario in which a) no more than five low-cost projects can be funded and b) the five highest-scoring projects must be funded.

# Of Projects Funded by Cost and Value

	Total (Cost)	Total (Value)	Funded (Cost)	Funded (Value)
Low (Cost/Value)	17	18	5	6
Med (Cost/Value)	23	19	22	16
High (Cost/Value)	16	19	9	14

More projects can get funded in this scenario than in scenario I; losing one high-value project allows the funding of many more medium range projects while still funding more top projects than the original optimization. Very few bad projects are funded.

III. Create different funding pools for small/medium/large projects

A way to adjust optimization without using tracking columns is by breaking the money down further by pool, earmarking amounts of money specifically for different groups of projects, whether by value, cost, or another piece of data. In this model, we can break the projects into the low, medium, and high cost brackets and then, as before, assign 50% of the requested funding to the respective budgets.

# Of Projects Funded by Cost and Value

	Total (Cost)	Total (Value)	Funded (Cost)	Funded (Value)
Low (Cost/Value)	17	18	11	9

Med (Cost/Value)	23	19	15	16
High (Cost/Value)	16	19	12	13

With a lot of money in the high-cost alternatives pool, this scenario funds the most large expensive projects with an overall portfolio similar to scenario II. This situation can also be easily adjusted to allow more or fewer projects of any type by moving the budget around between the different cost pools.

6. Limitations

None of these methods actually address the bias of the optimizer; they only override it to force it to behave in a more real-world fashion, and the results are necessarily somewhat arbitrary. If we make sure the optimizer funds three large projects, it will fund three large projects at a "cost" to the perceived portfolio value; we don't yet have a way to make the actual portfolio value calculation reflect reality in a more effective manner. Tracking columns and the use of pools allow us to massage the data in specific useful ways, but do not provide a universally generalizable solution.

7. Conclusions

The AHP lends itself well to linear optimization with a genetic algorithm, but there are elements of that approach that are suboptimal in real life situations because of a bias in favor of small poor-performing cheap projects. Using tracking columns and size-based pools introduces a greater form of control over the optimization by allowing us to fund more top performing projects, reduce the number of overall projects funded, or allocate resources more precisely to ensure a balanced portfolio. All of these methods help make the genetic algorithmic approach more viable in real life situations; however, they require a sophisticated knowledge of each individual problem on a case-by-case basis and are not easily applied universally. Further research in this area could focus on nonlinear methods of constructing portfolio value and other ways of calculating cost/benefit ratios.

8. Key References

List here only those 3 to 5 references that are key for the study at hand.

Adams, W.J. (2012). *Genetic Algorithm Description*. Arlington, VA. Decision Lens, Inc.

Adams, W.J. and Laughlin, Sean. (2006). *Decision Lens Suite Help Documentation*. Arlington, VA. Decision Lens, Inc.

Saaty, Thomas (2009). *Principia Mathematica Decemendi: Mathematical Principles of Decision Making.* Pittsburgh, PA. RWS Publications.