Spring 2022

Advances and applications in high-dimensional heuristic optimization

Samuel Alexander Vanfossan

Department: Engineering Management and Systems Engineering

ADVANCES AND APPLICATIONS IN HIGH-DIMENSIONAL HEURISTIC OPTIMIZATION

by

SAMUEL ALEXANDER VANFOSSAN

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

SYSTEMS ENGINEERING

2022

Approved by:

Dr. Suzanna Long, Advisor
Dr. Benjamin Kwasa, Co-Advisor
Dr. Cihan H. Dagli
Dr. Steven Corns
Dr. Sid Nadendla

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following three articles, formatted in the style used by the Missouri University of Science and Technology:

A Geometrically-Based Method for Efficient Many-Objective Decision-Making, found on pages 24–48, has been published in the Proceedings of the American Society for Engineering Management 2019 International Annual Conference in Philadelphia, PA, in October 2019.

Ideal Sort: A Terminable, Efficient Nondominated Sorting Algorithm, found on pages 49–99, is intended for submission to *IEEE Transactions on Cybernetics*.

Disaster Recovery Strategy Generation via Multiobjective Heuristic Optimization, found on pages 100–168, is intended for submission to *Natural Hazards Review*.

ABSTRACT

Applicable to most real-world decision scenarios, multiobjective optimization is an area of multicriteria decision-making that seeks to simultaneously optimize two or more conflicting objectives. In contrast to single-objective scenarios, nontrivial multiobjective optimization problems are characterized by a set of *Pareto optimal* solutions wherein no solution unanimously optimizes all objectives. Evolutionary algorithms have emerged as a standard approach to determine a set of these Pareto optimal solutions, from which a decision-maker can select a vetted alternative. While easy to implement and having demonstrated great efficacy, these evolutionary approaches have been criticized for their runtime complexity when dealing with many alternatives or a high number of objectives, effectively limiting the range of scenarios to which they may be applied. This research introduces mechanisms to improve the runtime complexity of many multiobjective evolutionary algorithms, achieving state-of-the-art performance, as compared to many prominent methods from the literature. Further, the investigations here presented demonstrate the capability of multiobjective evolutionary algorithms in a complex, large-scale optimization scenario. Showcasing the approach's ability to intelligently generate well-performing solutions to a meaningful optimization problem.

These investigations advance the concept of multiobjective evolutionary algorithms by addressing a key limitation and demonstrating their efficacy in a challenging real-world scenario. Through enhanced computational efficiency and exhibited specialized application, the utility of this powerful heuristic strategy is made more robust and evident.

ACKNOWLEDGMENTS

To allies and adversaries: a sincere and humble *Thank you*!

130:1:49

141:12:5

57:3:69

26:5:6

5:20:16

161:16:76

22:9:8

159:16:9

41:13:11

114:18:1

※

Additional acknowledgements are extended to my advisors, Dr. Suzanna Long and Dr. Benjamin Kwasa, for their encouragement and direction, and for making my graduate career a truly enjoyable experience. Thank you to Dr. Steven Corns and Dr. Cihan Dagli, my de facto co-advisors, and to Dr. Sid Nadendla for contributing unique perspectives and insights.

Unequivocally, I thank my family for their relentless love and support, and for the provision of every advantage I have ever received.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION Page

PAPER III

LIST OF TABLES

1. INTRODUCTION

1.1. SINGLE- VS. MULTIOBJECTIVE OPTIMIZATION

An appropriate applicant to many real-world decision scenarios, multiobjective optimization differs from single-objective optimization in a number of ways. A principal difference lies in the nature of the *resolution* to problems of these distinct classes. Single-objective optimization is concerned with the identification of a globally-optimum solution exhibiting the best possible performance as determined by a single objective function (Figure 1.1). Multiobjective optimization, however, is akin to problem formulations possessing a number of conflicting objective functions that disallow the existence of a single, globally optimal solution. Instead, the solution-space is characterized by a set of *Pareto optimal* solutions wherein no solution unanimously optimizes all objectives (Figure 1.2). Also called the *Pareto efficient set* or the *Pareto frontier* of the solution-space, a tradeoff exists between the members of this set. Each solution is simultaneously inferior to all other members by at least one objective and superior to all other members by another. Without additional preference information, no solution can be selected as the single best alternative.

These common scenarios of conflicting objectives manifest regularly in a myriad of situations. Indeed, the operations of science, business, and life itself recurrently face scenarios demanding the simultaneous optimization of disparate interests. Developing and improving methods to address these prevalent situations are thus areas of considerable research interest.

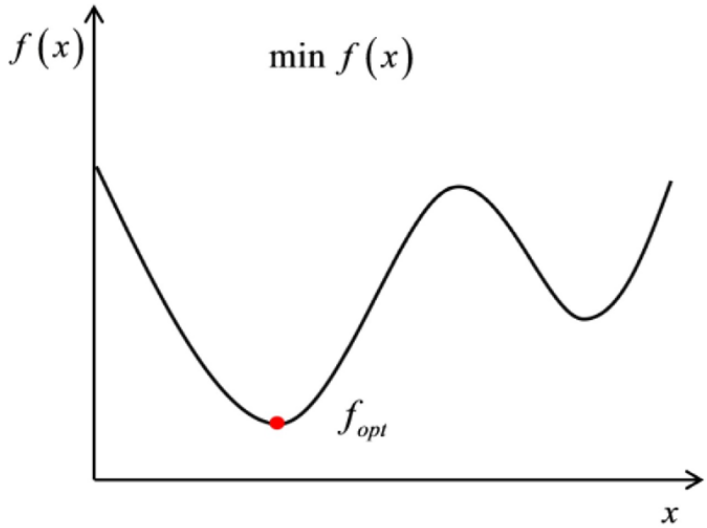


Figure 1.1. A single-objective optimization scenario with a globally optimum solution.

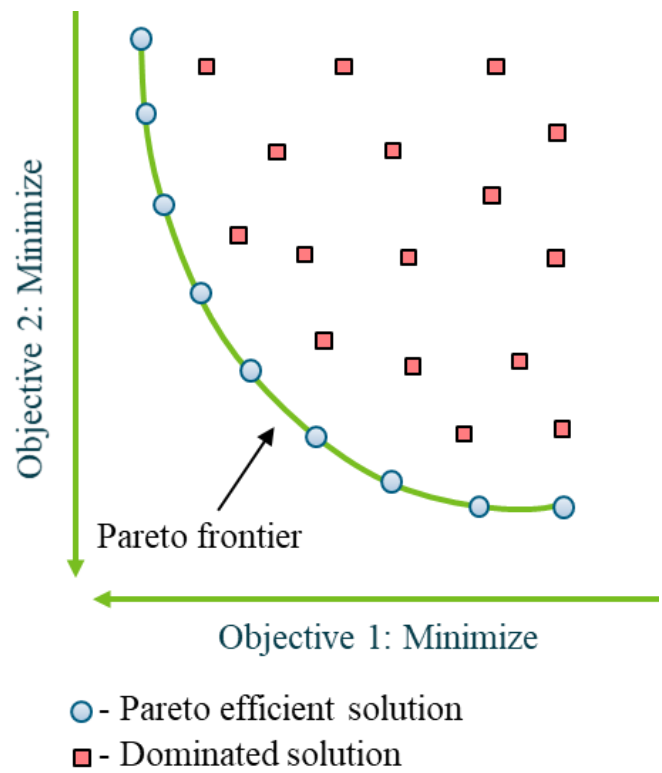


Figure 1.2. A multiobjective optimization scenario with two objectives.

1.2. AREAS OF APPLICATION

Multiobjective optimization has been applied to seemingly innumerable fields of study, occupations, industries, and circumstances. While this section does not scratch the surface of an exhaustive list of applications, a few examples are given to demonstrate the pervasiveness of this common decision-making situation.

1.2.1. Economics. Many aspects of macro- and microeconomics involve scenarios of competing simultaneous objectives. For instance, decisions along the production possibilities frontier describe the relative mix of products a society can produce (Hitch, 1953). Assuming full resource utilization, additional production of one product can only occur at the expense of another's production. A tradeoff then arises between the benefit associated with the increased production of the first product and the opportunity cost of producing less of another.

Governments and central banks use multiobjective optimization in establishing fiscal and monetary policy. Expanding upon the latter, institutions, such as the Federal Reserve in the United States, seek to establish policy that balances their stated objectives of price stability, low unemployment, and steady economic growth, among others (Federal Reserve Board, 2021; Dennis, 2002). While desired, achieving the optimum value of each of these objectives concurrently may not be feasible. Instead, the institution must enact policy promoting their preferred balance of these independent objectives.

1.2.2. Finance. A classic multiobjective optimization problem in the financial realm is the risk-return tradeoff of an investment portfolio (Mukerjee et al., 2002; Subbu et al., 2005). Specifically, investors desire their security portfolio to have a high excepted value of returns while also having low risk, usually measured by some variation metric,

such as standard deviation. However, these objectives are somewhat negatively correlated, with higher potential return values often associated with greater risk (Lundblad, 2007). Selecting a portfolio balancing this tradeoff is then an area of considerable interest for securities investors. A multitude of methods has been developed to generate and select well-performing alternatives along the Pareto frontier of these objectives (Mukerjee et al., 2002; Subbu et al., 2005; Chiam et al., 2007; Saborido et al., 2016).

1.2.3. Engineering Design. Multiobjective optimization has been applied regularly to engineering design problems spanning a range of disciplines. One such example occurs in aerospace design, where increased attention to the environmental impact of air travel has driven the design of commercial airplanes. Studies have sought to determine combinations of design and operational decision variables that simultaneously optimize greenhouse gas emissions and direct operating costs (Flores-Alsina et al., 2008; Sweetapple et al., 2014). *Greener* planes may produce less pollutants, but often come at the expense of increased operating costs which may be passed on to commercial customers. The strategies generated by these optimization models thus have substantial real-world environmental and economic impacts.

The design of sustainable buildings has employed multiobjective optimization in establishing well-performing renewable energy strategies (Kong et al., 2015). Given a set of renewable energy source alternatives, a multiobjective model is formulated, looking to concurrently optimize expected primary energy needs and resource investment costs. These models are used to guide decision-makers in determining the most advantageous array of renewable energy systems to deploy.

1.2.4. Medicine. A great variety of applications have been found for multiobjective optimization in the study of medicine. Drug design has used multiobjective optimization extensively in balancing numerous pharmaceutically important objectives (Nicolotti et al., 2011; Nicolaou et al., 2012; Nicolaou & Brown, 2013; Domenico et al., 2020). Conflicting directives such as effectiveness, safety, potency, and longevity, to name a few, are used in evaluating potential candidates for development. These optimization procedures search a vast decision-space of possible molecular structures, seeking to find solutions that perform well by each utilized objective.

Radiologic therapy plans are also developed using multiobjective optimization (Yu et al., 2000; Chan et al., 2014). Here, the amount of radiation interacting with different regions of the body is of critical importance. Concurrent objectives may look to minimize deviations from prescribed radiation levels for distinct regions including healthy tissues, critical tissues, and foreign masses (Aubry et al., 2006; Holdsworth, 2010). Finding an appropriate strategy, as evaluated by these objectives, can significantly bolster the procedure's chance of success while reducing the risk of unintended harm (Holdsworth et al., 2011).

This short survey of multiobjective optimization application demonstrates the amazing breadth of this paradigm's application. As application continues and the problems addressed become more challenging and complex, the efforts conducted to improve multiobjective optimization methods may become more vitally and broadly valuable.

1.3. MATHEMATICAL PRELIMINARIES

Here, several preliminary definitions are introduced to facilitate the understanding and discussion of multiobjective optimization terminology and properties.

1.3.1. Multiobjective Problem Definition. Principal is the definition of a (minimization) multiobjective optimization problem, given in Equation (1).

$$\min\big(f_1(x), f_2(x), \dots, f_k(x)\big) \quad s.t.\, x \in X \tag{1}$$

where the integer $k \geq 2$ is the number of objectives and the set X is the feasible set of decision vectors.

1.3.2. Pareto Dominance. As conflicting objectives likely rule out the existence of a globally optimal solution, a mechanism is needed to reason about solution performance. The principle of Pareto dominance is used to compare the attractiveness of multiobjective solutions. Defined in Equation (2), Pareto dominance is achieved when a solution-a performs just as well as another solution-b with respect to each objective of the optimization, while also performing better than solution-b with respect to at least one objective. In this scenario, solution-a is said to *Pareto dominate* solution-b and may be thought of as superior to the latter, as defined by the set of objective functions.

$$f_i(x^a) \leq f_i(x^b) \forall\, i \in \{1, 2, \dots, k\}\ and\ f_j(x^a) < f_j(x^b) \quad for\ at\ least\ one\ objective\ j \in \{1, 2, \dots, k\} \tag{2}$$

If the relationships of Equation (2) exist between two solutions (x^a and x^b) then x^a Pareto dominates x^b, assuming the minimization of each objective is aspired.

Solution-a's dominance of solution-b may be denoted as $x^a \prec x^b$. Alternatives between which a dominance relationship does not exist are called nondominating solutions.

1.3.3. Pareto Optimality and the Pareto Frontier. Solutions which are not dominated by any other solution in the set are regarded as Pareto optimal or Pareto efficient solutions. The set of all Pareto optimal solutions within the solution space is said to constitute the Pareto frontier of the optimization problem. Collectively, these solutions can be thought of as objectively better than all dominated solutions. However, no member of the Pareto frontier can be thought of as globally optimal. Pareto optimality and the Pareto frontier are defined in Equations (3) and (4), respectively.

$$x^a \in X \text{ is said to be Pareto optimal in } X \text{ iff } \nexists x^b \in X \text{ such that } x^b \prec x^a \tag{3}$$

$$\text{Those individuals that satisfy } \{x^a \in X | \nexists x^b \in X, x^b \prec x^a\} \text{ comprise the Pareto frontier} \tag{4}$$

Equipped with these definitions, the general proceedings of most multiobjective optimization algorithms can be understood and discussed.

