A Practical Approach to Subset Selection for Multi-Objective Optimization via Simulation

CHRISTINE S.M. CURRIE, University of Southampton, UK
THOMAS MONKS, University of Exeter, UK

We describe a practical two-stage algorithm, BootComp, for multi-objective optimization via simulation. Our algorithm finds a subset of good designs that a decision-maker can compare to identify the one that works best when considering all aspects of the system, including those that cannot be modeled. BootComp is designed to be straightforward to implement by a practitioner with basic statistical knowledge in a simulation package that does not support sequential ranking and selection. These requirements restrict us to a two-stage procedure that works with any distributions of the outputs and allows for the use of common random numbers. Comparisons with sequential ranking and selection methods suggest that it performs well and we also demonstrate its use analyzing a real simulation aiming to determine the optimal ward configuration for a UK hospital.

CCS Concepts: • **Computing methodologies** → *Simulation evaluation*;

Additional Key Words and Phrases: Ranking and selection, simulation, subset selection, chance constraints

ACM Reference Format:
Christine S.M. Currie and Thomas Monks. 2019. A Practical Approach to Subset Selection for Multi-Objective Optimization via Simulation . *ACM Trans. Model. Comput. Simul.* 100, 100, Article 100 (September 2019), 15 pages. https://doi.org/0000001.0000001

Authors' addresses: Christine S.M. Currie, University of Southampton, Highfield, Southampton, SO17 1BJ, UK, christine.currie@soton.ac.uk; Thomas Monks, University of Exeter, St Luke's Campus, Heavitree Road, Exeter, EX1 2LU, UK, t.m.w.monks@exeter.ac.uk.

XXXX-XXXX/2019/9-ART100 $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

We consider a practical method for optimization via simulation in which a decision-maker wishes to choose the best of a large set of alternative designs. In practice, the decision over which design to choose is generally not based on just one outcome measure but instead is a multi-objective problem in which some aspects of the decision may not be incorporated into the simulation model. In these large, complex problems simulation is frequently used to reduce the set of possible designs to a smaller subset of *good* designs, which the decision-maker(s) can consider in more detail to identify which will work best in practice.

The method that we describe here is designed to be straightforward to implement such that an analyst with only a basic understanding of statistics and simulation modeling could use it. This places a first restriction on the method, that it should work for any distribution of the outputs, including correlated outputs. Making this restriction assumes that good practice such as the use of common random numbers can be followed but also that there is no fine print for the experimenter to read and comply with before running the method. Algorithms that place few restrictions on the output are typically described as *general-purpose* and bootstrapping, as used in BootComp, is often used to avoid constraints on the output distributions [Lee and Nelson 2014, 2015, 2016].

We also make the assumption, based on practical experience, that communication with the simulation model can be computationally expensive and difficult for a simulation practitioner to incorporate efficiently. This places a second restriction on the method, that the choice of designs to replicate over and the number of replications to make of each design can only be changed once during the experiment. Doing so rules out the fully sequential ranking and selection procedures that have dominated the literature in recent years and effectively enforces a two-stage process. We acknowledge that there are off-the-shelf simulation packages that include fully sequential ranking and selection procedures but the majority do not and this procedure is designed to work well in those cases. The issue with building a ranking and selection algorithm that sits outside the software package is that there is likely to be a time delay for communication between the ranking and selection code and the simulation software, updating the model parameters and (re)initializing the model. The "hidden cost" of communication time with the simulation model has previously been described in [Kim and Nelson 2006a] but the difficulty of automating a simulation package to switch between the different designs being evaluated should not be underestimated, particularly where there are significant differences in the logic. In such situations, it is possible that the bootstrap ranking and selection procedures developed by Lee and Nelson may work well as they suggest conducting Δn observations at each step rather than just one [Lee and Nelson 2016]. An interesting practical problem would be to determine the optimal balance between the loss of efficiency that comes from using a high value of Δn with the computational cost of communicating more frequently between the optimization software and the simulation software when Δn is small.

A further benefit of the two-stage procedure is that it allows for efficient parallelization. Unlike the original sequential allocation algorithms, there is only a need to communicate results from different designs twice during the experiment. Where parallelization or cloud computing facilities are available this can result in a dramatic speed up in the computation times. Recent advances in sequential algorithms have addressed this issue (e.g. see [Pei et al. 2018]) but BootComp allows for a straightforward treatment of parallelization.

Multi-objective problems have been considered in three ways previously in the ranking and selection literature: (i) using a Pareto front approach [Lee et al. 2010]; (ii) via chance constraints [Hong et al. 2015]; and (iii) optimizing a weighted average of the multiple objectives, e.g. see [Swisher and Jacobsen 2002]. We use the second approach in which secondary objectives are reformulated as chance constraints, with the boundary values set based on discussions with system

experts. While the Pareto approach has obvious merits, as it avoids the subjectivity inherent in the selection of threshold values for the chance-constraints, it can be difficult to explain to decision-makers.

A Python implementation of the algorithm that we describe here is available for download [Monks and Currie 2019a]. The algorithm relies on bootstrapping and, given the split into two stages lends itself well to parallelization. An implementation using the cloud is available within the Github repository.

This article extends the work described in [Monks and Currie 2018] in which we introduced the basic ideas of the method, with the most significant advance being a thorough comparison of its performance with OCBA-m on a standard set of problems frequently used to measure performance. We also extend the treatment of the hospital simulation model that was considered previously.

We provide a formulation of the problem before the literature review and methods sections so that the reader has a clear understanding of what we are aiming to achieve.

1.1 Problem Formulation

Assume that there are in general $k \geq 2$ designs being compared and define X_{ij} to be univariate, real-valued output data coming from replication j for design i that represents the main performance measure. We make no assumptions about the distribution of the X_{ij} and allow for correlations between observations from different designs in the same replication, which may arise if CRN are being used. We assume that observations from different replications are independent and identically distributed. Our main performance measure is assumed to be the mean of the output for a particular design after n replications,

$$x_i = \sum_{j=1}^{n} X_{ij}/n,$$

and we assume here that small values of x_i are desirable.

