Challenges and Solutions with Exponentiation Constraints using Decision Variables via the BARON Commercial Solver

William N. Caballero, Alexander G. Kline, and Brian J. Lunday
Department of Operational Sciences
Air Force Institute of Technology, WPAFB, OH 45433

Abstract

We observe and explore a persistent issue encountered with the commercial global optimization solver BARON, wherein the solver falsely declares problem instances of a particular math programming formulation as infeasible. Problematic to BARON, the formulation contains constraints having exponentiation with decision variables in both the base and the exponent. We compare BARON’s performance for this math programming problem against other commercial solvers, explore the potential cause of the false infeasible termination, and demonstrate how to mitigate this error by perturbing the formulation.

Keywords
Global Optimization, BARON, Nonlinear Programming

1. Introduction

The Branch and Reduce Optimization Navigator (BARON) is a preeminent global optimization solver that utilizes “constraint propagation, interval analysis, and duality in its reduce arsenal with advanced branch-and-bound optimization concepts” to find optimal solutions to non-convex mathematical programming problems [1]. Foundational work with respect to its nonlinear function relaxations, range reduction strategies, and branching methods was developed by Tawarmalani and Sahinidis [2]. With specific regard to fractional programming problems (i.e., functions which can be decomposed as the sum and products of univariate functions), the authors discuss methods to develop a convex relation to a factorable function, and the creation of a polyhedral outer-approximation to this relaxation. That is, the described methods serve to create a linear approximation of some factorable function by first creating a convex approximation which is, in turn, approximated linearly with hyperplanes. The iterative application of these techniques incorporated into a branch-and-bound framework with the node partitioning and fathoming rules set forth by Tawarmalani and Sahinidis [2] form the foundation of BARON’s algorithmic procedure. The Sahinidis Group [3] provides a systematic comparison of BARON to four other leading global optimization solvers on 1740 test instances which shows BARON is able to solve instances much quicker than other commercial solvers. Extensive testing has also been conducted by Neumaier et al. [4] who concluded that BARON is the fastest and most robust global solver currently available.

For these reasons, BARON has earned its developers much recognition, including the 2004 INFORMS Computing Society Prize and the 2006 Beale-Orchard-Hays Prize from the Mathematical Optimization Society [5, 6]. It is available for use under a variety of algebraic modeling languages, including AIMMS, AMPL, and GAMS. Likewise, BARON has been utilized for supply chain design, integrated process water networks, scheduling, molecular design, manufacturing, and healthcare [7–11]. Its reputation as a reliable and effective global solver is such that BARON is commonly used as a benchmark for heuristics [12, 13].

Despite this success, Lastusilta et al. [14] suggest the solver AlphaECP outperforms BARON over the test instances available in MINLPLib. Likewise, Neumaier et al. [4] reference errors wherein BARON falsely reports an instance as infeasible. We expand upon these results by demonstrating a persistent issue encountered with BARON resulting in a false declaration of infeasibility for a class of problems which have a constraint with a decision variable exponentiated to another decision variable, henceforth referred to as a power program. In Section 2, we describe a specific power program formulation inducing the error, examine the potential cause, and provide programmatic perturbations to address it. Numerical tests in Section 3 illustrate the prevalence of these errors and the efficacy of our perturbations.
2. Rank Dependent Power Program
The presented power program is related to multiple individuals being offered many risky prospects with the objective to maximize the sum of perceived gain probabilities across all individuals. Concepts are borrowed from Cumulative Prospect Theory (CPT) in this formulation in terms of ranking outcomes and probability distortion, but the programs are meant to stand alone as an example and do not necessarily abide by the tenants of CPT [15]. Of note, we utilize neither the concept of a reference point nor the cumulative probability weighting function.

We consider a scenario wherein individuals in a set $I$ are offered a variety of prospects. Each individual $i \in I$ is offered $n_i$ prospects from a set $J_i$, each of which has $m_{ij}$ outcomes, indexed on the set $K_{ij}$. We assume an external decision maker is able to alter the offered outcomes' raw values and probabilities by some constant amount through some binary persuasion action. In doing so, this decision maker wishes to maximize the sum of perceived probabilities of positive outcome values across all individuals and prospects. We continue by introducing the requisite parameters and decision variables.

