SAND2019-2286C

Ultra-High Dimensional Optimization and Interactive Requirements Negotiation for New Acquisition Programs

Alexander I. Dessanti, John H. Gauthier, Stephen M. Henry, Matthew J. Hoffman
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-1178
Phone: 505-284-8087
Fax: 505-844-3321
adessan@sandia.gov

86th MORS Symposium
Working Group 27
27 February 2019

Abstract

The research described in this report was performed as part of the Sandia National Laboratories Laboratory Directed Research and Development (LDRD) program throughout Fiscal Years 2015 and 2016.

This LDRD project was conceived to address an issue that can present challenges for acquisition programs and ultimately contribute to delay or even cancellation: inconsistent or unachievable requirements. The result of this work produced ultra-high dimensional optimization algorithm enhancements (for problems with greater than 30 objective functions) and a new analytic capability called Advanced Requirements Integration and Exploration System (ARIES). ARIES provides developers of system requirements with analytical support for integrating requirements from different disciplines into a set of simultaneously achievable requirements by providing real-time feedback regarding the interactions between requirements. This feedback facilitates discussions to reconcile conflicts between requirements early in a program. ARIES was enabled by applying the novel ultra-high dimensional genetic algorithm optimization techniques that were developed during this research project.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2019-XXXX

CONTENTS

FIGURES

TABLES

NOMENCLATURE

1D	One-dimensional
2D	Two-dimensional
4D	Four-dimensional
ARIES	Advanced Requirements Integration and Exploration System
CDD	Capability Development Document
GA	Genetic Algorithm
GCV	Ground Combat Vehicle
ICD	Initial Capabilities Document
JEGA	John Eddy Genetic Algorithm
LDRD	Laboratory Directed Research & Development
MCoE	Maneuver Center of Excellence
MILP	Mixed Integer Linear Program
MPH	Miles per Hour
MRD	Mounted Requirements Division
NSGA-II	Non-dominated Sorting Genetic Algorithm-II
O	Objective
RPG	Rocket Propelled Grenade
S&T	Science and Technology
SME	Subject Matter Expert
SPEA	Strength Pareto Evolutionary Algorithm
T_D	Desired Threshold
T_P	Possible Threshold
UI	User Interface
US	United States
WSTAT	Whole System Trades Analysis Tool

1. PROBLEM OVERVIEW

Section 1 of this report outlines the problem being addressed, the motivation for it, and the general approach taken. In Section 2, the optimization challenges associated with this problem are detailed along with the novel algorithmic advances that address these issues. Section 3 describes the analytic capability that was developed to address the motivating problem, which was enabled by the algorithmic innovations. Lastly, in Section 4 transition opportunities and areas for future research are discussed.

1.1. Problem Description and Motivation

Early in the lifecycle of a government acquisition program (e.g., a new tank system, fighter jet, etc.), a highly complex and interdependent set of requirements is developed. These requirements define what the system being developed must do, typically with threshold (minimum desired) and objective (ideally desired) values for each requirement. For example, a military ground system's speed requirement might specify a 50 miles per hour (MPH) threshold level and a 75 MPH objective level – meaning that a prototype system must at least achieve 50 and ideally be able to travel at 75 MPH. These requirements are usually derived from user needs or current capability gaps; each requirement possesses strong rationale and analytical backing for its choice of the threshold and objective values.

A problem arises, however, when subsets of requirements (such as mobility or survivability requirements for a combat vehicle) are developed by independent teams of Subject Matter Experts (SMEs) who have a thorough understanding of their area of expertise, but lack an analytical methodology for ensuring their requirements interact appropriately with all other requirements. Thus, when all the individual requirements are integrated into a unified set, inconsistencies (such as requirement A cannot be met at the same time as requirement B) or unaffordable scenarios may arise. If these issues are not addressed early, they can present challenges to a program later in its lifecycle and ultimately contribute to programs being cancelled or delayed as the inconsistencies become known.

Traditionally, this requirements integration process is performed manually through negotiations between the relevant program stakeholders without analytic understanding of interactions. A capability to support these negotiations would reduce the risk of inconsistent or unaffordable sets of requirements persisting beyond the early stages of a program – guiding requirement threshold and objective levels towards a simultaneously achievable set. This capability would also help bring the analytic backing that individual requirements have to the integration negotiation process – providing defendable rationale to decisions that are made.

This requirements integration issue became starkly evident via repeated application of a related capability known as the Whole System Trades Analysis Tool (WSTAT). Developed by Sandia National Laboratories in conjunction with the United States (US) Army, WSTAT is a systems engineering tradeoff analysis tool that has informed numerous military acquisition programs – such as ground combat vehicles, unmanned robotic systems, aquatic transport vessels, and base camps. Broadly stated, WSTAT operates by decomposing a system into its constituent parts, mixing and matching the combination of these parts, and producing a spectrum of combinations

that best balance multiple competing design objectives. Extensive experience applying WSTAT to various programs led to the discovery of the requirements integration problem described above. While each program had unique reasons that caused this situation to arise (such as immaturity of technologies or a combination of requirements driving towards overly-expensive designs), the general problem was always present – the full set of requirements was not simultaneously feasible.

These observations led to the idea that an optimization approach considering both technological *and* programmatic constraints could be developed to facilitate integrating desired system capabilities into a simultaneously achievable set of requirements. Developing an analytic capability to inform how requirements should be set while a program is still early in its lifecycle would maximize the likelihood of successful fielding – making that the primary goal for this Laboratory Directed Research and Development (LDRD) project. The culmination of this research was the Advanced Requirements Integration and Exploration System (ARIES) capability – a unique combination of novel optimization techniques and collaborative decision support user interfaces that provides program stakeholders with a new analytic methodology for ensuring proper requirements integration from the outset of a project.

1.2. General Approach and Appropriate Optimization Technique

The goal of the ARIES capability is to guide stakeholders towards a single, optimal, simultaneously feasible set of requirements values (i.e., thresholds and objectives) for their program. Since each stakeholder will have a unique notion about which requirements are more important than others, ARIES must represent *all* possible choices of optimal requirements levels in an unbiased manner. In other words, ARIES must capture the *full requirements trade space* (those requirements values that best balance the potentially competing requirements metrics such as top speed, protection level, maneuverability, lethality level, etc.) so that an impartial, analytical examination of the trade space can be performed. In this section, we outline the general approach for capturing the full requirements trade space, as well as our rationale for the choice of underlying optimization methodology.

To start, it is fundamentally vital that every requirements solution presented by ARIES be *simultaneously feasible*. To ensure this feasibility, we borrow from the WSTAT framework with this key insight: the metrics measured for any *buildable combination of subsystems* implicitly represent a *feasible set of requirements values*. In other words, by optimizing over *buildable* combinations of technologies that optimally balance the requirements metrics, we also inherently optimize over *simultaneously feasible* requirements levels. Thus, emerging from this insight we can formulate the following necessities for the optimization technique. First, the technique must be able to address multi-objective optimization, since each requirement metric can be considered a unique objective function in which we desire good solutions in all metrics simultaneously. Second, the technique should be able to handle discrete decision variables (i.e., the choices of technology for each part combined into the system configuration). Third, the technique should handle any general requirements metric including non-linear and even non-closed-form calculations (since complex real-world system metrics are often themselves complicated derivations).

With these three necessities in mind, we chose evolutionary algorithms – in particular, a genetic algorithm (GA) – to perform the combinatorial requirements trade space optimization to find solutions that optimally balance a relatively large set of requirements (our system example described further in Section 1.3 has 37 unique requirements metrics). To justify this choice of technique, we spend the remainder of this section comparing GAs to another popular approach known as mixed-integer linear programming (MILP) – briefly presenting descriptions of these techniques and the basis for choosing a GA over a MILP in the context of the requirements integration problem.

Broadly speaking, GAs are meta-heuristics that mimic the biological process of evolution. A population of individuals or chromosomes evolve over a series of generations, during which the individuals mate and produce offspring. The individuals contain decision variables that describe a possible solution to the problem that is being optimized. In our example, an individual's decision variables describe the technologies that comprise a combat vehicle: engine, transmission, armor, weaponry, etc. As the GA progresses, offspring are created that differ from their parents by means of mutation (random alteration of one or more decision variables) and crossover (random combination of two or more parents' decision variables). Only the most "fit" offspring survive into future generations, and measures of fitness can vary by approach (selection methods for surviving solutions are discussed in Section 2). In this manner, the population evolves over the generations and approaches the best objective function value(s) – in our case, the combinations of subsystems that achieve good values in the requirements metrics. Since GAs carry forward an improving *population* of solutions, they naturally address multi-objective optimization which entails a spectrum of many different answers that balance the various objectives. Also, GAs place no restrictions on the structure of underlying metrics used to evaluate solutions; thus non-linear and even non-closed-form calculations can be readily handled.

By contrast, MILPs, perform a closed-form search procedure to produce a feasible, mathematically-verifiable solution to a single linear objective function (or a best-found solution with an associated optimality gap). In geometric terms, a MILP finds the largest (or smallest) point within a polyhedron in the direction defined by the linear objective function, where the polyhedron is formed by the intersection of linear inequalities that define the problem constraints. Technically speaking, the search space is the volume within the polyhedron intersected with the integrality restrictions of the decision variables. The search procedure (typically Branch and Bound, though many variations on this approach exist in practice) explicitly relies on the linearity of the objective function and bounding constraints. Thus, unless some reformulation of nonlinearities can be devised, the MILP approach does not naturally address the types of constraints and objectives typically seen in the requirements integration problem. It should also be noted that multiple objectives can be addressed using MILPs, but in practice this involves aggregating the objectives with a weighted sum, and sequentially solving multiple optimization problems – one for each unique weighting of the objectives.

Table 1 shows a summary comparison of the two techniques with potential pros and cons of each. GAs more naturally address non-linear objectives and multi-objective optimization – two qualities that are necessary for the requirements integration problem. Both GAs and MILPs can address the discrete variables used to define systems configurations. Thus, in total, the genetic-

algorithm technique is favored. For this project, the John Eddy Genetic Algorithm (JEGA), developed and widely applied by Sandia National Laboratories, was used as the baseline GA upon which testing and improvements were made.

