Measuring the effectiveness and efficiency of simulation optimization metaheuristic algorithms

Benjamin G. Thengvall; Shane N. Hall; Michael P. Deskevich

Journal of Heuristics (2025) 31:12
https://doi.org/10.1007/s10732-025-09549-2

Measuring the effectiveness and efficiency of simulation optimization metaheuristic algorithms

Benjamin G. Thengvall[1] · Shane N. Hall[2] · Michael P. Deskevich[1]

Received: 25 January 2024 / Revised: 19 December 2024 / Accepted: 8 January 2025 /
Published online: 30 January 2025

Abstract

Metaheuristic algorithms have proven capable as general-purpose algorithms for solving simulation optimization problems. Researchers and practitioners often compare different metaheuristic algorithms by examining one or more measures that are derived through empirical analysis. This paper presents a single measure that can be used to empirically compare different metaheuristic algorithms for optimization problems. This measure incorporates both the effectiveness and efficiency of the metaheuristic algorithm, which is especially important in simulation optimization applications because the number of simulation runs available to the analyst (i.e., the run budget) can vary significantly with each simulation study. Therefore, the trade-off between the effectiveness and efficiency of a metaheuristic algorithm must be examined. This single measure is especially useful for multi-objective optimization problems; however, determining this measure is non-trivial for two or more objective functions. Additional details for calculating this measure for multi-objective optimization problems are provided as well as a procedure for comparing two or more metaheuristic algorithms. Finally, computational results are presented and analyzed to compare the performance of metaheuristic algorithms using knapsack problems, pure binary integer programs, traveling salesman problems, and the average results obtained across a diverse set of optimization problems that include simulation and multi-objective optimization problems.

✉ Shane N. Hall
shane.hall1@montana.edu

Benjamin G. Thengvall
thengvall@opttek.com

Michael P. Deskevich
deskevich@opttek.com

1 OptTek Systems, Inc., Boulder, CO, USA

2 Jake Jabs College of Business and Entrepreneurship, Montana State University, Bozeman, MT, USA

Keywords Binary integer program · Efficiency and effectiveness of metaheuristics · Knapsack problem · Metaheuristic performance measures · Multi-objective optimization · Simulation optimization · Traveling salesman problem

1 Introduction

Measures to compare the effectiveness and efficiency of optimization algorithms are critical to determining the best methods to solve an optimization problem or class of problems. Many real-world optimization problems require complex model formulations with non-linear relationships and discrete decision variables. Exact algorithms that guarantee an optimal solution when given a feasible problem instance can be impractical even for simplified and/or relaxed models given the excessive execution time of these algorithms. Moreover, the models are often impractical if there is great uncertainty in the model parameters. Heuristic algorithms that relax the requirement for optimality offer a practical alternative to exact algorithms (Arts et al. 2003 and Reeves 1995). Likewise, simulations provide a means to model uncertainty in model parameters, offering a practical alternative to pure math programming formulations (Banks et al. (2009) and Law (2024)).

In the case of heuristic algorithms, the trade-off between finding very good solutions eventually (effectiveness) and finding good solutions quickly (efficiency) must constantly be balanced. Indeed, an enhancement to a heuristic, usually in the form of a metaheuristic (Glover 1986), in order to find better solutions almost always requires more computational overhead, which increases the time and memory to execute the algorithm. Metaheuristic techniques have strategies that allow them to escape local optima (solutions that are optimal for their immediate neighborhood) and search for a global optimum (solutions that are optimal over the entire solution space). Metaheuristics have consistently demonstrated their effectiveness and efficiency when a simulation is used to calculate the value of an optimization objective function (see Fu (2015), Fu et al. (2005), Lee et al. (2013), and Swisher et al. (2000)). Figure 1 summarizes a metaheuristic approach to simulation optimization where the metaheuristic sends input values to the simulation model, which is then executed in order to return output values back to the metaheuristic. In this paper, executing one set of

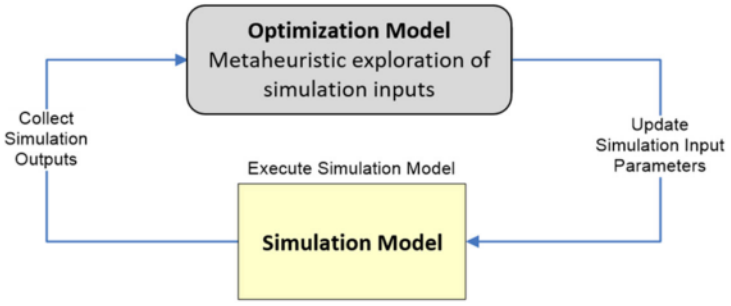

Fig. 1 Simulation optimization with metaheuristics

provided inputs from the metaheuristic will be referred to as one simulation trial (or iteration). This iterative cycle is executed until some stopping criteria is met and the input values that maximize (or minimize) the simulation outputs is returned. For example, a carwash owner may wish to determine the number of employees and service lanes that maximize profit or minimize customer wait time. A viable approach is to simulate the carwash with many different inputs (number of employees and service lanes), record the outputs (profit and average wait time) and then select the inputs that optimize the outputs.

In many practical applications of simulation optimization, executing the simulation model is computationally expensive. See Evans et al. (2017), Hall et al. (2019), Kannon et al. (2015), Onal et al. (2016), Ruane et al. (2023), Soykan and Rabadi (2022), Tekin and Sabuncuoglu (2004), Wade (2019), and Zais and Laguna (2017) for real-world simulation optimization applications. The number of simulation trials in Fig. 1 that can be run is limited based on computational resources available and the computational resources required for each trial. This computational limitation is aggravated further when Monte Carlo replications are required for the simulation model. Therefore, an analyst's simulation run budget will vary for different studies and can be limited by decision or analysis time, hardware, the number of optimization objectives, inputs and outputs, or required precision. This dependence emphasizes the importance of the trade-off between finding good solutions and finding these good solutions quickly. Additionally, different metaheuristic approaches may require different computational resources. A simple greedy approach may determine the next solution to evaluate very, very rapidly. However, a more complex approach, say one in which a metamodel is created to make response surface estimations, may take more time to choose the next solution to evaluate. The methodology and procedures described in this paper to compare algorithm effectiveness and efficiency can be applied in a similar manner using iterations or execution time. Using simulation trials to track effort controls for the many variables that dictate execution time and enables the comparison of metaheuristic algorithms between different researchers and across different computing environments. However, for research in a controlled and standardized computational environment, especially when comparing algorithms that have more variation in execution time as compared to the simulation models being optimized, the approaches in this paper may be applied using execution time rather than simulation trials for comparison.