1.4. MULTIOBJECTIVE SOLUTION SET QUALITY

Two considerations commonly describe the quality of a set of solutions to a multiobjective optimization problem (assuming the solution set aims to approximate the Pareto frontier of the solution space). Firstly, the convergence of the solution set's multiobjective alternatives to the actual Pareto frontier is evaluated. It is desired that the alternatives identified lie as close as possible to the true Pareto frontier. If successful, the identified alternatives are, indeed, part of the Pareto frontier and cannot be dominated by any feasible alternative in the objective space. Additionally, the distribution of the solution's multiobjective alternatives is a principal concern. A solution set is well-distributed if it represents the expanse of the Pareto frontier, and its constituent

alternatives are relatively equally spaced across this expanse. Well-distributed solution sets provide more information about the Pareto frontier and its associated tradeoffs, allowing for a more informed and vetted alternative consideration. Figure 1.3 describes a solution set that has a worse distribution than that of Figure 1.4. While both figures depict solution sets with 9 Pareto efficient alternatives, the solution set of Figure 1.4 is generally more valuable as it provides more diverse and complete information about the Pareto frontier.

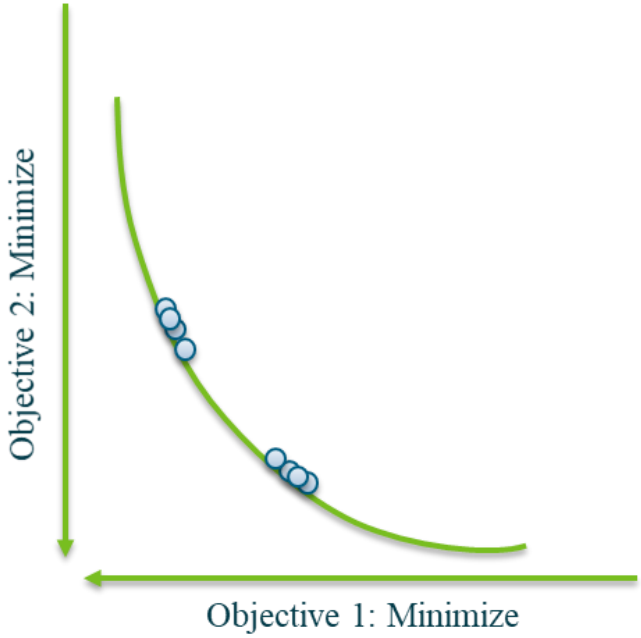


Figure 1.3. A poorly distributed multiobjective solution set with 9 alternatives.

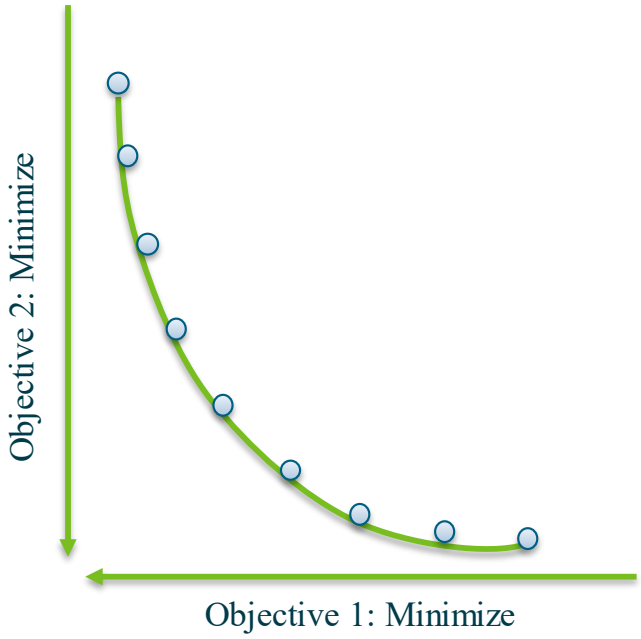


Figure 1.4. A well-distributed multiobjective solution set with 9 alternatives.

1.5. DISSERTATION ORGANIZATION

This section (SECTION 1) introduced the concept of multiobjective optimization and hinted at the scope of the paradigm's application. Additionally, some definitions and concepts were introduced which facilitate the understanding and discussion of multiobjective optimization procedures.

The remainder of this dissertation is organized as follows:

SECTION 2, *Multiobjective Optimization: Approaches and Critiques*, introduces several multiobjective optimization approaches from the literature, providing a brief critical analysis of select methods.

PAPER I, *A Geometrically-Based Method for Efficient Many-Objective Decision-Making*, presents a method of expediting the determination of the Pareto frontier of a

solution set, easing the restrictions this resource intensive procedure places on explorative rigor.

PAPER II, *Ideal Sort: A Terminable, Efficient Nondominated Sorting Algorithm*, extends the principles of Paper I, culminating in a NDS algorithm that achieves state-of-the-art performance in some cases. Further, the concept of terminability is introduced, a notion shown capable of improving the efficiency of other NDS algorithms from the literature.

PAPER III, *Disaster Recovery Strategy Generation via Multiobjective Heuristic Optimization*, applies a multiobjective evolutionary algorithm to disaster recovery strategy generation. This application demonstrates the feasibility of utilizing such a scheme in an incredibly challenging optimization scenario.

SECTION 3, *Conclusions and Future Work*, recounts the objective of these investigations: to advance the concept of multiobjective evolutionary algorithms by addressing a key limitation and demonstrating their efficacy in a challenging real-world scenario. Some directives for future research are also presented.

2. MULTIOBJECTIVE OPTIMIZATION: APPROACHES AND CRITIQUES

Multiobjective optimization methods have exhibited a number of forms and undergone a variety of transformations in approach across decades of research and innovation. While an exhaustive survey of proposed methods is impractical, some very prominent methods, their features, and acknowledged critiques are next discussed.

2.1. A PRIORI APPROACHES

A priori methods require the interjection of decision-maker preferences in effort to determine members of the Pareto frontier of a multiobjective optimization problem. These supplied preferences are generally incorporated within a priori methods to reflect the importance decision-makers place on varying objectives (Marler & Arora, 2004). These methods are then used to find the single Pareto efficient solution that optimizes the preference-adhering problem. In essence, these approaches look to transform a multiobjective problem into a single-objective problem by the introduction of the user-supplied preferences. While a sound means of finding a member of the Pareto frontier, this approach inherently introduces bias, as some objectives are assigned greater importance than their counterparts. This is problematic as the true significance of conflicting objectives may be very difficult to determine, variable with time, and/or disagreeable to different stakeholders. While the true spirit of multiobjective optimization is abandoned when following these a priori approaches, their ease of application, frequency of implementation, and capability to reliably identify solutions along the Pareto frontier warrant some further discussion.

2.1.1. Weighting and Scalarization Techniques. Weighting and scalarization techniques are perhaps the most simplistic means to address multiobjective optimization problems. These techniques use decision-maker preferences to form a single, parameterized objective function that can be optimized to find a Pareto optimal solution. Varying levels of complexity are applied to this objective function parameterization procedure, ranging from a simple linear weighting of independent objectives to more sophisticated product-based and exponential weighting strategies (Zadeh, 1963; Steuer, 1989; Yoon & Hwang, 1995; Saaty, 1977; Rao & Roy, 1989; Athan & Papalambros, 1996; Bridgman, 1992; Gerasimov & Repko, 1978). Each of these strategies introduce decision-maker preferences that impact the optimal solution identified by the parameterized optimization model.

2.1.2. Distance Function Methods. Distance function methods cast a multiobjective problem as a single-objective counterpart, looking to minimize the objective function distance between an optimal solution and some supplied multiobjective aspiration point (Charnes et al., 1955). When the aspiration point is unattainable (that is, an objectively better solution than can be attained given the problems objective functions and constraints) the identified optimal solution is Pareto optimal (Wierzbicki, 1986; Marler & Arora, 2004). Several evolutions of this distance function approach have emerged with varying levels of intricacy and utility (Charnes et al., 1955; Charnes & Cooper, 1957; Ijiri, 1965; Charnes et al., 1967; Charnes & Cooper, 1977; Hwang & Md. Masud, 2012; Gembicki, 1974; Ogryczak, 1994). It is important to note that the aspiration point utilized has ramifications on the optimal solution generated; in this way, bias is more subtly introduced by these distance function methods.

2.1.3. Constraint Methods. Constraint methods, sometimes called *bounded objective function methods*, seek to optimize the objective function identified as most important while ensuring that the other objective functions are within some range of acceptability. In this way, the constraint methods transform a multiobjective problem into a partially representative single-objective problem with additional constraints (Marler & Arora, 2004). User preference is thus interjected in both the selection of a single objective to optimize, and the acceptability bounds identified for objective functions translated into constraints. While preference bias is introduced, Pareto optimal solutions can be precipitated by constraint method application (Hwang & Md. Masud, 2012; Miettinen, 2012). Effort has been expended to develop and refine variations of these constraint methods and provide guidance toward the selection of appropriate acceptability bounds (Haimes et al., 1971; Goicoechea et al., 1976; Cohon, 2004; Stadler, 1988; Carmichael, 1980; Lin, 1976; Stadler & Dauer, 1992; Dauer & Krueger, 1980; Wendell & Lee, 1977; Corley, 1980).

2.2. NO-PREFERENCE METHODS

While the discussions of Section 2.1 make clear the inherent bias introduced by a priori preference methods, a warning is issued about the apparent remedy of no-preference methods. No-preference methods operate under the premise that the relative importance of objective functions cannot be accurately defined (Marler & Arora, 2004). This argument leads to a desire to treat each objective function as equally important and identify optimal solutions under this egalitarian prescription. Methods have been developed following this premise, utilizing various tactics to heed each objective function

equally in determining a Pareto optimal solution (Yoon, 1980; Stadler, 1988; Hwang et al., 1993; Mazumdar et al., 1991; Cheng & Li, 1996; Rao, 1987; Rao & Freiheit, 1991). However, the assumption that each objective function is similarly important is another form of bias, akin to assigning the same weight to each objective when scalarizing multiple objectives functions into a single measure. Therefore, variations of no-preference methods should not be regarded as an objective means to perform multiobjective optimization, even if their inherent subjectivity is more subtle.

2.3. MATHEMATICAL A POSTERIORI METHODS

The a priori and no-preference methods introduced previously are, as mentioned, not entirely veracious to the spirit of multiobjective optimization. Instead of considering multiple objectives independently, these instead find a means to convert the original problem into a single-objective problem that is, in some way, reflective of the multiobjective form. Because a single-objective nature is instilled, solving the modified problem usually results in the determination of a single optimum solution. This may be problematic as a single solution is not informative about the tradeoffs that can be made along the Pareto frontier of the solution space.

To remedy this shortcoming, a posteriori methods seek to supply the decision-maker with a set of Pareto efficient alternatives from which a preferred solution can be selected (Messac & Mattson, 2002). By delaying the articulation of decision-maker preference, a final solution can be identified with a greater knowledge of the Pareto efficient alternatives achievable. A posteriori methods are also helpful when the decision-maker finds it difficult to make an explicit articulation of objective preference a priori.

2.3.1. Weighting Methods. To provide a set of Pareto efficient solutions, a weighting method may be recursively applied with varying a priori preferences imposed. In doing so, the final solution identified by each run can similarly vary. In this way, the interjection of bias is used as a mathematical tool to generate alternatives, instead of a means to impose decision-maker preference (as in a priori methods). If repeated enough times, a nice set of Pareto efficient alternatives can be generated, providing some information about the Pareto frontier (Marler & Arora, 2004). While straight-forward, successfully implementing this multi-run strategy may be challenging. Specifically, it may be difficult to vary the imposed weights in a manner that produces a good representation of the Pareto frontier (Das & Dennis, 1997). Some mechanism to ensure a good distribution of the identified solutions about the Pareto frontier would lead to a much more informative set of alternatives for a posteriori consideration.

2.3.2. The Normal Boundary Intersection Method. The Normal Boundary Intersection Method provides a means to obtain an evenly distributed set of Pareto efficient points. Specifically, the method seeks to identify the portion of the boundary of feasible objective space that contains Pareto optimal points (Das & Dennis, 1998). To do so, a series of points along the convex hull of individual minima (CHIM) are systematically selected. These points are then projected normally to the CHIM toward the origin until they intersect the boundary of the feasible objective space (Das & Dennis, 1998). The determination of these intersections is completed algebraically by solving respective optimization problems. This method, however, can return points that are not Pareto optimal if the feasible region of the objective space is not convex. Further, this

method may overlook some Pareto optimal points when the number of objectives is greater than two (Das & Dennis, 1998)

2.3.3. The Normalized Normal Constraint Method. The Normalized Normal Constraint Method, like the Normal Boundary Constraint Method, seeks to determine a well-distributed set of Pareto optimal solutions. This procedure first determines the utopia point of the objective space and normalizes each objective (Messac et al., 2003). The individual exhibited minima of each normalized objective are then used to construct the utopia hyperplane of the objective space. A systematic weighting procedure then identifies a supplied number of points along this plane, which are next projected onto the boundary of the feasible objective space by solving respective optimization problems (Messac et al., 2003). Casually resembling the Normal Boundary Intersection Method to this point, the Normalized Normal Constraint Method finishes with a Pareto filter to ensure that only Pareto efficient points are returned by the procedure.

2.3.4. Brief Discussion. While the a posteriori methods presented are admirable in their ability to generate a set of Pareto efficient solutions for further consideration, they are generally encumbered by a few considerations. Firstly, these methods require the successive solution of single-objective sub-problems, used to reason about, and glean information from, the true multiobjective solution space. With each run, at most a single Pareto optimal solution is identified. While the process can be systematized to some extent, reasonable mathematical formulation and solution efforts may need to be expended. Further, these efforts, and the computational expense they incur, may become problematic as the number of objectives becomes large. Additionally, some of these

methods rely on knowledge of the utopia point of the objective space, an entity which may be difficult to determine in many instances (Wang et al., 2018.)

2.4. HEURISTIC A POSTERIORI METHODS

While the a priori and mathematical a posteriori methods previously introduced have sought to modify the formulation of a multiobjective optimization problem into an emblematic single-objective counterpart, several heuristic methods have been developed to solve multiobjective optimization problems directly (Marler & Arora, 2004). These heuristic approaches are often inspired by natural processes and consider each of a problem's objectives simultaneously, in quest of finding an approximation to the Pareto frontier. Many of the developed approaches maintain multiple solutions throughout their progression, finding natural application to multiobjective optimization where the Pareto frontier is generally comprised of many nondominated alternatives. While not guaranteed to find globally Pareto optimal solutions, these methods have proven adept at finding multiple objectively excellent solutions in a single algorithmic run (Deb et al., 2002; Hu & Eberhart, 2002; Bandyopadhyay et al., 2008). While a multitude of heuristic approaches have been developed, three very prominent approach avenues are described briefly, next.

2.4.1. Evolutionary Algorithms. Several multiobjective evolutionary algorithms have been developed, seeking to mimic evolutionary processes such as genetic crossover and natural selection to *evolve* a set of solutions toward the Pareto frontier of a solution space (Schaffer, 1985; Murata & Ishibuchi, 1995; Deb et al., 2002). In general, a random set of solutions defined by their decision variables are first generated and evaluated.