Table 1. Characteristics of GAs and MILPs, two possible optimization techniques for determining the optimal requirements space (bold indicates a more desirable characteristic)

Characteristic	GA	MILP
Functional form of objectives and constraints	**Linear, non-linear, non-closed form**	Must be linear (non-linear can sometimes be accommodated via reformulations)
Number of objectives	**Typically up to ~5**	1 objective (2-3 can be handled by sequentially solving the problem under weighted sums of the objectives)
Certificate of optimality	None	**Optimality gap**
Variable types (continuous, binary, integer)	Any, though best for binary or small integer	**Any**
Parallel evaluations	Yes (though domination check can be a bottleneck)	Yes (though Branch and Bound syncing can be a bottleneck)

1.3. Example Application

To test the novel capabilities investigated in this LDRD, an exemplar case study needed to be selected. The US Army's Ground Combat Vehicle (GCV) acquisition program provided an ideal test case for several reasons. The GCV program was cancelled in early 2014, so it was not an active program during this research project (providing a stable set of requirements), but was recent enough that the target audience (requirements developers) would be familiar with the program. Second, the GCV program was cancelled at least in part because it had become too expensive to meet all the requirements for the system – typifying the problem being addressed in this LDRD. Finally, Sandia National Laboratories had previously provided WSTAT analytic support to the GCV program and therefore was familiar with the system and associated requirements metrics.

2. OPTIMIZATION CHALLENGES AND ADVANCES

In this section, we discuss the unique mathematical challenges associated with optimization via evolutionary metaheuristics in the context of requirements trade space analysis. As mentioned in Section 1.2, GAs exhibit several beneficial properties in that they naturally address discrete decision variables under multiple non-linear objective functions and constraints – inherent properties of the requirements trade space. Despite these benefits, however, difficulties arise when addressing problems with more than about five objective functions. The remainder of this section outlines the unavoidable need for ultra-high dimensional optimization (greater than 30 objective functions) as well as the novel mitigations taken to enable this capability.

2.1. Problems with Aggregation

While it may seem natural at first to think of requirements as *constraints* on the design of a system, it is more helpful to recognize that all requirements have an inherent direction of "goodness" and can instead be thought of as design *objectives*. For example, in the context of military systems, a higher top speed, a higher armor score, and a lower purchase price would be illustrations of improving directionality of mobility, protection, and cost requirements. Traditionally, requirements are abilities that a system must achieve to be valuable. Recognizing that there is significant subjectivity in the value of a system, one can argue that a viable solution that meets "depressed" requirements is better than an unviable program with unachievable requirements. The design challenge, then, is to make all requirements as good as simultaneously possible. Thus, when we refer to the "requirements trade space," we are referring to the set of Pareto optimal threshold levels for all requirements such that one threshold cannot be improved without also making one or more other thresholds worse. It is then a matter of negotiation to decide exactly which set of thresholds to use for the program.

The problem that quickly arises from this approach is that typical military systems have many dozens of independent requirements; capturing the Pareto optimal requirements trade space in this many objectives is still an open problem in the optimization literature. A common method for dealing with this many objective functions focuses on reducing their number via aggregation (i.e., weighted summation) into a smaller, more manageable set. Philosophically, the aggregation approach can be thought of as incentivizing compromise – often stated as a variation of "a small improvement in one objective at a large expense to another is unacceptable." Unfortunately, as will be shown, a compromise-based approach can severely obfuscate the tradeoffs between individual requirements – something that cannot happen if our capability is to be of use to real-world requirements developers.

To explicitly see the problem with objective function aggregation, even in a small number of objectives, consider the following simple numerical experiment. First, randomly generate k points inside an n-dimensional unit hypercube, with each dimension representing an objective whose direction of improvement is towards larger values. Furthermore, when generating these points, enforce that for each pair of dimensions, the generated points must fall beneath a pairwise "tradeoff limit" indicated by the dashed curves in Figure 1 (where panel one exhibits a "lax" tradeoff between dimensions 1 and 2 while panel two exhibits a "strict" tradeoff between dimensions 1 and 3). This mimics empirically-observed phenomena where physical laws and

technology availability limit simultaneous goodness of a solution in two objectives. Each point in Figure 1 represents a realization of possible threshold levels in *n* requirements, and the goal is to capture those points that represent the best possible tradeoff in all the requirements (shown by the blue lines).

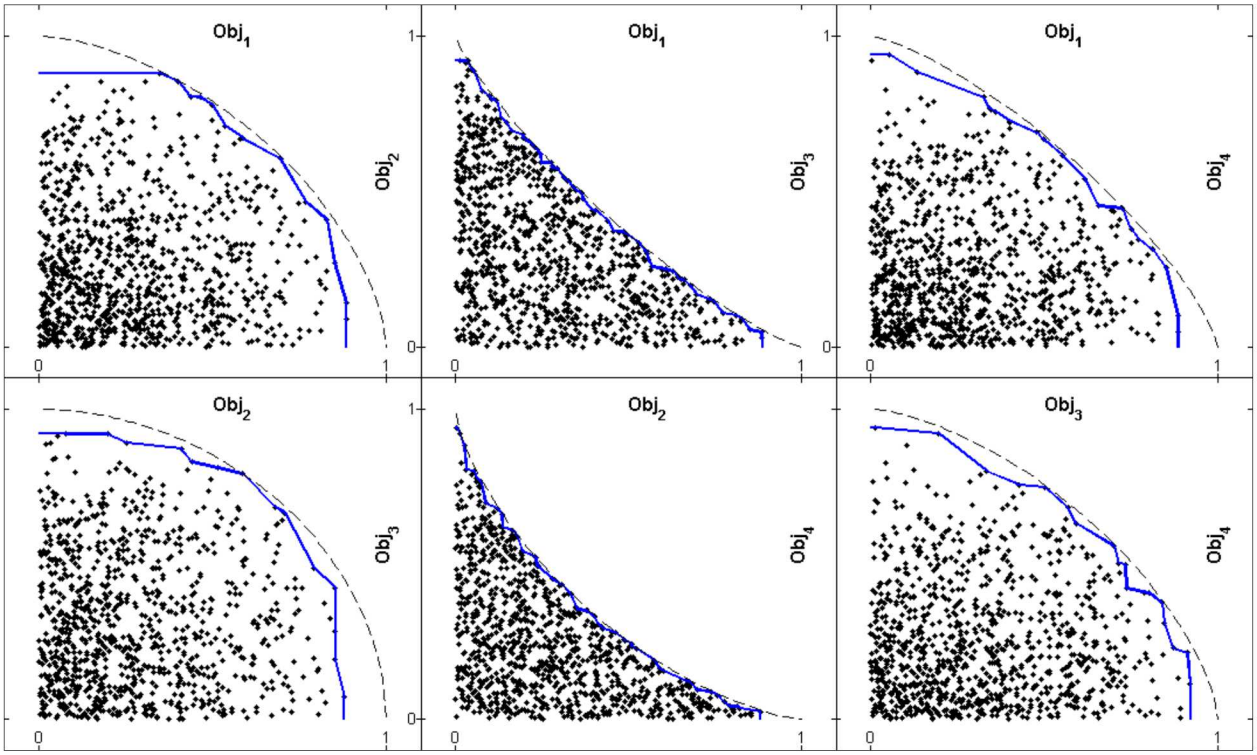

Figure 1. *k*=1000 random points in *n*=4 dimensions satisfying pairwise tradeoff limits

However, suppose that instead of performing optimization in the original *n* objectives, these dimensions are instead aggregated via simple addition and Pareto solutions are selected from the resulting lower-dimensional space. For example, Figure 2 displays the 1000 points from Figure 1 where the first two and the last two dimensions have been aggregated to reduce the space from four-dimensional (4D) down to two-dimensional (2D). While the 2D space may be more amenable to traditional multi-objective optimization techniques, the aggregate space is not guaranteed to preserve the tradeoffs that exist in the original 4D space.

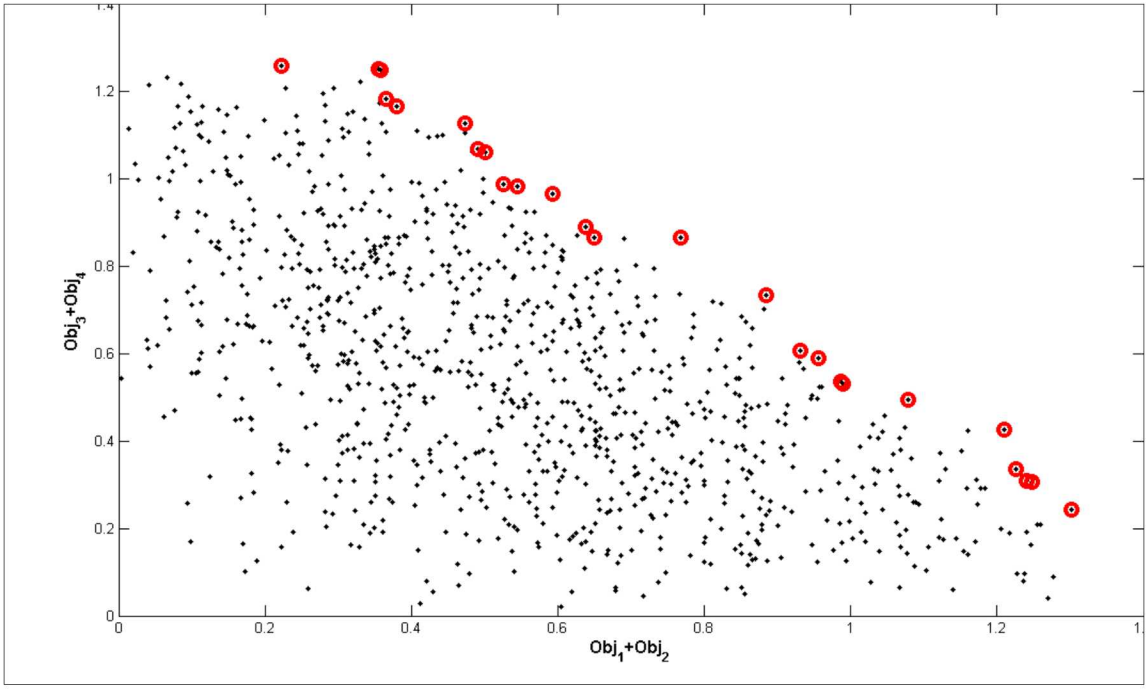

Figure 2. Pareto optimal solutions (red) in the aggregate 2D solution space

This loss of tradeoff information can be seen by comparing the red Pareto optimal solutions from the aggregated space in Figure 2 to these same points remapped back to the original 4D space as shown in Figure 3. Note that in some cases (i.e., the middle panels) the red points selected from the aggregated Pareto provide a passable representation of the pairwise tradeoff (the blue line). In general, however, the aggregated Pareto solutions do not capture a representation of the pairwise tradeoff. It follows that if the simplest 2D interactions are not generally well represented, then neither are higher-order interactions between three or more dimensions. For this reason, objective function aggregation is unfortunately not a viable means of acquiring the complex interactions of the requirements trade space; instead we must understand and address the challenges of optimization in 30 or more dimensions.