Comparative measures that balance both the effectiveness and efficiency of a metaheuristic algorithm are especially important in simulation optimization applications. In many cases, reported results provide only the global best, the best-known solution, or the best solution found in a certain number of simulation trials. However, for a general-purpose algorithm these measures alone are insufficient. What is needed is a way to measure the best result obtained after any number of trials. Finally, it is important to note that a primary benefit of simulation model optimization is the ability to consider uncertainty. For stochastic simulations, replications are run and statistical measures of goodness (i.e., mean, median, percentile, standard deviation, etc.) are collected for each simulation trial. Commonly for simulation optimization problems, mean output values are used to measure performance of the modeled system or process. These same mean output values are used for metaheuristic algorithm solution

comparison in this paper. A researcher should consider the variability in simulation output values when considering how many replications to run at each trial point for algorithm comparison. Confidence intervals around mean values can be constructed based on the responses at each iteration, and in the figures that follow, one could create bands rather than lines illustrating a chosen confidence width. Of course, these confidence widths can be made arbitrarily small with enough replications performed at each trial point. A researcher will have to carefully consider their desired accuracy, the observed variability, and the best number of replications to run for a particular analysis.

This paper presents a comparative measure that incorporates the effectiveness and efficiency of a metaheuristic algorithm and examines this measure empirically. The paper is organized as follows. Section 2 provides a review on comparative measures in the metaheuristic literature. Section 3 discusses the limitations of the most common effectiveness measure and presents a single measure that considers both effectiveness and efficiency and provides the methodology for computing this measure. Section 4 provides illuminating computational results for different types of optimization problems, and, finally, Sect. 5 concludes the paper and discusses future research.

2 Literature review

This section discusses the common measures that are used to compare the effectiveness and efficiency of metaheuristic algorithms and summarizes the related literature for simulation optimization and multi-objective performance evaluation. It also briefly reviews other approaches to simulation optimization.

Coffin and Saltzman (2000) describe two common approaches for evaluating an algorithm's performance: (1) theoretical analyses of worst-case and average case performance expressed as a "big-O" asymptotic function of the problem size and (2) empirical analyses of implementing the algorithm and measuring its actual running time and solution quality for a selected set of problem instances. Hooker (1994) establishes a need to evaluate algorithmic performance empirically (or computationally) versus worst-case or average-case deductive theoretical results because these results offer little insight into how the algorithm will perform in practice. This is particularly true for metaheuristic algorithms where deductive theoretical results are not particularly useful to the practitioner. Therefore, this paper demonstrates an empirical analysis approach for comparing metaheuristic algorithms used for single-objective, multi-objective, and simulation optimization problems.

Greenberg (1990) discusses the role of computational testing in reporting original research results in operations research (OR) and computer science (CS). Metaheuristic algorithms fit squarely in this OR-CS space because they, by definition, solve operations research problems using computers. Greenberg (1990) suggests the reasons for computational testing are to demonstrate correctness, solution quality, computational speed, and robustness of an algorithm. Empirical comparisons of algorithms in the literature primarily use measures that address these areas put forth by Greenberg, especially solution quality and computational speed. See Laguna and Martí (2005), Jacobson et al. (2005), Hall et al. (2008), Fuellerer et al. (2009), Singhal et al.

(2011), Kumbharana and Pandey (2013), Antosiewicz et al. (2013), Melouk et al. (2014), McNabb et al. (2015), Geyer et al. (2016), and Asih et al. (2017) for a sample of research articles that empirically compare algorithm performance for metaheuristic and simulation optimization methods. Many of these articles provide empirical studies for computationally challenging optimization problems such as the Traveling Salesman Problem (TSP) and the related Vehicle Routing Problem (VRP). Finally, Barr et al. (1995) presents a comprehensive list of attributes that are meaningful when comparing heuristics such as fast, accurate, robust, simple, high-impact, generalizable, innovative, revealing, and theoretical. The authors define these terms and provide several additional literature references that demonstrate these attributes. This paper emphasizes the fast and accurate attributes given their supreme importance in practice.

Notable recent and related work in performance metrics for simulation optimization and multi-objective methods is now summarized. Eckman et al. (2023a) presents stochastic area under progress curves as a performance measure for comparing simulation optimization algorithms for single-objective problems. Hunter and Ondes (2023) analyze performance indicators for assessing exact algorithms for solving multi-objective simulation optimization (MOSO) problems and emphasizes the challenges in developing performance measures for MOSO problems. Audet et al. (2021) provide a comprehensive review of 57 performance indicators to measure the quality of an approximate Pareto frontier, which includes a hypervolume indicator. This paper extends the work of Eckman et al. (2023a), Audet et al. (2021), and Hunter and Ondes (2023) with an implementation of the area under the progress curve performance measure for metaheuristic algorithms and provides a procedure for calculating this measure and comparing multiple metaheuristic algorithm, which have proven effective in solving MOSO problems (see Hall et al. (2019), Melouk et al. (2014), and Wade (2019)) even if not to global optimality as explored in Hunter and Ondes (2023). Finally, computational testing requires a broad set of test problems. Many online test suites exist and offer researchers access to well-known problems (see Surjanovic and Bingham (2024) for the Virtual Library of Simulation Experiments, Pasupathy and Henderson (2006) for a motivation of the Simulation Optimization Library (SOL) (2024) test suite, and Eckman et al. (2023b) for the most recent description of the SOL test suite). The results provided in Sect. 4 rely on a custom test suite collected and developed by OptTek Systems, Inc. that includes over 3000 single-objective, multi-objective, and simulation optimization problems.

In addition to metaheuristic algorithms, common approaches to simulation optimization also include brute force methods such as trial-and-error and enumeration (Hall et al. 2024), design of experiments (Montgomery 2019), and derivate-free algorithms (Conn et al. 2009 and Audet and Hare 2017). Brute force approaches are only computationally tractable for problems with a small number of decision alternatives, and hence, are less practical for real-world applications. Design of experiments is often used to characterize simulation model sensitivities and behaviors and to develop surrogate (or meta) models of the decision space. Surrogate models can be coupled with metaheuristic or derivative-free algorithms to provide approximate solutions for simulation optimization problems. Common derivative-free algorithms use interpolation and surrogate models to find good solutions to a related optimization problem.

In summary, solution quality metrics generally measure how close the best objective function value found by the metaheuristic algorithm is to the best-known objective function value for the problem instance. When comparing metaheuristics, the metaheuristic with the best solution quality is victorious. Therefore, solution quality is the most common effectiveness measure for metaheuristic comparison. Likewise, computational speed generally refers to the running time of the metaheuristic algorithm. When comparing metaheuristics, the metaheuristic requiring the least time is triumphant; and hence, computational speed is the most common efficiency measure.

3 Comparative measure for effectiveness and efficiency

This section discusses the limitations of considering solution quality alone when evaluating a metaheuristic algorithm's performance and presents a single measure that considers both solution quality (effectiveness) and computational speed (efficiency). Details for calculating this measure for single- and multi-objective optimization problems are provided, and, finally, a procedure for comparing two or more metaheuristic algorithms is also presented.