Some protocol is then used to *select* a subset of these solutions which will be used in the creation of a set of *offspring* solutions added to the population. Through user defined mechanisms, each of these offspring are created by combining the decision-variable information of two or more of the solutions selected from the initial population. In this way, new solutions may be generated that share some characteristics of the solutions used in their formation. Additionally, a defined mutation operator may be applied that randomly modifies some decision variable(s) within select offspring solutions to introduce entirely new characteristics. The new population (the solutions selected from the original population and the offspring they produced) are then subjected to a subsequent round of selection and the process repeats itself. When a selection mechanism is utilized that appropriately encourages the proliferation of well-performing solutions, the population can migrate toward the Pareto frontier of the multiobjective space. After meeting some stopping criteria, the algorithm is terminated, and a set of Pareto efficient solutions can be identified from the final generated population. While several mechanisms for the mentioned selection procedure have been developed, those employing some sort of Pareto dominance ranking scheme have emerged as some of the best performing and most widely applied (Srinivas & Deb, 1994; Horn et al., 1994; Zitzler & Thiele, 1999; Zitzler et al., 2001; Deb et al., 2002; Deb & Jain, 2013).

2.4.2. Particle Swarm Optimization. Particle swarm optimization is another biologically inspired optimization procedure that has been modified to solve multiobjective optimization problems (Hu & Eberhart, 2002). Generally, particle swarm optimization algorithms initiate a set of random solutions and maintain a mechanism to individually migrate these points toward better performing regions of the decision space

(Kennedy & Eberhart, 1995). Naturally inspired, this method mimics the behavior of flocking birds, schooling fish, and other swarm instances in their activities to find food, avoid predators, and optimize environmental parameters (Kennedy & Eberhart, 1995). Utilizing individual and *population-held* knowledge, challenging optimization scenarios can be effectively explored via the swarms managed by these approaches. Initially developed for single-objective optimization, alterations have been made to avoid the convergence of the method's agents upon a single solution. Often using clustering or some other diversity-preserving mechanism, these modified particle swarm optimization methods have shown the ability to discover well-distributed representations of the Pareto frontier in multiobjective space (Hu & Eberhart, 2002; Janson & Merkle, 2005; Coello et al., 2004; Pulido & Coello, 2004).

2.4.3. Simulated Annealing Approaches. Simulated annealing seeks to mimic the controlled cooling of metals and other materials to manipulate their physical properties (Kirkpatrick et al., 1983). Whereas metallurgic annealing is concerned with the heating and slow cooling of a metal to remove internal stresses and toughen the material, simulated annealing adopts analogous techniques to perform global optimization (Černý, 1985). In the most general sense, an arbitrary initial solution is set as the current *state* and is evaluated with respect to the objective function. A neighboring solution is then selected and similarly evaluated. If the neighboring solution is more optimal than the current state, the algorithm *moves* to this neighboring solution, setting it as the new current state. If, however, the neighbor is not more optimal than the current state, the algorithm may still elect to move to the neighbor by some probabilistically driven mechanism. While not described in detail here, this mechanism considers both the objective function difference

between the two solutions and a descending *temperature* parameter. When the temperature is high, there is a greater chance the algorithm can move to a less optimal solution; as the temperature lowers, a move of this kind becomes much less likely. At each step of the algorithm, this temperature is reduced by a supplied convention, resembling the cooling of a metal undergoing annealing. This ability to move to less optimal solutions allows the algorithm to escape local optima in search of a globally optimal solution. This procedure is repeated until the temperature is reduced to some predefined level or other stopping criteria are met. Originally designed for single-objective optimization, this method has been modified to handle multiobjective problems as well (Bandyopadhyay et al., 2008; Suppapitnarm et al., 2000). Typically, this involves the incorporation of some sort of Pareto archive to keep track of Pareto efficient solutions discovered and provide information about the relative performance of compared solutions.

2.5. INTERACTIVE METHODS

Interactive methods have carved out a niche within the taxonomy of multiobjective optimization solution approaches (Mäkelä & Miettinen, 2006; Branke et al., 2008; Miettinen et al., 2008). This iterative solution process periodically requests decision-maker input to guide the search toward a preferred solution. This interactive approach provides some utility as it incorporates preferences important to the decision-maker, but does so gradually, allowing a continued search process that gives the decision-maker updated information about the range of solutions attainable before requiring all preferences be made. While a creative and effective tool to learn about the solution space

and identify preferred solutions, the results of these methods are inherently subjective and require considerable effort from decision-makers to obtain. A survey of interactive method concepts, variations, and utilizations was conducted by Xin et al. (2018).

2.6. METHOD PREVALENCE AND RESEARCH DIRECTIONS

Each of the above methods have received considerable attention and application, obvious by their possession of the notoriety requisite of inclusion in this very brief survey. While a priori methods using simple scalarization techniques have enjoyed much usage —owing their simplistic and expedient implementation—, their deviation from a truly multiobjective consideration and the inherent bias they introduce before any solutions are generated have limited their applicability, veracity, and performance (Srinivas & Deb, 1994).

Examining a posteriori alternatives, evolutionary algorithms have emerged as some of the most widely applied, rigorously examined, and well-performing multiobjective optimization methods (Deb et al., 2002; Zitzler et al., 2001; Srinivas & Deb, 1994; Zitzler & Thiele, 1999). These techniques have shown the ability, in repeated evaluations, to converge to a well-distributed set of Pareto optimal points, producing an approximation to the Pareto frontier in a single algorithmic run (Corne et al., 2001; Deb et al., 2002; Zitzler et al., 2001). Further, they do not require much in the way of mathematical transformation or reformulation; all that is needed is a means to determine the objective function values achieved by programmatically generated solutions. These approaches also consider all objectives simultaneously, retaining the true spirit of multiobjective optimization. Indeed, evolutionary algorithms have emerged as some of

the premier methods to solve multiobjective optimization problems in the literature and in practical application.

While celebrated and well-performing, some criticisms of these biologically inspired algorithms have emerged. Principally, the computational complexity of their processes has been a major critique. Often, this complexity has limited the scale and scope of problems to which these algorithms may be applied. Therefore, substantial effort has been dedicated to improving the efficiency of these well-adopted optimization tools (Deb et al., 2002; Tang, Cai, & Zheng, 2008; McClymont & Keedwell, 2012; Wang & Yao, 2014; Zhang et al., 2015; Roy, Islam, & Deb, 2016; Mishra et al., 2018; Roy, Deb, and Islam, 2019). These investigations have considerably, and advantageously, improved the computational efficiency of multiobjective evolutionary algorithms, greatly broadening their range of applicability.

Accordingly, two cooperative research directives are here identified, relating specifically to enhancing the state and scope of multiobjective evolutionary algorithms:

1. Improve the computational efficiency of multiobjective evolutionary algorithms
2. Demonstrate the utility of multiobjective evolutionary algorithms by applying them to challenging optimization scenarios

These directives are complimentary in that achievement with respect to one sponsors effort and potential achievement in the other. For instance, improving the computational efficiency of multiobjective evolutionary algorithms enables their successful application to a more complete and challenging set of multiobjective optimization scenarios. Reciprocally, the successful application of multiobjective evolutionary algorithms to challenging optimization scenarios demonstrates their range

and utility, further sponsoring efforts to make them more efficient and widely applicable. Researches demonstrating achievement toward either of these directives serve well to further the study of multiobjective evolutionary algorithms and the broader paradigm of multiobjective optimization.

PAPER

I. A GEOMETRICALLY-BASED METHOD FOR EFFICIENT MANY-OBJECTIVE DECISION-MAKING

Samuel Vanfossan

Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, MO 65409

ABSTRACT

Practitioners of the systems engineering discipline are increasingly asked to make decisions from large sets of alternative solutions while considering the conflicting interests of diverse system stakeholders. Formulated as many-alternative, many-objective optimization problems, a posteriori methods are often applied to these scenarios to determine the solution alternatives that are objectively best performing according to the diverse stakeholder preferences. Frequently operating under computational and temporal constraints, decision-makers are often forced to consider fewer alternatives or incorporate a smaller number of stakeholder preferences due to the inefficiencies of current a posteriori methods. Utilizing a geometric comparison to the *ideal point* of the solution-space, a method is proposed that seeks to reduce the computational and temporal expense of determining the set of objectively superior solutions. In a numerical comparison to current methods, the proposed was shown to exhibit improved efficiency across a range of many-objective test-classes. Equipped with these efficiency allowances, systems engineering decision-makers can consider more alternatives and a greater number of stakeholder preferences without violating computational or time restrictions. These

liberties enable a more complete and tailored search of the solution-space, permitting the identification of more thoroughly vetted and scrutinized objectively superior engineering solutions.

Keywords: Pareto frontier, Pareto efficient set, many-objective decision-making, geometric presort, ideal point comparison

1. INTRODUCTION

The systems engineering discipline frequently demands its practitioners make decisions from large sets of alternative solutions while serving the conflicting interests of diverse stakeholders (Crawley, Cameron, & Selva, 2016). Sponsoring this demand has been the increased employment of model-based systems engineering. This technique allows systems designers to develop an ever-greater number of solution alternatives quickly, while avoiding many of the inhibitive costs associated with traditional system development approaches (Ramos, Ferreira, & Barcelo, 2012). While this increased exploration of the solution-space has facilitated marked advancement in creativity and innovation, it has greatly increased the burden associated with systems engineering decision-making. Decision-makers are now expected to consider a large number of solution candidates in justifiably selecting the preferred alternative (Crawley, Cameron, & Selva, 2016). Further complicating the selection process are the alluded diverse interests that nominated system solutions must serve. As engineering endeavors become more complex and global in scale, the parties contributing to a system's creation, and those expecting service from its enaction, greatly increase in number (Leybourne,

Kanabar, & Warburton, 2010). These stakeholders often house conflicting interests and dissimilarly perceive system value, creating scenarios where different alternatives are the preferred solution of different interest parties. Consider the simple example of two stakeholders interested in the development of an aircraft. The first stakeholder may desire the vehicle have as large a carrying capacity as possible, while the second may desire the craft's fuel economy be maximized. Because of the noncooperative nature of the desires, it is unlikely that a single solution is the globally preferred choice of both stakeholders. As the number of conflicting stakeholders increases, the likelihood that a universally preferred alternative exists decreases (Marler & Arora, 2004) Instead, the decision-maker will likely be faced with a set of alternatives that are variably attractive to different stakeholders in accordance with their respective conflicting preferences.

The described scenario, characterized by a high-volume of solution candidates and diverse stakeholder preferences, is indicative of a many-alternative, many-objective optimization problem. This class of optimization problems is defined by the use of many (more than 3; whereas a multiobjective problem has 2 or 3) objectives in the selection of alternatives from a set of many candidate solutions (IEEE, 2018). In solving many-objective optimization problems, two method classes are used. The first method class, *a priori methods*, requires preference information be expressed prior to conducting the optimization (Hwang & Masud, 1979). These methods typically involve the scalarization of the multiple objective functions into a single function based on the information supplied by the decision-maker. Strategies of this class include weighting or utility function techniques, lexicographic methods, and goal programming (Srinivas & Deb, 1994). While this class of methods serves well to reduce the set of candidate solutions to

a fitting alternative, each variation requires extensive user input that is inherently subjective, detracting from the veracity and robustness of generated solutions. The second method class, *a posteriori methods*, operates to provide the decision-maker with the set of alternatives objectively performing the best with regard to the independent objective functions (Marler & Arora, 2004). This provision proves extremely useful to the systems engineer as it reduces the consideration set of solution candidates to a condensed set of well-performing alternatives. Further, the members of this condensed set are superior to removed candidates according to their performance at the stated objectives. Constructing this reduced superior set allows for greater attention and scrutiny to be paid to the remaining candidates. This aids the decision-maker in selecting a more informed and well-vetted alternative, while ensuring that objectively superior alternatives are not overlooked. No preference information is required of decision-makers to establish the superior set, insulating it from the subjectivity that plagues a priori methods.

Comprised primarily of mathematical programming-based and heuristic strategies, the crux of many a posteriori methods is the pairwise and objective-wise comparison of candidate alternatives to determine those constituting the superior set (Marler & Arora, 2004). These comparisons are the mechanism enabling certification that members of the reduced set are both relatively well-performing and not inferior to any member of the original set of candidates. However, this mechanism, particularly when applied to many-alternative and many-objective problems, is computationally and temporally expensive (Roy, Islam, & Deb, 2016). This expense arises as more comparisons are required to reduce the original candidate set to the set of superior performers. This shortcoming proves problematic, particularly as systems engineers seek

to optimize the stakeholder-determined value of complex contemporary systems selected from large pools of candidate alternatives. The effect of this deficiency is that decisions made under computational or time constraints cannot be as thorough in their search of the solution-space. Instead, decision-makers must reduce the quantity of alternatives examined or the number of objectives considered when selecting a candidate solution. Simply, the computational expense of this comparative mechanism institutes a tradeoff between the scrupulousness of the solution search conducted and the resources required to construct the superior set.

The author herein proposes a technique based on solution-space geometry that can be used within a posteriori optimization methods to reduce their computational and temporal expense. Armed with this technique, systems engineers employing a posteriori methods can include considerably more solution alternatives and an increased number of objective functions in determining the set of superior alternatives. This permits a more exhaustive, incorporating, and tailored search of the candidate solution-space without violating computational or time constraints. Through a more complete search, decision-makers are granted the potential to identify and select solutions that more aptly and globally satisfy the interests of the system's diverse stakeholders.

To introduce the proposed technique, a few definitions describing many-objective considerations are first presented. Current methodologies used in constructing the superior set are then examined, in addition to the provision of commentary on their efficiency and contributing factors. The proposed technique is then presented, highlighting the logic precipitating its formulation. The proposed and current methods are then applied to a variety of many-alternative, many-objective solution sets, testing the

computational and temporal expense of each. The comparative performance of the proposed technique is then discussed, providing some explanation for its aptitude across the different testing scenarios. Finally, a brief discussion of the technique's limitations and future work suggestions are provided.

2. PRELIMINARIES

To facilitate discussion of current methods and the technique proposed by the author to hasten the discovery of the superior set, a few basic concepts are first established. Principal is the definition of a many-objective optimization problem, given in Equation (1).

$$\min\big(f_1(x), f_2(x), \dots, f_k(x)\big) \quad s.t.\, x \in X \tag{1}$$

where the integer $k > 3$ is the number of objectives and the set X is the feasible set of decision vectors.