We also define a set of $l = 1, \ldots, L$ secondary outputs, Y_{ijl} coming from replication j for design i. The Y_{ijl} are real-valued and we assume that their mean values,

$$y_{il} = \sum_{j=1}^{n} Y_{ijl}/n,$$

are the secondary objectives. Our problem is one of subset selection with chance constraints on the secondary outputs. We wish to identify a shortlist or subset of designs $\mathbf{S}^*$ that are all within a proportion β of the mean of the best feasible design, x^* with a probability $1-\alpha$, and fail the chance constraints with probability less than γ. Note that we use a non-standard measure of proximity to the best in this work, preferring to use a proportion of the mean of the best design instead of the standard absolute difference. A design is said to satisfy the chance constraints if $\mathbb{P}\{\mathbf{y}_i \geq \mathbf{0}\} \geq 1-\gamma$, where we use $\mathbf{y}_i$ to represent the vector of sample means $y_{il}, l = 1, \ldots, L$. Combining, our problem can be written as one of determining $\mathbf{S}^*$ such that

$$\begin{aligned} \mathbb{P}\left\{\bigcap_{i \in \mathbf{S}^*}(x_i \leq x^*(1+\beta))\right\} &\geq 1-\alpha, \\ \mathbb{P}\{\mathbf{y}_i \geq \mathbf{0}\} &\geq 1-\gamma,\ i \in \mathbf{S}^* \end{aligned} \tag{1}$$

The user and the decision maker set m to be the maximum length of the shortlist. If the size of $\mathbf{S}^*$, $|\mathbf{S}^*| > m$, the top m designs in a ranked list of mean outputs of the key measure will be included in the shortlist. We make no claims that these will be the top m methods, but only that they all lie within β of the best feasible design and that there is a high probability that the designs we include

in the final set are feasible. From a statistical perspective, this is the least satisfactory part of the method, but it helps in situations where there are a large number of suitable designs. An alternative would be to reanalyze the results with smaller values for β or α.

2 LITERATURE REVIEW

Selection procedures aim to find the best design or designs from a finite set of possibilities. Observations of each design are subject to stochastic variability and the best design is generally that with the maximum or minimum mean. [Branke et al. 2007] provides a thorough review of the topic and splits the approaches into three categories: the indifference zone approach; expected value of information procedures (VIP) and optimal computing budget allocation (OCBA). While indifference zone procedures such as [Kim and Nelson 2006b] suggest sequential sampling plans that guarantee the probability of correct selection (PCS), OCBA procedures first introduced by [Chen 1996] aim to maximize PCS with a fixed computation budget; and VIP methods choose which design to sample from based on the expected gain in the value of information [Chick and Inoue 2001b].

Parallelization and advances in cloud computing mean that there is a great deal of scope for speeding up these sequential selection procedures. For example, preliminary work in [Pei et al. 2018] describes a framework Parallel Adaptive Survivor Selection (PASS), designed to exploit parallelization when solving ranking and selection problems with large numbers of designs. Instead of using PCS as an objective, PASS aims to minimize the expected false elimination rate, a statistic that scales better as the number of designs being tested increases. Methods that are able to easily transfer to parallel computing seem likely to dominate in the future.

The remainder of the literature review focuses on selection procedures designed for similar optimization problems to the one we consider here.

2.1 Subset Selection

Much of the literature on optimal subset selection aims to identify the best m designs out of a total of k possibilities, where the quality of a design is measured by its average output over all of the simulation replications. This raises an interesting question for users of the method as to what an appropriate value for m might be and more flexible measures might be more appropriate, e.g., all designs within some percentage of the best. The difficulty with this latter approach is identifying the borderline for the subset when the mean output of the best design is unknown.

Early approaches to this problem used a two-stage approach. [Sullivan and Wilson 1989] use a two-stage procedure to find a subset of designs such that the subset contains a design with mean that is less than δ from the best mean, with probability greater than or equal to a user-defined threshold. The two-stage procedure in [Koenig and Law 1985] finds the subset of size m that contains the best m designs with probability at least equal to a threshold, providing that the difference between the objective of the $m+1$th and mth designs is no less than an indifference zone parameter. More recently, [Wang et al. 2011] develop a sequential algorithm that finds a subset of designs which are either desirable or acceptable but not within the elimination region. These categories, which draw on ideas from indifference zone methods, are taken from [Andradóttir and Kim 2010].

In the numerical examples considered later, we compare our algorithm with OCBA-m [Chen et al. 2008], which aims to find all of the top m out of k designs. There is a subtle difference between this procedure and earlier subset selection methods such as [Gupta 1965], which aim to find a subset of designs that contain the best design with a pre-specified probability. The OCBASS algorithm introduced in [Gao and Chen 2016] solves the same problem as OCBA-m, allocating replications to the design that is most likely to be "incorrectly observed". While this principle is similar to the standard OCBA algorithm, the application is subtly different as the question is now whether a design is within the boundary of the optimal subset or outside of it, rather than whether a design

is the best or not. Numerical results suggest that OCBASS is more efficient than OCBA-m, with improvements in efficiency of around 29.2 and 42.5%. [Chingcuanco and Osorio 2013] consider the same problem but have a different approach in which they use all of the possible subsets of size m as the different designs in a ranking and selection problem.

[Gao and Chen 2015] instead minimize the Expected Opportunity Cost (EOC), measured as the difference in expected performance between the selected designs and the best designs. The use of EOC as a measure of the quality of a ranking and selection procedure was first introduced by [Chick and Inoue 2001b] and has a clear practical benefit.

2.2 Multi-Objective Optimization via Ranking and Selection

[Hunter et al. 2019] provide a very clear introduction to multi-objective simulation optimization (MOSO), which has a wider scope than simply multi-objective ranking and selection problems. Within their discussion of ranking and selection procedures, they assume that these aim to identify the globally Pareto optimal designs, considering both fixed budget (e.g. MOCBA, [Lee et al. 2010]) and fixed precision procedures. Globally Pareto optimal designs are considered to be those in the *efficient set*, i.e., no other design corresponds to an objective vector that is at least as small on all objectives, and strictly smaller on at least one objective.