3.1 Limitations of solution quality measure

There are two key limitations to considering solution quality alone. First, solution quality gives no credence to the running time of each metaheuristic algorithm since in most empirical research, each metaheuristic algorithm is run for some set time and the algorithm that finds the best objective function value is determined the best; however, determining this time can be difficult and often seems arbitrary. When comparing metaheuristic algorithms for general simulation models, the researcher has no knowledge about how much time an analyst is willing to dedicate to finding a good solution to their specific simulation optimization problem. Therefore, it is important to have a comparative measure that rewards metaheuristic algorithms for finding the best solutions possible at any point in their execution. For complex, practical simulation models, the time spent in a simulation optimization process is generally dominated by the time it takes to execute the requested simulation trials. That is, the metaheuristic algorithm that generates simulation trials to execute takes a fraction of the time that the simulation takes to execute. Therefore, measuring the quality of the best solution found after any number of simulation trials is a way to compare results without having to account for differences in the hardware on which the experiments were executed, the complexity of the problem instance, or type and fidelity of the simulation model. For example, simulation model execution times vary significantly with some models performing a trial in seconds while other models take several hours to complete one trial. Furthermore, as noted in Sect. 1, simulations are usually stochastic, and hence, one trial often includes many Monte Carlo replications that measure the variation in the simulation outputs (or objective function values in a simulation optimization problem).

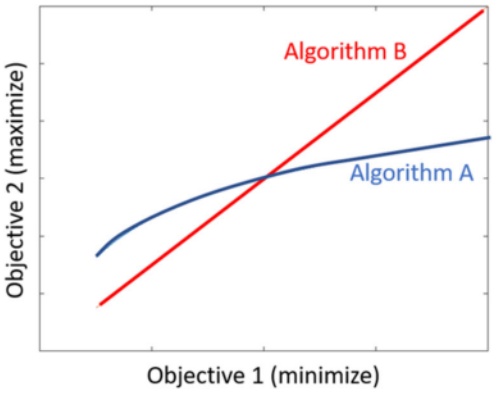

Fig. 2 A pair of incompatible Pareto frontiers

Second, considering solution quality alone does not generalize to multi-objective optimization problems which results in a Pareto frontier of solutions that are "equally" good (Ehrgott 2000). Most simulation optimization applications are multi-objective in nature since there are many simulations outputs that may be optimized. To illustrate, consider the example in Fig. 2 that displays two estimated Pareto frontiers from two different metaheuristic algorithms, Algorithm A (represented by the blue curve) and Algorithm B (represented by the red line). The goal in this multi-objective problem is to maximize Objective 2 and minimize Objective 1 (solutions which are up and to the left are preferred), which means some Pareto solutions from Algorithm A are preferred to many Pareto solutions from Algorithm B and vice versa. Therefore, two estimated Pareto frontiers can, in general, be incomparable, which highlights the importance of having a comparative measure that also applies to multi-objective problems. Furthermore, the Pareto front may change as better solutions are discovered, and hence, to measure algorithmic efficiency and effectiveness, it is desired to have the best Pareto frontier after any number of simulation trials.

3.2 Single comparative measure for effectiveness and efficiency

To define a single measure to compare metaheuristic algorithms, consider an optimization problem O with p objective functions. Transforming O to a new optimization problem O^+ with $p + 1$ objective functions, where the additional objective function is to minimize the simulation trials required to find a solution of a given quality for optimization problem O overcomes the first limitation by accounting for run time. However, the second limitation of comparing the quality of Pareto frontiers remains.

To address this second limitation, an "area under the curve" measure is proposed. As motivation, consider a single-objective minimization problem. Adding the additional objective to minimize the simulation trials (as a measure for computational speed) to find a solution of a given quality, it is now a two-objective problem yielding a Pareto frontier. Figure 3 shows an example of a hypothetical run of two different metaheuristic algorithms, Algorithm 1 (red) and Algorithm 2 (blue).

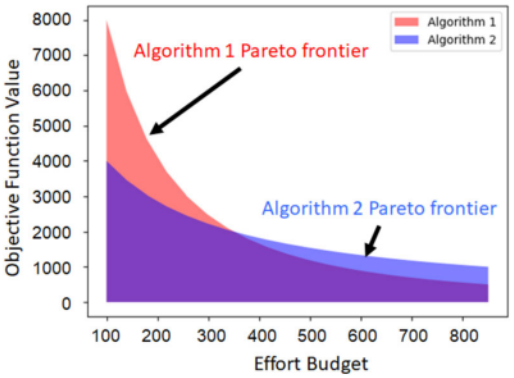

Fig. 3 Example areas for two metaheuristic algorithms

Initially, the blue algorithm (Algorithm 2) is finding much better solutions than the red algorithm (Algorithm 1). However, given enough effort (as measured by iterations or execution time), Algorithm 1 eventually finds better solutions than Algorithm 2. Therefore, while neither algorithm strictly dominates the other, it is possible to compare the two algorithms by comparing the area of the red region for Algorithm 1 with the area of the blue region for Algorithm 2. The algorithm with the smallest area is deemed best from a solution quality and computational speed perspective since this area provides a composite metric that encompasses both criteria.

This measure can be calculated for any general optimization problem O, since any objective function, $f(x)$, to be maximized may be minimized using $-f(x)$. In Fig. 3, the objective function value is bounded below by zero; however, the area under the curve measure can always be determined by selecting a suitable lower bound to ensure every objective function value lies above this bound. Furthermore, this lower bound must not be too small to obscure the variation between the algorithms. Using the smallest known value, the true minimum value, or a lower bound found by solving a relaxation of the optimization problem are viable options for a suitable lower bound for each objective function.

For multi-objective optimization problem O with $p \geq 2$, the area under the objective function values vs. run-time curve equates to computing the volume under a $(p + 1)$-dimensional surface for problem O^+. Note the volume is a hypervolume when $p \geq 3$. While the computations are more complicated for volume and hypervolumes, the interpretation is the same: an algorithm with the smallest area, volume, or hypervolume is the best from a solution quality and computational speed perspective.

In a single-objective optimization, the area is the Riemann sum of the best-known objective value at each trial. By definition, the sequence of best-known values is monotonic (decreasing for minimization, increasing for maximization problems), and the sum of the area of each rectangle is the "area under the curve" measurement. The rectangles do not overlap, making the area calculation straightforward. Computing the (hyper)volume is non-trivial for $p \geq 2$. The (hyper)volume under the frontier is a measurement of the non-dominated volume (which will be a cuboid) of each frontier point which do not necessarily have monotonic behavior in all dimensions,

and hence, the non-dominated region (or cuboids) of multiple frontier points may have gaps resulting in missing volume and/or overlap and that volume must not be double counted. To account for this, each pair of non-dominated cuboids (determined by each frontier point) needs to be considered. For $p \geq 2$, a modified line-scan algorithm presented by Shamos and Hoey (1976) is adapted for each objective in succession to accumulate the non-dominated volume under the frontier.

Additional details for this modified line-scan algorithm follow and assume minimization of objective functions with each objective function normalized to a [0, 1] scale. The line-sweep algorithm in two dimensions (when $p = 2$) is first described to show that it is identical to the Riemann sum of rectangles. This line-scan algorithm is easier to visualize in higher dimensions if dominated volume is computed and then this volume is subtracted from one (the total volume of the normalized bounding cube) to get the non-dominated volume. By convention, let the x-axis be the first objective function (denoted by O1) and the y-axis be the second objective function (denoted by O2). Further, let $X = 1–O1$ and $Y = 1–O2$, which are the complement axes for O1 and O2. The area under the curve in this coordinate system is the dominated volume, V_D, and hence, the non-dominated volume, $V_{ND} = 1–V_D$. For example, consider the Pareto frontier consisting of the nine non-dominated points displayed together with its complement axes X and Y as displayed in Fig. 4 and observe visually that $V_{ND} = 1–V_D$.

