Methodology for Comparison of Algorithms for Real-World Multi-objective Optimization Problems: Space Surveillance Network Design

Troy B. Dontigney
United States Air Force
Laurence D. Merkle
Air Force Institute of Technology
Richard G. Cobb
Air Force Institute of Technology
John M. Colombi
Air Force Institute of Technology
Gary B. Lamont
Air Force Institute of Technology

ABSTRACT

Geosynchronous Earth Orbit (GEO) is a valuable orbital regime containing many mission-critical national and civilian assets. As the region becomes increasingly congested, the risk of collision between resident space objects (RSOs) grows as well. There is an imminent need for a space surveillance network (SSN) capable of resolving GEO targets with greater resolution and frequency. Many research efforts have aimed at developing computationally feasible models and methods for finding the SSN architectures capable of detecting sufficiently small RSOs often enough to maintain an accurate picture of the state of objects in GEO at the lowest possible price. Several recent efforts investigated using high-performance computing (HPC) and evolutionary algorithms (EAs) to identify near-optimal solutions in this vast search space. Stern and Wachtel proposed a 28-dimensional model of SSNs, coupled with the Non-dominated Sorted Genetic Algorithm II (NSGA-II) as one such method. Their model includes four classes of optical sensors: ground-based telescopes (GBTs), equatorial observation satellites (obsats), sun-synchronous obsats, and near-GEO obsats. Within their model, GBTs may be placed at zero or more user-defined locations on Earth with one or more telescopes at each location. If included in an architecture, there may be only one constellation of each class of obsat, appearing at one of several possible altitudes, with a variable number of obsats in the constellations. Each constellation or GBT site may use different aperture diameters, but must use the same diameter within the given constellation or site. Their methodology can be broadly summarized in four steps: 1) simulate 813 targets using AGI's Systems Toolkit (STK) to approximate a realistic catalog of GEO satellites, 2) compute access, range, angular parameters, and illumination conditions pairwise between each sensor and target for each single-class configuration possible in the model, 3) combine that data to determine performance of any of the more than 2×10^{21} architectures possible in the model, and 4) use NSGA-II to perform a non-exhaustive exploration of the solution space. For this application, the search is orders of magnitude more computationally expensive than the simulations.

Extending that effort, the current research increases efficacy of those methods by presenting a methodology for selecting the most effective from a selection of multi-objective (MO) search algorithms. Four different methods of comparing Pareto fronts are examined and applied to a selection of common MO search algorithms in order to determine the strengths and weaknesses of the methods when applied to this search space. Additionally, a quantitative investigation of the tradeoffs associated with using a subset of the simulation data to reduce the computational load of evaluating multiple algorithms is presented. Given the substantial cost of running a single search, and the projected growth of RSOs in GEO, it is desirable to reduce computational cost while evaluating multiple algorithms. Therefore, subsets of 20, 81, 203, and 407 targets (2.5%, 10%, 25%, and 50% of the full set, respectively) are analyzed statistically, compared with the original 813 RSO set, and used to repeat the comparisons between algorithms. The results are analyzed to demonstrate the tradeoffs involved in using subsets of data to quickly evaluate multiple algorithms, recommending a way forward for analyzing similar large-scale problems of interest to the DoD.

1. INTRODUCTION

When humanity first developed spaceflight capabilities, little attention was given to the issue of congestion. In the intervening decades, much has changed. Technology has progressed at an astonishing rate, and the domain that was once in reach only for global superpowers is now more accessible than ever before. In particular, with the rise of miniaturization, technologies such as CubeSats [5] have opened the space to such a degree that the use of space assets is now achievable by commercial entities and even high schools [3]. Concurrently, space has become an indispensable domain for the US Government (USG) and military, enabling myriad capabilities such as communication, surveillance, reconnaissance, navigation, and weather forecasting at levels that would not be otherwise possible [15]. In particular, with its 24-hour orbital period, the Geostationary Earth Orbit (GEO) regime allows satellites to loiter over particular points on the surface of the Earth [11], and has proven to be a particularly valuable orbital regime for the USG. However, as the number of nations and industries dependent on space increases, the once untapped domain of space is becoming ever-more congested, competitive, and contested [15].

Efforts have been made in the past to apply evolutionary algorithms (EAs) to the problem of optimizing components of space surveillance networks (SSNs), as well as to the design of entire SSN architectures. These efforts have enjoyed some success, but they have not addressed the facts that there are an infinitude of possible EAs and that not all EAs are equally well suited for a given problem. So while there have been results produced by these efforts, there is no way of knowing if the search algorithms used are appropriate to the task.

In this paper, a methodology is presented to compare the performance of EAs on real-world problems for which the true optimal solution is not known. The methodology for finding near-optimal SSN architectures developed by Stern and Wachtel [18] is adapted to use multiple search algorithms, and five representative search algorithms are applied to the problem. The results are compared to one another pairwise using four binary comparison methods and relative rankings are produced, revealing the relative strengths and weaknesses of the algorithms. This procedure is repeated using subsets of the underlying simulation data and multifidelity analysis is performed to determine if similar results can be obtained at lower computational cost by simulating fewer target satellites in GEO.

This paper is organized in five sections after this introduction. Section 2 presents a brief background of the concepts used in this paper, including the four binary comparison methods used to evaluate the effectiveness of algorithms. Section 3 describes the methodology used to conduct the computational experiments. Section 4 describes the computational experiments and summarizes the key parameters used. Section 5 presents the results of the experiments as well as the analysis and interpretation of those results. Finally, the paper concludes with a brief summary in Section 6.

2. BACKGROUND

2.1 Space Situational Awareness (SSA) and Space Surveillance Networks (SSN)

SSA is central to a nation's ability to operate in space, and is defined by the Joint Chiefs of Staff as "the requisite foundational, current, and predictive knowledge and characterization of space objects and the [operational environment] upon which space operations depend" [19]. Numerous sensor technologies are available with which to obtain such knowledge by observing objects in orbit (see, e.g., the 2008 report of the Joint Defense Science Board Intelligence Science Board Task Force on Integrating Sensor-Collected Intelligence [12]), but only ground- and space-based electro-optical telescopes are considered by the Stern and Wachtel model [18] used in this research.