An alternative method for dealing with multi-objective problems is to instead reduce the dimensionality of the objective by introducing a multi-attribute variable that is a weighted average of the different objectives. A particularly relevant paper describing this approach [Swisher and Jacobsen 2002] aims to optimize the medical personnel staffing and facility size for a family practice clinic. Similar to the case study we describe at the end of this article, the measures of effectiveness of a policy include several elements and the authors identify appropriate weights to find a sum that best balances the tradeoffs between them before using the NM method [Nelson and Matejcik 1995] to carry out the optimization.

As discussed in the Introduction, we use chance constraints to account for secondary objectives, an approach that works well for many practical problems and can be easily explained to a decision-maker. As far as we are aware, there are no other examples of algorithms that apply a two stage procedure to identifying the optimal subset of designs with chance constraints when the output can take any distributional form. The main focus of the remainder of our review is on other algorithms that use chance constraints as these are likely to be most closely related to the BootComp algorithm we describe here.

[Hong et al. 2015] identify two distinct problems within optimization with chance constraints: Expectation Constrained Selection of the Best (ECSB) and Chance Constrained Selection of the Best (CCSB). In ECSB, the expected value of one or more outputs of the simulation model must be below some fixed value while in CCSB, the constraints imposed by the decision-maker must be satisfied with a given probability. [Andradóttir and Kim 2010], like us, uses the expected value of the secondary output measure in the constraint equation, making this an ECSB problem. Their procedure uses an indifference zone to identify the best design that satisfies a single constraint. While they assume that the main and secondary outputs are normal, the results presented seem relatively robust to non-normality. Their approach to the chance constraint relies on Bechhofer's indifference zone structure [Bechhofer 1954] in which three different regions are identified for the value of the chance constraint: desirable, acceptable and unacceptable.

[Hong et al. 2015] instead solve the CCSB problem and imposing this structure on the constraints facilitates checking the feasibility of designs. Their approach also assumes normality for the simulation outputs and selects just one optimal value. It allows for common random numbers (CRN) and finds a relatively conservative solution for the number of replications due to its assumption that all chance constraints must be satisfied subject to the Bechhofer indifference zone structure.

Chance constraints have also been included in algorithms that aim to optimally allocate a fixed simulation budget in ranking and selection problems. For example, [Pasupathy et al. 2014] make an elegant use of bilevel optimization to determine an algorithm SCORE to solve the ECSB problem; and [Hunter and Pasupathy 2013] put forward a sequential algorithm that maximizes the asymptotic rate of decay of the probability of incorrect selection, reliant on knowing the underlying distribution of the output data.

2.3 Generalized Ranking and Selection

Many of the methods described in the previous sections make some assumption about the simulation outputs, the most common being independence and normality. Several authors have considered how ranking and selection can be achieved for more general distributions. [Lee and Nelson 2016] describe their methods as *general-purpose ranking and selection*, and they allow for different output distributions for different designs; and measuring statistics other than the mean, as well as accounting for dependent observations (as observed when using CRN) and non-normality. In order to provide guarantees of correct selection, [Lee and Nelson 2016] use bootstrapping to estimate the probability that the observed best design is the best. When this exceeds the threshold, sampling can stop. Bekki et al. [2010] have a slightly different aim of developing a flexible ranking and selection procedure that allows comparisons to be made on any distributional property. Again the authors use non-parametric bootstrapping as a key part of their approach.The algorithm that we develop here also makes use of non-parametric bootstrapping as, although computationally intensive, bootstrapping is a straightforward way of allowing for general distributions.

2.4 Two-Stage Ranking and Selection Methods

Articles on two-stage optimization methods were prevalent during the 1990s and the early 2000s following the publication of [Koenig and Law 1985] that developed a two-stage procedure for subset selection. Of particular relevance to this work are [Nelson and Matejcik 1995] and [Chick and Inoue 2001a], which describe two-stage procedures that account for CRN but for selection of a single design, rather than a subset. Both procedures assume normality in the outputs with [Chick and Inoue 2001a] having a slightly less restrictive assumption about the covariance structure existing between the outputs of the different designs. The procedure described by [Chick and Inoue 2001a] has more similarities to our work in that it works with a restricted budget and does not include designs in the second stage that it would be of no benefit to sample further. In contrast, the *NM* procedure of [Nelson and Matejcik 1995] aims to estimate the number of replications needed in stage 2 to achieve a threshold probability of correct selection. [Nelson and Matejcik 1995] provide estimates of the difference between the best design and each of the other designs as a way of presenting a fuller set of results, allowing the decision-maker to consider other inferior designs that may have benefits not included in the simulation model. This goes some way towards addressing our assumption that decision-makers prefer to see a few possibilities rather than just one final result. Neither of these two-stage procedures considers multi-objective optimization, chance constraints or non-normal output distributions.

3 METHODOLOGY

The method relies on two bootstrap routines, which are both run twice: first in stage one and again at the end of stage two. The first, **Constraints Bootstrap**, is used to determine the probability of a particular design violating the constraints, while the second, **Quality Bootstrap**, identifies the designs with means within a proportion β of the mean of the best remaining designs. We begin by describing the whole procedure, which is also described in Figure 1, before giving more detailed descriptions of the two bootstrap routines.

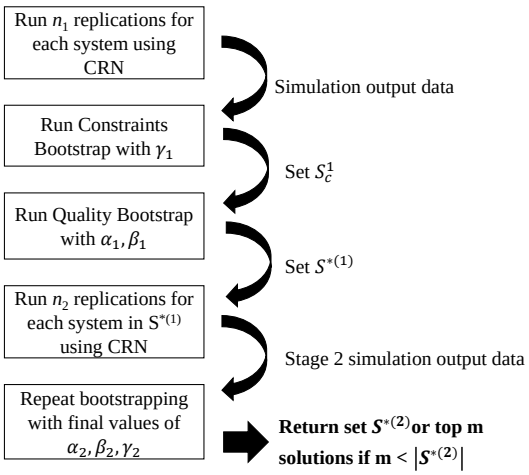

Fig. 1. An overview of the methodology.