The line-scan algorithm starts by sorting all non-dominated points in increasing order by X such that $X_i < X_j$ when $i < j$. This is an $O(N \log(N))$ operation for N non-dominated points. The area under the curve is accumulated by scanning a line from X_N to X_1 as shown in the following pseudocode:

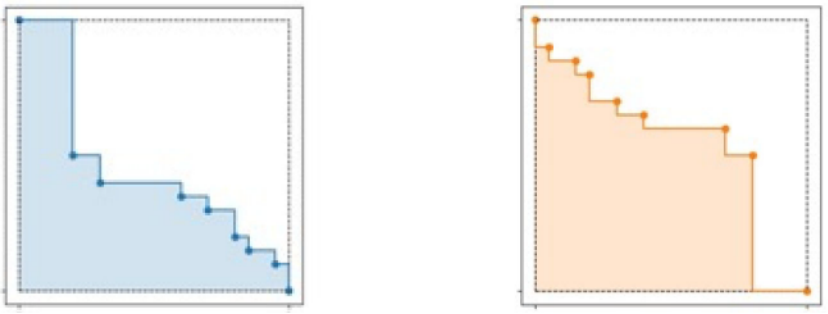

Fig. 4 Example 2-dimensional Pareto frontier with complementary axes

***Line-scan Algorithm* (when $p = 2$)**

```
Sort(X, Y)          sort the non-dominated points so they are in the order of increasing X
V_D = 0             initialize dominated area
i = N               start from the last non-dominated point
While i > 1         calculate the area between non-dominated points and add V_D
    dx = X_i – X_i-1    find the width of the i^th rectangle
    dy = Y_i            find the height of the i^th rectangle
    V_D = V_D + (dx*dy) accumulate the area of the i^th rectangle
    i = i – 1           increment to next rectangle.
```

Referring to Fig. 4, the line-scan algorithm can be visualized as a vertical line with a height of dy starting at 0 and sweeping out an area between X_i–$X_{i\text{-}1}$ (or a width of dx) and accumulating the area of each Reimann rectangle.

This algorithm generalizes to p-objective functions. The pseudocode for three objective functions and complement axes X, Y, and Z follows.

***Line-scan Algorithm* (when $p = 3$)**

```
Sort(X, Y, Z)       sort the non-dominated points so they are in the order of increasing X
V_D = 0             initialize dominated volume
i = N               start from the last non-dominated point
While i > 1         calculate the area between non-dominated points and add V_D
    dx = X_i – X_i-1    find the X-length scanned
                        sort the last non-dominated points so they are in the order of increasing Y
    Sort(Y(i,…N) , Z(i,…,N))
    Area = 0            initialize the area of Y-Z plane
    dz = 0              initialize the length of the swept line in Z dimension
    j = N               start from the last non-dominated point (as before)
    While j > 1         calculate the area of the Y-Z plane
        dy = Y_j – Y_j-1    find the Y-length swept
                            accumulate the Z-length swept
        dz = max(dz, Z(j))
                            accumulate the area of the Y-Z plane
        Area = Area + (dy*dz)
        j = j – 1           increment to next plane
                            accumulate the volume
    V_D = V_D + (Area*dx)
    i = i – 1           increment to next cuboid.
```

The three-dimensional line-scan algorithm starts with a line scan from X_N to X_1 yielding a plane along the Y dimension creating a nested inner while loop. The sort operation for Y and Z is only needed for the last non-dominated points from i to N in

the order of increasing Y making the inner while loop another line scan for Y. Since the algorithm performs the inner loop for each non-dominated point, the Y sort is done by keeping the list sorted and inserting the new non-dominated point in $O(\log(N))$ time resulting in an algorithm computational complexity of $O(N^2 \log(N))$. As more objective functions are considered (i.e., for $p > 3$), the line-scan algorithm can recursively be generalized to p dimensions with an algorithm computational complexity of $O(N^{p-1} \log(N))$, which is computationally tractable to recompute for every simulation trial when p is reasonably small. Fortunately, in practice, most multi-objective optimization problems only have two or three objective functions since more objectives make the analysis and interpretation of the Pareto frontier challenging.

Therefore, a procedure for comparing two or more metaheuristic algorithms using a composite metric that captures both the effectiveness (solution quality) and efficiency (computational effort) of the algorithms is as follows:

1. Run each metaheuristic algorithm A_i, $i \geq 2$, for T simulation trials (T should be enough trials to represent most practical applications). At each trial $t \leq T$ when a new best solution is found (a solution that has a better objective function value than any previous objective function value or for multi-objective problems, a new non-dominated solution), record new best solution and trial t.
2. Select a bounding $(p+1)$-dimensional cube that encompasses every solution found in step 1.
3. For each metaheuristic algorithm A_i, compute the volume (or hypervolume), V_i, using the line-scan algorithm described above between the Pareto frontier (including the trial-of-solution objective) of all non-dominated solutions found by the algorithm and the bounding cube in step 2.
4. (Optional) Normalize by the volume (or hypervolume) of the bounding cube, V_{Cube}, in step 2, i.e., V_i/V_{Cube} for all i.
5. Define the metaheuristic algorithm A_i with the smallest volume V_i (or V_i/V_{Cube}) as the best.

To conclude this section, it is worth noting that this measure and calculation procedure is applicable to other exact or heuristic optimization algorithms whether deterministic or stochastic. However, the tradeoff between solution quality and computational efficiency is most relevant when the requirement for optimality is relaxed which, by definition, applies to heuristics; and hence, different metaheuristic algorithms can find very different solutions even when only minor changes in algorithmic design exist. The focus of this paper on metaheuristic algorithms is based on extensive empirical evidence that these types of algorithms are an effective solution method for deterministic optimization problems, single-objective simulation optimization problems, and MOSO problems. This paper also demonstrates a practical way for metaheuristic and optimization tool developers to systematically evaluate if new modifications and improvements to their tools show promising results for a diverse set of problem types of varying complexity. It is worth reiterating that analytic caution is warranted when using this methodology for models that measure uncertainty in output such as stochastic simulation models. In these cases when a statistic output value is used for metaheuristic algorithm solution comparison, researchers and practitioners should consider the variability in simulation output values when considering

how many replications to run at each trial point for algorithm solution quality comparison. Confidence intervals around statistical output values should be constructed based on the responses at each iteration and then used to determine if two solutions are statistically different from one another.