Contrary to their single-objective counterparts, many-objective optimization problems incorporate numerous distinct, and often competitive, objective functions that can be optimized independently. Constituting the essence of many-objective decision-making, the optimization of one objective often occurs at the expense of others. This culminates in the nonexistence of a feasible solution that concurrently minimizes all objective functions (Srinivas & Deb, 1994). The mechanism of *Pareto dominance* is instead used to compare the attractiveness of candidate solutions, given the absence of a universally optimizing feasible solution (Steuer, 1989).

Defined in Equation (2), Pareto dominance is achieved when a solution-a performs just as well as another solution-b with respect to each objective of the optimization, while also performing better than solution-b with respect to at least one objective. In this scenario, solution-a is said to Pareto dominate solution-b and may be thought of as superior to the latter, as defined by the set of objective functions.

$$\begin{gathered} If\ f_i(x^a) \leq f_i(x^b) \forall\ i \in \{1, 2, \dots, k\} and\ f_j(x^a) < f_j(x^b) \\ for\ at\ least\ one\ index\ j \in \{1, 2, \dots, k\} \end{gathered} \tag{2}$$

then x^a is said to *Pareto dominate* x^b.

Solution-a's dominance of solution-b may be denoted as $x^a \prec x^b$. Alternatives between which a dominance relationship does not exist are called non-dominating solutions (Steuer, 1989).

Solutions which are not dominated by any other solution in the set are regarded as *Pareto optimal* or *Pareto efficient* solutions (Pareto, 1906). The set of all Pareto optimal solutions within the candidate set is said to constitute the *Pareto frontier* of the set. This set describes the collection of solutions that are superior to the dominated solutions while non-dominating to other members of the Pareto frontier (Horn, Nafpliotis, & Goldberg, 1994). Pareto optimality and the Pareto frontier are defined in Equations (3) and (4), respectively. The Pareto frontier is synonymous with the *superior set* described in the previous section.

$$\begin{gathered} x^a \in X\ is\ said\ to\ be\ Pareto\ optimal\ in\ X\ iff\ \nexists x^b \in X \\ such\ that\ x^b \prec x^a \end{gathered} \tag{3}$$

$$\begin{gathered} Those\ individuals\ that\ satisfy\ \{x^a \in X | \nexists x^b \in X, x^b \prec x^a\} \\ comprise\ the\ Pareto\ frontier \end{gathered} \tag{4}$$

These conventions work well to describe the responsibility of systems engineering decision-makers facing the conflicting interests of diverse stakeholders. Treating stakeholder preferences as independent objectives, the decision-maker may formulate solution selection as a many-objective optimization problem void of a globally optimizing alternative. It is then the task of the decision-maker to determine the set of objectively superior (Pareto efficient) solutions which will receive increased scrutiny before a final selection is made.

3. CURRENT METHODS

To discern the set of Pareto efficient solutions various methodologies have been developed. Each of these require the pairwise and objective-wise comparison of alternative solutions as discussed previously. As the number of candidate solutions increases, these comparisons make the computational and temporal expense of constructing the Pareto frontier much more burdensome. While an increased number of comparisons is inevitable as solution sets expand, the scheme used in determining the Pareto frontier can have substantial implications on the number of comparisons necessitated.

The first methodology, which will henceforth be called the *traditional method*, compares each member of the candidate solution set (S) to every other member. At each comparison, it is determined whether the focal solution (x^{ℓ}) is Pareto dominated by the current comparison solution (x^{w}). If this relationship exists, the index of the comparison solution (w) is added to the index set of counterpart solutions dominating the focal

solution (DS^{ℓ}). Upon the comparison of each counterpart solution to the focal, the focal solution is added to the set of Pareto efficient solutions (PF) if its index set of dominating solutions is empty.

After each candidate has been examined for domination as the focal solution, the resulting set of Pareto efficient solutions describes the Pareto frontier of the solution set. The algorithmic description of the traditional method is presented in Algorithm 1.

Algorithm 1: **Traditional Method**

Description: The following procedure determines the set of Pareto efficient solutions, PF, within any given set of solutions, S.

```
Let x^ℓ_i denote the i^th objective function value of the ℓ^th solution of S such that
i ≤ | S_ℓ | and ℓ ≤ | S |.
Let DS^ℓ denote the index set of solutions Pareto dominating solution ℓ.
Set DS^ℓ = ∅ ∀ ℓ
Set PF = ∅
Set ℓ = 1
While ℓ ≤ | S |
    Set w = 1
    While w ≤ | S |
        If w = ℓ, set w=w+1
        Else if x^w ≺ x^ℓ, set DS^ℓ = DS^ℓ ∪ w, w=w+1
        Else set w=w+1
    End
    Set ℓ = ℓ+1
    If DS^ℓ = ∅, set PF = PF ∪ x^ℓ
End
Return PF
```

Methodologies following the scheme of the traditional method have been used in several popular publications, perhaps most notably Srinivas and Deb's work on non-dominated sorting in genetic algorithms (1994). These algorithms, among others,

commonly utilize structures resembling the traditional method for various optimization processes including evolutionary survival and parent selection. Employed in many applications for the facilitation of non-dominated sorting, the use of the traditional method solely for the determination of the Pareto frontier may not make best use of expended computation as redundant comparisons frequently occur. Several adaptations have been developed to address this inefficiency, one of which will be introduced following a discussion of the factors contributing to the traditional method's improvidence.

While the traditional method produces the Pareto frontier, the number of comparisons required for this production can be reduced, enabling a more efficient algorithm. Following the traditional method, the relationship between a solution-a found to be dominated by a solution-b will later be reexamined during solution-b's tenure as the focal solution. Indeed, no additional information is found through this reexamination, a redundancy that will occur between every pairwise comparison within the set. Other inefficiencies are also present. For instance, consider a solution-a found to be dominated by a solution-b while subsequently found to dominate a solution-c. In this scenario, $x^b \prec x^a \prec x^c$, implying that $x^b \prec x^c$. While this implied relationship could have been inferred from the information gained during solution-a's tenure as the focal solution, it is not reconciled by the traditional method until solution-c is investigated as the focal.

The second methodology, which will henceforth be called the *dynamic method*, addresses both of the previous inefficiencies. This method begins by setting the first solution of the candidate set (x^1) as the focal solution (x^ℓ) and comparing it to the second ($x^{\ell+1}$) solution of the candidate set, denoted as x^w. If x^w is dominated by x^ℓ, x^w is removed

from the candidate set and x^{w+1} becomes the new x^w. If, instead, x^ℓ is dominated by x^w, x^ℓ is removed from the solution set. Upon the removal of x^ℓ, $x^{\ell+1}$ is set as the new x^ℓ and the new $x^{\ell+1}$ is set as x^w. In the event that no dominance relationship exists or x^ℓ and x^w are identical, no removal is made and the index w is set to $w+1$. Repeating this process until w exceeds the number of solutions in the candidate set ($|S|$), the index ℓ is then set to $\ell+1$ and w is reset to the new $\ell+1$. The indexing of ℓ is then repeated as dictated by w until ℓ exceeds one less than the number of solutions in the candidate set ($|S|-1$). When ℓ reaches this stopping criterion, the algorithm is terminated and the solutions remaining in the candidate set comprise the Pareto frontier of the original set. The algorithmic description of the dynamic method is presented in Algorithm 2.

Algorithm 2: **Dynamic Method**

Description: The following procedure determines the set of Pareto efficient solutions, *PF*, within any given set of solutions, *S*.

```
Let x^ℓ_i denote the i^th objective function value of the ℓ^th solution of S such that
i ≤ | S_ℓ | and ℓ ≤ | S |.
Set ℓ = 1
While ℓ ≤ | S |−1
    Set w = ℓ + 1
    While w ≤ | S |
        If x^ℓ_i = x^w_i for all i, set w=w+1
        Else if x^ℓ ≺ x^w, set S = S \ {x^w}
        Else if x^w ≺ x^ℓ, set S = S \ {x^ℓ}, w= ℓ+1
        Else set w=w+1
    End
    Set ℓ = ℓ+1
End
Set PF = S
Return PF
```

The dynamic method, as described, addresses the inefficiencies of the traditional method by removing dominated solutions from the consideration set upon the recognition of their domination. This does not affect the output of the algorithm as any solutions that would have been dominated by a removed solution are assuredly dominated by the solution instigating the latter's removal. Additionally, the dynamic method employs two-way dominance checks, determining if the focal solution is dominated by or dominates the pairwise solution at every comparison. Through these mechanisms the effects of the second discussed inefficiency are reduced, enabling a smaller number of comparisons to construct the Pareto frontier. Further, the index advancement conventions of the dynamic method ensure that all necessary comparisons needed to ensure the Pareto efficiency of nonremoved solutions are made, without performing the directly redundant comparisons described by the first discussed inefficiency.

Methodologies following a scheme resembling the logic of the dynamic method have been used in many well-cited algorithms including the Normalized Normal Constraint Method (Messac, Ismail-Yahaya, & Mattson, 2003) and Deductive Sort (McClymont & Keedwell, 2012). The use of this approach has shown to be effective in reducing the number of comparisons required to generate the Pareto frontier (McClymont & Keedwell, 2012). This allows for the consideration of more candidate solutions and a greater number of objective functions without violating computational or time constraints. Building upon its efficacy, an adjustment to the algorithm of the dynamic method is now presented, aimed at further reducing the number of comparisons required in constructing the Pareto frontier.

4. A GEOMETRICALLY INTELLIGENT METHODOLOGY

To understand the logic of the proposed methodology the following scenario is first presented:

Let a set of candidate solutions be composed of four alternatives: solution-a, solution-b, solution-c, and solution-d. Let non-dominating relationships exist between solution-a, solution-b, and solution-c, while each of these solutions are dominated by solution-d. If the solutions are placed in alphabetical order, the following operations are completed by following the dynamic method algorithm:

1. Solution-a compared with solution-b, no removal
2. Solution-a compared with solution-c, no removal
3. Solution-a compared with solution-d, solution-a removed
4. Solution-b compared with solution-c, no removal
5. Solution-b compared with solution-d, solution-b removed
6. Solution-c compared with solution-d, solution-c removed

This constitutes six pairwise comparisons to determine solution-d as the only Pareto efficient solution within the candidate set. However, if solution-d is placed as the first alternative in the candidate set, the same algorithm is able to produce the Pareto frontier in half the amount of comparisons:

1. Solution-d compared with solution-a, solution-a removed
2. Solution-d compared with solution-b, solution-b removed
3. Solution-d compared with solution-c, solution-c removed

The reduced number of required comparisons arises as the solution dominating the highest number of counterparts (the *most dominant solution*) is placed first in the set. This allows it, serving as the original focal solution, to remove dominated counterparts early, relieving the proceedings of the non-dominating relationships examined in the first sequence. This recognition sponsors the desire to place the most dominant solutions of the candidate set first, increasing the likelihood that inferior candidates are removed early, along with the unnecessary comparisons they may solicit.

Seeking to place the most dominant solutions of the candidate set at the set's beginning, a surrogate dominance metric must be established. This surrogate is developed as determining the exact number of counterparts that each candidate solution dominates requires the use of an algorithm with complexity similar to the traditional method. The application of an algorithm of this scale would then make any efficiencies granted by knowledge of the true domination count irrelevant, as the Pareto frontier could have been established for the same computational expense. In establishing this proxy metric, a consideration of the location on the solution-space that would be the most dominant is made. To approach this location the *ideal point* of the candidate set is identified. The ideal point is comprised by the optimum exhibited value of any solution in the candidate set for each objective function. Defined by Equation (5) for a candidate solution set with X members and k objectives, the ideal point exhibits dominance over every alternative within the candidate set.

$$x_i^{ideal} = \inf_{x \in X} f_i(x) \; \forall \; i = 1, \ldots, k \tag{5}$$

The ideal point, however, is likely not a real member of the candidate solution set, as optimizing all objectives simultaneously is difficult in practice. Consider, for example,

the simplistic multiobjective problem of maximizing performance while minimizing cost. Intuitively, it is very unlikely that the ideal point of maximum performance at minimal cost is exhibited by any real alternative.

The illusory nature of the ideal point does not, however, mean that it is not useful. Instead, it can be used as a measuring stick to anticipate the dominance that any real solution will exhibit. This expectation is achieved by determining the scaled Euclidean distance between the real candidate solution and the ideal point, as shown for candidate solution-a with *k* objectives in Equation (6).

$$D^a = \sqrt{\sum_{i=1}^{k} \left(\frac{x_i^a - x_i^{ideal}}{x_i^{ideal}} \right)^2} \tag{6}$$

Determining the distance to ideal point (*D*) value for all members of the candidate solution set, those with the smallest *D*-values can be regarded as most geometrically similar to the ideal point, x^{ideal}. Sorting the candidate solution set by ascending *D*-value, the algorithm shown in Algorithm 3 seeks to use this dominance surrogate to place the most dominating alternatives first. This proposed methodology, henceforth called the *geometrically intelligent dynamic method* (*GIDM*), attempts to take advantage of dominance ordering and reduce the number of comparisons required in producing the Pareto frontier.

Identical to the dynamic method following the *D*-value sorting initiated in *Step 0*, the GIDM attempts to remove poor performing alternatives early in the procedure by subjecting them to frontloaded comparisons with highly dominating focal solutions. With the less dominating solution candidates removed, the redundant comparisons they incite

are also removed from the procedure, enabling a more efficient construction of the Pareto frontier.

Algorithm 3: **GIDM**

Description: The following procedure determines the set of Pareto efficient solutions, *PF*, within any given set of solutions, *S*.

```
Let x^ℓ_i denote the i^th objective function value of the ℓ^th solution of S such that i ≤ | S_ℓ | and ℓ ≤ | S |.
Let D^ℓ denote the scaled Euclidean distance of the ℓ^th solution of S from the Ideal Point x*.
Set S as the set S sorted by the ascending D-value of each solution.
Set ℓ = 1
While ℓ ≤ | S |−1
    Set w = ℓ + 1
    While w ≤ | S |
        If x^ℓ_i = x^w_i for all i, set w=w+1
        Else if x^ℓ ≺ x^w, set S = S \ {x^w}
        Else if x^w ≺ x^ℓ, set S = S \ {x^ℓ}, w= ℓ+1
        Else set w=w+1
    End
    Set ℓ = ℓ+1
End
Set PF = S
Return PF
```

5. NUMERICAL ANALYSIS

To test the efficacy of the GIDM at reducing the resources required in establishing the Pareto efficient set, a series of random many-objective solution sets was created. Comprising 25 test-classes, solutions sets were developed incorporating a variable number of objectives (5, 7, 10, 12, and 15) and a variable number of candidate

solutions (1,000, 5,000, 10,000, 15,000, and 25,000). Objective function values for each solution were randomly selected from the uniform distribution between zero and one. The three defined methodologies were then presented identical solution sets within each class. The number of solution comparisons and the algorithmic runtime required by each method to determine the Pareto frontier was then recorded. This procedure was repeated 50 times within each class, totaling 1,250 distinct scenarios presented to each method.