(1) Run n_1 replications of the simulation model for all designs using CRN if available to generate a set of primary outputs X_{ij} and secondary outputs Y_{ijl}, where $j = 1, \ldots, n_1$; $i = 1, ..., k$; $l = 1, \ldots, L$.
(2) Remove any designs that have a probability of failing the chance constraints of greater than γ_1 leaving a set of designs $\mathbf{S}_c^1$ after the first stage of sampling. The set $\mathbf{S}_c^1$ is identified via bootstrapping using the **Constraints Bootstrap** described below in Section 3.1 with inputs $\mathbf{S}$, the complete set of designs, $\mathbf{Y}$ and γ_1.
(3) Run the **Quality Bootstrap** described below in Section 3.2 with inputs $\alpha_1, \beta_1, \mathbf{S}_c^{(1)}$ and the n_1 by k matrix of results for the main output measure $\mathbf{X}$. This outputs the set of designs $\mathbf{S}^{*(1)}$ that have means within a proportion β_1 of the best mean in set $S_c^{(1)}$ with probability greater than $1 - \alpha_1$.
(4) Run n_2 replications of the simulation model for all designs in the set $\mathbf{S}^{*(1)}$ using CRN if available.
(5) Follow steps 2 and 3 using data from both stages in $\mathbf{X}$ and $\mathbf{Y}$ and parameters $\gamma_2, \alpha_2, \beta_2$. This results in a final shortlist of feasible and high quality designs $\mathbf{S}^{*(2)}$.
(6) If the number of designs is greater than the desired number m, pick the top m in a non-decreasing list of means of the main output measure. Ties are decided using secondary output measures $\mathbf{Y}$.

We use different values for stage 1 and stage 2 parameters in the numerical results. These parameters are set by the user and while we provide recommendations below, we acknowledge that more work is needed on determining optimal values.

3.1 Constraints Bootstrap

Our aim in the Constraints Bootstrap is to identify designs that are likely to violate the chance constraints. We assume that we have more than one chance constraint and use a fairly conservative

measure of satisfying the chance constraints, as suggested in [Hong et al. 2015] in which we insist that *all* of the constraints are satisfied simultaneously.

The routine takes as inputs $\mathbf{Y}$, the secondary outputs of the simulation model such that Y_{ijl} is the value of the lth chance constraint, $l = 1, \ldots, L$ in the jth replication for the ith design. Here, $i \in \mathbf{S}$ and $j = 1, \ldots, n$. Note that $n = n_1$ in stage 1 and $n = n_2$ in stage 2. Also input is γ, the threshold probability for failing the chance constraints.

As is standard with bootstrapping, we assume that $\mathbf{Y}$ can be viewed as a sample from a true multivariate distribution G and use bootstrapping to sample from an approximation to this distribution, $\hat{G}$. The bootstrapping proceeds as follows.

(1) Generate B bootstrap samples of size n using a non-parametric bootstrap in which we sample with replacement from $\mathbf{Y}$. This yields a set of bootstrap samples $\mathbf{Y}^{\star(1)}, \mathbf{Y}^{\star(2)}, \ldots, \mathbf{Y}^{\star(B)}$ and for each of these we calculate $y_{il}^{\star(b)}, l = 1, \ldots, L$, where $y_{il}^{\star(b)} = \frac{1}{n}\sum_{j=1}^{n} Y_{ijl}^{\star(b)}$.

(2) Include designs in the final feasible set $\mathbf{S}_c$ if

$$\frac{1}{B}\sum_{b=1}^{B}\prod_{l=1}^{L} I\left\{y_l^{\star(b)} \geq 0\right\} \geq 1-\gamma, \tag{2}$$

where $I(\cdot)$ is the indicator function, which takes a value of 1 if the condition inside the parentheses is true and 0 otherwise.

(3) Return $\mathbf{S}_c$.

3.2 Quality Bootstrap

The quality bootstrap aims to identify a set of designs S^* that all have means within a distance β of the mean of the best sampled design in S^* with a probability $1-\alpha$. It takes as inputs α, β, the set of feasible designs $\mathbf{S}_c$, and $\mathbf{X}$, a set of outputs for the main output measure X_{ij}, $i \in \mathbf{S}_c$, $j = 1, \ldots, n$, where n is the number of replications being considered in the bootstrap (n_1 in stage 1 and n_2 in stage 2).

(1) Define a new variable $\mathbf{d}$ with elements $d_{ij}, i \in \mathbf{S}_c, j = 1, \ldots, n$, such that

$$d_{ij} = x_j^* - X_{ij},$$

where $x_j^* = x_{i^*j}, i^* = \arg\max x_i$ is the value of the main output measure for the best, feasible design in stage 1, in replication j of the simulation. We can treat the set of d_{ij} as a sample from the true multivariate distribution for the differences from the best, F.

(2) Generate B bootstrap samples of size n from $\hat{F}$, our approximation to F. We do this using a non-parametric bootstrap in which we sample with replacement from the set of d_{ij}. This gives us a new set of samples $\mathbf{d}^{\star(1)}, \mathbf{d}^{\star(2)}, \ldots, \mathbf{d}^{\star(B)}$, where each $\mathbf{d}^{\star(b)}$ is of the same dimensions as the original set of differences, $\mathbf{d}$.

(3) Identify $\mathbf{S}^*$ such that it is the biggest set for which

$$\frac{1}{B}\sum_{b=1}^{B}\prod_{i\in\mathbf{S}_c} I\left\{\frac{1}{n}\sum_{j=1}^{n} d_{ij}^{\star(b)} \geq -\beta x^*\right\} \geq 1-\alpha. \tag{3}$$

This final step is carried out by ordering the designs in non-increasing order of the main output measure and working down the list until 3 is true for the set of designs at the top of the list but would not be true if the next design was included in the set.

(4) Return $\mathbf{S}^*$.

3.3 Python Implementation

An implementation of the optimization procedure, *BootComp* [Monks and Currie 2019a], was developed in Python 3.7.2 with dependencies NumPy 1.16.2, Pandas 0.24.1, SciPy 1.2.1 and Numba 0.42.0. Matplotlib 3.0.2 and Seaborn 0.9.0 were used for visualization in the analysis. All numerical examples, including a cloud executable example of using *BootComp*, are available online via GitHub and Binder (https://github.com/CLAHRCWessex/BootComp). For local installation it is recommended that users install an Anaconda Distribution (https://www.anaconda.com/download/). Our code implementation includes an environment file that will install the appropriate dependencies. Users unfamiliar with version control or GitHub can simply download the repository and de-compress the files. Details of the applied example are found in the Jupyter notebook `BootComp_Tutorial.ipynb`.