4 Computational results

Using the single comparative volume measure just presented, this section provides computational results for several different optimization problems that demonstrate its value in comparing metaheuristic algorithms. These examples use a general-purpose metaheuristic for simulation optimization called OptQuest (OptTek Systems 2024). OptQuest is a commercially available simulation optimization library that is included in many commercial simulation packages. This metaheuristic employs scatter-search (Glover et al. 2000) and tabu search (Glover and Laguna 1997) techniques along with several other solution generators depending on the characteristics of the problem presented (Laguna 2011). To test the robustness of this general-purpose metaheuristic, a test suite of over 3000 problems consisting of simulation models and linear, integer, combinatorial, and non-linear optimization problems is used. This metaheuristic is routinely updated to address performance shortfalls found during testing and incorporate proven innovations published in the literature. When considering changes to OptQuest, the performance of the updated metaheuristic, denoted the new metaheuristic, is tested using the single volume measure presented in Sect. 3. Specifically, the following examples compare the performance of the original metaheuristic, denoted the old metaheuristic, with an updated new metaheuristic using the single volume measure. Once the normalized single volume measure is calculated for the new and old metaheuristic (denoted V_{New} and V_{Old}, respectively), the percent improvement is calculated (i.e., $(V_{Old} - V_{New})/V_{Old}$) and is used as the final measure of whether the new metaheuristic outperforms the old metaheuristic. Note a negative percent improvement implies the new metaheuristic does not outperform the old metaheuristic for the given problem since $V_{New} > V_{Old}$.

When considering changes to a mature algorithm that improves performance across a wide variety of test problems, it is very difficult to find enhancements that improve all test cases. As will be demonstrated in the next section, when just measuring the best value found in a fixed number of trials, significant changes in performance can be overlooked. Using this volume under the curve approach to measure algorithmic changes offers a more holistic approach that considers the very important practical necessity to find the best solutions possible after any number of simulation trials.

4.1 Knapsack problem examples

This example analyzes two multi-dimensional knapsack problem instances where, for a given set of n items, each with a value v_j and a weight w_j, $j = 1,2,\ldots,n$, and m knapsacks, each with capacity W_k, $k = 1,2,\ldots,m$, the objective is to select the number of each item, x_j, an integer, to place into each knapsack such that the total value is

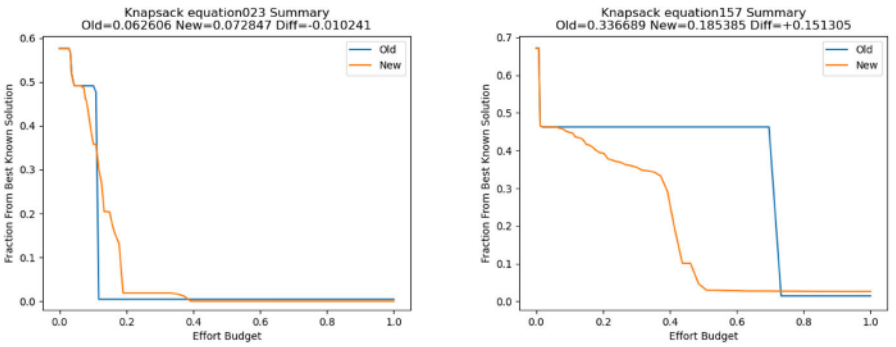

Fig. 5 Solution quality vs. computational effort for knapsack problem instances

maximized (i.e., $\sum_{j=1}^{n} v_j x_j$) subject to each knapsack being within capacity (i.e., $\sum_{j=1}^{n} w_j x_j \leq W_k$ for all $k = 1,2,\ldots,m$). For these test cases, the objective function value is evaluated as the result of a simulation trial, such that these problems cannot be solved directly as a mixed integer program.

Figure 5 shows the single measure comparison for two knapsack problem instances drawn from the authors' test library. The y-axes in Fig. 5 (and for the following computational figures) show the fraction of the distance from the best known solution based on the range between the best known and worst known solutions. For example, the y-axis for the left graph (problem instance 23) in Fig. 5 ranges from 0.0 to 0.6. The x-axes in Fig. 5 (and for the following computational figures) show the effort budget, normalized for each test case and displayed from 0 to 1. As discussed in Sect. 1, this effort budget can represent iterations or execution time. The problem instance on the left (problem instance 23) has 49 items and 5 knapsacks, whereas the problem instance on the right (problem instance 157) has 18 items and 3 knapsacks. For problem instance 23, the old and new metaheuristic performed equally well early on, then for a period the new metaheuristic outperformed the old, then for a period the old outperformed the new, and then around 40% of the way through the test the new metaheuristic found the best known solution which the old metaheuristic never found. Using best solution value found, the new metaheuristic outperforms the old; however, using the percent improvement in volume that considers solution quality and computation speed (by effort budget), the old metaheuristic slightly outperforms the new metaheuristic. For problem instance 157, both metaheuristics are initially equivalent, and the old metaheuristic finds the best overall solution, but the new metaheuristic finds much better solutions for large portions of the test. The new metaheuristic performs better than the old metaheuristic on this test case with a more than 15% improvement using the proposed metric for comparison. The values displayed above each graph in Fig. 5 (and for the following computational figures) display the values for New = V_{New}, Old = V_{Old}, and Diff = the percent improvement = $(V_{Old} - V_{New})/V_{Old}$.

Generating these graphs also provides visual information to compare how algorithmic changes impacted the best solution available after any level of effort. For example, in problem instance 23 the old metaheuristic provided better performance

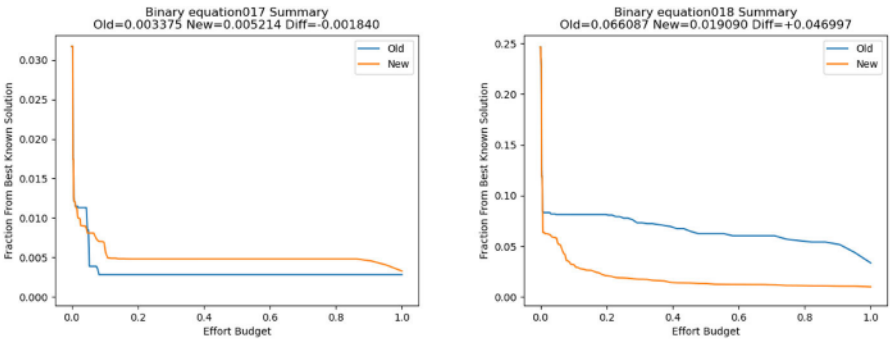

Fig. 6 Solution quality vs. computational effort for binary IP problem instances

between about 12% and 38% of the effort budget, but the new metaheuristic found a better overall solution. The visual depiction of the results may lead the researcher to explore whether there is a way to combine the good early performance of the old metaheuristic with the consistent improvement of the new metaheuristic.

4.2 Pure binary integer program (IP) examples

This example compares two pure binary IP instances where all decision variables are required to be 0 or 1. As with the knapsack problems, the objective function value for each test case is evaluated as the result of a simulation trial.

Figure 6 shows the single measure comparison for two pure binary integer program instances. The problem instance on the left (problem instance 17) has 11 binary decision variables with 8 constraints, whereas the problem instance on the right (problem instance 18) has 84 binary decision variables with 34 constraints. For problem instance 17, the new metaheuristic outperforms early in the test, but then the old metaheuristic finds better solutions all the way until the end. However, both metaheuristics find solutions very close to the best known solution very early in the search and so the final distance in the performance metric is less than 0.2%. The results for problem instance 18 show that the new metaheuristic finds better solutions very early in the test and outperforms the old metaheuristic throughout the test. The percent improvement in volume for problem instance 18 over the test is approximately 4.7%.