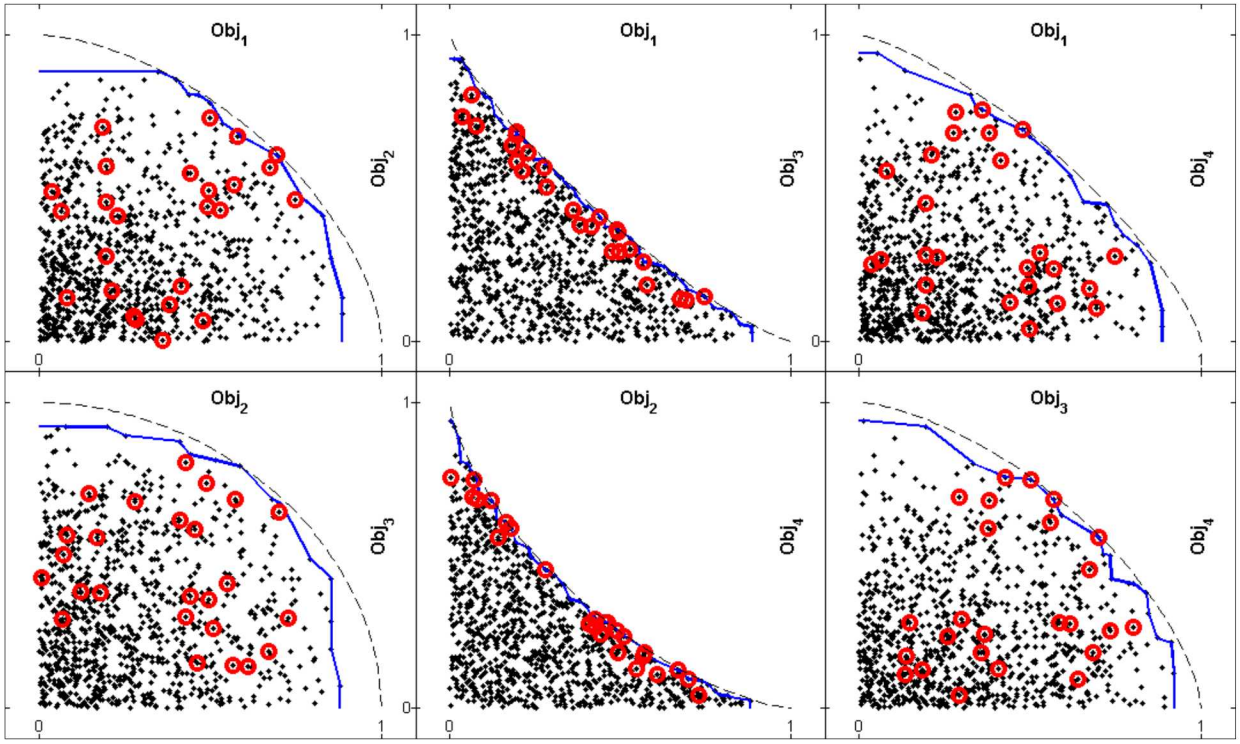

Figure 3. Aggregated Pareto points (red) do not represent pairwise tradeoffs (blue)

2.2. Challenges with Ultra-High Dimensional Optimization

Optimization in very many dimensions is rarely attempted in the current literature.[1] Most often, applications are amenable to the compromise-based philosophy mentioned above and do not explicitly require optimization that treats each metric as an independent objective function. In addition, it is very difficult to visualize and derive insight from very high-dimensional results (a challenge treated in more detail in Section 3). Thus, most operations researchers address these applications via methods that aggregate multiple metrics into one, or a small handful, of objectives. Furthermore, the difficulties with many-objective optimization are well known and while it is an active area of research (as evidenced in recent papers such as (Deb 2014)) there are not general solutions to these difficulties. One of the primary technical reasons for avoiding ultra-high dimensionality is that the traditional Pareto dominance criteria loses its ability to differentiate good solutions from bad. This is not to say that all solutions are equally good in a many objective problem, but rather, the probability that a solution is dominated by its peers (worse than its peers in all objectives) in a finite population becomes vanishingly small. In other words, as the number of objectives n grows, so too does the probability that a fixed number of solutions will all be relatively non-dominating to each other (each solution is likely to be equal or better in at least one objective than the other solutions).

This phenomenon can be demonstrated by generating k random solution vectors within a hypercube of dimension n, for several values of n, and calculating the percentage of non-dominated solutions, as shown in Figure 4 which was adapted from (Ishibuchi 2008). In this

example with $k = 200$ solution vectors, for $n \geq 14$ nearly all candidate solutions are non-dominated.

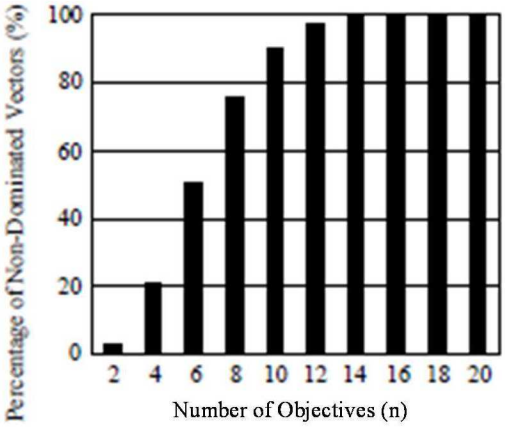

Figure 4. Average percentage of non-dominated vectors among *k*=200 vectors randomly generated in the *n*-dimensional unit hypercube $[0,1]^n$ (Ishibuchi 2008)

This situation occurs because the volume of the solution space grows much faster with n than the volume of any given solution's domination cone, so the probability of one solution dominating another solution shrinks drastically.

There are several cascading consequences of having most, if not all, solutions in a finite population being non-dominated. First, if Pareto-dominance is the sole criteria for culling the population size for a given GA, then this could lead to unbounded population growth beyond the fixed limits of system memory or human interpretability (i.e., there would be no solutions to discard). On the other hand, if the population size is bounded, Pareto dominance is non-discriminating and some other selection criteria needs to be carefully devised to encourage populations to both improve as the generations progress and eventually achieve convergence.

Existing many-objective methods introduce some other selection pressure to mitigate these issues, but they often incentivize compromise, whether explicitly (as methods based on relations *prefer* or *favor* (Sulflow 2007, Drechsler 2001), which score solutions higher if they are better in more objectives) or implicitly, such as with aggregation methods discussed in Section 2.1.

In addressing these challenges, several guiding principles/goals are considered. First is to efficiently characterize as much of the full spectrum of tradeoff information as possible. This means abandoning the traditional philosophy of compromise in the process of selecting which solutions to keep. Showing *only* compromises over-represents the "knee" of the tradeoff while providing little or even misleading tradeoff information about the full-dimensional trade space (as explained in Section 2.1). Instead, we desire information about the extremes of the n-dimensional trade space as well as the "knee" – and everything in between. Another goal is to avoid unnatural clumps (overrepresentation of an area) or holes (underrepresentation of an area) in the trade space depiction; we want the solutions presented to the user to be well-spaced from one another to give the fairest, most-representative information about the full spectrum of requirements interactions. These concepts are illustrated in Figure 5.

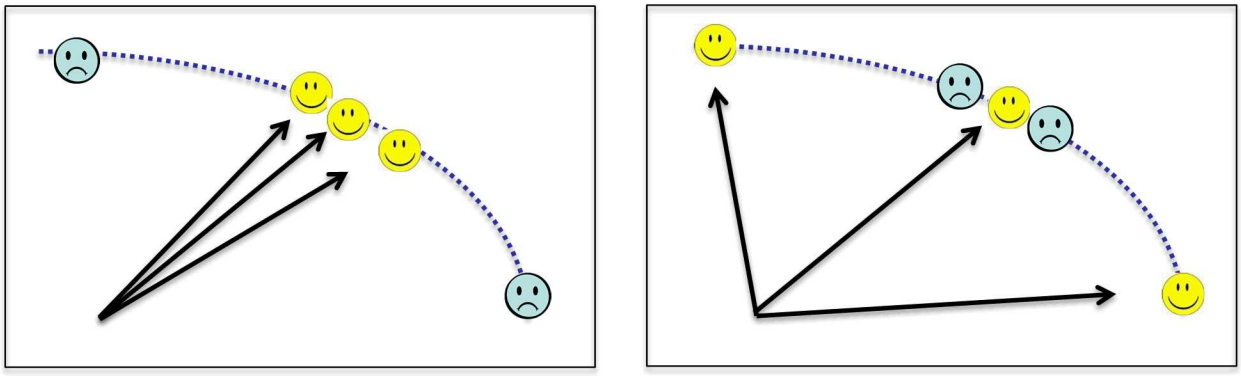

Figure 5. Comparison of compromise-based (L) and tradeoff-focused (R) philosophies

Understanding that the full richness of the trade space cannot possibly be represented, a solution set that includes each dimension's optimum and a well-spaced subset of points in between those dimensional optima is sought, giving significant insight into tradeoffs despite a limited population size. This philosophy is rather distinct from multi-objective optimization in the typical sense, in that we do not seek to find the highest-*performing* solutions, but the most *diverse* subset of non-dominated points.