4 NUMERICAL EXAMPLES

We begin this section by demonstrating that the algorithm performs well on a set of test problems considered in other articles introducing subset selection algorithms. It should be stressed that these test problems do not contain chance constraints and we compare BootComp with a sequential algorithm, OCBA-m. As a result, we do not necessarily expect BootComp to be the top performer. Nonetheless, we feel that including the comparison provides a measure of its quality.

4.1 Comparison to Single Objective OCBA-m

We first compare our two-stage procedure to the Optimal Computing Budget Allocation (OCBA-m) procedure for returning the *top m* designs [Gao and Chen 2016]. OCBA-m is a state-of-the-art fully sequential single objective subset selection method. In our first test bed there are ten competing designs that are normally distributed with mean $\mu_1 = 1$, $\mu_2 = 2$ ... $\mu_{10} = 10$ and unit variance. In each experiment, the procedures are set to select the top three largest designs. The use of Common Random Numbers (CRN) to reduce the variance between competing designs is standard within commercial off-the-shelf simulation software packages. We compare BootComp and OCBA-m over a range of different computational budgets (total number of replications), carrying out 10,000 experiments with and without the use of CRN for each budget value. Respectively, this gives a sense of the best and worst case performance of BootComp.

The second test bed is the well known inventory control problem introduced by [Koenig and Law 1985] and also considered in [Gao and Chen 2016]. In this example, we simulate 10,000 experiments of nine competing system designs in the Law inventory problem. CRN have been employed, achieving 89% variance reduction between simulated designs. We consider two different optimization problems with this example: first, returning the three designs with the largest mean and second, the two designs with the lowest mean.

Code and data to reproduce the numerical results are available in GitHub [Monks and Currie 2019b]

4.1.1 Ten Designs with CRN. Both OCBA-m and BootComp were run with 20 initial replications for each design ($n_1 = 20$). In all of the experiments we use the following values for the BootComp parameters $\alpha_1 = 0.2, \alpha_2 = 0.05, \beta_1 = 0.6, \beta_2 = 0.3$. We tested budgets between 300 and 2000 and found that when competing designs use perfectly synchronized CRN, BootComp correctly selected the top three designs in all 10,000 experiments for each budget. This result held with a more restrictive $\beta_1 = 0.4$. OCBA-m has high PCS in all experiments and selects the top three designs in all experiments when budgets exceed 800 replications.

4.1.2 Ten Designs without CRN. Both OCBA-m and BootComp were run with 20 initial replications for each design ($n_1 = 20$). In all of the experiments we use the following values for the BootComp

parameters $\alpha_1 = 0.2, \alpha_2 = 0.05, \beta_1 = 0.6, \beta_2 = 0.3$. Again using 10,000 experiments, we tested budgets of between 600 and 5000 replications. The comparison is illustrated in Figure 2. The results illustrate the importance of CRN in variance reduction and, as expected, BootComp loses its advantage over OCBA-m in examples where samples are independent. Without CRN, BootComp was unable to match OCBA-m under any of the budget assumptions we included. This is not unexpected as OCBA-m is a sequential algorithm and BootComp includes only two stages. We make two further observations. First, budgets above 1400 replications yielded a PCS with more utility for decision making. Second selecting a less conservative β_1 often yielded better results than holding β_1 constant and increasing the replication budget.

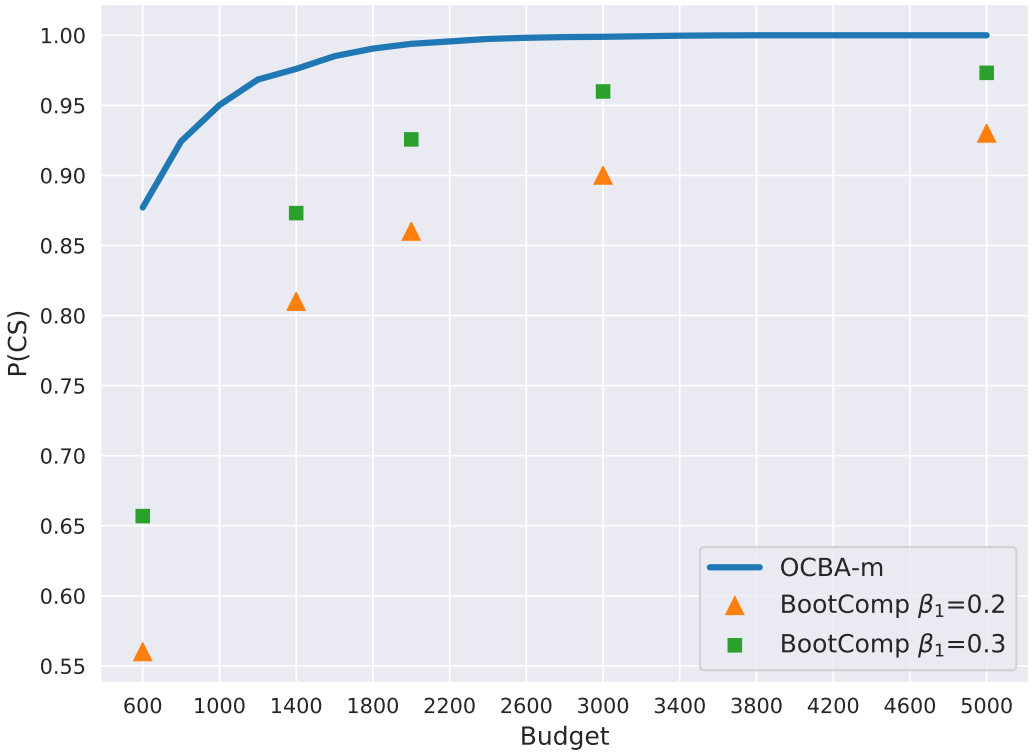

Fig. 2. Probability of Correct Selection of the Top Three Designs in the Ten Designs without CRN case (10,000 experiments).

4.1.3 Inventory Simulation Model. In the inventory example, full synchronization has not been achieved, but CRN has still induced a high positive correlation across the nine designs. Both OCBA-m and BootComp select the top three designs (highest mean) in all 10,000 experiments for each budget. This result held when the problem was changed into a minimization to select the smallest two designs.