4.3 Traveling salesman problem (TSP) examples

TSPs are ordering problems with a set of c cities $j = 1,2,\ldots,c$. Distances or times between all city pairs are provided and distances or times can be different each direction between a pair of cities. For example, d_{12} would represent the distance from city 1 to city 2 and d_{21} would represent the distance from city 2 to city 1. The goal is to find the shortest circuit that visits each city once and returns to the starting city. Vehicle routing problems (VRP) are generalizations of TSP problems where there is a single depot, and multiple vehicles are used to create circuits starting and ending at

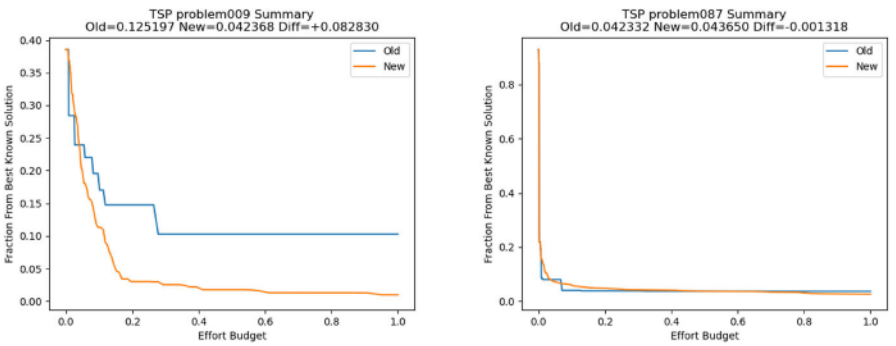

Fig. 7 Solution quality vs. computational effort for TSP problem instances

the depot to visit all cities. Many custom metaheuristics exist for the TSP and VRP due to their computational intractability with exact methods (see Glover and Laguna (1997), Fuellerer et al. (2009), Singhal et al. (2011), Kumbharana and Pandey (2013), McNabb et al. (2015), and Asih et al. (2017)).

Figure 7 shows the single measure comparison for two TSP problem instances. Problem instance 9 has 9 cities with 362,880 solutions and problem instance 87 has 87 cities with more than 2.1×10^{132} solutions. For problem instance 9, the old metaheuristic outperforms very early in the test, but then the new metaheuristic finds better solutions all the way until the end. The results for problem instance 87 show both metaheuristics performing very similarly with the old metaheuristic slightly outperforming (with a smaller area) even though the new metaheuristic finds a better solution at the very end.

4.4 Average performance results for a diverse problem set

These results show the average performance over a test suite of 60 different optimization problems. This demonstrates an analysis that examines the robustness of the metaheuristic algorithm by evaluating its performance on a broad collection of optimization problems that include linear, integer, and non-linear problems. This collection of problems is a diverse subset of the types of test problems used for internal testing of the OptQuest metaheuristic.

Figure 8 compares the new and old metaheuristic average solution quality performance across the test set of 60 problems. The results from the test problems are aggregated so that fair comparisons can be made across the set. Specifically, the problems are aggregated into a mean performance metric for the new and old metaheuristic by creating a new pseudoproblem where the value at each unit of effort is the mean volume under the frontier computed across all problems at the same level of effort. As discussed in Sect. 3.2, the objective values are normalized on [0, 1] between the best and worst known values, so aggregating the mean volume provides a useful mean performance metric of the metaheuristic algorithm across all problems. When comparing problems of varying complexity, some easier problems require less effort than others.

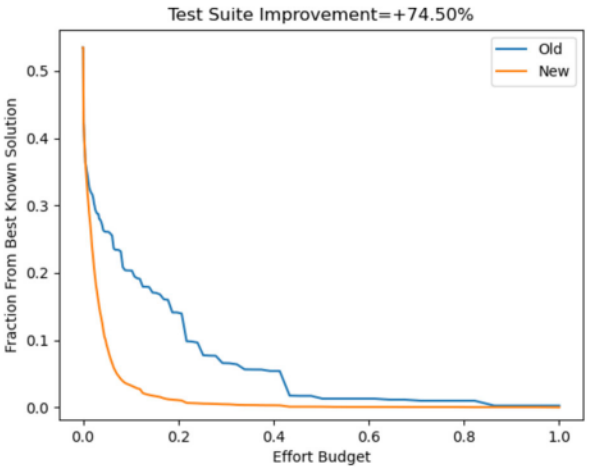

Fig. 8 Average performance of problem set

In this case, a problem that requires less effort does not contribute to the mean for the full effort shown on the x-axis. For example, if effort is being measured by iterations, and two problems were run to 1000 iterations and one was only run for 500 iterations, then the mean volume for the first half of the x-axis would be the mean of all three problems, and then for the second half, the mean volume would be only the mean of the two remaining problems. An alternative approach for aggregation is to extend the iterations of the problem run for 500 iterations to 1000 by replicating the objective value at iteration 500 for iterations 501 to 1000, but that tends to over-weight easier problems that require fewer iterations. Once the pseudoproblems for the new and old metaheuristics are created, the composite volume is computed as a function of effort and the percent from best known as described above.

In a set of 60 problems, an algorithmic change can result in improvements to some problems and worse performance in others. Using normalized, average behavior is one way to determine if algorithmic changes lead to an overall improvement. In this case, the old and new metaheuristic begin with near-identical performance, but the performance of the new metaheuristic quickly diverges with the old metaheuristic performing much better in aggregate across the test suite.

4.5 Results for multi-objective and simulation optimization problems

This section concludes the computation results by showing results for multi-objective and simulation optimization problems. Figure 9 shows the average performance of two different metaheuristic algorithms on two multi-objective problems. Both problem instances involve continuous variables and problem instance 27 also includes constraints. The new metaheuristic starts by mapping out the problem space with a bias towards diversification and the initial solutions are not very good compared to

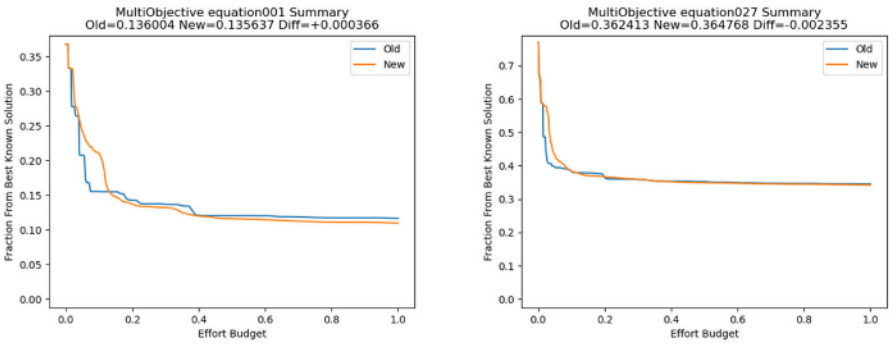

Fig. 9 Solution quality vs. computational effort for multi-objective problem instances

the old metaheuristic algorithm. However, the extra information gained in the diversity search provides a better basis for the intensification phase of the metaheuristic algorithm and the undominated area quickly shrinks yielding a better frontier more quickly. Ultimately, both the new and the old metaheuristics converge finding the same frontier, but the new metaheuristic algorithm converges more quickly, and hence, is preferred.