Ground-based telescope observations are currently limited to the hours of darkness, while space-based telescopes are limited by the transit of objects such as the Earth in front of the target RSO and of light sources such as the Moon and Sun behind it. Thus, the relative motion of the Sun, Moon, and Earth makes it necessary to employ multiple telescopes at spatially diverse locations to enable sufficiently frequent observation of all RSOs of interest in the catalog. The typical use of an SSN is to schedule all sensors to routinely scan through some subset of observable RSOs in an efficient manner to maintain current data on the current orbit of each RSO.

2.2 Stern and Wachtel's Methodology for Designing Near-Optimal SSNs

Stern and Wachtel [18] proposed a methodology for determining near-optimal SSN architectures along with a refined model for representing executable architectures. The model is a 28-dimensional representation of an SSN that can include space-based telescopes in three different orbital regimes (sun-synchronous, equatorial Low Earth Orbit (LEO),

and near-GEO) and nine ground-based locations. The methodology consists of two general phases. The first phase involves performing simulations in advance, simulating 813 RSOs in GEO and each possible sensor within the model. Over two simulated 24-hour periods, data about line-of-sight, illumination conditions, and relative positions of the Sun, Moon, Earth, and each RSO, relative to each sensor, is collected at 30-second intervals. Data is collected separately for each sensor to allow for the combination of individual data sets to obtain results for any possible architecture without requiring individual simulations for each of the more than 2×10^{21} architectures possible in the model.

The second phase is optimization. This phase applies a well-known multi-objective evolutionary algorithm (MOEA), the Non-dominated Sorted Genetic Algorithm II (NSGA-II) [7], to the problem, minimizing three objectives: cost, latency, and detection size. Cost, which is fairly self-explanatory, includes estimates of acquisition costs, operation costs (for ten years), maintenance costs, and launch costs for space-based telescopes. Formally,

$$C(X) = \sum C_{Sat} + \sum C_{Tel} + \left(\sum C_{SatOp} + \sum C_{TelOp}\right) \times 10yrs + \sum C_{Launch}, \tag{1}$$

where X is the architecture under evaluation.

Latency is defined as

$$L(X) = \frac{\sum_{RSO=1}^{numTgt} \left[\max_{1 \le o \le numObs} (t_1 - t_{start}, t_{o+1} - t_o, t_{end} - t_{numObs}) \right]_{RSO}}{numTgt}, \tag{2}$$

where $numTgt$ is the number of simulated target RSOs and $numObs$ is the number of observations made on a given target. It is the mean of the maximum observation time gap (MMOTG) across all simulated RSOs and represents the longest gap in observation for an average target satellite.

Finally, Detection Size is defined as

$$S(X) = \frac{\sum_{RSO=1}^{numTgt} \left(\frac{\sum_{o=1}^{numObs} size_o}{numObs} \right)_{RSO}}{numTgt}, \tag{3}$$

which is the average minimum RSO size that a given architecture is predicted to be able to detect in GEO. The details of both the SSN model and the formulation of the problem are thoroughly discussed in [18] and omitted here.

The computational burden of this phase is very high, and is necessarily distributed using a high performance computer (HPC). For each generation, the 96 possible architectures are evaluated in parallel, distributed across several HPC nodes. The optimization process is repeated several times, applying a single objective approach to an equally weighted linear combination of the objectives, an unconstrained MO approach, and a constrained MO approach.

2.3 Binary Comparison of Pareto Fronts and Multi-objective Algorithms

For most problems, a multi-objective optimization algorithm will produce a set of distinct solutions, each of which represents a unique, optimal compromise between the objectives, known as a Pareto front [8]. This solution set consists of the subset of all evaluated solutions for which no other solution is better in any objective without deterioration in another objective. That is to say, no other evaluated solution *dominates* any solution in the Pareto front, which is sometimes referred to as the non-dominated solution set.

Unfortunately, not every algorithm is equally effective on the same problem [20]. An algorithm that is exceptionally effective on one problem will prove to be rather ineffective on some other problems, and vice versa. Therefore, for problems with high stakes, such as the multi-billion-dollar expansion of the SSN, it is not enough to pick a well-known search algorithm and obtain results. One must determine the appropriateness of an algorithm for the problem. For common benchmark problems with a known solution, or true Pareto front, one can simply run the problem on an algorithm and compare how close it comes to the solution. For real-world problems where the search space is only understood in more general terms, one has much less to inform the selection of a search algorithm. For these problems, one may elect to apply several algorithms to the problem (or a scaled down version) and evaluate which proves to be the most effective.

Evaluating the quality of Pareto fronts is a complicated issue on which much research exists [6]. As this paper focuses on real-world problems with no known true Pareto front against which to compare, only binary[1] comparison methods

[1] Binary, in this usage, refers to the comparison of two items, not the numbering system. This is in contrast to unary indicators, which characterize a single solution set in isolation (e.g. the hypervolume indicator) [6].

are employed. Several binary comparison methods are available for comparing solution sets, and with multiple runs of two algorithms A and B, these methods can be extended to compare those algorithms, thereby elucidating their benefits and shortcomings. Four such techniques, described below, are applied. For each of these methods, running the comparison in both directions, that is both A to B and B to A, is necessary to obtain the fullest understanding of the relationships possible with these comparison methods.

The Binary Hypervolume (I_{H2}) comparison method compares two solution sets based strictly on each set's hypervolume (HV) [21]. It is described by the following equation

$$I_{H2}(A,B) = HV(A+B) - HV(B), \tag{4}$$

where $HV(A+B)$ is the hypervolume of the union of Pareto front A and Pareto front B (the fronts obtained by algorithms A and B, respectively). Thus, $I_{H2}(A,B)$ indicates the volume of decision space weakly dominated by solution set A, but not by solution set B, which Zitzler proposes as a method to compare two solution sets using hypervolumes.