4.2 Hospital Simulation Model

As a final demonstration of the *full* method, including the chance constraints bootstrap, we apply BootComp to a problem of hospital ward design coming from a real project with the UK National

Health Service (NHS). The project investigated the design of an NHS community hospital rehabilitation ward in order to minimize delays in the transfer of care of elderly in-patients from a large acute hospital. The new rehabilitation ward would be created from the merger of two geographically separate wards. The context and simulation model are described in detail in [Penn et al. 2018] and the model built in Simul8 Professional 2018 is freely available online [Penn and Monks 2018]

NHS England mandate single sex accommodation for patients. To accommodate this rule and provide some flexibility, beds within the wards could be put within single rooms or grouped in bays. Single rooms are in one sense the most efficient way to improve flow; however, single rooms are more expensive to safely staff. Bays of beds offer a good compromise to single rooms in terms of staffing costs and patient flow. An empty bay is assigned a gender corresponding to the gender of the first patient to be admitted. As patients are admitted and discharged from a ward, patients can be moved and the gender of the bay flipped to admit waiting patients. As there are no differences in process times for patients in a single room or within a bay, bays are less efficient for flow than single rooms. For example, a male patient requiring rehabilitation may not be able to be placed on a rehabilitation ward that has a free bed if the free bed is within a female designated bay. The simulation study, therefore, analyzed, the mix of single rooms and size/number of bays that were required to keep waiting times for admission low and bed utilization acceptable.

The primary output of the model is the average waiting time for transfer to rehabilitation. Secondary outputs are utilization (occupancy) of beds, and the number of patient transfers between gender specific bays. Patient transfers are required when it is not possible to fit the current set of patients in the ward without breaking the constraint that a bay includes patients of the same gender. The client requested a *set* of good designs as opposed to a single optimal design.

4.3 Experiments and Comparisons

In stage one the model was used to conduct 1151 initial simulation experiments (stage 1 replications $n_1 = 5$; time unit = days; run length = 365 days). The replications can be found in three .csv files in the Git repository (`data\replications_wait_times.csv`; `data\replications_util.csv`; `data\replications_transfers.csv`).

Table 1 lists the parameters used in the two-stage optimization. Clients preferred an average utilization of at least 80% and the upper threshold for the mean number of patient transfers between bays was set to 50. The number of replications in stage 1 is based on the pragmatic recommendations from Law and Kelton [2000]; taking the upper bound of their recommended range of *at least 3-5 replications*. In the applied example, it took approximately 2.5 hours to generate the stage one replications for all 1151 designs, using commercial software and a machine of a similar specification that would be found in use in industry (Simul8 Professional 2017; Dell Laptop; 16GB RAM; i7 processor). Chernick [2007] warns of the limitations of the bootstrap for sample sizes below 10. We found that results were consistent if stage one replications were extended to 10 per design (with the associated 5 hour model run time) and consequently report the results with just 5 replications. The stage 1 and stage 2 values for γ were chosen to allow for the fact that errors will be higher in stage 1 due to the smaller number of replications, while α was set to 0.05 to mimic accepted practice in ranking and selection procedures and β to 0.05 based on the clients' preferences. We set n_2 based on the total computation time available and the number of designs being passed through to stage 2 $|S^{*(1)}|$.

4.4 Chance Constraints

Figure 1 illustrates an initial informal analysis of the conflict between the mean waiting time (Figure 1a), which is the main output measure, and the first chance constraint, which requires utilization

Table 1. Parameters used in the optimization.

Parameter	Stage 1	Stage 2
Utilization constraint	$\geq 80\%$	$\geq 80\%$
Patient transfers	≤ 50	≤ 50
Replications	5	50
γ	0.7	0.95
α	0.05	0.05
β	0.3	0.05

to be greater than 80% (Figure 1b). For reasons of simplicity, the chart only includes results for designs that have exclusively single rooms and with the number of beds in the range 43-55.

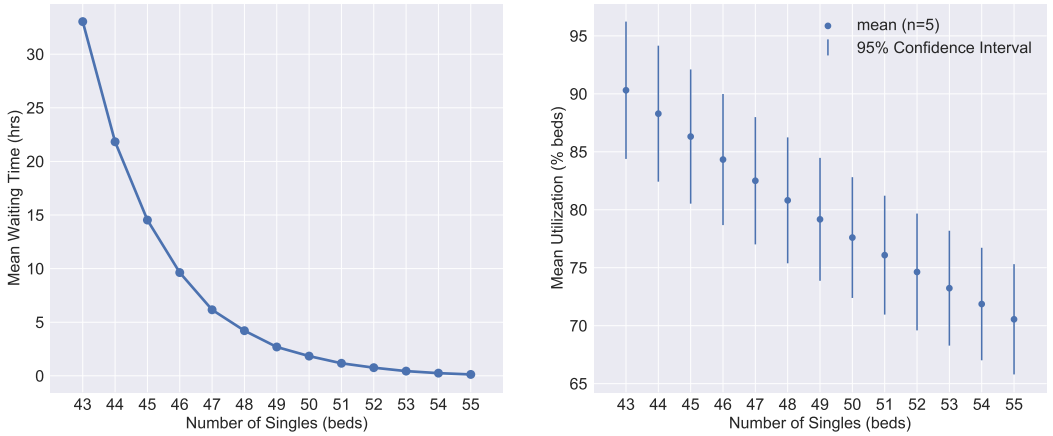

Fig. 3. Illustration of Stage 1 Utilization Chance Constraint.

4.5 Optimization Procedure

Figure 4 illustrates our optimization algorithm applied to the rehabilitation model. Stage 1 reduces the number of designs from 1151 to 34. Most of this reduction in designs is delivered by the chance constraints – reducing the feasible set to 177. A further 45 replications were conducted in stage 2, resulting in a total of 7285 individual runs of the model in stages 1 and 2. Stage 2 reduced the number of designs from 34 to 28. Table 2 lists the top 10 final elite designs. These all have 48 beds, and suggest that bay sizes of 3 or 4 are optimal.