Table 1 describes the average number of alternative-to-alternative comparisons required by each method for each of the 25 test-classes.

Table 1. Average Required Comparisons by Method and Test-Class.

Comparisons Required - Traditional Method					
Alternatives	Objectives				
	5	7	10	12	15
1,000	999,000	999,000	999,000	999,000	999,000
5,000	24,995,000	24,995,000	24,995,000	24,995,000	24,995,000
10,000	99,990,000	99,990,000	99,990,000	99,990,000	99,990,000
15,000	224,985,000	224,985,000	224,985,000	224,985,000	224,985,000
25,000	624,975,000	624,975,000	624,975,000	624,975,000	624,975,000
Comparisons Required - Dynamic Method					
Alternatives	Objectives				
	5	7	10	12	15
1,000	38,590.28	144,541.82	357,664.28	442,560.24	488,874.48
5,000	233,321.22	1,407,972.50	5,903,314.96	9,089,724.70	11,592,595.84
10,000	455,354.58	3,404,493.48	18,527,981.38	32,012,691.14	44,614,498.56
15,000	652,536.94	5,836,478.12	36,175,657.88	65,583,607.00	97,336,392.38
25,000	1,049,823.84	10,339,660.94	79,241,345.59	158,592,549.69	257,951,730.67
Comparisons Required - GIDM					
Alternatives	Objectives				
	5	7	10	12	15
1,000	19,347.60	95,772.16	305,812.82	414,567.86	482,103.46
5,000	95,539.16	769,490.30	4,451,584.96	7,890,052.52	11,145,182.40
10,000	186,794.50	1,777,218.44	13,406,838.66	26,714,548.06	42,225,936.20
15,000	267,671.00	3,010,697.90	25,136,390.40	53,564,087.90	90,848,672.90
25,000	417,666.70	4,882,014.28	53,112,571.97	123,950,115.49	238,085,324.53

While a consideration of the number of comparisons required by each method is a good rudimentary measure of computational efficiency, temporal considerations are perhaps more valuable. This value advantage arises as the latter is a more tangible

measure and also incorporates the resource expense of the presorting mechanism utilized by the GIDM. While the improved runtime of structures using the dynamic method over the traditional method has been well demonstrated in the literature (McClymont & Keedwell, 2012; Wang & Yao, 2014; Roy, Islam, & Deb, 2016), a comparison of the dynamic method to the GIDM is now made. Table 2 describes the average algorithmic runtime required by these methods for each of the 25 test-classes. The tests were conducted on a 3.00 GHz Intel Core i9-9980XE processor with 64 GB of RAM, running Windows 10.

From the data in the Table 2, Table 3 displays the average reduction in runtime required by the GIDM from the dynamic method. This data is displayed both as the number of seconds reduced and the percent runtime reduction achieved. Reading the table, the GIDM required 0.073 fewer seconds than the dynamic method for the 5-objective, 1,000-alternative test-class, recognized as a 46.642% runtime reduction. Table 4 details the average percent of candidate solutions within each test-class that exist as non-dominated points. These solutions, which comprise the Pareto frontier, constitute a greater proportion of the original candidate set with an increased number of objectives and a diminished number of alternatives. The former phenomenon occurs as domination is harder to achieve with an increased number of objective criteria (Deb & Jain, 2014). The latter arises as an increased number of randomly generated solutions increases the prevalence of highly-dominating and highly-dominated solutions.

Table 2. Average Required Algorithmic Runtime by Method and Test-Class.

Runtime Required - Dynamic Method (Seconds)					
Alternatives	Objectives				
	5	7	10	12	15
1,000	0.157	0.580	1.420	1.755	1.971
5,000	0.985	5.665	23.666	36.508	46.125
10,000	2.030	13.714	73.646	127.699	178.259
15,000	3.074	23.973	144.912	264.594	391.121
25,000	5.608	45.251	315.921	638.298	1,030.798

Runtime Required - GIDM (Seconds)					
Alternatives	Objectives				
	5	7	10	12	15
1,000	0.084	0.385	1.214	1.630	1.935
5,000	0.454	3.111	17.726	31.550	44.207
10,000	1.003	7.281	53.384	106.010	168.809
15,000	1.650	12.676	101.865	215.996	363.978
25,000	4.338	24.637	214.845	493.108	950.274

Table 3. Average GIDM Algorithmic Runtime Reduction from Dynamic Method.

GIDM Runtime Reductions - From Dynamic Method (Seconds)					
Alternatives	Objectives				
	5	7	10	12	15
1,000	0.073	0.196	0.206	0.124	0.036
5,000	0.531	2.554	5.941	4.958	1.917
10,000	1.027	6.433	20.263	21.689	9.450
15,000	1.423	11.298	43.047	48.597	27.143
25,000	1.270	20.614	101.076	145.190	80.524

GIDM Runtime Percent Reductions - From Dynamic Method					
Alternatives	Objectives				
	5	7	10	12	15
1,000	46.642%	33.705%	14.481%	7.089%	1.832%
5,000	53.879%	45.083%	25.101%	13.582%	4.157%
10,000	50.606%	46.910%	27.513%	16.985%	5.301%
15,000	46.305%	47.126%	29.706%	18.367%	6.940%
25,000	22.649%	45.555%	31.994%	22.746%	7.812%

Table 4. Average Percent of Candidate Solutions Non-dominated by Test-Class.

Percent of Candidate Solutions Non-dominated					
Alternatives	Objectives				
	5	7	10	12	15
1,000	14.990%	41.420%	78.510%	90.560%	97.960%
5,000	6.558%	21.748%	57.118%	76.900%	93.880%
10,000	4.286%	16.679%	49.322%	71.199%	90.502%
15,000	3.253%	13.599%	44.719%	67.371%	89.186%
25,000	2.420%	10.654%	38.169%	61.601%	85.972%

6. DISCUSSION

Examining Table 1, it is shown that the GIDM is able to reduce the average number of comparisons required to establish the Pareto frontier in relation to the traditional and dynamic methods for all test-classes. The superiority of the dynamic method over the traditional method is similarly confirmed. Shifting focus to Table 2 and Table 3, the GIDM is additionally shown to reduce the algorithmic runtime required to produce the Pareto frontier in comparison to the dynamic method. It is also observed that for each objective function value tested, the GIDM generally enjoys increased runtime superiority over its counterpart methods with an increased number of alternatives. For example, examining 1,000 alternatives and 10 objectives, the GIDM exhibits an average runtime reduction of 0.206 seconds from the dynamic method; however, when 25,000 alternatives are examined at the same objective function level, the reduction improves to over 101 seconds. Similar advantage trends can be discerned for the required number of comparisons by an analysis of Table 1. The only figure not in compliance with this tendency is the runtime reduction value for 5 objectives and 25,000 alternatives within Table 3. This value is smaller than its 15,000-alternative counterpart within the objective function level, perhaps indicating a point where the resource requirements of the presorting mechanism outweigh the benefits they enable.

In a further review of Table 3, it appears that the runtime percent reductions are diminished as the number of objective functions increases. While the percent reductions are reduced, the GIDM still maintains some savings even in the most troublesome case of 15 objectives and 1,000 alternatives. Additionally, the apparently contracted efficiency of

the GIDM with an increased number of objective functions is a product of the nature of the candidate solution set. As discussed previously and confirmed by Table 4, a greater number of objective functions generally corresponds to a greater proportion of the initial candidate set exhibiting Pareto efficiency. With an expanded Pareto frontier, less eliminations are made throughout the process, waning the efficiency that any Pareto-set-identifying method could achieve. While an examination of the percent reductions for these high-objective test-classes may appear underwhelming, attention is reverted to direct runtime reduction values of Table 3, describing the concrete and not-insignificant savings achieved by the GIDM.

With these allowances, systems engineering decision-makers can consider more alternatives and incorporate a greater number of objective functions without violating the constraints imposed by computational or time limitations. Utilizing this freedom, decision-makers can claim and conduct a more complete and tailored search of the solution-space; strengthening their assertations of diligence and consideration when presenting recommendations to the diverse stakeholders from which objective functions stem. Simply, the GIDM equips systems engineering practitioners with a tool to make better decisions when computational and temporal availability are at a premium. This method helps assuage the need to ignore, or develop combinatory surrogates for, stakeholder preferences or reduce the number of alternatives that receive consideration. Consider a decision-maker using the dynamic method whose constraints only allow for the consideration of 10,000 alternatives when 7 objective functions are used; had this practitioner used the GIDM, the number of alternatives that could have been examined exceeds 15,000 (see Table 2). As consideration limitations are removed, decision-makers

can more confidently select alternatives that are scrutinized by a greater number of stakeholder preferences and selected from an even larger pool of candidate alternatives. These liberties enable a more justifiable and robust recommendation to these system interest parties.

7. CONCLUSIONS AND FUTURE WORK

Decision-makers of the systems engineering discipline are increasingly asked to make solution selections from large sets of alternatives while serving the interests of diverse stakeholders. Constituting a many-alternative, many-objective optimization problem, these scenarios confront practitioners with a set of stakeholder preferences to which no single solution alternative is globally preferred. Instead, the decision-maker is faced with a set of solutions that are variably attractive to the different interest groups. A posteriori methods help alleviate this burden by reducing the complete set of candidate alternatives to those objectively best performing, the Pareto frontier. The determination of this reduced set allows greater scrutiny to be granted to each member in making a final decision. As the number of alternatives and objective functions used in making a decision increases, however, the computational and temporal expense required to establish the Pareto frontier increases dramatically. This increased resource demand limits the number of alternatives that can be examined and the number of objective functions that can be utilized in producing the Pareto frontier under computational and time constraints. Decision-makers are thus forced to consider fewer alternatives or evaluate using fewer

objective functions, culminating in a less complete or less tailored search of the solution-space.

Seeking to remediate some of the inefficiencies of current a posteriori methods, a geometrically intelligent dynamic method of establishing the Pareto frontier was proposed. A comparative study was then conducted to examine the method's efficacy at reducing the computational and temporal load of establishing sets of Pareto efficient solutions. In repeated trials, across a range of candidate solution set sizes and employed objective functions, the proposed method was shown to reduce the resources required to produce the Pareto frontier in comparison to current methods. This improved efficiency is useful to the systems engineering decision-maker as it allows more alternatives and a greater number of stakeholder-preference-driven objectives to be considered within the same computational and time restrictions. These allowances permit a more thorough and considerate search of the solution-space, enabling the contemplation and selection of more globally preferred and competitively examined solutions. Using this methodology, decision-makers are more well equipped to handle the many-alternative, diverse-interest-serving nature of contemporary solution selection scenarios.

The author notes that the numerical analysis performed used random values from the uniform distribution to create solution objective function values. The ability of this method should further be tested with objective values stemming from more complex and challenging distributions or creation functions. A number of many-objective test suites exist, serving as tremendous starting points for investigations of the kind. Several other a posteriori methodologies also exist that were not examined in this work. It is the immediate focus of the author to compare the GIDM's computational expectancy against

these nonincluded methodologies, incorporating solution sets generated by the mentioned many-objective test suites.

REFERENCES

Crawley, E., Cameron, B., & Selva, D. (2016). Reasoning about Architectural Tradespaces. In *System architecture: Strategy and product development for complex systems* (pp. 346-372). Boston: Pearson.

Deb, K., & Jain, H. (2014). 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*(4), 577-601.

Horn, J., Nafpliotis, N., & Goldberg, D. (1994). A niched Pareto genetic algorithm for multiobjective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence,* 82-87.

Hwang, C., & Masud, A. S. (1979). Methods for Multiple Objective Decision Making. In *Multiple Objective Decision Making - Methods and Applications, Lecture Notes in Economics and Mathematical Systems*, 21-97. Berlin Heidelberg: Springer-Verlag.

IEEE Computational Intelligence Society. (2018). *IEEE CIS Task Force on Many-Objective Optimisation*. Retrieved March 23, 2019, from http://www.cs.bham.ac.uk/~limx/MaOP.html

Leybourne, S. A., Kanabar, V., & Warburton, R. D. H. (2010). Understanding and overcoming communications complexity in projects. Paper presented at PMI® Global Congress 2010—North America, Washington, DC. Newtown Square, PA: Project Management Institute.

Marler, R. T., & Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*(6), 369-395.

McClymont, K., & Keedwell, E. (2012) Deductive sort and climbing sort: New methods for non-dominated sorting. *Evolutionary Computation*, *20*(1), 1-26.

Messac, A., Ismail-Yahaya, A., & Mattson, C.A. (2003). "The normalized normal constraint method for generating the Pareto frontier". *Structural and Multidisciplinary Optimization*. *25*(2), 86–98.

Pareto, V. (1906). *Manuale di economia politica*. Milano: Societa Editrice.

Ramos, A. L., Ferreira, J. V., & Barcelo, J. (2012). Model-Based Systems/ Engineering: An Emerging Approach for Modern Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(1), 101-111.

Roy, P. C., Islam, M. M., & Deb, K. (2016). Best Order Sort: A New Algorithm to Non-dominated Sorting for Evolutionary Multi-objective Optimization. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion - GECCO 16 Companion*, 1113-1120.

Srinivas, N., & Deb, K. (1994). Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, *2*(3): 221-248.

Steuer, R. E. (1989). Multiple Criteria Optimization: Theory, Computation, and Application. Malabar: Robert E. Krieger Publishing.

Wang, H., & Yao, X. (2014). Corner sort for pareto-based many-objective optimization. *IEEE Transactions on Cybernetics*, *44*(1), 92-102.