Because this comparison method depends on the relative position of solutions in hyperspace, Binary Hypervolume is vulnerable to differences in scale among the objectives of a problem. It gives greater weight to differences between solution sets with respect to objectives with large magnitudes, relative to objectives with smaller magnitudes. As such, it is recommended to normalize objective values to ensure that each objective is given equal weighting with this comparison method. It also tends to favor solution sets that "explore" the search space more effectively; every point in solution set A that is more extreme that the most extreme points in solution set B increases the comparison value (in favor of A) without regard to B in any way. Thanks to this phenomenon, this is the only comparison method that can give a hint that one algorithm explored better than another, especially when its results contrast with the findings of the Coverage indicator.

For solution sets A and B, the coverage indicator $I_C(A,B)$ is the fraction of points in B that are weakly dominated by points in A [21, 24]

$$I_C(A,B) = \frac{|\{b \in B \mid \exists\, a \in A : a \preceq b\}|}{|B|}. \tag{5}$$

Unlike binary hypervolume, which deals with volumes of objective space, the coverage indicator relies on pairwise comparisons of objective vectors. Consequently, it is more susceptible to skewing by a few exceptional solutions in an otherwise mediocre solution set. It is not, however, vulnerable to differences in scale and will render the same results with or without normalization of objective values. Zitzler [21] recommends using I_{H2} and I_C as complementary tools to build a fuller picture of the relationship between solution sets.

Zitzler et al. [25] introduce the ε-Indicator (I_ε) to allow one to not only make claims about the binary relationship between two solutions sets, but to also quantify the degree of certain relationships. For example, in addition to determining if A strictly dominates B, it can also quantify the factor by which it dominates B. This capability is very useful for the objective comparison of algorithms, as it can be combined with other characteristics of the algorithms, such as running time, to quantify the amount of qualitative gain or loss in contrast with the non-qualitative factors.

In simple terms, $I_\varepsilon(A,B)$ represents A's ε-dominance of B, where ε-dominance is defined as follows: for a minimization problem with n positive objectives ($Z \subseteq R^{+^n}$), an objective vector $z^1 = \left(z_1^1, z_2^1, \ldots, z_n^1\right) \in Z$ ε-dominates another objective vector $z^2 = \left(z_1^2, z_2^2, \ldots, z_n^2\right) \in Z$, written $z^1 \preceq_\varepsilon z^2$, if and only if

$$\forall i \in \{1,2,\ldots,n\}(z_i^1 \leq \varepsilon \cdot z_i^2) \tag{6}$$

for a given $\varepsilon > 0$. The binary ε-indicator I_ε for solution sets A and B is defined as follows:

$$I_\varepsilon(A,B) = \inf_{\varepsilon \in R} \left\{ \forall z^2 \in B \left(\exists z^1 \in A(z^1 \preceq_\varepsilon z^2) \right) \right\}, \tag{7}$$

i.e. the infimum of values by which every objective value in B can be multiplied such that solution set B is weakly dominated by solution set A.[2]

In the same paper, Zitzler et al. propose the Binary Additive ε-Indicator, which is based on an additive ε-dominance relation ($\preceq_{\varepsilon+}$) for minimization problems with n real objectives ($Z \subseteq R^n$) and is otherwise analogous to the Binary

[2] Assuming, along with a finite number of objectives, finite cardinalities of A and B (which is certainly the case in this work), this is equivalent to the maximum value by which every objective value in B can be multiplied such that B is *not* weakly dominated by A.

ε-Indicator, with analogous capabilities and limitations. Specifically, it is defined as the infimum of values that can be added to every objective value in B such that B is weakly dominated by A, i.e.

$$I_{\varepsilon+}(A,B) = \inf_{\varepsilon \in R} \left\{ \forall z^2 \in B \left(\exists z^1 \in A (z^1 \preceq_{\varepsilon+} z^2) \right) \right\}, \tag{8}$$

where $z^1 \preceq_{\varepsilon+} z^2$ if and only if

$$\forall i \in \{1,2,\ldots,n\} (z_i^1 \leq \varepsilon + z_i^2). \tag{9}$$

Despite its similar definition, this indicator tends to result in different relative rankings than the standard ε-Indicator. The use of addition instead of multiplication (a translation rather than a scaling) means that this indicator compares two Pareto fronts differently based on the shapes of the two fronts. Like Binary Hypervolume, is influenced by differences in scale among objectives, so objective values should be normalized to ensure each objective is weighted equally.

Each of these methods presents a partial picture of how one solution set relates to another, and each is able to describe only some of the important possible relations between solution sets. Table 1 lists those relations and the corresponding indicator values necessary to assert such relations hold between two solution sets.

Table 1: Binary indicators and their capabilities. The relations are: strictly dominates ($\lessdot$), dominates ($\prec$), better than ($\triangleleft$), weakly dominates ($\preceq$), equals ($=$), and incomparable ($\|$). Adapted from Zitzler et al. [25]