When comparing metaheuristics algorithms across a set of multi-objective problems instances, it is useful to look at the distribution of percent change to see if the new metaheuristic algorithm is better for the entire set of problem instances. Figure 10 shows a histogram of percent improvement of the new metaheuristic algorithm versus the old metaheuristic for 171 distinct multi-objective optimization problem instances. The problem instances in this test set vary in the number of objectives (two and three objectives), decision variables, and constraints. Of the 171 problem instances, the

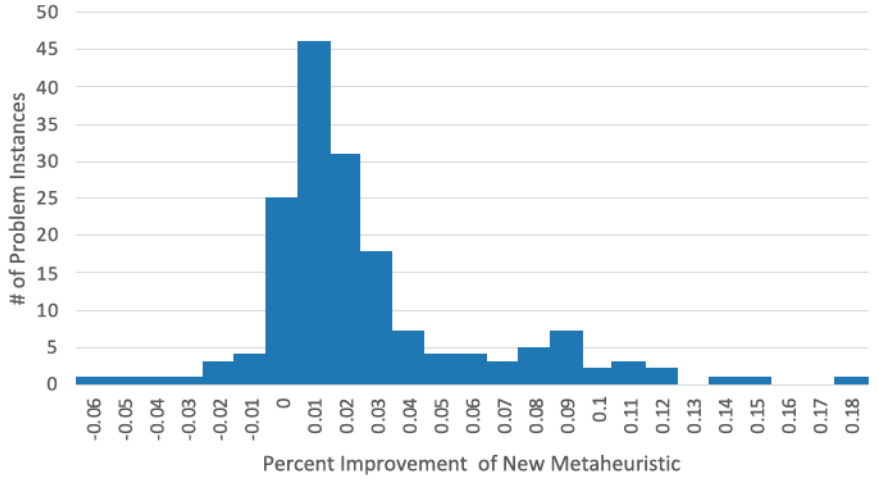

Fig. 10 Histogram of percent improvement for multi-objective problem set

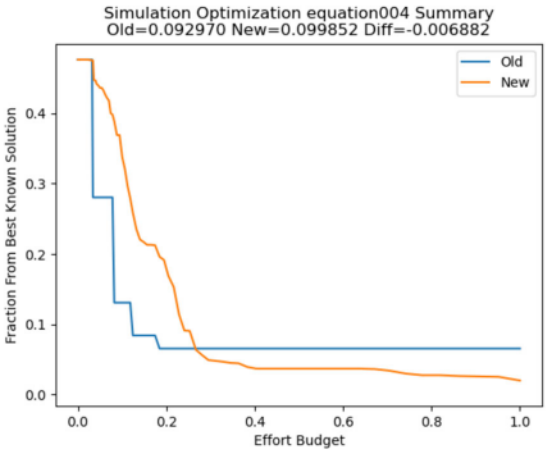

Fig. 11 Solution quality vs. computational effort for simulation optimization problem instance

new metaheuristic algorithm has the same performance (0% percent improvement for 25 problem instances) or positive percent improvement (for 135 problems instances) when compared with the old metaheuristic algorithm as shown in Fig. 9. These results provide empirical evidence that the new metaheuristic algorithm is better.

Finally, when optimizing a simulation, a more computationally expensive simulation may constrain the number of iterations that may be explored and finding a better result more quickly is critical. Figure 11 shows the results for a simulation optimization problem using a computationally expensive discrete-event combat simulation. In this example, the simulation model executes a combat scenario to evaluate the performance of a military system given different resources. The simulation model is computationally expensive requiring several minutes to complete a single iteration. The new metaheuristic algorithm starts off worse than the old metaheuristic but then finds a better solution before the effort budget is exhausted.

5 Conclusion and future work

The objective of this study was to implement a single measure that incorporates the solution quality (or effectiveness) and computational effort (or efficiency) of a metaheuristic algorithm that can be used to empirically compare it with other metaheuristic algorithms for single- and multi-objective optimization problems. A review of the comparative measures within the literature and the limitations of using a simple solution quality measure was also presented. Since metaheuristic algorithms have proven their utility in simulation optimization and an analyst's simulation run budget can vary significantly with each simulation study, it is especially important to balance the trade-off between the effectiveness and efficiency of the metaheuristics being used for simulation optimization applications. Finally, this paper provided computational results of

how the defined volume measure can be used to compare the performance of metaheuristic algorithms using knapsack problem instances, pure binary integer program instances, traveling salesman problems, the average results across a diverse problem set, and for multi-objective and simulation optimization problem instances.

As demonstrated in Sect. 4, the volume measure defined in Sect. 3 offers an intuitive and direct way to quantify solution quality and computational effort in a single metric and can be used to quickly assess the impact of changes to a metaheuristic algorithm. Work is in progress to examine changes and enhancements that account for multi-threaded implementations of a general-purpose metaheuristic algorithm. This research will help quantify the benefits of high-performance computing for effectively and efficiently solving complex simulation optimization problems.

Acknowledgements Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of OptTek Systems, Inc., or Montana State University.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

Aarts, E., Lenstra, J.: Local Search in Combinatorial Optimization. Princeton University Press, Princeton (2003)

Antosiewicz, M., Koloch, G., Kaminski, B.: Choice of best possible metaheuristic algorithms for the traveling salesman problem with limited computational time: quality, uncertainty and speed. J. Theor. Appl. Comput. Sci. 7(1), 46–55 (2013)

Asih, A., Sopha, B., Kriptaniadewa, G.: Comparison study of metaheuristics: empirical application of delivery problems. Int. J. Eng. Bus. Manag. **9**, 1–12 (2017)

Audet, C., Hare, W.: Derivative-Free and Blackbox Optimization. Springer, Cham (2017)

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., Salomon, L.: Performance indicators in multiobjective optimization. Eur. J. Oper. Res. **292**(2), 397–422 (2021)

Banks, J., Carson, J., Nelson, B., Nicol, D.: Discrete-Event System Simulation, 5th edn. Pearson, Hoboken (2009)

Barr, R., Golden, B., Kelly, J., Resende, M., Stewart, W.: Designing and reporting on computational experiments with heuristic methods. J. Heuristics **1**, 9–32 (1995)

Coffin, M., Saltzman, M.: Statistical analysis of computational tests of algorithms and heuristics. INFORMS J. Comput. **12**(1), 24–44 (2000)

Conn, A., Scheinberg, K., Vicente, L.: Introduction to Derivative-Free Optimization. SIAM, New Delhi (2009)

Eckman, D., Henderson, S., Shashaani, S.: Diagnostic tools for evaluating and comparing simulation-optimization algorithms. INFORMS J. Comput. **35**(2), 350–367 (2023a)