The process that was settled upon over the course of this research is as follows. First, find extremal points via one-dimensional (1D) optimization for each of the objectives, and create an initial population from these extremal points. Next, run a multi-objective optimization heuristic, starting from this initial population, that 1) preserves extremes, 2) incentivizes/preserves diversity, and 3) stabilizes over time.

There were several research challenges in developing this process and particularly in modifying a GA to have the desired properties. First, it became apparent that there could be many alternate optima for a given single dimension, so we would need to determine a method for choosing *which* of these optima to preserve. Secondly, we would need to figure out *how* to incentivize diversity as the population evolved. The following subsections describe in more detail the steps taken to address these issues and why.

2.2.1. Seeding with Results of One-Dimensional Optimizations

Finding solutions that are optimal or "near-optimal" for each dimension is trivial within a multi-objective optimization heuristic – simply run it with only one dimension at a time for each dimension 1 through *n*. The points selected by this process then become the initial population for the multi-objective GA.[2] Determining the optima for each dimension up front has several benefits. First, it provides a significant amount of initial diversity since the "genes" that are optimal for one dimension are likely to be very different from those that are optimal for another. Secondly, knowing (approximately) the best possible value for each objective has analytic value of its own; it bounds expectations and, as discussed later, it can be used in discussions with SMEs for that functional area to vet the input data and evaluation method for that dimension. Finally, it guarantees that at least the most basic tradeoff information has been captured; for any

given dimension we at least know the best possible value for that dimension, and have many solutions that are suboptimal in that dimension because they are optimal in others. This provides a crude understanding of tradeoffs: which dimensions must "lose" for another to "win". At this point we have not explicitly concerned ourselves with the possibility of multiple optima for any given objective; that becomes important in a later step.

Once the initial population is created using the 1D optima, there are two major changes made to selection criteria within the GA to incentivize diversity.

2.2.2. Extrema Preservation

The previous section overviews the acquisition of 1D optimal values for each dimension and details how they are used as seed solutions for the full-dimensional requirements trade space optimization. It would be 1) a poor use of computational effort and 2) a betrayal of the tradeoff-focused philosophy if these extrema solutions were quickly forgotten during the full-dimensional evolution. Hence, we need a mechanism to ensure that the GA always preserves dimensional extremes from generation to generation – ensuring that 1D tradeoff information is not lost (and so that incremental improvements on the 1D extremes might be acquired).

At the onset of this research, as is demonstrated in the top panel of Figure 6, these extremes were not preserved in the baseline implementation of the JEGA algorithm. To see this, note that each of the 37 lines in the figure represents the best value of that corresponding dimension from generation to generation. Since the lines are generally non-monotonic – increasing and decreasing chaotically from generation to generation – this implies that the evolution is continually discovering and then forgetting the best solutions in each individual dimension.

To address this issue, we employ the following strategy to ensure certain solutions cannot be culled from the population. During each generation, for each dimension $i \in \{1, \dots, n\}$, we select the solutions(s) that have the best score in that dimension. Of these solutions that are best in i, we then preserve one point for each dimension $j \neq i$ that provides the best score for dimension j. This hierarchical preservation mechanism will keep at most $n(n-1)$ points, which is typically (for $n \approx 40$) much smaller than the maximum population size. This approach guarantees that we capture some of the most basic 2D tradeoff information – in essence asking "given that we are optimal in one dimension, how well can we still do in another dimension?" As can be seen in the bottom panel of Figure 6, the lines for each dimension are monotonically improving from generation to generation with this new approach – proving that the best solutions for each individual dimension are not being lost, and indeed are being incrementally improved upon in many cases.

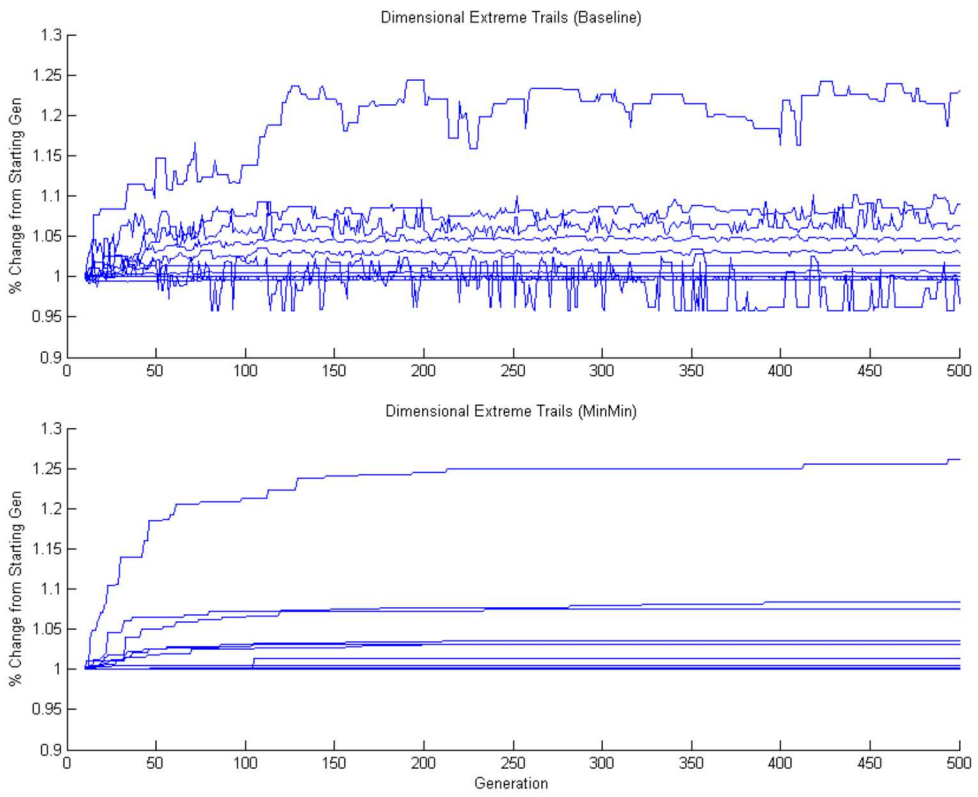

Figure 6. Trace of dimensional optima without (top) and with (bottom) extrema preservation

2.2.3. Space-Filling Niche Operator

Finally, without Pareto dominance as a driving selection pressure (and with inferior/infeasible points trivially removed), we adopt a new selection approach whose goal is to measure and dis-incentivize crowding. This is the purpose of a niche operator. We found that the existing Maximum Designs niche operator within JEGA (intended to limit the population size to stay within computer memory limitations, with selection based on a combination of domination and crowding scores) led to undesirable clumping behavior, which was readily apparent in 2D test problems as demonstrated in Figure 7. In this figure, the top chart shows the progress of a 2D Pareto frontier (black points) towards the true frontier (blue points) after 5000 generations. Under the existing niching methodology, emergent clumping becomes apparent in the black solutions. This is also evident in the lower graph of Figure 7, which plots a histogram of the Euclidean distance between all points and exhibits "spikiness" in the distribution. This strongly indicates that over-representation of some areas coupled with under-representation of others is being observed.

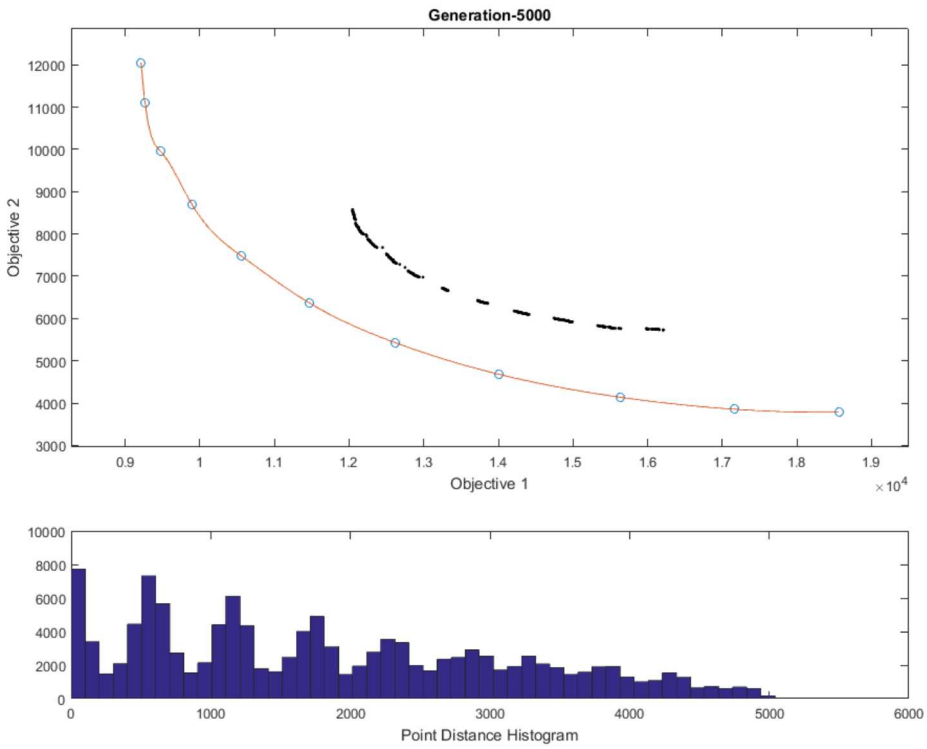

Figure 7. Emergent clumping in 2D toy problem with JEGA Maximum Designs

Upon investigation we were able to determine the cause of the problem. The Maximum Designs niche operator functioned based on the count of solutions within a given Euclidean neighborhood of each solution; the higher the count for a given solution, the less desirable it is to keep that solution since this is interpreted as significant local crowding. While intuitively this seems correct, we found that this led to an undesirable emergent property. Clumps of solutions just small enough that their penalties allowed them to stay within the population would emerge, and when they emerged close enough to each other, the niche operator reinforced the existing clumped population structure. As depicted in Figure 8, solutions *between* clumps – which intuitively seem desirable – actually receive worse scores than the solutions in the clumps themselves. This leads to such solutions being removed and the clumps persisting with gaps in between them.