Indicator	can determine relation:					
	$\lessdot$	$\prec$	$\triangleleft$	$\preceq$	$=$	$\|$
Epsilon Indicator (I_ε)	$I_\varepsilon(A,B) < 1$	n/a	$I_\varepsilon(A,B) \leq 1$ $I_\varepsilon(B,A) > 1$	$I_\varepsilon(A,B) \leq 1$	$I_\varepsilon(A,B) = 1$ $I_\varepsilon(B,A) = 1$	$I_\varepsilon(A,B) > 1$ $I_\varepsilon(B,A) > 1$
Additive Epsilon Indicator ($I_{\varepsilon+}$)	$I_{\varepsilon+}(A,B) < 1$	n/a	$I_{\varepsilon+}(A,B) \leq 0$ $I_{\varepsilon+}(B,A) > 0$	$I_{\varepsilon+}(A,B) \leq 0$	$I_{\varepsilon+}(A,B) = 0$ $I_{\varepsilon+}(B,A) = 0$	$I_{\varepsilon+}(A,B) > 0$ $I_{\varepsilon+}(B,A) > 0$
Coverage (I_C)	n/a	$I_C(A,B) = 1$ $I_C(B,A) = 0$	$I_C(A,B) = 1$ $I_C(B,A) < 1$	$I_C(A,B) = 1$	$I_C(A,B) = 1$ $I_C(B,A) = 1$	$0 < I_C(A,B) < 1$ $0 < I_C(B,A) < 1$
Binary Hypervolume (I_{H2})	n/a	n/a	$I_{H2}(A,B) > 0$ $I_{H2}(B,A) = 0$	$I_{H2}(A,B) \geq 0$ $I_{H2}(B,A) = 0$	$I_{H2}(A,B) = 0$ $I_{H2}(B,A) = 0$	$I_{H2}(A,B) > 0$ $I_{H2}(B,A) > 0$

3. METHODOLOGY

The computational experiments presented here consist of four general stages. The core methodology consists of the simulation and optimization stages, closely mirroring the methodology of Stern and Wachtel [18]. The simulation phase consists of generating all simulation data in advance using STK [2] on a high-end workstation. Simulation data is collected not only for the full catalog of 813 RSOs, but also for subsets of 407, 203, 81, and 20 RSOs (50%, 25%, 10%, and 2.5% of the full catalog, respectively). The optimization phase consists of running multiple search algorithms separately on the full, 813-RSO data set. Each algorithm is run five independent times. In the third stage, analysis, the binary comparison methods are applied to the resulting solution sets to determine relative rankings of the algorithms' effectiveness according to each method. The fourth and final stage is multifidelity analysis. Here, optimization and analysis are repeated using each of the reduced subsets of simulation data, and the results are compared and analyzed.

Generation of simulation data is accomplished using a Python script and a randomized list of GEO RSOs to configure a 24-hour scenario in STK with up to 813 RSOs and 99 ground- and space-based optical telescopes possible within the model. Data is calculated regarding line-of-sight, illumination conditions, and relative position to the Earth, Sun, and Moon for each sensor every 30 seconds of the scenario. Using STK's object model [1], this data is collected and partitioned into the 77 possible "sties" (i.e. ground-based locations or space-based constellations) allowed by Stern and Wachtel's model. Finally the data is serialized into 77 individual files using Python's *Pickle* module to preserve data structure and native data types. These data files provide all of the necessary information for SSN evaluation by the optimization algorithms, eliminating the need for further simulations during the optimization phase.

Optimization consists of using the multi-objective search algorithms separately to explore objective space. For each algorithm, SSN architectures are generated and evaluated according to the algorithm's heuristics. Evaluation is accomplished by combining the simulation data for each site (of the 77 possible) included in each generated architecture and evaluating the three objectives specified by Stern and Wachtel [18] on the composite data set. Each algorithm terminates with the output of a set of non-dominated architectures.

In the analysis phase, the relative performance of the algorithms is analyzed with respect to each binary comparison method. The methods compare solution sets, so algorithms are compared based on their aggregate solution sets. Specifically, the solutions sets from each run of a given algorithm are first consolidated into a single solution set and the resulting Pareto front extracted. The aggregate Pareto fronts are compared pairwise using each of the comparison methods. Since there are five algorithms, the comparisons are directional, and there are four comparison methods, this is a total of $5 \times 4 \times 4 = 80$ comparisons considered in this phase. The results indicate relative rankings of algorithms with respect to each comparison method.

Multifidelity analysis, the final phase of this methodology, consists of repeating the optimization and analysis phases using the four subsets of the full simulation data. This results in a grand total of $80 \times 5 = 400$ comparisons considered in this phase. Reducing the number of simulated target satellites does not impact the cost objective, but it does in general alter the other two objective values for any particular architecture. However, the question addressed here is not whether one can evaluate architectures as a lower computational cost. Rather, it is whether one can evaluate *algorithms* for their effectiveness on this problem at a lower computational cost. For example, in preliminary testing for this research, the average evaluation time for an architecture using the full data set was in excess of five minutes, while using the smallest data set averaged about eight seconds. Thus, the question is whether the same relative performances are observed with smaller sets of simulated targets, thus producing similar relative rankings. For computationally expensive problems such as this one, this would facilitate a preliminary study to assess a variety of algorithms before performing an intensive search using the best of the evaluated algorithms.

Subsets were selected by drawing from a randomized list of the RSOs to be simulated. The list was randomized once and used for all experiments. For each subset, the selection begins with the first RSO in the list and continues sequentially until the desired number of targets have been selected. Thus, subsets are constant and reproducible.

3.1 Algorithms Used

With this methodology, any number of MO algorithms can be used. For this research, two classical algorithms and three well-known evolutionary algorithms are selected for evaluation. Taken as a whole, they represent a spectrum of possible search algorithms that might be used on a problem with a search space of this size, and can be used to inform future efforts in this area.

The process of determining the best combination of parameters for a given problem is known is parameter tuning, and is a field of research unto itself. Many recent research projects have been devoted solely to the determination of ideal parameters [16, 17] or automating the tuning of parameters [13, 14], which has been a staple of the Evolution Strategies branch of EA research since its inception. Parameter tuning is not the primary focus of this research. Values proposed by Stern and Wachtel [18] are for the most part retained here.