II. IDEAL SORT: A TERMINABLE, EFFICIENT NONDOMINATED SORTING ALGORITHM

Samuel Vanfossan and Benjamin Kwasa

Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, MO 65409

ABSTRACT

Nondominated sorting is a key procedure in the operations of many multiobjective evolutionary algorithms. Comprising most of the time required by these biologically inspired procedures, considerable attention has been dedicated to improving the efficiency of this critical process. Here presented is a novel, terminable nondominated sorting algorithm, Ideal Sort, that utilizes a pre-comparison solution ordering mechanism based on squared Euclidean distance to the *ideal point* of the population. The algorithm is further enhanced by the incorporation of a termination procedure, potentially reducing the number of fronts the algorithm has to determine. Both variations of this method (vanilla and terminable) exhibit a worst-case time complexity of $O(MN^2)$ and demonstrate strong experimental performance. Across a wide range of multiobjective datasets, Ideal Sort is shown to outperform other terminable nondominated sorting algorithms and achieves state-of-the-art performance in some instances. The investigation additionally highlights the importance and benefit of terminability within nondominated sorting procedures, a property demonstrated to enable considerable efficiency improvements. These findings make the case for the restructuring of efficient nondominated sorting procedures not equipped for terminability and the subsequent integration of this valuable property.

Keywords: Evolutionary algorithm, genetic algorithm, ideal sort, many-objective optimization, multiobjective optimization, nondominated sorting, terminable nondominated sorting, Pareto ranking

1. INTRODUCTION

An appropriate applicant to many real-world decision scenarios, multiobjective optimization differs from single-objective optimization in a number of ways. A principle difference lies in nature of the *resolution* to problems of these distinct classes. Single-objective optimization is concerned with the identification of a globally optimum solution exhibiting the best possible performance as determined by a single objective function. Multiobjective optimization, however, is akin to problem formulations possessing a number of conflicting objective functions that disallow the existence of a single, globally optimal solution [1]. Instead, the solution-space is characterized by a subset of solutions superior to all other alternatives of the space, but from which no member can be considered globally preferable. Called the set of Pareto-optimal solutions (or Pareto efficient set, or nondominated set, or Pareto frontier), these alternatives constitute those solutions that perform objectively-best according to the objective functions while being unable to claim universal superiority over any other member of the Pareto efficient set [2]. The focus of many multiobjective optimization techniques is the provision of this Pareto efficient set, from which the receiving decision-maker can make a final selection.

Operating on populations of solutions, evolutionary algorithms (EAs) have emerged as a premier and natural approach avenue to handle multiobjective optimization problems. These heuristic methods seek to mimic natural selection in evolving a set of solutions toward the Pareto frontier of a problem [3]. Methodologies utilizing nondominated sorting (NDS) as a mechanism for selection have emerged as some of the most well-performing and widely applied of the published multiobjective EAs (NPGA [4]; MOGA [5]; PAES [6]; NSGA-II [7]; SPEA2 [8]; PESA-II [9]; NSGA-III [10]). While demonstrating aptitude, these algorithms are temporally and computationally constrained by the resource complexity of the NDS algorithm used [11]. This dependency culminates in the resource expense of NDS-employing EAs growing rapidly as the number of objective functions and candidate alternatives examined increases.

An elevated resource complexity thus limits the number of objective functions and candidate alternatives that can be utilized under the presence of temporal and computational constraints. This limitation impacts not only the scale of optimization problems that these methods can feasibly address, but also their applicability within limited-resource, time-sensitive environments. Therefore, substantial effort has been expended to develop more efficient NDS algorithms, aimed at improving the time and computational requirements of employing multiobjective EAs [12]. These works have focused on reducing the number of solution-to-solution comparisons and the algorithmic runtime required to complete the NDS procedure [11]. While much progress has been made, any efforts capable of further improving the resource complexity of NDS procedures should receive considerable attention and application.

The contributions of this paper are summarized as follows:

1) A novel, terminable NDS algorithm (Ideal Sort) is presented, along with the theoretical inspirations for its development.
2) The importance and benefit of terminability within NDS algorithms is discussed and demonstrated.
3) The presented algorithm's performance across a wide range of multiobjective test sets is compared to that of several state-of-the-art methods. Ideal Sort is shown to generally outperform other terminable methods and achieves state-of-the-art performance in some instances.
4) A new, scalable multiobjective test set creation procedure for evolved datasets is presented (Algorithm 2) and utilized. This procedure looks to mimic a common scenario to which NDS algorithms are applied within multiobjective EAs.

In the next section, some fundamental multiobjective optimization definitions and concepts are quickly introduced. Subsequently, a review of published algorithms focused on improving the efficiency of NDS is presented. A new methodology is then proposed, followed by a brief examination of its theoretical complexity. The method's experimental performance is then compared to that of several state-of-the-art algorithms, followed by a discussion of the results achieved. In a final section, concluding remarks and recommendations for future work are extended.

2. PRELIMINARIES

A (minimization) multiobjective optimization problem with *M* objectives is defined by Equation (1), where $f_j(x^a)$ is solution *a*'s performance with respect to objective function *j* and *X* is the set of feasible solutions.

$$min(f_1(x), f_2(x), \dots, f_M(x)) \quad s.t. x \in X \tag{1}$$

As these objectives may be conflicting and eliminatory of a globally optimal solution, "Pareto dominance" is used to compare solutions. Solution *a* is Pareto dominated by another solution *b* in *M* objectives if the conditions of Equation (2) are met. Solution *b*'s dominance of *a* is denoted by $x^b \prec x^a$.

$$f_i(x^b) \leq f_i(x^a)\ \forall\ i \in 1,2,\dots,M\ and f_j(x^b) < f_j(x^a) \quad for\ at\ least\ one\ j \in \{1,2,\dots,M\} \tag{2}$$

Solutions that are not dominated by any solution in the set are said to be Pareto efficient or nondominated. Collectively, all nondominated members of a set comprise the set's first Pareto front. Solutions dominated only by one or more member of the first Pareto front constitute the second Pareto front, and so on. NDS slates solutions of the set into their associated Pareto fronts in this manner.

3. RELATED WORK

3.1. COMPLETE COMPARISON METHODS

Among the first prevalent multiobjective EAs to utilize NDS was Srinivas and Deb's *Non-dominated Sorting Genetic Algorithm* (NSGA) [13]. Receiving much

attention, this algorithm emerged as one of the seminal works on the topic of multiobjective EAs. Capable of maintaining a well-distributed set of points while converging to the Pareto frontier of the solution-space, NSGA employs an NDS mechanism (which has come to be called the *naïve* or *brute force method*) that compares each solution in the population to every other solution. At each comparison, the dominance relationship between the pair is determined, recording an incrementing tally to the manifest of a dominated solution. After the final pairwise evaluation is completed, those solutions not dominated by any of their counterparts are recognized as the Pareto efficient solutions of the population. These solutions are then temporarily removed from the population, the dominance tallies are reset, and the process is repeated to find the next Pareto front. While able to successfully sort the population into their nondominated fronts, the algorithm is resource intensive, maintaining a worst-case computational complexity of $O(MN^3)$ -where M is the number of objective functions and N is the size of the population.

Emerging as one of the key criticisms of NSGA, its computational complexity was addressed by one of the algorithm's authors in a successor approach, *NSGA-II* [7]. The most well-cited multiobjective optimization procedure to date, NSGA-II utilizes an approach called Fast Non-dominated Sorting (FNDS) that reduces the worst-case computational complexity of the sorting procedure to $O(MN^2)$. This efficiency is enabled by an expanded bookkeeping mechanism that stores the dominance relationships between each pair of solutions in the population. Comparing each solution to every other, those shown to be not dominated by a counterpart are recognized as members of the first Pareto front. Editing the expanded domination records, the domination effects of these solutions

are then disregarded. Solutions not dominated by any member in the updated ledger are then set as the second Pareto front. This process is repeated until every solution is assigned to a front. Owing the dominance relationship traceability of this mechanism, each solution only needs to be compared to each of its counterparts once. This eliminates several of the directly redundant comparisons required by the naïve method as subsequent fronts are found. This computational savings does come at the expense of storage, increasing the space complexity to $O(N^2)$ from the $O(N)$ required by its predecessor. The improved computational complexity of NSGA-II allows it to handle large and complex optimization problems much more feasibly than NSGA and other naïve-method-driven algorithms. As these two methods compare each solution to every other in the population in determining the set of Pareto fronts, they belong to a class of NDS algorithms called *complete comparison methods*.

3.2. INFERRED DOMINANCE METHODS

Recognizing the increased applicability of NDS EA's when granted improved efficiency, several subsequent methodologies were developed to further improve the computational complexity of NDS procedures. An approach based on Arena's Principle was proposed, capable of reducing computational complexity to $O(MN\sqrt{N})$ [14]. This method uses dominance relationships to dictate a solution's tenure as an arena host. During its tenancy, the Arena host is compared to all other solutions to determine if it is dominated by any of its counterparts. If it is not, it is added to the Pareto efficient set and the next arena host is selected. If the host is dominated by a counterpart, it is removed from consideration and the dominating solution becomes the new host. Solutions

dominated by the arena host are additionally removed from consideration. This procedure is continued until only the set of Pareto efficient solutions remain. Nondominated solutions are then ignored and the process is repeated to find the next Pareto front. Similar to FNDS, Arena's Principle exhibits a worst-case computational complexity of $O(MN^2)$ while boasting a lesser space complexity of $O(N)$.

McClymont and Keedwell [15] proposed two algorithms, Climbing Sort and Deductive Sort, aimed at reducing the computational burden of NDS while maintaining a simple procedure to allow easy integration within existing EAs. The proposed Climbing Sort algorithm begins by comparing solutions of the population until a dominance relationship is established. When dominance occurs, the dominated solution is marked and discarded. If the focal solution of the comparison is the dominated solution, its dominating counterpart becomes the focal solution and the algorithm continues. This process is repeated until a nondominated focal solution is found. The algorithm then moves to the next focal solution and continues until only the set of nondominated solutions remain. The logic of this algorithm is that once a solution has been dominated, it cannot be a member of the efficient set, so any further comparisons to it are redundant. Additionally, any solutions that would have been dominated by the removed solution will be dominated by the solution prompting the latter's removal. Called *inferred dominance*, this compound idea is the crux of the Climbing Sort and Deductive Sort algorithms. Inferred dominance lends to the improved efficiency of Arena's Principle, as well, though the authors did not explicitly mention the term in their proceedings. Deductive Sort assesses each solution based on a fixed population order. By this procedure, a focal solution is compared to solutions that occur after it within a set list of solutions. Solutions

dominated by the focal solution are flagged and ignored. If a solution dominates the focal solution, the focal is similarly flagged and the focus is shifted to the counterpart immediately below the focal solution in the ordered list. This process is completed until all solutions have been examined, defining the nondominated set as those solutions not marked with a flag. Climbing Sort and Deductive Sort, like Arena's Principle must be repeated to find subsequent Pareto fronts. Deductive Sort is demonstrated by the authors to generally outperform Climbing Sort and Arena's Principle, while all maintain a best- / worst-case complexity of $O(MN\sqrt{N})$ / $O(MN^2)$ and a space complexity of $O(N)$. Corner Sort provides some improvement to Deductive Sort by ensuring that the next focal solution when a transition is made is in the current Pareto front [16]. This is achieved by selecting the remaining solution touting the best value at any one objective. While Corner Sort has been shown to outperform Deductive Sort in many scenarios, the algorithms' computational and space complexity are identical.

These four algorithms, Arena's Principle, Climbing Sort, Deductive Sort, and Corner Sort, comprise prominent members of the set of NDS procedures called the *inferred dominance methods*. Heralding Deductive Sort and Corner Sort as their most efficient members, these algorithms successively determine Pareto fronts and can be terminated after a desired number of fronts are found.

3.3. CONSTRUCTIVE FRONT METHODS

Zhang et al. [17] took a different approach to NDS in their proposition of the *Efficient Non-dominated Sort* (ENS) algorithms. Instead of comparing each solution to all other unflagged solutions in the set, the ENS algorithms compare focal solutions only to

the solutions already assigned to a Pareto front. If a focal solution is dominated by any member of a front it is moved to the next front for similar testing. If the focal solution is found to be nondominating with all members of the front, then it is added to the front and the next solution is examined. This process is repeated until all members of a population are assigned to a front. Presorting the population in ascending order by the first objective function value (and lexicographically in the case of ties), there will never be a focal solution that is dominant of any previous solutions already added to the set of Pareto fronts. This enables the algorithms to disregard the possibility of having to move solutions to different fronts once they are placed during their focal tenure. The algorithmic variance between the two proposed ENS methodologies lies only in the first front to which the focal solution is compared. The *Sequential Search* algorithm (ENS-SS) begins by comparing the focal solution to the first front and progresses by the described mechanism accordingly. The *Binary Search* algorithm (ENS-BS) begins the comparisons at the median established front, seeking to bypass the comparisons stemming for common demotions arising as the algorithm progresses to the latter parts of the sorted population. While ENS-BS typically outperforms ENS-SS, the latter may exhibit superiority in some instances, particularly when a small number of fronts are present. Maintaining a worst-case complexity of $O(MN^2)$, ENS-SS exhibits a best-case computational complexity of $O(MN\sqrt{N})$ while ENS-BS achieves $O(MN\log N)$ complexity. These algorithms are advantaged by a space complexity of $O(1)$ but require that all solutions be examined before even the first Pareto front can be considered complete.

Furthering the pre-comparison sorting procedures, Roy, Islam, and Deb [11] proposed an NDS algorithm entitled *Best Order Sort* (BOS). This method first sorts the

population according to their performance on each independent objective, assigning the set of partial ranks that define each solution. The algorithm then iterates through the sorted objective columns in a row-wise fashion, placing solutions into respective fronts as they are discovered. The sorting procedure (like with the ENS algorithms) ensures that the front to which a solution is added will not need to be altered after the initial assignment. Further, when a solution is discovered it only needs to be compared to solutions which have a higher partial rank with respect to the currently examined objective. This reduces many of the unnecessary comparisons executed by the ENS methods. BOS is able to achieve a best-case computational complexity of $O(MN\log N)$ and requires $O(MN)$ storage. The method described by the authors, however, is not able to handle duplicate solutions within the population. This shortcoming is addressed by Mishra et al. [18] in their proposition of the *Generalized Best Order Sort* (GBOS) algorithm. Their alterations enable the handling of duplicate solutions while retaining the computational and space complexities of BOS. Two variants of GBOS are posed, sequential search (GBOS-SS) and binary search (GBOS-BS), mirroring the strategy and performance characteristics applied to ENS. Roy, Deb, and Islam [12] expanded upon the BOS algorithm in their creation of *Bounded Best Order Sort* (BBOS). BBOS uses adaptive binary trees to cut down on the number of fronts a solution needs to pass through before finding the front to which it belongs. This algorithm is shown to demonstrate great performance when the number of fronts is very large and retains the space and time complexities of BOS.

These six algorithms, ENS-SS, ENS-BS, BOS, GBOS-SS, GBOS-BS, and BBOS comprise members of the set of NDS procedures called the *constructive front methods*.