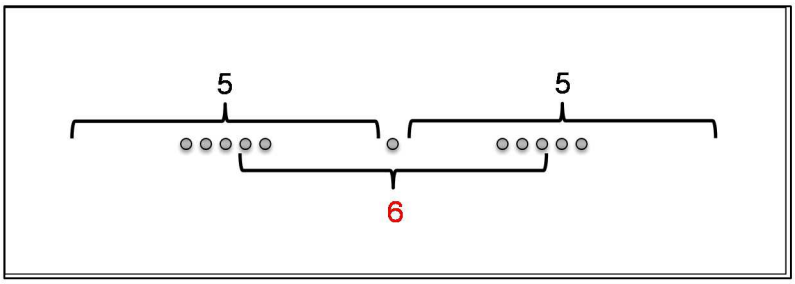

Figure 8. Solutions between emergent clumps are dis-incentivized

While theoretically a user could calculate an appropriate neighborhood size to mitigate, though not eliminate, this issue, it would likely be 1) difficult to determine and 2) problem-specific. Furthermore, such a neighborhood size would likely defeat the purpose of the niche operator. For instance, a very small neighborhood would eliminate the problem, but all solutions would be equal in the eyes of the niche operator; without a secondary selection criterion there would be no rationale for keeping any solutions over any others. A similar issue arises with very large neighborhoods. In between the very large and very small, changing the neighborhood size merely fine-tunes the size and spacing of the clumps.

To ensure a diverse, well-spaced solution set, we built a new selection heuristic. Given a maximum number of points to keep, M, this heuristic greedily tries to maximize the Euclidean distance (in normalized solution space) between selected points. The algorithm is summarized as follows:

- mark at least one seed solution for preservation into the next generation,
- find the unmarked solution whose minimum distance to any marked solution is greatest, mark this solution,
- continue the second step until M solutions are marked.

The progression of this heuristic for the first ten iterations is demonstrated on an example solution set in Figure 9. Gray points denote solutions in the larger population that have not yet been selected. The red point is the seed, and blue points are those that have been selected so far. Figure 10 shows the results for the same sample problem after 200 points have been marked.

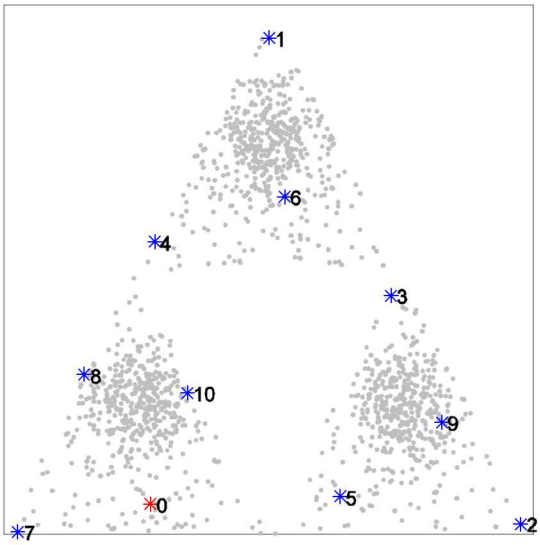

Figure 9. Sample progression of the space-filling heuristic

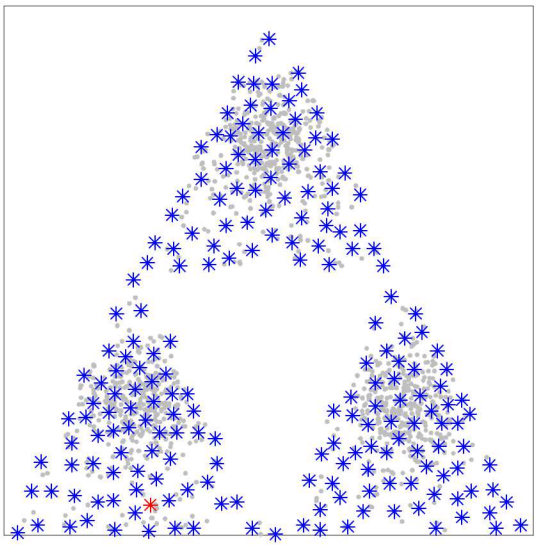

Figure 10. Result of space-filling heuristic with 200 points selected

In our application of this heuristic, since we intend for extrema to be preserved, these extrema are typically marked as our seed solutions. However, in general, final solution spacing quality is largely invariant to choice of the initial seed solution(s), and the heuristic almost invariably picks the extrema anyway due to their large distance from other points, as seen in Figure 9.

It should be noted that this example in Figure 9 and Figure 10 was chosen purely for demonstration of the heuristic; no 2D Pareto set would have this appearance, as the vast majority of these points would be dominated for any choice of two objectives.

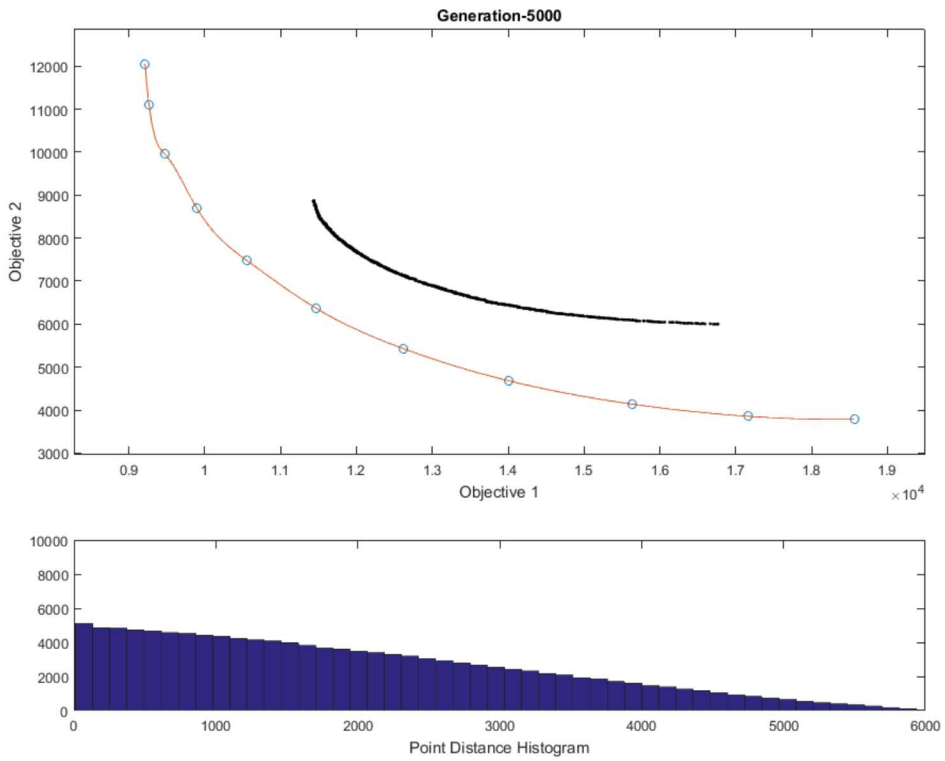

Figure 11. Improved spacing in 2D toy problem with Space-Filling Nicher

While the greedy heuristic cannot guarantee an optimally spaced set of points, it appears to work quite well and quite efficiently. As seen in Figure 11, it completely removes the clumping issues seen in the 2D toy problem, originally shown in Figure 7. Note that finding the *truly* optimally-spaced set of points is itself a very difficult and expensive optimization problem. In our tests, the heuristic achieved nearly the same diversity performance as a true optimization for small n and small population, in a much shorter amount of time. It remained solvable for larger problems, whereas the true optimization did not. Its complexity is $O(n^2)$.

As shown in Figure 12 and Figure 13, some dimensions may have much less variability in their outcomes, yet the space-filling heuristic still returns well-spaced subsets of these solutions. This is of importance because often the variation in the Pareto set can be represented in a basis of dimension less than n.

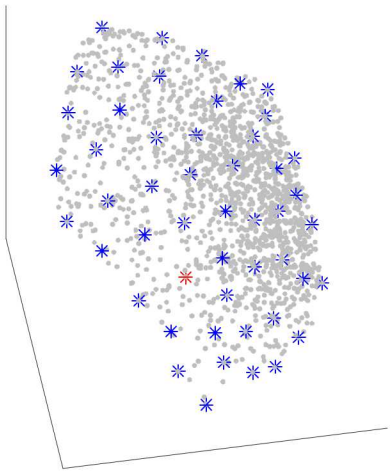

Figure 12. Space-filling niche operator performance with primarily 2D variability in 3-space

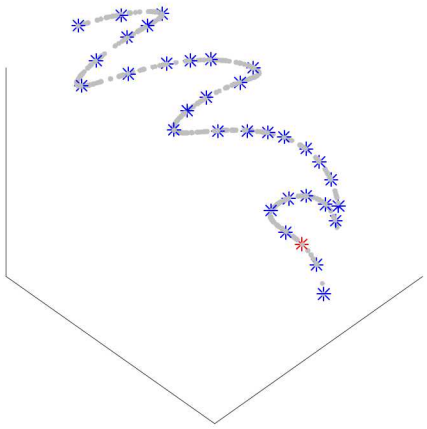

Figure 13. Space-filling niche operator performance with primarily 1D variability in 3-space

It should be noted that there are other niche operators in the literature, many of which do not exhibit the clumping issue observed with Maximum Designs. However, we believe the space-filling operator to be ideally suited to our application, as well as competitive with existing operators. For comparison, let us consider the niche/selection operators of two significant benchmark GAs: the modified Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb 2002) and the Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler 1999). An accessible summary of the selection processes for these two GAs can be found in (Deb 2005) but we summarize here.