The classical algorithms used are parallelized versions of Random Search (RS) and Random Restart Hill Climber (RRHC) [4]. Random search simply generates a pre-determined number of unique architectures, distributes equal shares of the workload to all available nodes for evaluation, collects the results, and extracts the Pareto front from the solution set. Random Restart Hill Climber performs an independent search on each available node, performing classic hill climbing on a randomly generated architecture until a user-defined number of consecutive, non-improving mutations are encountered. At that point, a new random architecture is generated and the process repeats. Each node maintains a local Pareto front and, after completing its portion of the workload, returns the local front to the master node. The master node combines the fronts and removes the dominated solutions to produce the final Pareto front.

The EAs used in this research were selected both because they each are well-known MOEAs and because they each use binary tournament selection as the parent selection operator, thereby limiting the number of parameters that might affect performance. The Non-dominated Sorted Genetic Algorithm II (NSGA-II) [7] uses dominance and a crowding function as its survivor selection operators. The Indicator Based Evolutionary Algorithm (IBEA) [22] uses binary quality indicators as its survivor selection operator. Finally, the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [23] uses an external archive of non-dominated solutions as its survivor selection operator. The archive has a maximum size, and uses a truncation function to maintain the archive once the maximum size is reached.

3.2 Hardware and Software Used

A variety of hardware and software are used. Simulations are run on a Dell T5600 workstation running Windows 10 and featuring two 8-core Xeon processors and 128 GB of memory (memory usage for full simulations exceeded 100 GB.) Optimization algorithms are run on Air Force Research Laboratory's (AFRL) HPCs Mustang and Thunder. Mustang has a total of 56,448 compute cores, features 244 terabytes (TB) of memory, and is rated at 4.88 peak petaflops (PFLOPS). Thunder features 125,888 compute cores, has 460 TB of memory, and is rated at 5.62 peak PFLOPS. Both HPCs run on Linux.

Scripting of the simulations, optimization, and data analysis are all done using Python 3.6. The Platypus library [10] provides the EAs used. PyGMO's [9] hypervolume tool is adapted for the binary hypervolume indicator. Simulations are conducted using AGI's Systems Toolkit (STK) software [2].

4. COMPUTATIONAL EXPERIMENTS

The five algorithms are run five times each for each of the five data sets. Altogether, 25 optimization runs are performed with each data subset, producing a total of 125 Pareto fronts. Each run terminates after evaluating 25,000 architectures (classical search algorithms) or 250 generations with a population size of 100 architectures per generation (EAs). For this population size, the takeover time of binary tournament selection without replacement is $\lceil \log_2(100) \rceil = 7$ generations [8]. Thus, 250 generations would allow for sequential optimization of $\lfloor 250/7 \rfloor = 35$ genes. Heuristically, this suggests a nontrivial probability of optimizing the 28 genes in the SSN architecture representation used in these experiments. The high-level parameters used for each algorithm are summarized in Table 2.

Table 2: The high-level parameters used for each algorithm considered. No parameter tuning was performed in this research.

Algorithm	Rate of Mutation	Rate of Crossover	Population Size	Generations	Evaluations
RS	n/a	n/a	n/a	n/a	25000
RRHC	n/a	n/a	n/a	n/a	25000
NSGA-II	.05	1.0	100	250	n/a
IBEA	.05	1.0	100	250	n/a
SPEA2	.05	1.0	100	250	n/a

5. RESULTS

This section summarizes the results of the computational experiments performed during this research. It also offers summary observations applicable to these results, and not guaranteed to hold in general. The results produced using the full set of simulation data are also used in the multifidelity analysis. For the sake of space, all data is presented together in Section 5.2 (Tables 5 through 8).

There is a tremendous amount of data in Tables 5 through 8. Each value is a comparison of two solution sets which are generally interpreted in this paper as how algorithm A (left column) performed relative to algorithm B (top row). In that sense, the values should be interpreted as follows: larger values indicate better relative performance by A with Binary Hypervolume and Coverage, and smaller values indicate better relative performance by A with both ε-indicators.

5.1 Comparison of Algorithms on the Full Simulation Data Set

Table 4 presents descriptive statistics for each of the aggregate Pareto fronts. These statistics indicate that IBEA tends to produce solutions with the best (lowest) detection sizes and costs, and with among the best latencies. However, the statistics provide no information on whether any solutions (nearly) achieve all of these conditions simultaneously. At the other extreme, they indicate that NSGA-II produces solutions across the largest ranges for each objective. A plausible explanation for both sets of results is that IBEA exploits the (apparent) Pareto front more effectively, while

NSGA-II explores it most broadly. Accordingly to this explanation, the remaining algorithms are intermediate to IBEA and NSGA-II in terms of both exploration and exploitation. However, more information is needed to assess the veracity of these claims. The binary comparison results (Tables 5, 6, 7, and 8) contain a great deal of such information that both confirms and expands on the relationships suggested by the descriptive statistics.

The column of Table 5 labeled "IBEA" shows that none of the other algorithms covers more than a few of IBEA's solutions, suggesting that those solutions tend to be close to the true Pareto front. However, the row labeled "IBEA" shows that IBEA covers (only) between 24% and 67% of the other methods' solutions, suggesting that significant portions of objective space close to the Pareto front are not dominated by IBEA solutions. Together, these observations further suggest that IBEA's solutions are tightly spaced. Assuming this is the case and taking into consideration the previous observation that IBEA's solutions tend to have the best detection sizes and costs and among the best latencies leads to the following conclusion: IBEA's solutions tend to lie near the outer edges of the Pareto front, where it approaches the planes minimizing the individual objectives.

IBEA outperforms the other algorithms in Binary Hypervolume despite not exploring large regions of the Pareto front that its competitors do. This fact implies that its performance in the regions it explores is sufficiently superior to compensate for the regions it does not explore. This is meaningful in two ways. First, it confirms that IBEA does not explore the search space very broadly, as suspected from the descriptive statistics. Second, it suggests that the other algorithms, at least in the space explored by IBEA, have significant room for improvement with regard to the quality of solutions found.

In agreement with observations based on the descriptive statistics, IBEA tends to do very well with respect to the binary comparison methods. In fact, it does the best with all comparison methods except Coverage, for which it is outperformed by both SPEA2 and NSGA-II.