These algorithms each look to add solutions to their respective fronts individually and must examine all solutions before the algorithm can be terminated. The BOS and BBOS algorithms boast the best experimental efficiency of the constructive front methods and are considered state-of-the-art NDS procedures.

A table summarizing the computational and space complexities of the discussed methods is shown in Table 1.

Table 1. Time and Space Complexity of Existing Nondominated Sorting Methods.

Method	**Time Complexity**		**Space Complexity**	**Method Class**
	Base-Case	**Worst-Case**		
Naïve / Brute Force	$O(MN^2)$	$O(MN^3)$	$O(N)$	Complete Comparison
Fast Non-dominated Sort	$O(MN^2)$	$O(MN^2)$	$O(N^2)$	Complete Comparison
Arena's Principle	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$	Inferred Dominance
Climbing Sort	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$	Inferred Dominance
Deductive Sort	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$	Inferred Dominance
Corner Sort	$O(MN\sqrt{N})$	$O(MN^2)$	$O(N)$	Inferred Dominance
ENS-SS	$O(MN\sqrt{N})$	$O(MN^2)$	$O(1)$	Constructive Front
ENS-BS	$O(MN\log N)$	$O(MN^2)$	$O(1)$	Constructive Front
BOS / GBOS	$O(MN\log N)$	$O(MN^2)$	$O(MN)$	Constructive Front
BBOS	$O(MN\log N)$	$O(MN^2)$	$O(MN)$	Constructive Front

4. THE PROPOSED ALGORITHM*:* IDEAL SORT

4.1. ALGORITHMIC INSPIRATION

To understand the logic of the algorithm to be proposed, the following example is first presented:

Consider a population of four solutions: solution-a, solution-b, solution-c, and solution-d (Figure 1). While the first three solutions are nondominating to each other, each is dominated by solution-d. Maintaining their alphabetical ordering and employing a terminable inferred dominance method (in this case, Deductive Sort, see [15]), the

following domination comparisons may be conducted to determine the Pareto front of the set:

1) solution-a to solution-b; no removal
2) solution-a to solution-c; no removal
3) solution-a to solution-d; solution-a removed
4) solution-b to solution-c; no removal
5) solution-b to solution-d; solution-b removed
6) solution-c to solution-d; solution-c removed

Here, six two-way dominance comparisons will be made to determine solution-d as the only member of the first Pareto front. Had solution-d been placed at the head of the ordered solutions list, the Pareto frontier could have been produced in half the number of comparisons:

1) solution-d to solution-a; solution-a removed
2) solution-d to solution-b; solution-b removed
3) solution-d to solution-c; solution-c removed

This reduced number of comparisons is enabled by the placement of the *most dominant* solution at the head of the ordered list. This placement allows solution-d to remove its inferior counterparts early, avoiding the unnecessary comparisons they sponsor. This capability yields a general desire to place the most dominant solutions of a candidate set first, seeking to sidestep unnecessary comparisons, as illustrated by the example.

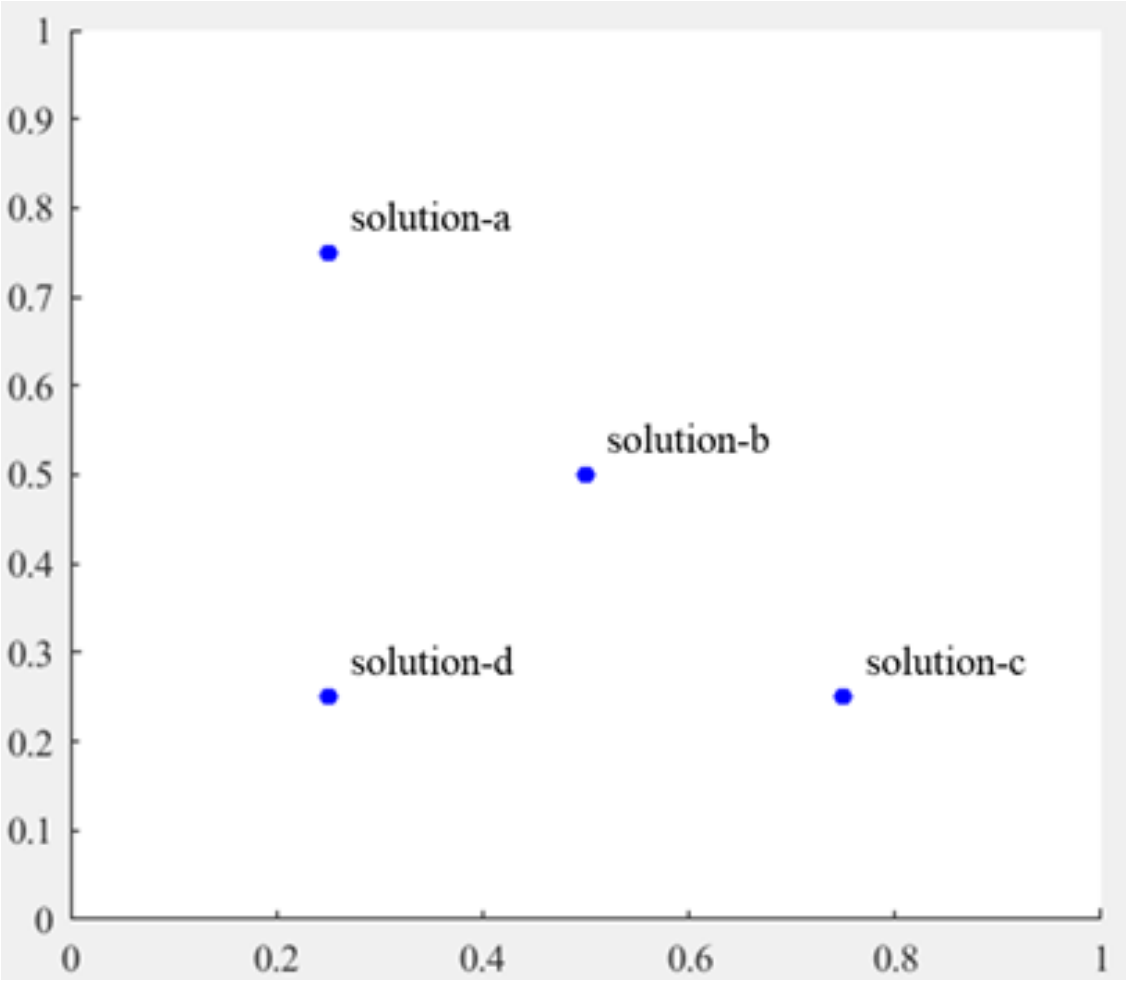


Figure 1. 4-solution example population demonstrating effects of population order.

4.2. A DOMINANCE SURROGATE

Knowledge of a solution's true dominance, however, cannot be affordably attained. Determining the number exact number of solutions any one solution of the population dominates would require a computational expense on the order of a complete comparison method. A surrogate is, therefore, proposed to estimate the dominance a solution may exhibit.

First, consider a point in the solution-space that is globally optimum, that is, optimum by every objective. This solution would then exhibit dominance over every other solution occupying the population. Called the *ideal point*, this location in the solution space is composed of the optimum exhibited value by any solution for each objective considered. Assuming minimization, as in (1), the ideal point ($\vec{x}^{ideal}$) is defined by Equation (3) where *M* is again the number of objectives and *X* is the population of possible solutions.

$$\vec{x}^{ideal} \ni \vec{x}_i^{ideal} = \inf_{x \in X} f_i(x) \; \forall \; i = 1, \ldots, M \tag{3}$$

While a solution matching the ideal point is not likely to exist within a multiobjective population, the knowledge of its location can be used in establishing an expected dominance surrogate. Here proposed is the use of the squared Euclidean distance of a solution from the ideal point as an estimate of the solution's Pareto dominance. Each objective's contribution to the distance is also divided (or scaled) by the corresponding ideal point term to handle varying scales. The calculation of this proxy metric for a solution-a (E^a) with *M* objectives is detailed in Equation (4).

$$E^a = \sum_{i=1}^{M} \frac{\left(f_i(x^a) - \vec{x}_i^{ideal}\right)^2}{\vec{x}_i^{ideal}} \tag{4}$$

Determining the squared distance to the ideal point (*E*) for all members of the population, those closest are considered most dominant by the surrogate measure.

4.3. ALGORITHM DESCRIPTION

Described by Algorithm 1, the proposed method makes use of the established dominance surrogate and is relatively simple to implement. The procedure begins by

determining the ideal point of the population, as defined in Equation (3). Once determined, the distance of each solution to the ideal point is calculated and the population is sorted according to this value in an ascending fashion. The algorithm then selects the first solution in the sorted list, marks it as a member of the current –initially, the first– front, and begins comparing it to its counterparts in an orderly manner. Progressing down the sorted list, if a solution is dominated by the selected solution, the dominated candidate is marked as such and will not be a member of the current front.

Note that considerations do not need to be made for the selected solution being dominated by a subsequent counterpart, due to the nature of the pre-comparison sorting. For a solution to exhibit a lesser *E*-value than a counterpart solution, the former must be better (smaller-valued) than the latter by at least one objective. Called a *one-way* domination comparison, this characteristic is also intrinsic to Corner Sort [16], and further implies that any solution enduring a tenure as the selected solution will be present within the current front. These factors are largely responsible for the improved experimental performance of Corner Sort over other inferred dominance methods.

Once a selected solution has been compared to all unmarked members of the population, the next unmarked solution is selected, ranked within the current front, and is compared to its remaining unmarked counterparts. When all solutions have been marked (as dominated or members of the current front), marks are cleared from the dominated solutions and the procedure is repeated to find the next front of the population.

These operations are repeated until every solution has been assigned to a front (Algorithm 1: line 10) or the number of solutions assigned to fronts satisfies what is needed for selection by the employing multiobjective EA (Algorithm 1: line 25).

Algorithm 1: Ideal Sort

Inputs: Population *P* with *N* solutions and *M* objectives; Boolean *T* defining if the algorithm is to be terminated after a sufficient number of solutions are ranked.

Functions: *ideal(X)*: Determines ideal point of population X, as defined by Equation (3); *SSED*(*y*, *z*): determines squared scaled Euclidean distance between two points *y* and *z*, as defined by Equation (4).

Output: Pareto front *Rank* of desired number of solutions

```
I = ideal(P)                                   // Determine ideal point
for n = 1 to N                                 // For all solutions
  E^n = SSED(p^n, I)                           // Determine SSED from n^th solution of P to I
end
P ← Sort(P, E)                                 // Sort P by ascending E-value
front = 1                                      // Initialize front to 1
Rank[1 : N] = null                             // Initialize Rank to null for all solutions
ranked = 0                                     // Initialize number of ranked solutions to 0
marked = 0                                     // Initialize number of marked solutions to 0
while ranked < N                               // While not all solutions are ranked
  current = 1                                  // Initialize current solution to first
  while marked < N                             // While not all solutions are marked
    while Rank[current] != null                // While the current solution is marked
      current += 1                             // Increment current
    end
    Rank[current] = front                      // Rank current as front
    ranked += 1                                // Increment ranked
    marked += 1                                // Increment marked
    for i = current+1 : N                      // Rank current as front
      // If p^i is not marked and is dominated by p^current
      if Rank[i] == null && p^current ≺ p^i then
        Rank[i] = inf                          // Mark p^i as dominated
        marked += 1                            // Increment marked
      end
    end
  end
  // If terminating and a sufficient number of solutions are ranked
  if T == True && ranked ≥ N/2 then
    ranked = N                                 // Set ranked to N to terminate sorting
  end
  marked = ranked                              // Set marked to only ranked
  Rank[Rank==inf] = null                       // Set Rank to null for dominated solutions
  front += 1                                   // Increment front
end
return
```

4.4. THE BENEFITS OF TERMINABILITY

The latter method by which the algorithm may be concluded is allowable by the terminability property of Ideal Sort. Maintained similarly by the other inferred dominance methods introduced, this property allows the algorithm to be terminated following the determination of any single front defining the population.

To demonstrate the benefits of this property, reference to Figure 2 is made. This graphic provides a high-level overview of the general selection operations that an NDS multiobjective EA may follow. For each iteration, an initial population is comprised of solutions surviving a previous iteration's selection and the generated offspring they produce. Applying NDS, the fronts defining this population are then determined and used as the primary ranking mechanism for selection. In the likely event of a tie in NDS rank between solutions seeking to survive selection (such as those solutions comprising the third front in Figure 2), a diversity procedure is often used to keep the required number of solutions that maintain the best diversity about the population. Those selected by this procedure are then passed to the generation procedure used to form the initial population of the next iteration.

This procedure can solicit some inefficiency during the NDS operations. Consider first that the number of solutions surviving from one iteration to another is generally fixed (and even more generally, known). Once this desired number of solutions has been ranked, the operations performed to rank solutions into subsequent fronts provides no value to the larger procedure. Returning to Figure 2: after the solutions belonging to the first, second, and third fronts have been identified, the effort dedicated to ranking solutions into the fourth and fifth fronts is computationally and temporally wasteful.

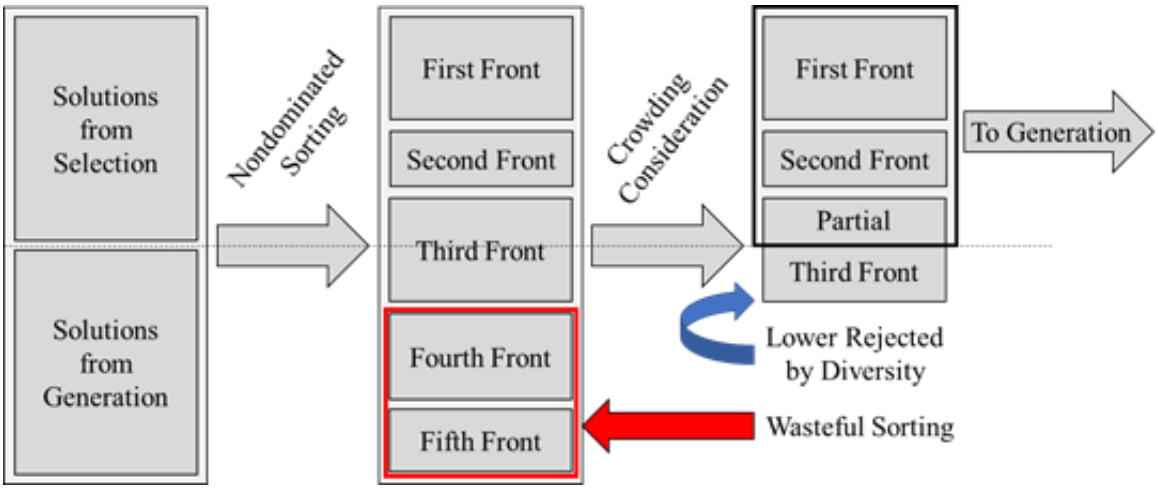


Figure 2. General selection operations of a generic NDS multiobjective EA.