Parameters
- $\hat{x}_{ijk}$: Baseline raw value for $k$-th outcome of prospect $j$ for individual $i$ before persuasion
- $\hat{p}_{ijk}$: Baseline probability of $k$-th outcome of prospect $j$ for individual $i$ before persuasion
- $\varepsilon$: Arbitrary sufficiently small positive real number
- $M$: Arbitrary sufficiently large real number
- $\hat{f}_{ijk}$: Persuasion effect on outcome $k$ raw value for prospect $j$ and individual $i$
- $\hat{g}_{ijk}$: Persuasion effect on outcome $k$ probability for prospect $j$ and individual $i$

Primary Decision Variables
- $T_{ijk}^+$: Equal to 1 if $x_{ijk}$ is the $(m_{ij} - 1 + k')^{th}$ greatest gain for $k = 1, ..., m_{ij}$, and 0 otherwise; defined for all $(i, j)$ combinations
- $T_{ijk}^-$: Equal to 1 if $x_{ijk}$ is the $(k')^{th}$ greatest loss for $k = 1, ..., m_{ij}$, and 0 otherwise; defined for all $(i, j)$ combinations
- $\gamma_i$: Gain distortion coefficient for individual $i$ after persuasion
- $a_i$: Binary variable indicating action taken against individual $i$.
  - Equals one if action taken, zero otherwise.

Intermediate Decision Variables
- $x_{ijk}$: Gain/loss for individual $i$ for $k$-th outcome of prospect $j$ after persuasion
- $p_{ijk}$: Probability of $k$-th outcome of prospect $j$ for individual $i$ after persuasion
- $t_{ijk}^+$: Ascending rank based list of $x_{ijk}$ gains corresponding with mapping $T_{ijk}^+$
- $t_{ijk}^-$: Ascending rank based list of $x_{ijk}$ losses corresponding with mapping $T_{ijk}^-$
- $b_{ijk}^+$: Corresponding probabilities for sorted $t_{ijk}^+$ outcomes
- $b_{ijk}^-$: Corresponding probabilities for sorted $t_{ijk}^-$ outcomes
- $\pi_{ijk}^+$: Distorted gain probability for $i$ on $k$th outcome of prospect $j$ after persuasion

Many of the intermediate decision variables could be eliminated and substituted with their explicit functional form in the constraints. However, they are maintained for tractability purposes and to facilitate the performance of GAMS/BARON in accordance with published documentation [16]. Using these sets, parameters, and decision variables, we define our Rank Dependent Power Program.
Constraints (1f)–(1o) to assign more free variables or sort them in ascending order of their associated outcome. (1b), (1c), (1d), and (1p) assign a value to free decision variables, and (2) A subset of these assignments is used in the remaining intermediate decision variables can then be found by simple calculation. Furthermore, this formulation γ a observing that the decision variables a converting the formulation into a power program.

Constraints (1l) and (1m) enforce the positivity and negativity of gains and losses, respectively. Constraint (1p) serves (1n) and (1o) do likewise for probabilities. Constraints (1j) and (1k) ensure the values are in ascending order, and through (1o) ensure all prospects are sorted in ascending order and labeled as a gain or a loss depending on the values +′ and T − . Constraints (1f) through (1m) create the mappings, T + jkk and T − jkk. Constraints (1f) and (1g) enforce a bijective mapping. Constraints (1h) and (1i) perform the actual mapping calculation for outcomes, and Constraints (1n) and (1o) do likewise for probabilities. Constraints (1j) and (1k) ensure the values are in ascending order, and Constraints (1l) and (1m) enforce the positivity and negativity of gains and losses, respectively. Constraint (1p) serves as an intermediate decision variable of the terms summed in the objective function. This equality is responsible for converting the formulation into a power program.

For a given set of a decision variables, Constraints (1b) – (1p) are completely determined. This is readily noted by observing that the decision variables a dictate the values of x jk and p jk, which in turn control all remaining decision variables except γ. The values T + jkk and T − jkk are easily found by sorting the x jk-values and observing their signs. The remaining intermediate decision variables can then be found by simple calculation. Furthermore, this formulation always has a feasible solution, regardless of parameter values. This can be observed from the following: (1) Constraints (1b), (1c), (1d), and (1p) assign a value to free decision variables, and (2) A subset of these assignments is used in Constraints (1f)–(1o) to assign more free variables or sort them in ascending order of their associated outcome.
Using these results and observing that increasing $\gamma$ always decreases the objective function value based on the domain of $h_{ijk}^+$, finding an optimal solution to this program can be simplified to searching among all possible combinations of $a_i$ decision variables. This combinatorial construct is not necessarily helpful for large instances, but allows us to easily find optimal solutions to small instances of the program for testing purposes.

BARON is unable to directly handle a function $x^\gamma$ when both $x$ and $y$ are decision variables [17]. In order to process such functions, GAMS/BARON transforms the function to $e^{\gamma \log(x)}$. However, the domains of the two functions are not equivalent. For example, the original function is defined at $x = 0$, whereas the transformation is not.