Furthermore, in all cases, SPEA2 ranks higher than NSGA-II, which is in addition to the three measures in which IBEA ranks higher than both SPEA2 and NSGA-II. This suggests that, without parameter tuning, NSGA-II is probably not as appropriate an EA for the problem. Both competing EAs are able to produce higher-quality solutions, and SPEA2 is able to do so without sacrificing much in terms of its ability to explore the search space.

Finally, the Random Restart Hill Climber performed quite well in both variants of the ε indicator, actually outperforming all but IBEA. This suggests that the problem itself may not require techniques as sophisticated as EAs to explore the search space.

Table 3: Relative rankings of algorithm effectiveness based on the results of each binary comparison method.

Rank	Binary Hypervolume	Coverage	ε-Indicator	Addative ε-Indicator
1	IBEA	SPEA2	IBEA	IBEA
2	SPEA2	NSGA-II	RRHC	RRHC
3	NSGA-II	IBEA	SPEA2	RS
4	RRHC	RRHC	NSGA-II	SPEA2
5	RS	RS	RS	NSGA-II

5.2 Multifidelty Analysis Results

In order to determine if algorithms can be evaluated against this problem at a lower computational cost, the procedures in the previous section are repeated on each of the simulation data subsets. The resulting data is compiled in Tables 4 through 8. It's important to reiterate that the goal here is not to obtain similar objective values, as the number of simulated targets is integral to two of the objective functions. Rather, the goal is to determine if similar relative rankings can be obtained while simulating fewer targets.

The tables show that, though values do tend to change as the number of simulated RSOs vary, relative ranking of algorithms from one subset to the next tends to remain consistent for each of the comparison methods. For example, in Table 5, the Random Search rows, which show how Random Search compared with the other four algorithms in each subset of data, shows that it performed best against Random Restart Hill Climber and second best against IBEA across all data sets, and that it performed the worst against SPEA2 in three data sets and against NSGA-II in the other

two. Similarly, in the SPEA2 row of Table 8, the relative ranking of how well SPEA2 competed against the other four algorithms is the same across all data sets (from best to worst, NSGA-II, RS, IBEA, RRHC). While not every group is so conclusive, the larger trend is for an ordering to be apparent across all subsets.

An implication of the consistency in algorithm ranking is that, though objective values obtained with smaller subsets of simulation data do not correctly predict performance of a given SSN architecture, the *relative ranking* of algorithms using any one of the subsets tends to be representative of the relative rankings produced using the full set of simulation data. In other words, *algorithms* can be tested against this problem faster and at a lower computational cost with reasonable confidence using far fewer simulated RSOs.

Table 4: Aggregate Pareto front descriptive statistics (normalized objective values)

		Detection Size		**Latency**		**Cost**		
Algorithm	**RSOs**	**min**	**max**	**min**	**max**	**min**	**max**	**Hypervolume**
RS	**20**	0.00	0.01	0.00	0.64	0.01	0.55	9.85e-01
	81	0.00	0.01	0.00	0.73	0.00	0.56	9.83e-01
	203	0.00	0.13	0.01	0.54	0.01	0.43	9.78e-01
	407	0.00	0.27	0.01	0.56	0.01	0.47	9.68e-01
	813	0.00	0.38	0.02	0.86	0.01	0.43	9.56e-01
RRHC	**20**	0.00	0.01	0.00	0.42	0.06	0.47	9.34e-01
	81	0.00	0.01	0.00	0.19	0.03	0.34	9.62e-01
	203	0.00	0.13	0.01	0.42	0.02	0.34	9.64e-01
	407	0.00	0.13	0.01	0.50	0.03	0.34	9.52e-01
	813	0.00	0.21	0.02	0.42	0.01	0.39	9.58e-01
NSGA-II	**20**	0.00	0.05	0.00	1.0	0	0.48	9.91e-01
	81	0.00	0.05	0.00	1.0	0	0.60	9.89e-01
	203	0.00	0.27	0.01	1.0	0	0.49	9.85e-01
	407	0.00	0.42	0.01	1.0	0	0.48	9.79e-01
	813	0.00	0.65	0.02	1.0	0	0.47	9.67e-01
IBEA	**20**	0.00	0.00	0.00	0.70	0.00	0.19	9.89e-01
	81	0.00	0.01	0.01	0.86	0.00	0.15	9.83e-01
	203	0.00	0.01	0.01	0.62	0.00	0.21	9.82e-01
	407	0.00	0.12	0.01	0.51	0.00	0.18	9.79e-01
	813	0.00	0.11	0.02	0.79	0.00	0.16	9.66e-01
SPEA2	**20**	0.00	0.01	0.00	0.85	0.00	0.51	9.91e-01
	81	0.00	0.02	0.00	0.86	0.00	0.47	9.89e-01
	203	0.00	0.19	0.01	0.86	0.00	0.55	9.85e-01
	407	0.00	0.51	0.01	0.88	0.00	0.55	9.78e-01
	813	0.00	0.54	0.02	1.0	0	0.44	9.66e-01

Table 6: Summary of Coverage comparisons across all simulation data subsets (normalized objective values). In the context of the comparison methods, algorithm *A* is listed in the first column, and algorithm *B* is listed in the top row. Larger values indicate that *A* outperformed *B*.