5 CONCLUSION

Our aim in this work was to develop a tool for multi-objective simulation optimization that is simple-to-use, fits within typical simulation practice, does not require the user to implement code to interact with the simulation models, and yet draws on recent research in ranking and selection. The result is a two-stage method, which reduces simulation effort by removing designs that are likely to be infeasible or poorly performing at the end of the first stage, and goes on to return a subset of designs that satisfy one or more chance constraints and are within a fixed tolerance of the best design.

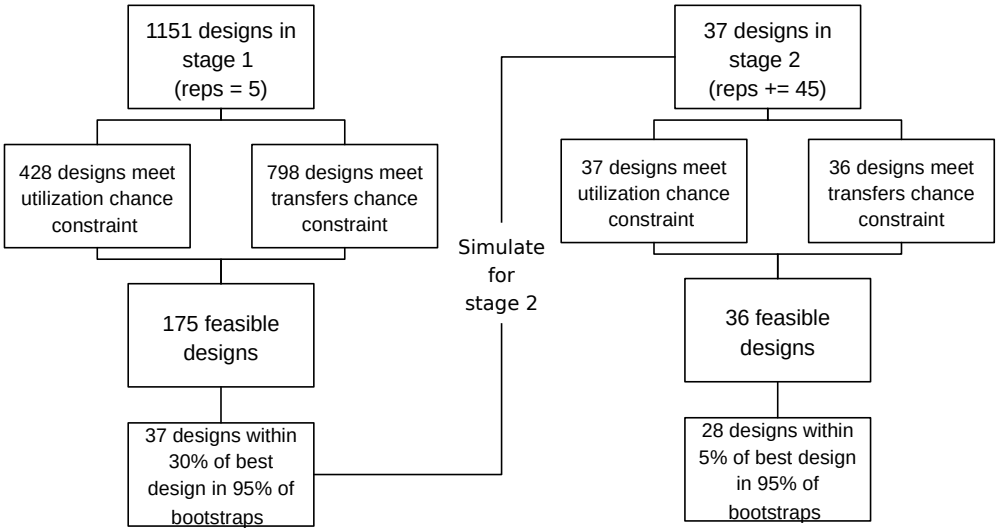

Fig. 4. Optimization procedure in applied example.

Table 2. Details of the top ten scenarios.

Total Beds	Bay Size	Number of Bays	Number of Singles	Waiting Time (hours)	Utilization	Transfers
48	0	0	48	4.9	81.0	0.0
48	3	3	39	4.9	81.0	12.7
48	3	4	36	4.9	81.0	16.5
48	3	5	33	4.9	81.0	20.9
48	3	6	30	4.9	81.0	24.4
48	3	7	27	4.9	81.0	29.9
48	3	8	24	4.9	81.0	35.3
48	3	9	21	4.9	81.0	40.2
48	4	2	40	4.9	81.0	12.5
48	4	3	36	4.9	81.0	17.3

The paper includes a full comparison of the method on test problems within subset selection and these results suggest that BootComp performs well in simulation studies where CRN have been employed and have either fully synchronized or induced a positive correlation. Our experience is that many simulation practitioners use commercial simulation software that employs CRN. We have also demonstrated our procedure's use on a complex simulation from health care, showing how it can be used in practice. The first stage appears to be very effective at significantly reducing the number of designs in the second-stage comparison.

We have made available an efficient Python implementation of BootComp. We hope that this will encourage further testing of the method and subsequent improvements as well as its use in practice. The framework exploits rudimentary parallelism and for large solution spaces (or for

those where the user wishes to run a very high number of bootstraps) it can make use of multi-core CPUs to reduce runtime. Parallelization of the Constraints Bootstrap is straightforward as it deals with marginal probabilities whereas parallelization of the Quality Bootstrap is more difficult. Parallelizing these bootstrap routines does not give substantial increases in efficiency compared with parallelizing the simulation.

One of the disadvantages of the method in its current form is that the user is required to decide on a number of different parameters for threshold probabilities, both for satisfying chance constraints and for being within a given tolerance of the best design. In the second stage these parameters have some intrinsic meaning but setting their values in the first stage is not straightforward and may require some experimentation. We suggest $\gamma_1 < \gamma_2$. Parameters α_1 and β_1 should generally be chosen to be greater than α_2 and β_2 to ensure that competitive designs are not discarded in stage 1. In the applied example considered here, we found that increasing the value of γ_1 (the threshold value for failing the chance constraints in stage 1) from 0.7 to 0.8 led to a different set of elite designs. More experimentation is needed to determine more clearly how the choice of these parameters affects the final design. One of the benefits of this method is that there is scope for experimentation with stage 1 parameters before running stage 2 replications; for example, if the initial choice of parameters suggest either very few or very many designs remain in the process.

Further work could consider how this two-stage approach could be adapted to identify the Pareto set of designs rather than treating the secondary objectives as chance constraints. There is some appeal in doing this, particularly for problems in which there is no objective that is clearly more important than the others.

REFERENCES

S. Andradóttir and S-H. Kim. 2010. Fully Sequential Procedures for Comparing Constrained Systems via Simulation. *Naval Research Logistics* 57 (2010), 403–421.

R.E. Bechhofer. 1954. A Single-Sample Multiple Decision Procedure for Ranking Means of Normal Populations with Known Variances. *Annals of Mathematical Statistics* 25 (1954), 16–39.

J.M. Bekki, B.L. Nelson, and J.W. Fowler. 2010. Bootstrapping-Based Fixed-Width Confidence Intervals for Ranking and Selection. In *Proceedings of the 2010 Winter Simulation Conference*, B. Johansson et al. (Eds.). IEEE, Piscataway, New Jersey, 1024–1033.

J. Branke, S.E. Chick, and C. Schmidt. 2007. Selecting a Selection Procedure. *Management Science* 53 (2007), 1916–1932.

C.-H. Chen. 1996. A Lower Bound for the Correct Subset-Selection Probability and its Application to Discrete Event Simulations. *IEEE Trans. Automat. Control* 41 (1996), 1227–1231.

C-H. Chen, D. He, M. Fu, and L.H. Lee. 2008. Efficient Simulation Budget Allocation for Selecting an Optimal Subset. *INFORMS Journal on Computing* 20 (2008), 579–595.

M.R. Chernick. 2007. *Bootstrap Methods: A Guide for Practitioners and Researchers.* John Wiley & Sons, Inc, New York.