In an effort to resolve this conflict, we introduce the reformulations A1 through A4. Each perturbation is based on the hypothesis that pre-processing bounds for $p_{ijk}$ are not communicated to $b_{ijk}$ and an error is triggered when BARON observes the possibility of taking $\log(0)$. A1 removes the intermediate decision variable, against the general guidance of GAMS documentation. However, by removing the intermediate decision variable, any error associated with its bounds should be eliminated. A2 and A3 attempt to add a small real number to the intermediate decision variables in (1p) to avoid a domain violation in the transformation. A4 provides a similar solution by lower bounding the sorted probabilities by some very small real number. That is, $b_{ijk}$ is explicitly stated as greater than zero. Strictly speaking, such a lower bound makes the program infeasible. However, if chosen small enough (e.g., $1 \times 10^{-16}$), the lower bound is essentially treated as a roundoff error while remaining defined in the transformed power function.

**Alternative or Additional Constraints**

A1 : $\pi_{ijk}^+ = \sum_{k=1}^{m_{ij}} T_{ijk}^+ \theta_{ijk}$ substituted for (1p)

A2 : $\pi_{ijk}^+ = (b_{ijk}^+ + \varepsilon)^\gamma$ substituted for (1p)

A3 : $\pi_{ijk}^+ = (b_{ijk}^+ + \varepsilon)^\gamma$ substituted for (1p)

A4 : $b_{ijk}^+ \geq \varepsilon$, $b_{ijk}' \geq \varepsilon$ added to the base model

### 3. Testing and Analysis

We compare BARON’s performance on 100 instances of the base Rank Dependent Power Program with two individuals each being offered two prospects, for which there respectively exist two potential outcomes, to that of five other solvers. Of these five solvers, one is a global solver (i.e., SCIP) while the other four (i.e., DICOPT, LINDO, SSB, and AlphaECP) are primarily used for solving convex MINLPs. The 100 random instances draw parameters from the following distributions such that $x_{ijk} \in [-6500, 6500]$, $p_{ijk} \in [0.01, 0.99]$, and $e^{\gamma \log(h_{ijk}^+)}$ is always defined: $\hat{x}_{ijk} \sim U[-5000,5000]$, $\hat{p}_{ijk} \sim U[0.25,0.75]$, $\hat{f}_{ijk} \sim U[-1500,1500]$, and $\hat{g}_{ijk} \sim U[-0.249,0.249]$. To calculate the optimality gap, the action space in each instance is enumerated and its effect on (1b) – (1p) manually calculated as previously discussed.

Testing is performed on an HP ZBook equipped with a 2.70 GHz Intel i7-4800MQ processor and 32GB of RAM. Each solver is provided a relative optimality termination criteria of 0.001, an iteration limit of 3000, and a time limit of five minutes. Table 1 details the termination criterion invoked for each solver over the 100 random instances. Only AlphaECP correctly identifies all instances as feasible. The global solvers BARON and SCIP struggle to find feasible solutions and terminate with a conclusion of infeasibility for 100 and 85 instances, respectively. DICOPT does not label any of the instances as infeasible but is only able to return a solution for 16 of the instances. SSB and LINDO incorrectly terminate by designating 8 and 12 instances as infeasible, respectively. Taken collectively, the global solvers appear to underperform their convex MINLP counterparts.

With regard to BARON, we compare the alternative formulations and examine the termination criterion. It can be observed in Table 1 that A1, A3, and A4 all conclude that 95 instances are feasible. Surprisingly, alternative A2 concludes that only 2 instances are feasible. However, upon examination, the evaluation of the objective function is found to be incorrect. These two observations are excluded from further analysis. We postulate the difference in performance between A2 and A3 derives from the pre-processing technique utilized for the absolute value function.

Although the alternative formulations A1, A3, and A4 greatly improve performance, erroneous infeasible terminations still occur. AlphaECP is the only solver which correctly identifies the feasibility of all instances. However, Table 1 shows that, for the instances BARON identifies as feasible, it has the lowest average optimality gap.
4. Conclusions
We have examined a problem which belongs to the very challenging class of power programs. The program analyzed proves to be problematic for the GAMS/BARON solver combination. In its base form, an incorrect infeasible termination is reached in the Rank Dependent Power Program for each of the hundred instances examined which is the worst result over all solvers considered. These results are believed to be due to the GAMS/BARON transformation utilized for $x^y$ when both $x, y$ are decision variables. Accounting for this transformation, alternative formulations are examined which yield promising results in terms of solution quality, but still conclude with incorrect infeasible terminations for some instances.

Although, BARON is advertised as not requiring an initial seeded solution, for problems such as ours, it may be a helpful tactic to mitigate the observed error. Since AlphaECP seemingly is the most effective software in terms of finding a feasible solution, a combined approach of using AlphaECP to find a starting point which is then fed into BARON may prove efficacious.

Fortunately, there always exists a feasible solution for the examined power program in this research. However, our results demonstrate the potential for BARON to yield a false infeasible termination, which is problematic for instances without a guarantee of feasibility. In order to mitigate this issue, we have provided a collection of programmatic mitigation techniques for practitioner use.

References