Algorithm	RSOs	RS	RRHC	NSGA-II	IBEA	SPEA2
RS	**20**		0.87	0.28	0.0	0.14
	81		0.57	0.16	0.00	0.07
	203		0.73	0.14	0.0	0.03
	407		0.41	0.04	0.00	0.02
	813		0.65	0.03	0.0	0.00
RRHC	**20**	0.38		0.04	0.0	0.06
	81	0.33		0.13	0.0	0.08
	203	0.19		0.07	0.0	0.0
	407	0.26		0.03	0.00	0.02
	813	0.23		0.05	0.01	0.01
NSGA-II	**20**	0.57	0.62		0.0	0.16
	81	0.48	0.45		0.01	0.20
	203	0.39	0.36		0.00	0.16
	407	0.57	0.60		0.0	0.15
	813	0.81	0.84		0.01	0.09
IBEA	**20**	0.09	0.32	0.26		0.31
	81	0.27	0.43	0.27		0.34
	203	0.33	0.37	0.20		0.19
	407	0.33	0.35	0.27		0.25
	813	0.45	0.67	0.28		0.24
SPEA2	**20**	0.65	0.75	0.48	0.01	
	81	0.72	0.73	0.46	0.01	
	203	0.83	0.94	0.48	0.02	
	407	0.85	0.87	0.40	0.01	
	813	0.96	0.99	0.52	0.04	

Table 5: Summary of Binary Hypervolume comparisons across all simulation data subsets (normalized objective values). In the context of the comparison methods, algorithm *A* is listed in the first column, and algorithm *B* is listed in the top row. Larger values indicate that *A* outperformed *B*.

Algorithm	RSOs	RS	RRHC	NSGA-II	IBEA	SPEA2
RS	**20**		5.06e-02	9.97e-05	2.14e-03	1.18e-04
	81		2.10e-02	3.68e-06	5.87e-03	9.19e-07
	203		1.41e-02	1.61e-05	3.48e-03	4.68e-05
	407		1.70e-02	5.50e-06	5.37e-04	6.25e-07
	813		1.67e-03	2.10e-06	7.41e-04	1.33e-07
RRHC	**20**	6.59e-05		1.12e-05	1.60e-03	1.40e-07
	81	7.37e-06		1.12e-06	5.62e-03	2.98e-07
	203	2.05e-06		7.76e-06	2.86e-03	1.45e-08
	407	2.45e-04		1.36e-05	5.16e-04	8.32e-08
	813	3.82e-03		2.79e-05	5.46e-04	5.48e-06
NSGA-II	**20**	6.71e-03	5.71e-02		2.85e-03	5.26e-04
	81	6.19e-03	2.72e-02		6.85e-03	7.74e-04
	203	6.82e-03	2.09e-02		4.05e-03	7.22e-04
	407	1.04e-02	2.72e-02		9.37e-04	1.27e-03
	813	1.10e-02	8.87e-03		1.05e-03	5.61e-04
IBEA	**20**	6.69e-03	5.66e-02	7.90e-04		5.42e-04
	81	5.91e-03	2.66e-02	6.99e-04		7.21e-04
	203	7.36e-03	2.08e-02	1.12e-03		1.22e-03
	407	1.13e-02	2.81e-02	1.31e-03		1.59e-03
	813	1.13e-02	8.99e-03	6.47e-04		8.17e-04
SPEA2	**20**	6.85e-03	5.72e-02	6.55e-04	2.73e-03	
	81	5.97e-03	2.69e-02	5.59e-04	6.66e-03	
	203	6.61e-03	2.06e-02	4.87e-04	3.91e-03	
	407	9.98e-03	2.68e-02	8.31e-04	7.74e-04	
	813	1.08e-02	8.67e-03	3.80e-04	1.04e-03	

Table 8: Summary of Additive ε-Indicator comparisons across all simulation data subsets (normalized objective values). In the context of the comparison methods, algorithm *A* is listed in the first column, and algorithm *B* is listed in the top row. Smaller values indicate that *A* outperformed *B*.

Algorithm	RSOs	RS	RRHC	NSGA-II	IBEA	SPEA2
RS	**20**		0.22	0.07	0.35	0.03
	81		0.54	0.07	0.41	0.09
	203		0.12	0.06	0.21	0.10
	407		0.14	0.05	0.29	0.06
	813		0.01	0.44	0.06	0.07
RRHC	**20**	0.06		0.05	0.27	0.06
	81	0.03		0.05	0.19	0.04
	203	0.04		0.05	0.13	0.08
	407	0.03		0.03	0.16	0.37
	813	0.08		0.05	0.23	0.08
NSGA-II	**20**	0.36	0.58		0.30	0.15
	81	0.27	0.81		0.44	0.14
	203	0.46	0.58		0.38	0.14
	407	0.44	0.50		0.49	0.12
	813	0.39	0.65		0.55	0.31
IBEA	**20**	0.05	0.27	0.00		0.00
	81	0.13	0.67	0.01		0.00
	203	0.08	0.20	0.00		0.00
	407	0.00	0.02	0.00		0.00
	813	0.00	0.38	0.00		0.00
SPEA2	**20**	0.20	0.42	0.05	0.32	
	81	0.13	0.67	0.05	0.31	
	203	0.32	0.44	0.08	0.34	
	407	0.33	0.41	0.09	0.39	
	813	0.27	0.58	0.04	0.43	

Table 7: Summary of ε-Indicator comparisons across all simulation data subsets (normalized objective values). In the context of the comparison methods, algorithm *A* is listed in the first column, and algorithm *B* is listed in the top row. Smaller values indicate that *A* outperformed *B*.