Eckman, D., Henderson, S., Shashaani, S.: A testbed for simulation-optimization experiments. INFORMS J. Comput. **35**(2), 495–508 (2023b)
Ehrgott, M.: Multicriteria Optimization. 2nd edition Springer, Berlin-Heidelberg (2005)
Evans, L., Bae, K., Roy, A.: Single and multi-objective parameter estimation of a military personnel system via simulation optimization. In: Chan, W., D'Ambrogio, A., Zacharewicz, G., Mustafee, N., Wainer, G.E., Proceedings of the 2017 Winter Simulation Conference, pp. 4058–4069. Institute of Electrical and Electronics Engineers, Inc., Las Vegas, NV (2017)
Fu, M. (ed.): Handbook of Simulation Optimization, vol. 216. Springer, Cham (2015)
Fu, M., Glover, F., April, J.: Simulation optimization: a review, new developments, and applications. In: Kuhl, M.E., Steiger, N.M., Armstrong, F.B., Joines, J.A. (eds.) Proceedings of the 2005 Winter Simulation Conference, pp. 83–95. Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ (2005)
Fuellerer, G., Doerner, K., Hartl, R., Iori, M.: Ant Colony optimization for the two-dimensional loading vehicle routing problem. Comput. Oper. Res. **36**(3), 655–673 (2009)
Geyer, A., Hall, S., Moore, J.: Operations-focused Optimized theater weather sensing strategies. Mil. Oper. Res. **21**(3), 51–71 (2016)
Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control Cybern. **29**(3), 653–684 (2000)
Greenberg, H.: Computational testing: why, how and how much. ORSA J. Comput. **2**(1), 94–97 (1990)
Hall, S., Jacobson, S., Sewell, E.: An analysis of pediatric vaccine formulary selection problems. Oper. Res. **56**(6), 1348–1365 (2008)
Hall, S., Wade, B., Thengvall, B.: Simulation optimization. In: Scala, N., Howard, J. (eds.) Handbook of Military and Defense Operations Research, 2nd edn. Chapman and Hall/CRC, Boca Raton (2024)
Hall, S., Thengvall, B., Schauer, R.: Simulating a maritime anti-air warfare scenario to optimize a ship's defensive system. In: Mustafee, N., Bae, K., Lazarova-Molnar, S., Rabe, M., Szabo, C., Haas, P., Son Y. (eds.) Proceedings of the 2019 Winter Simulation Conference, pp. 2536–2547. Institute of Electrical and Electronics Engineers, Inc., National Harbor, MD (2019)
Hooker, J.: Needed: an empirical science of algorithms. Oper. Res. **42**(2), 201–212 (1994)
Hunter, S., Ondes, B.: Properties of several performance indicators for global multi-objective simulation optimization. In: Corlu, C., Hunter, S., Lam, H., Onggo, B., Shortle, J., Biller, B. (eds.) Proceedings of the 2023 Winter Simulation Conference, pp. 3577–3588. Institute of Electrical and Electronics Engineers, Inc., San Antonio, TX (2023)
Jacobson, S., Hall, S., Mclay, L., Orosz, J.: Performance analysis of cyclic simulated annealing algorithms. Methodol. Comput. Appl. Probab. **7**(2), 183–201 (2005)
Kannon, T., Nurre, S., Lunday, B., Hill, R.: The aircraft routing problem with refueling. Optim. Lett. **9**(8), 1609–1624 (2015)
Kumbharana, S., Pandey, G.: A comparative study of ACO, GA, and SA for solving traveling salesman problem. Int. J. Soc. Appl. Comput. Sci. **2**(2), 224–228 (2013)
Laguna, M.: OptQuest: Optimization of Complex Systems. OptTek Systems Inc., Boulder (2011)
Laguna, M., Martí, R.: Experimental testing of advanced scatter search designs for global optimization of multimodal functions. J. Global Optim. **33**(2), 235–255 (2005)
Law, A.: Simulation Modeling and Analysis, 6th edn. McGraw Hill, New York (2024)
Lee, L., Chew, E., Frazier, P., Jia, Q., Chen, C.: Advances in simulation optimization and its applications. IIE Trans. **45**(7), 683–684 (2013)
McNabb, M., Weir, J., Hill, R., Hall, S.: Testing local search move operators on the vehicle routing problem with split deliveries and time windows. Comput. Oper. Res. **56**, 93–109 (2015)
Melouk, S., Fontem, A., Waymire, E., Hall, S.: Stochastic resource allocation using a predictor-based heuristic for optimization via simulation. Comput. Oper. Res. **46**, 36–48 (2014)
Montgomery, D.: Design and Analysis of Experiments, 10th edn. Wiley, New York (2019)
Onal, H., Woodford, P., Tweddale, S., Westervelt, J., Chen, M., Dissanayake, S., Pitois, G.: A dynamic simulation/optimization model for scheduling restoration of degraded military training lands. J. Environ. Manag. **171**, 144–157 (2016)
OptTek Systems, Inc.: OptQuest. http://www.opttek.com/products/optquest. Accessed 23 Jan 2024

Pasupathy, R., Henderson, S.: A testbed of simulation-optimization problems. In: Perrone, L.F., Wieland, F.P., Liu, J., Lawson, B.G., Nicol, D.M., Fujimoto, R.M. (eds.) Proceedings of the 2006 Winter Simulation Conference, pp.255–263. Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ (2006)

Reeves, C. (ed.): Modern Heuristics Techniques for Combinatorial Problems. McGraw Hill, London (1995)

Ruane, P., Walsh, P., Cosgrove, J.: Using simulation optimization to improve the performance of an automated manufacturing line. Proc. Comput. Sci. **217**, 630–639 (2023)

Shamos, M., Hoey, D.: Geometric intersection problems. In: 17th Annual Symposium on Foundations of Computer Science (sfcs 1976), pp. 208–215 (1976)

Simulation Optimization Library: https://github.com/simopt-admin/simopt. Accessed 23 Jan 2024

Singhal, S., Goyal, S., Goyal, S., Bhatt, D.: A comparative study of a class of nature inspired algorithms. In: Proceedings of the 5th national conference; INDIACom, pp. 611–617. New Delhi (2011)

Soykan, B., Rabadi, G.: A simulation-based optimization approach for multi-objective runway operations scheduling. Simulation **98**(11), 991–1012 (2022)

Surjanovic, S., Bingham, D.: Virtual library of simulation experiments. https://www.sfu.ca/~ssurjano/optimization.html. Accessed 23 Jan 2024

Swisher, J., Hyden, P., Jacobson, S., Schruben, L.: A survey of simulation optimization techniques and procedures. In: Joines, J.A., Barton, R.R., Kang, K., Fishwick P.A. (eds.) Proceedings of the 2000 Winter Simulation Conference, pp. 119–128. Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ (2000)

Tekin, E., Sabuncuoglu, I.: Simulation optimization: a comprehensive review on theory and application. IIE Trans. **36**(11), 1067–1081 (2004)

Wade, B.: A multi-objective optimization of ballistic and cruise missile fire plans based on damage calculations from missile impacts on an airfield defended by an air defense artillery network. J. Def. Model. Simul. Appl. Methodol. Technol. **16**(2), 103–117 (2019)

Zais, M., Laguna, M.: A simulation-optimization approach to estimate workforce requirements. Mil. Oper. Res. **22**(1), 19–39 (2017)