NSGA-II uses a crowding approach (like Maximum Designs), but measured in distances rather than number of points within a neighborhood. NSGA-II first sorts the solution set by each objective, and then for each solution calculates the difference between the two closest neighboring solutions *for each objective*. The overall crowding score for the solution is then the sum of objective-wise differences. Note that for higher dimensions this is not generally equivalent to any standard measure of distance, as the "closest neighbors" in one objective are not generally closest neighbors in other objectives. SPEA uses an $O(n^3)$ iterative clustering technique. It starts by considering all solutions to be their own clusters, and iteratively merges the closest clusters until the desired number of clusters is achieved. A representative solution from each cluster is then chosen for selection. While the NSGA-II crowding method is more efficient at $O(n \log n)$, SPEA achieves much better diversity for more than two objectives as discussed in (Deb 2005, Deb & Thiele 2002, and Khare 2003). Although we have not had the opportunity for head-to-head computational comparisons, we believe the space-filling niche operator will have competitive diversity preservation compared to SPEA despite its lower complexity of $O(n^2)$. Furthermore, unlike other recently suggested niching/selection operators (such as those described in Deb 2014), it does not require the user to determine appropriate algorithm parameters or the *a priori* provision of search directions or reference points to preserve diversity.

3. ANALYSIS FRAMEWORK AND SUPPORTING PROCESSES

The ARIES capability provides insight to help stakeholders in the requirements development process integrate all the functionality desired for the system into a consistent, achievable set of requirements. In addition to being achievable, the requirement sets generated by this capability are also intended to be challenging (there should not be infinite ways to meet the targets) yet robust (there should be multiple ways to meet the targets).

The overall process associated with an ARIES analysis effort is shown in Figure 14. The initial steps, and most of the work in setting up an ARIES analytic capability, involve understanding the needs of the system being procured. Once the problem is well defined, a two-stage optimization is used to explore alternatives for the integrated requirement set with the first stage bounding the search space and the second stage identifying a representative set of alternative requirement sets. Following the two optimizations, optimal requirement sets are available and can be interrogated in real-time by the stakeholders to understand the available options and gain insight into the tradeoffs between different courses of action during the negotiation panel. Additional detail on each of the steps in the process is provided in the following sections.

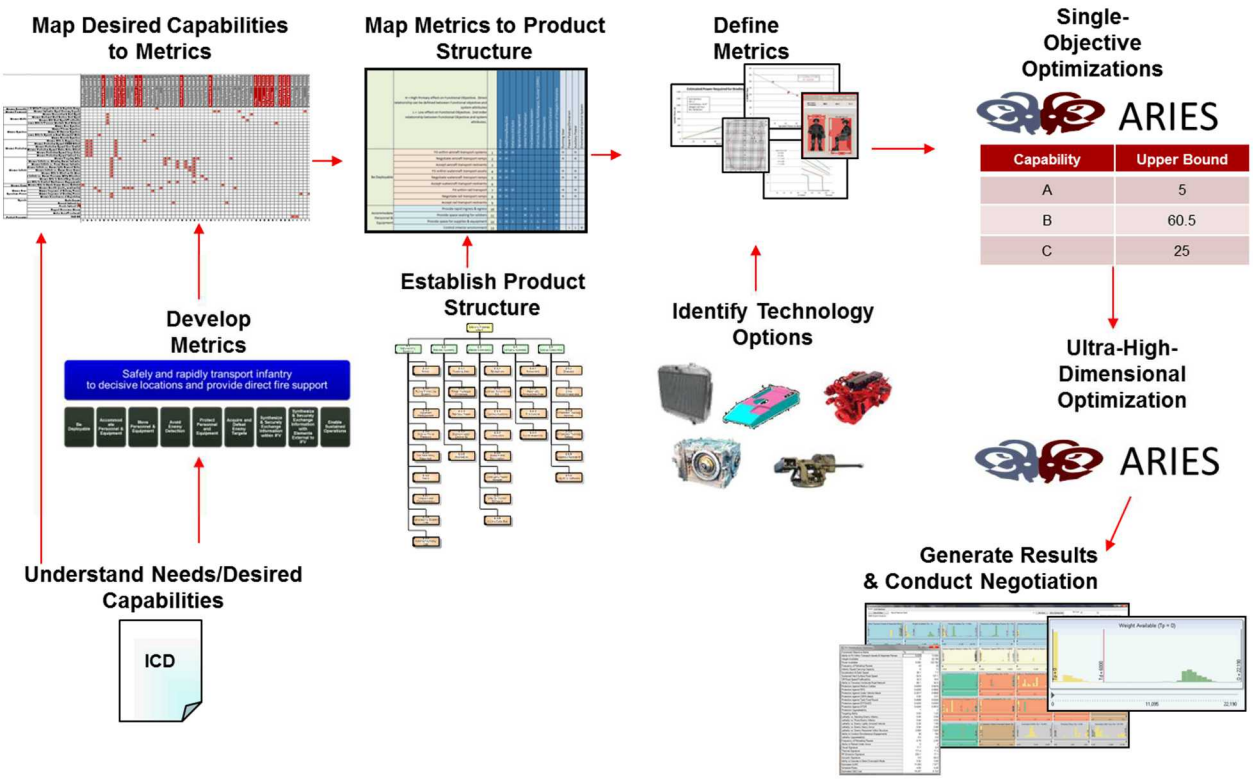

Figure 14. Depiction of Process Associated with an ARIES Analysis

Then result of the ARIES process is a set of threshold and objective values for each requirement where, at a minimum, all threshold values are simultaneously achievable. The outcome is based on compromises made amongst the stakeholders and provides the best solution within the budgetary, schedule, and technological constraints that were known. The resulting threshold and

objective values can be used to inform a Capability Development Document (CDD) or other requirements document.

3.1. Analysis Problem Definition

The first step in setting up an ARIES analytic capability is to understand the needs and desired capabilities for the system being developed. In defense acquisition these are typically laid out in an ICD, which guides requirements developers in defining requirements to satisfy the identified capability gaps. Once the requirements developers have determined which requirements are necessary to cover the needs of the system and the gaps identified in the ICD in preparation for creating a draft CDD, ARIES can be employed to help integrate and set threshold and objective values for the requirements.

Upon identification of which requirements to include, metrics that quantify the requirements and capture their essence must be conceptualized. These metrics will be the decision criteria for the ARIES optimization.

Next, the structure of the system being developed must be understood. By decomposing the system into a product structure, the requirements can be tied to design decisions.[3] This enables the requirements to be set with technological constraints adequately accounted for, ensuring that the requirement levels decided upon can be achieved or that focus can be shifted to investments in technologies that will allow the requirement levels to be achieved.

The last step in problem definition is to identify the technology options that are relevant to the system being designed, being sure to include even those technologies that have not yet been fully developed if there is a chance they could be of value to the program. This ensures that investments in revolutionary technologies that are beneficial and instrumental to achieving a desirable set of requirements can be appropriately prioritized against other technology development efforts. All this setup work to define the problem feeds into the two-stage optimization discussed in Section 2.2 and the associated subsections.

3.2. Negotiation Process and Interface

While previously discussed challenges were primarily numeric/analytic in nature, informing the requirements negotiation process is principally a human challenge in that the trade space must be presented in an intuitive manner that is agile to the needs of a real-time multi-stakeholder adjudication panel.

Typically for a military acquisition program during the early drafting of the CDD, a collection of stakeholders gathers and decides upon the official threshold and objective values that will be used for the program going forward. This is a critical juncture in the program lifecycle. As previously discussed, if the requirement levels are not set in a mutually feasible manner, then it is often discovered only later in the acquisition process and requires a costly (and sometimes programmatically fatal) recalibration of expectations and materiel solutions.

3.2.1. Elicit Desired Thresholds

An important precursor to the requirements negotiation is level-setting the expectations of the individual panel members by understanding their assumptions and predictions about the outcome of particular requirements. This task ensures that the "art of the possible" resulting from the ARIES tool for each requirement generally matches with the expectations of the SMEs who will be negotiating the threshold levels in the adjudication panel.

Another part of this interaction is capturing the SMEs best estimate for a desired threshold level. Again, this desired value needs to be in line with the art of the possible reported by the ARIES tool, and will serve as an important reference point for the unified negotiation process, as described further in the next section.

3.2.2. The ARIES Grid

The main ARIES user interface (UI) presents the requirements trade space to the negotiation panel and enables real-time interrogation and mediation of threshold values. Key challenges for this UI and the underlying negotiation process are to 1) display the complex, high-dimensional trade space in an intuitive manner to users not versed in Operations Research methodology, 2) provide a repeatable process that is not dependent on the order of who goes first in the negotiation, 3) disallow systematic exploitation by panel members to set their own threshold levels in such a way as to disadvantage everyone else, and 4) avoid situations where the negotiation becomes "stuck" or locked into a repeating cycle.

Keeping these four goals in mind, observe that Figure 15 presents the ARIES grid along with details for each "distribution panel" therein. The grid is laid out with one panel for each dimension in the requirements trade space, with groupings of related requirements subtly shaded and situated contiguously in the grid to aid usability. Each individual panel holds a "Requirement Distribution" plot – essentially a histogram of the values for that requirement that exist in the full requirements trade space. Along with this histogram, the plot details the following additional pieces of information:

A) A filter slider that points in the direction of improvement and allows users to entirely remove solutions that fall below the value of the slider from consideration. This is one of the primary means of interrogating the trade space, as a filter in one panel will affect the solutions visible in all other panels.
B) The current Possible Threshold (T_P) value given the filter settings on all panels. This is a key requirements value reported by ARIES, and all T_P values are guaranteed to be simultaneously feasible.
C) The Desired Threshold (T_D) value provided by the users, which represents their requested threshold value for that requirement. This is a static value elicited before the negotiation panel.
D) The current possible Objective (O) value given the filter settings on all panels. This is the second key requirements value reported by ARIES, and while not all O values can simultaneously be achieved, it is at least guaranteed that there is some combination of technologies that are able to achieve the value for each individual dimension.