Algorithm	RSOs	RS	RRHC	NSGA-II	IBEA	SPEA2
RS	**20**		3.01	2.15	6.78	2.14
	81		3.80	2.51	4.51	2.40
	203		3.10	1.98	17.18	2.68
	407		2.11	2.00	15.13	2.02
	813		4.01	2.42	11.69	2.41
RRHC	**20**	2.49		1.90	5.24	2.37
	81	1.57		2.16	3.26	1.94
	203	2.83		2.61	14.56	2.37
	407	1.63		2.12	10.23	2.33
	813	1.54		1.93	10.11	1.99
NSGA-II	**20**	10.94	13.43		18.56	7.53
	81	9.24	12.48		16.05	8.16
	203	2.18	3.01		20.44	5.86
	407	2.33	4.14		16.39	2.13
	813	2.44	6.88		12.21	1.68
IBEA	**20**	1.08	1.94	1.09		1.07
	81	1.30	4.47	1.31		1.30
	203	1.32	1.49	1.15		1.36
	407	1.08	1.15	1.11		1.15
	813	1.36	2.47	1.13		1.21
SPEA2	**20**	3.32	3.69	1.47	5.29	
	81	2.10	4.47	1.65	4.37	
	203	1.80	2.90	1.95	14.31	
	407	2.83	5.04	1.39	15.17	
	813	2.02	6.88	1.48	12.77	

5.3 Interpretation

These results do not provide a simple, definitive answer to the question of which algorithm to use on the problem in question. Instead, they present a multifaceted exploration of the strengths and weaknesses of the algorithms evaluated. These results can enable a researcher to make a decision based on the performance of algorithms, further informed by his or her understanding of what the solution must accomplish and what behaviors he or she finds desirable in an algorithm for the problem. In the case of SSN architectures, in light of the multi-billion dollar nature of the problem, and the long-term use of any solution, a researcher would likely choose SPEA2 as the best algorithm, in light of its balance between thorough exploration and aggressive exploitation of the search space, for parameter tuning and/or use on the full-scale problem.

The multifidelity analysis reveals that this specific problem can produce similar relative performances from algorithms even with very small subsets of the full simulation data set. This does not mean that the problem can be solved using reduced data sets; the objective values are dramatically skewed in two of the three objectives. It does mean that algorithms can be tested against this problem using very small data sets and, therefore, at a much lower computational cost, and still get reliable relative rankings of algorithm performance for the binary comparison methods presented.

6. CONCLUSION

The ability to evaluate the performance of multi-objective search algorithms on real world problems is an important tool in many disciplines. This research proposes a methodology for comparing the relative effectiveness of multiple multi-objective search algorithms on a problem for which the True Pareto Optimal Set is not known, as demonstrated on the SSN optimization problem. It further demonstrates one method of multifidelity analysis to determine if the methodology can be applied *a specific problem* at a lower computational cost. The methodology does not act as a black box, definitively stating which algorithm is best, but rather presents a multifaceted description of each evaluated algorithm's performance. This allows a researcher to make an informed decision based on the needs of the problem.

The multifidelity analysis reveals two important findings specific to this problem. First, SPEA2 appears to be the best algorithm for the SSN design problem. However, it is clear that each of the EAs have shortcomings that could benefit from parameter tuning, which might reveal one of them to be a much clearer winner. Second, this problem does tend to produce similar relative rankings with each subset of simulation data. This means that this methodology can be used to evaluate algorithms at a much lower computational cost than what is required to actually solve the problem.

Like many real-world problems, GEO SSA is an important issue with implications across many industries and national missions. This research provides another tool to understand such problems and apply the best methods possible to their solutions. Application of these techniques will surely enable researchers in this problem area, as well as others, to produce better results at lower computational costs than would have been otherwise possible.

REFERENCES

[1] AGI. STK Programming Interface, 2018.

[2] AGI. Systems Toolkit, 2019.

[3] Tom Berg. Irvine students launch second satellite in a month, 2018.

[4] Jason Brownlee. *Clever Algorithms*. 2011.

[5] Jamie Chin, Roland Coelho, Justin Foley, Alicia Johnstone, Ryan Nugent, Dave Pignatelli, Savannah Pignatelli, Nikolaus Powell, Jordi Puig-Suari, William Atkinson, Jennifer Dorsey, Scott Higginbotham, Maile Krienke, Kristina Nelson, Bradley Poffenberger, Creg Raffington, Garrett Skrobot, Justin Treptow, Anne Sweet, Jason Crusan, Carol Galica, William Horne, Charles Norton, and Alan Robinson. CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers. Technical Report October, 2017.

[6] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, 2 edition, 2007.

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 2002.

[8] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[9] ESA ACT. Pagmo / Pygmo / ACT / ESA.
[10] David Hadka. Playtypus (Version 1.0.3), 2015.
[11] Elizabeth Howell. What Is a Geosynchronous Orbit?, 2015.
[12] Joint Defense Science Board Intelligence Science Board Task Force. Integrating Sensor-Collected Intelligence. 2008.
[13] Oliver Kramer. Evolutionary self-adaptation : a survey of operators and strategy parameters. *Evolutionary Intelligence*, 3(2):51–65, 2010.
[14] Mohammadsadegh Mobin, Seyed Mohsen Mousavi, Mohammad Komaki, and Madjid Tavana. A hybrid desirability function approach for tuning parameters in evolutionary optimization algorithms. *Measurement: Journal of the International Measurement Confederation*, 114(June 2017):417–427, 2018.
[15] NASIC. Competing in space. Technical report, 2018.
[16] Moshe Sipper, Weixuan Fu, Karuna Ahuja, and Jason H. Moore. Investigating the parameter space of evolutionary algorithms. *BioData Mining*, 11(1):1–15, 2018.
[17] S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pages 399–406, 2009.
[18] Jordan Stern, Steven Wachtel, John Colombi, David Meyer, and Richard Cobb. Multi-objective optimization of Geosynchronous Earth Orbit space situational awareness system architectures. In *15th Annual Conference on Systems Engineering (CSER)*, 2017.
[19] U.S Joint Cheifs of Staff. Joint Publication 3-14 Space Operations, 2013.
[20] David H. Wolpert and William G. Macready. No-Free-Lunch Theorem. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
[21] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications (PhD dissertation)*. PhD thesis, Institut für Technische Informatik und Kommunikationsnetze, 1999.
[22] Eckart Zitzler and Simon Künzli. Indicator-Based Selection in Multiobjective Search. In *International Conference on Parallel Problem Solving from Nature*, pages 832–842, Heidelberg, 2010. Springer.
[23] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, 103, 2001.
[24] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.
[25] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert. Performance Assessment of Multiobjective Optimizers : An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.