S.E. Chick and K. Inoue. 2001a. New Procedures to Select the Best Simulated System Using Common Random Numbers. *Management Science* 47 (2001), 1133–1149.

S.E. Chick and K. Inoue. 2001b. New Two-Stage and Sequential Procedures for Selecting the Best Simulated System. *Operations Research* 49 (2001), 732–743.

F. Chingcuanco and C. Osorio. 2013. A Procedure to Select the Best Subset Among Simulated Systems using Economic Opportunity Cost. In *Proceedings of the 2013 Winter Simulation Conference*, R. Pasupathy et al. (Eds.). IEEE, Piscataway, New Jersey, 452–462.

S. Gao and W. Chen. 2015. A Note on the Subset Selection for Simulation Optimization. In *Proceedings of the 2015 Winter Simulation Conference*, L. Yilmaz et al. (Eds.). IEEE, Piscataway, New Jersey, 3768–3776.

S. Gao and W. Chen. 2016. A New Budget Allocation Framework for Selecting Top Simulated Designs. *IIE Transactions* 48 (2016), 855–863.

S.S. Gupta. 1965. On some Multiple Decision (Selection and Ranking) Rules. *Technometrics* 7 (1965), 225–245.

L. J. Hong, J. Luo, and B.L. Nelson. 2015. Chance Constrained Selection of the Best. *INFORMS Journal on Computing* 27 (2015), 317–334.

S.R. Hunter, E.A. Applegate, V. Arora, B. Chong, K. Cooper, O. Rincón-Guevara, and C. Vivas-Valencia. 2019. An Introduction to Multiobjective Simulation Optimization. *ACM Trans. Model. Comput. Simul.* 29, 1, Article 7 (2019), 36 pages. https://doi.org/10.1145/3299872

S.R. Hunter and R. Pasupathy. 2013. Optimal Sampling Laws for Stochastically Constrained Simulation Optimization on Finite Sets. *INFORMS Journal on Computing* 25 (2013), 527–542.

S-H. Kim and B.L. Nelson. 2006a. Selecting the best system. In *Handbooks in Operations Research and Management Science: Simulation*, Shane G. Henderson and Barry L. Nelson (Eds.). Vol. 13. Elsevier B.V., North Holland.

S-H Kim and B.L. Nelson. 2006b. Selecting the Best System. In *Elsevier Handbooks in Operations Research and Management Science: Simulation*, S. G. Henderson and B.L. Nelson (Eds.). Elsevier, Waltham, MA.

L.W. Koenig and A.M. Law. 1985. A Procedure for Selecting a Subset of Size m Containing the l Best of k Independent Normal Populations. *Communications in Statistics - Simulation and Computation* 14 (1985), 719–734.

A.M. Law and W.D. Kelton. 2000. *Simulation Modeling & Analysis* (3rd ed.). McGraw-Hill, Inc, New York.

L.H. Lee, E.K. Pen Chew, S. Teng, and D. Goldsman. 2010. Finding the Non-Dominated Pareto Set for Multi-Objective Simulation Models. *IIE Transactions* 42 (2010), 656–674.

S. Lee and B.L. Nelson. 2014. Bootstrap Ranking and Selection Revisited. In *Proceedings of the 2014 Winter Simulation Conference*, A. Tolk et al. (Eds.). IEEE, Piscataway, New Jersey, 3857–3868.

S. Lee and B.L. Nelson. 2015. Computational Improvements in Bootstrap Ranking and Selection Procedures via Multiple Comparison with the Best. In *Proceedings of the 2015 Winter Simulation Conference*, L. Yilmaz et al. (Eds.). IEEE, Piscataway, New Jersey, 3758–3767.

S. Lee and B.L. Nelson. 2016. General-Purpose Ranking and Selection for Computer Simulation. *IIE Transactions* 48 (2016), 555–564.

T. Monks and C.S.M. Currie. 2018. Practical Considerations in Selecting the Best Set of Simulated Systems. In *Proceedings of the 2018 Winter Simulation Conference*, M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson (Eds.). IEEE, Piscataway, New Jersey, 2191–2200.

T. Monks and C.S.M Currie. 2019a. CLAHRCWessex/BootComp: v1.0.0. https://doi.org/10.5281/zenodo.3381945

T. Monks and C.S.M Currie. 2019b. CLAHRCWessex/subset-selection-problem: v1.0.0. https://doi.org/10.5281/zenodo.3382147

B.L. Nelson and F.J. Matejcik. 1995. Using Common Random Numbers for Indifference-Zone Selection and Multiple Comparisons in Simulation. *Management Science* 41 (1995), 1935–1945.

R. Pasupathy, S.R. Hunter, N.A. Pujowidianto, L.H. Lee, and C-H. Chen. 2014. Stochastically Constrained Ranking and Selection via SCORE. *ACM Transactions on Modeling and Computer Simulation* 25 (2014), 1–26.

L. Pei, S. Hunter, and B.L. Nelson. 2018. A New Framework for Parallel Ranking & Selection Using an Adaptive Standard. In *Proceedings of the 2018 Winter Simulation Conference*, M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson (Eds.). IEEE, Piscataway, New Jersey, 2201–2212.

M.L. Penn and T. Monks. 2018. Generic Ward Configuration Simulation Model. https://doi.org/10.5281/zenodo.1468288

M.L Penn, T. Monks, A. Kazmierska, , and M. Alkoheji. 2018. Designing and Redeveloping Generic Models in Healthcare. In *Proceedings of the OR Society Simulation Workshop 2018*, A. Anagnostou et al. (Eds.). OR Society, Stratford-Upon-Avon, UK.

D.W. Sullivan and J.R. Wilson. 1989. Restricted Subset Selection Procedures for Simulation. *Operations Research* 37(1) (1989), 52–71.

J.R. Swisher and S.H. Jacobsen. 2002. Evaluating the Design of a Family Practice Healthcare Clinic Using Discrete-Event Simulation. *Health Care Management Science* 5 (2002), 75–88.

Y. Wang, L. Luangkesorn, and L.J. Shuman. 2011. Best-Subset Selection Procedure. In *Proceedings of the 2011 Winter Simulation Conference*, S. Jain et al. (Eds.). IEEE, Piscataway, New Jersey, 4315–4323.