E) The distribution of requirements solutions below Desired Threshold. These are colored yellow to visually indicate they are below the users' expectations.
F) The distribution of requirements solutions above Desired Threshold. These are colored green to visually indicate they meet or exceed the users' expectations.

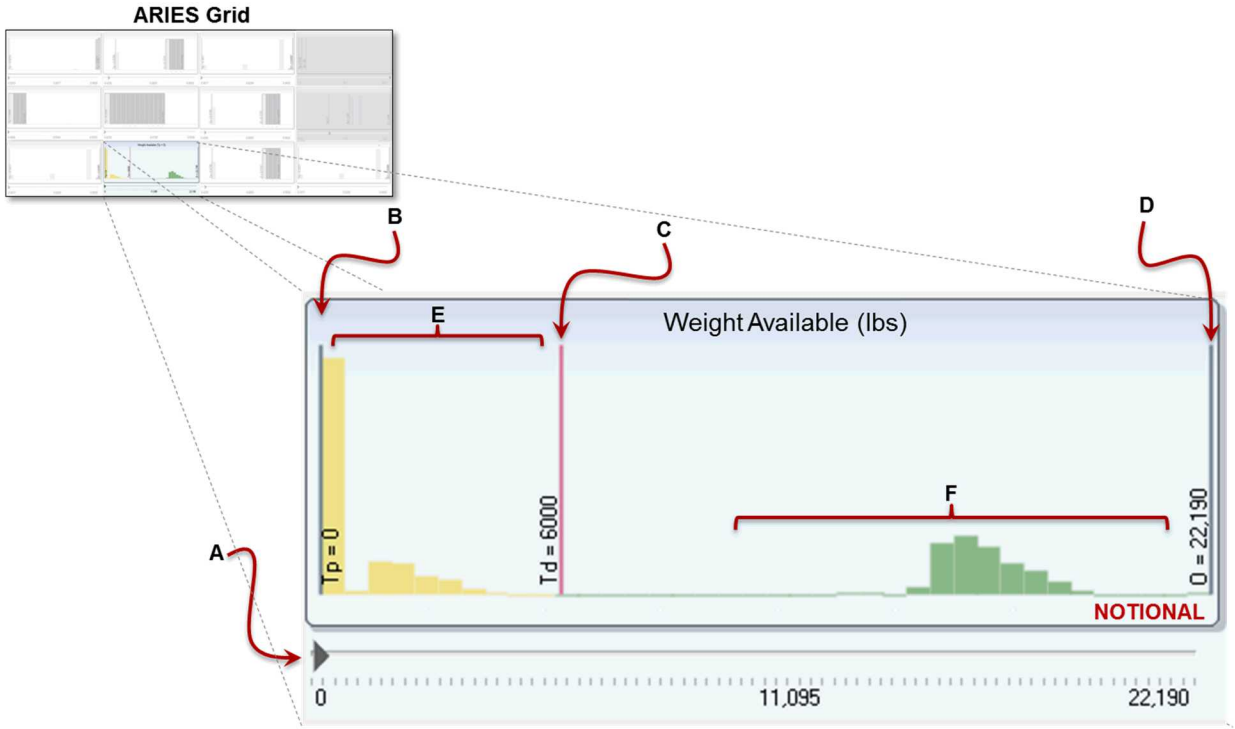

Figure 15. An example panel from the ARIES grid showing the distribution of values for the Weight Available requirement

The ARIES grid is designed to inform a negotiation panel by allowing individual members to perform filtration actions on their requirements panels to bring their possible threshold, T_P, as close as possible to their desired threshold, T_D. While the grid presents all information about all requirements given the filter settings, it is advised that individual panel members focus primarily on the few requirements panels of primary interest to them to avoid becoming overwhelmed by the full grid.

Inevitably, as the process proceeds and members perform more and tighter filtration actions to achieve their desired thresholds, some actions will harshly affect the remaining solutions available in one or more other requirement panels. It is important that ARIES has a built-in mechanism that 1) intuitively informs when these harsh actions occur, and 2) provides a method of "backtracking" the action in the least disruptive manner possible and without putting the process in a loop of repeated undo's and redo's. Naively, one might expect that simply undoing the last filtration event would suffice. If the unacceptable condition only emerged due to the last action, then this may work. However, given the human element of the negotiation process, it is possible the unacceptable condition went unnoticed for several rounds or most likely, that the unacceptable condition is a cumulative consequence of *all* prior filtration events, not merely the most recent one.

Figure 16 shows an example from the GCV test problem of a potentially unacceptable condition in the Rocket Propelled Grenade (RPG) Protection requirement panel on the ARIES grid. In this instance, notice that the direction of improvement is towards a higher protection score and the desired threshold was elicited at 0.8752 – above not only the possible threshold, but also above the currently best possible objective score of 0.7737. This situation has arisen due to filtration actions in other panels, which have inadvertently removed all solutions at or near the desired RPG Protection score. ARIES highlights this condition by coloring the panel red in the region where the best possible solution falls below the desired level; a larger red region indicates a larger gap between objective and desired. This red region provides immediate visual feedback when a filter action is performed and may itself inform or modify the filter action that produced it (i.e., relaxing filter(s) slightly to readmit solutions that meet the desired criteria).

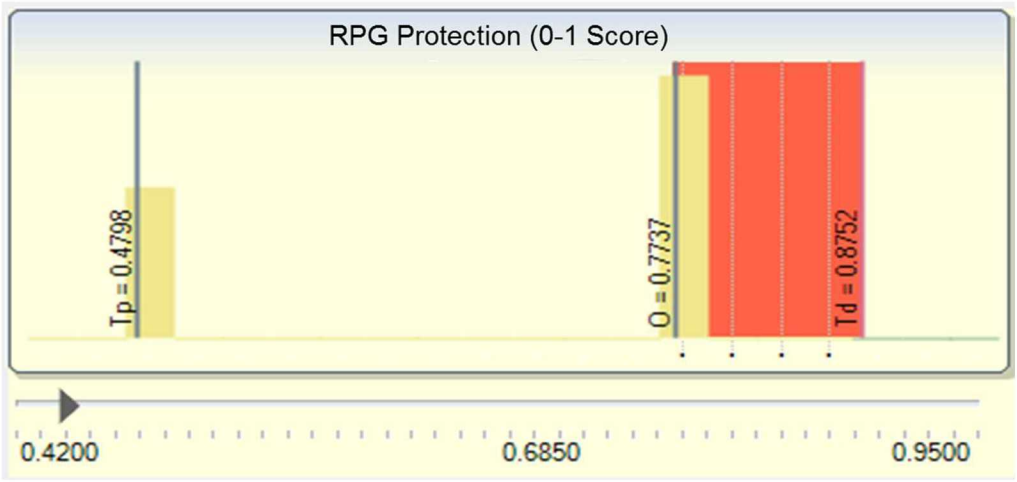

Figure 16. A single distribution panel from the ARIES grid showing a red region that highlights the gap between currently best available and desired levels

In addition to this visual cue, a ranking algorithm was also devised that chooses the best filtered-out point(s) to readmit in order to bring back desirable solutions into a panel, such as Figure 16, while having the least disruptive impact in other panels. To do this, solutions are ranked based on a specially-devised modification to the standard Euclidian distance metric. Dubbed "anti-optimal distance," this metric determines the distance between a solution and a target – only accumulating distance for those objectives where a solution is *worse* than the target.

Determining solutions with the lowest anti-optimal distance value allows identification of points that require the smallest relaxation of filters for readmission; if the current filter settings are considered the target, then the closest solution in anti-optimal distance will be admitted, moving the filters in the least disruptive manner.

Another usage of the anti-optimal distance ranking algorithm is to find the closest solution(s) to a desirable reference point – such as the point given by T_D in all dimensions, or a utopic point representing the best possible score in each capability area. By feeding this point to the ranking algorithm, one can trivially find solution(s) that are closest to such utopic points. If the histograms are filtered to admit only these near-utopic points, that allows for an alternate

negotiation mechanism where filters are progressively relaxed to add trade space, rather than progressively tightened to restrict it.

In summary, the ARIES grid provides a step-by-step method whereby a group of users can systematically tighten or relax filter settings to achieve a set of simultaneously achievable threshold values (T_P) while also having insight into how much better or worse each threshold value is compared to the elicited desired level (T_D). Immediate visual cues indicate when certain filtration actions create undesirable conditions (i.e., no remaining solutions can achieve the desired threshold). Figure 17 displays the full ARIES grid for 35 objective dimensions of the GCV example model. When an individual requirement panel has all remaining solutions better than the elicited desired level, the panel turns green. The ideal end state for the negotiation process would be to arrive at a set of filter positions for which *all* panels turn green, but this will be nearly impossible in practice for any real system. While SMEs must still choose which requirements are deprioritized compared to others, ARIES ensures the deprioritized thresholds are set in the best possible manner.

Figure 17. Full ARIES grid showing 35 requirement panels for the GCV model, with similar requirements clustered and similarly shaded

3.3. Process Vetting Activities

3.3.1. MCoE Discussion

SMEs and leadership at the US Army Maneuver Center of Excellence (MCoE) Mounted Requirements Division (MRD) were engaged early in the project to help the team fully understand the current requirements integration process. Initial engagements were in May 2015, where the premise of the research and vision for the end state was presented by the research

team. The audience's initial reaction was skeptical since the proposed capability was a significant departure from what had been done previously.

Upon discussion, the SMEs and leadership began to concede that there are challenges in the requirements integration process, especially once technological and programmatic constraints are considered, since it is such a complex problem. Eventually, there was consensus that an analytic capability could be of great benefit to the requirements development community and they expressed interest in partnering going forward and offered to help test the capability and process once it was developed.

This meeting provided the research team with a much better understanding of the existing requirements integration and negotiation process. This understanding proved invaluable to shaping the eventual outcome of the project as decisions could be made regarding the capability while considering the likely value and impact to real-world requirements developers.

3.3.2. Mock Panel

Initial vetting of the analysis process was performed on April 4, 2016 using a mock panel conducted with staff members at Sandia National Laboratories who were not involved in the development of the capability. The mock panel consisted of approximately twenty staff members playing the roles of requirements developers for the example problem identified in Section 1.3.

This mock panel focused on testing the functionality of the analysis capabilities included in the prototype tool, with the intent of identifying stumbling blocks and getting feedback on additional capabilities that users might find useful.

The participants quickly understood the intent and rapidly began to understand the insight being provided by the visuals. The tool proved intuitive and relatively easy for the participants to use. Most of the feedback received was on subtle ways to improve the user interaction with the tool, such as allowing the moderator to type in specific values rather than relying on sliding the filter to enable finer granularity, and have been incorporated as appropriate.

4. CONCLUSION

The result of this research included mathematical advances in ultra-high dimensional optimization via the development of innovative seeding, extreme preservation, and space filling niching techniques implemented with a prototype analytic tool that provides a unique requirements integration capability. The optimization advances are broadly applicable to GAs, while the ARIES analytic capability is generally applicable to requirements development efforts. The success of this project led to significant interest from potential transition partners and to several future application possibilities.

Despite the success of this research, multiple opportunities for future work to enhance this capability exist. Three primary focus areas are detailed within this section. Some thought was given to each topic, but budget and time constraints prevented thorough investigation.

4.1. Solution Inbreeding

Preliminary tests where 1D optimization results for each objective were seeded into the initial population for a pathologically difficult 2D toy problem showed a tendency to explore only locally near the seeded points. The observed behavior is caricatured in Figure 18, where the child solutions (round points) are only observed to be clustered around the initial seed points (star points) and do a poor job thoroughly representing the true Pareto frontier (dashed line).

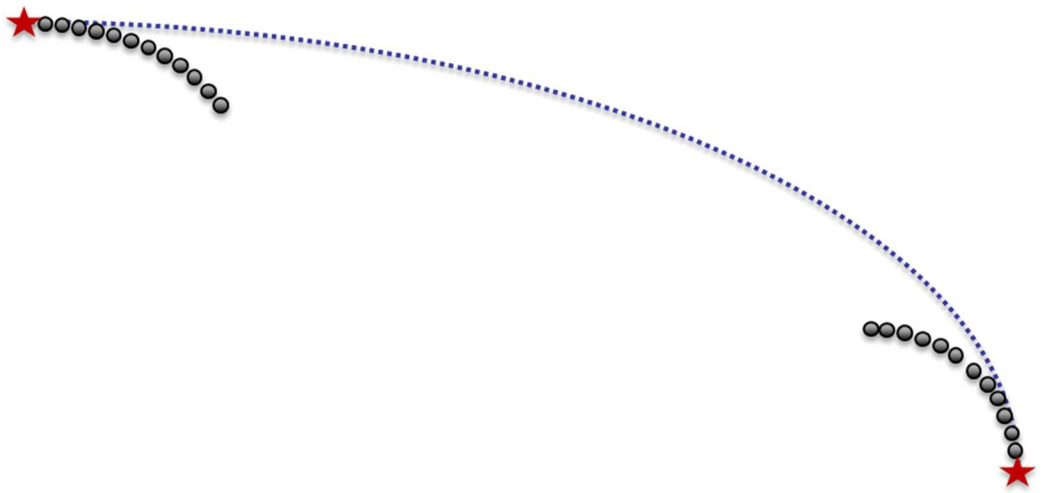

Figure 18. Demonstration of Inbreeding Behavior Observed

The primary hypothesis for this behavior is that other randomly seeded solutions whose offspring would eventually fill out the trade space tend to be prematurely dominated and killed off by the 1D optimal seeds. Though this is likely less of an issue in ultra-high dimensional problems where domination is harder to achieve, broadly speaking the issue can be summarized as later populations are largely "inbred" solutions from the initial 1D optimal seeds since those solutions already have a "leg up" over all other early solutions. Seeding 1D optimal solutions at knee points in addition to the extremes has a positive effect, but there is still a tendency for later populations to cluster around the seeded points. It quickly becomes intractable to seed an

adequate number of points to eliminate any unnatural gaps in the Pareto frontier, since each seed requires a separate optimization. Fortunately, a potential mitigation technique is an algorithm developed at Sandia National Laboratories that dynamically heals solutions that violate genetic constraints (technology compatibility dependencies) in an unbiased manner (this healing was not employed in the 2D toy problem). Healing ensures that child solutions are not eliminated simply because they violate problem constraints and allows large, but feasible changes in chromosomes. While we believe this would alleviate most issues with inbreeding, time did not permit the full investigation into realistic many-objective problems.

4.2. Data Visualization/Human Cognition Enhancements

As mentioned in Section 3.2, a significant hurdle to utilizing the results of an ultra-high dimensional optimization is in effectively presenting the data and overcoming human cognition limitations. While aspects of this were addressed within the scope of this project and a collaborative workspace was developed to utilize ARIES results, there is significant room to enhance the user experience.

One idea is for each stakeholder in the requirements integration process to have a tablet that displays the ARIES results. The user could then display only those requirements that are of importance to them on the tablet while the full set of requirements is projected to the entire group. This would enable each user to focus on the impacts of changes to their area of interest while a moderator focuses at a high level on the interactions between different users' requirements. Functionality to allow the moderator to designate control to a specific user would allow users a hands-on experience while communication between each tablet and a central system would still allow the entire group to see the impacts of each user's changes in real time.

4.3. Metrics to Aid Requirements Integration Process

An observation made while carrying out the mock ARIES requirements integration panel was that quantifying various aspects of the process would be beneficial. Specifically, developing metrics to inform which requirement should be addressed next, when to stop the negotiation process, and how robust the final set of requirement values is would provide great utility.

How to analytically determine which requirement should be addressed next (i.e., have its filter adjusted), is a remaining challenge to the flow of the process. It is not always evident which users are in the "best shape" and which have over-compromised their desired thresholds. This is complicated since users or groups of users are responsible for differing numbers of requirements. If one group is responsible for five requirements, of which one meets the desired threshold while the other four do not, and another group is responsible for one requirement, which does not meet the desired threshold, which group is in worse shape? The magnitude by which a user's requirements are unmet is also potentially of significance. Developing a fair metric to capture these considerations would promote improved negotiation flow.

Similar issues exist related to determining when the negotiation process should cease. Negotiation could continue indefinitely if users are unsatisfied and unwilling to accept the current values for their requirements. Quantifying the state of affairs throughout the process

would help inform whether the compromises being made are having a net positive or negative effect on the overall set of requirements. Being able to associate a value to the status of the negotiation could indicate to a moderator that progress is no longer being made and there is no reason to continue negotiating (i.e., the situation will get no better than it currently is).

Additionally, quantifying the robustness of the result could indicate if the consensus set of requirement values will be acceptable or if it will pose too much risk. A brittle solution would be one where there are only limited options for achieving the consensus set of requirement values (e.g., a specific developmental technology is required) and poses a significant risk to the eventual success of the program. A robust solution provides the greatest chance that the program will succeed. Measuring the robustness of the consensus decision would enable a moderator to raise the question of risk to the group of requirements developers and facilitate a discussion to determine if the risk is acceptable before finalizing the set of requirement values.

5. NOTES

[1] Definitions of "many" differ, but generally more than three is "many" and more than eight is quite rare.

[2] Randomly generated points can also be added to round out the population size and provide additional diversity.

[3] The product structure is comprised of major subsystem categories (e.g., engine, transmission, armor). This is distinct from technology options, which are unique technologies that can fulfill a subsystem role. For example, a 450 horsepower diesel engine could be one technology option for the engine product structure element.

6. REFERENCES

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6, no. 2 (2002): 182-197.

Deb, Kalyanmoy, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. "Scalable multi-objective optimization test problems." In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1, pp. 825-830. IEEE, 2002.

Deb, Kalyanmoy, Manikanth Mohan, and Shikhar Mishra. "Evaluating the ε -Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions." *Evolutionary computation* 13, no. 4 (2005): 501-525.

Deb, Kalyanmoy, and Himanshu Jain. "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints." IEEE Transactions on Evolutionary Computation 18, no. 4 (2014): 577-601.

Drechsler, Nicole, Rolf Drechsler, and Bernd Becker. "Multi-objective optimisation based on relation favour." In International conference on evolutionary multi-criterion optimization, pp. 154-166. Springer Berlin Heidelberg, 2001.

Ishibuchi, Hisao, Noritaka Tsukamoto, and Yusuke Nojima. "Evolutionary many-objective optimization: A short review." In IEEE congress on evolutionary computation, pp. 2419-2426. 2008.

Khare, Vineet, Xin Yao, and Kalyanmoy Deb. "Performance scaling of multi-objective evolutionary algorithms." In International Conference on Evolutionary Multi-Criterion Optimization, pp. 376-390. Springer Berlin Heidelberg, 2003.

Sülflow, André, Nicole Drechsler, and Rolf Drechsler. "Robust multi-objective optimization in high dimensional spaces." In International conference on evolutionary multi-criterion optimization, pp. 715-726. Springer Berlin Heidelberg, 2007.

Zitzler, Eckart, and Lothar Thiele. "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach." IEEE transactions on Evolutionary Computation 3, no. 4 (1999): 257-271.

7. DESCRIPTORS

- Optimization
- Genetic Algorithm
- Multi-Objective
- Ultra-High Dimensional
- Decision Analysis
- Requirements