Therefore, algorithms exhibiting terminability may be able to see substantial improvements to their computational and runtime efficiencies should this property be invoked. The merit of these assertions will be evaluated in Section 7.

5. TIME AND SPACE COMPLEXITY

The time complexity of the proposed method is contributed to by four general procedures:

1) Determining the ideal point of the population
2) Determining each solution's *E*-value, describing proximity to the ideal point
3) Sorting the population by ascending *E*-value
4) Performing domination comparisons to determine Pareto fronts

Assuming minimization, finding the ideal point of a population with N solutions and M objectives is equivalent to finding the minimum of a 1xN array, M times. As each solution must be examined to find the minimum of each 1xN array, the number of comparisons required will be $O(N)$ per objective. Completing this procedure for each of the M objectives yields a contribution of $O(MN)$ to the overall complexity. Determining the E-value of each solution similarly requires $O(MN)$ operations as each of the N solutions must be compared to the ideal point with respect to M objectives. Once the E-value of each solution is found, the population can be sorted in-place via Heapsort with a complexity of $O(N\log N)$ [19].

Having established the sorted population, the time complexity of the nondominated sorting procedure should be investigated for the worst- and best-case. The worst-case complexity for Ideal Sort occurs when all solutions of the population are in a single front. In such a scenario, each solution in the sorted population will be compared to each subsequent solution. With no solution being marked as dominated, a total of $\frac{1}{2}M(N^2 - \frac{N}{2})$ comparisons will be required for a population with N solutions and M objectives. This worst-case comparison complexity mirrors that of Deductive Sort [15] but is faster as it only requires one-way domination comparisons. This worst-case complexity also arises for the vanilla (non-terminating) implementation of Ideal Sort when each solution occupies its own front; the terminable implementation of Ideal Sort performs better in this case as the second half of the sorted population is not ranked into fronts. Examining these contributing processes, the algorithm is shown to exhibit a worst-case complexity of $O(MN^2)$.

The best-case for the vanilla Ideal Sort algorithm bears a relationship to the triangular number sequence, where the nth triangular number (T_n) is the sum of the natural numbers from 1 to n. When a population with T_n solutions is comprised by n fronts and the number of solutions in incrementing fronts reduces by one (see Figure 3) the number of comparisons required to rank the solutions can be reduced to the value defined by Equation (5). Herein, C_N is the number of comparisons required for a population of N solutions with M objectives where N is the nth triangular number. This best-case complexity additionally requires that the first encountered solution (according to E-value sorting) in each front dominates every solution of each subsequent front. Populations of this structure may be created at varying scales and dimensionality by following the procedure for fixed-front dataset generation described in [15].

$$C_N = M \sum_{i=1}^{n} i(i-1) \tag{5}$$

For populations with a number of solutions (N) between the nth and $n+1^{th}$ triangular number, the best-case number of comparisons required occurs in select instances where a nonincreasing number of solutions comprise incrementing fronts. In such a case, the comparisons required (C_N) is a linear interpolation between that required for the nth and the $n+1^{th}$ triangular number. This is defined by (6) where $\hat{C}_n$ is minimum number of comparisons required for a population size equal to the nth triangular number (see Equation (5)) and the number of objectives is M.

$$C_N = M\left(\hat{C}_n + \frac{N - T_n}{T_{n+1} - T_n}\left(\hat{C}_{n+1} - \hat{C}_n\right)\right) \tag{6}$$

This required number of comparisons can be improved by incorporating terminability, a claim evidenced by the experimental results of the subsequent section.

Ideal Sort requires $O(N)$ space to track which solutions are dominated and the rank assigned to each solution.

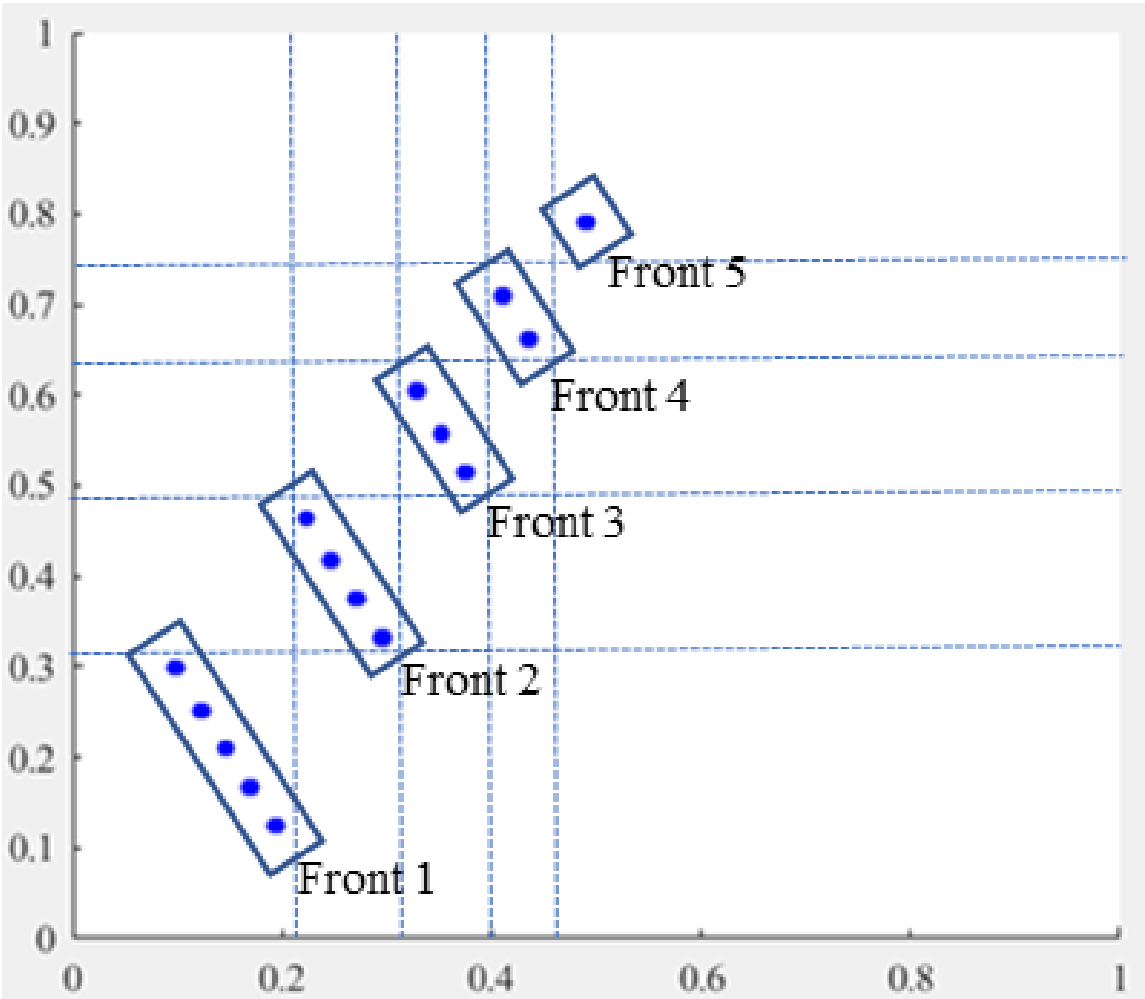


Figure 3. Population with T5 solutions enabling best-case computational performance by the Ideal Sort algorithm.

6. EXPERIMENTAL RESULTS

This section compares the performance of Ideal Sort to four other NDS algorithms: Deductive Sort [15], Corner Sort [16], Best Order Sort [11], and Bounded Best Order Sort [12]. Additionally, simple terminability modifications are made are to Deductive Sort and Corner Sort to examine this property's effect on performance characteristics. These modifications mirror that described for Ideal Sort in Algorithm 1: Lines 25-27. As members of the constructive front class of NDS methods, the current implementations of Best Order Sort and Bounded Best Order Sort must examine all solutions before any front can be considered complete. Therefore, these algorithms have not been modified to include an early termination procedure. Thus, the total number of algorithms examined is eight, with Deductive Sort, Corner Sort, and Ideal Sort being implemented under terminable and vanilla statuses.

These algorithms are presented with a variety of multiobjective datasets, described within the next subsection. The number of objective comparisons to determine domination and the total algorithmic runtime required are used to evaluate the performance of each NDS procedure. 30 instances of each dataset are created and the average performance of each algorithm on each dataset is presented in the *Experimental Results* subsection. Each algorithm is implemented in Java and experiments are conducted on a PC with a 3.00GHz Intel Core i9-9980XE CPU with 64 GB of RAM.

6.1. EXPERIMENTAL DATASETS

To compare the performance of Ideal Sort with these state-of-the-art methods, a test suite of multiobjective datasets is developed. Instances of these datasets are presented to each algorithm under similar conditions, enabling a comparison of the discussed performance measures. Two standard multiobjective dataset creation methods are utilized: cloud dataset generation and fixed front dataset generation. Additionally, a new generation scheme is presented and implemented, aimed at emulating a common EA scenario.

6.1.1. Cloud Dataset Generation. Cloud dataset generation methods create random populations of solutions with objective values pulled from the uniform distribution between zero and one. The datasets can be tailored to a desired population size and number of objectives using the procedure defined in [16]. These datasets reflect the scenarios incumbent of NDS procedures at the beginning of an EA implementation as initial populations are often generated randomly [12]. Two series of cloud datasets were created. The first maintains a constant population level of 10,000 solutions while incrementing the number of objectives by 1 from 2 to 30 (Figure 5). The second series conducted incrementation along two axes, increasing the population size from 1,000 to 10,000 in steps of 1,000 solutions, while also incrementing the number of objectives from 5 to 20 by a step size of 5 (Figure 7). In total, 69 different cloud dataset classes were produced, each owning 30 unique instances for an examination total of 2,070 unique cloud populations.

6.1.2. Fixed Front Dataset Generation. Fixed front dataset generation methods offer further tailoring of the dataset by allowing the number of fronts within the

population to be controlled. Utilizing the strategy described in [16], a reasonably similar number of solutions are placed within each of the designated number of fronts, enabling the investigation of examined methods' performances across a variety of dominance scenarios. As different methods may inherently perform better given a different number of Pareto fronts, it is important that a range of front-counts be examined. In doing so, the demonstrated strength of consistently well-performing algorithms can be considered more robust.

A two-axis incrementation was similarly employed in the creation of fixed front datasets, incrementing the number of fronts by 1 from 1 to 15 in one direction and the number of objectives by 5 from 5 to 20 in the other (Figure 8). Each population contains 10,0000 solutions. 60 classes culminate in the examination of 1,800 unique fixed front populations.

6.1.3. Evolved Dataset Generation. The final dataset generation strategy employed seeks to mimic the datasets presented to an NDS algorithm by EAs during general iterations. Any non-initialization iteration of a generic EA will have a population comprised by two classes of members. The first class constitutes those solutions from the previous iteration that survived the selection process. The second is composed of offspring solutions generated by combining characteristics from the surviving members and incorporating the effects of mutation. This procedure within multiobjective EAs enables the gradual convergence of the maintained population to the Pareto front of a decision-space [3]. This gradual migration is emulated by Algorithm 2 as an initial population is used to make modified "offspring" that are better or worse by one objective. For each added solution, a root solution of the original population, used to create the new

member, will either dominate or be dominated by the added solution. Highly customizable, the generated dataset and the level of migration its added members exhibit are dependent on a few input parameters defined next.

Algorithm 2: Evolved Dataset Generation

Inputs: Population (P) dimensions: N solutions and M objectives; a: the number of new solutions added to P as a fraction of N; b: the best objective performance improvement exhibited by an added solution as a fraction

Functions: *randi*(y, z): generates a random number from the uniform distribution between y and z

Output: Evolved dataset with $N+N*a$ solutions and M objectives

```
    // Create initial front of population
1   for i = 1 to N                                          // For all solutions
2     P(i, 1) = randi(0, 1)                   // Set first objective value between 0 and 1
3     for j = 2 to M-1                                      // For objectives 2 to M-1
4       P(i, j) = randi(0, 1)*sum(P(i, 1:j-1))
5     end
6     P(i, M) = 1-sum(P(i, 1:M-1))                          // Ensure nondomination
7   end
    // Add new solutions emulating offspring
8   for k = 1 to N*a                                        // For new solutions
9     modCol = randi(1, M)                          // Select column for modification
10    modVal = randi(1-b, 2-b)                // b% improvement to (100-b)% regression
11    P(N+k, :) = P(k, :)                                   // Add new solution
      // Modify new solution's modCol^th objective value according to modVal (smaller
         is better)
12    if P(k, modCol) ≥ 0                                   // If root is nonnegative
13      P(N+k, modCol) = P(k, modCol)*modVal
14    else                                                  // If root is negative
15      P(N+k, modCol) = P(k, modCol)*(2-modVal)
16    end
17  end
18  return
```

This procedure first creates a single front of N nondominating solutions with M objectives, as defined by user input (Algorithm 2: Lines 1-7). The user additionally supplies a desired number of additional solutions to be added to the population (a) as a fraction (can be improper) of N. Further supplied by the user is a parameter, b, defining

the superiority a new solution can have when compared to a member of the original population. Entered as a fraction, *b* describes the maximum improvement at any one objective a new solution can exhibit when compared to the root solution used to create it (i.e. a *b*-value of 0.15 indicates a maximum 15% improvement). This value further defines the proportion of added solutions that will be present within the first front of the expanded dataset. All four parameters are used in Lines 8-17 to add $N*a$ additional *evolved* solutions. An example evolved dataset utilizing the following parameters is shown in Figure 4:

- $N = 1{,}000$
- $M = 2$
- $a = 0.25$
- $b = 0.15$

Figure 4 clearly shows the incorporation of a handful of evolved solutions that are closer than the initial front to the optimal values of each objective. Each of these solutions will dominate at least one member of the original front and belong to the new Pareto efficient set.

An identical creation scheme to that used for cloud datasets was employed to develop 2,070 unique evolved datasets for examination (Figure 6; Figure 10). These populations were developed using an *a*-value of 1 (doubling the population as EA reproduction procedures generally do) and a *b*-value of 0.1.

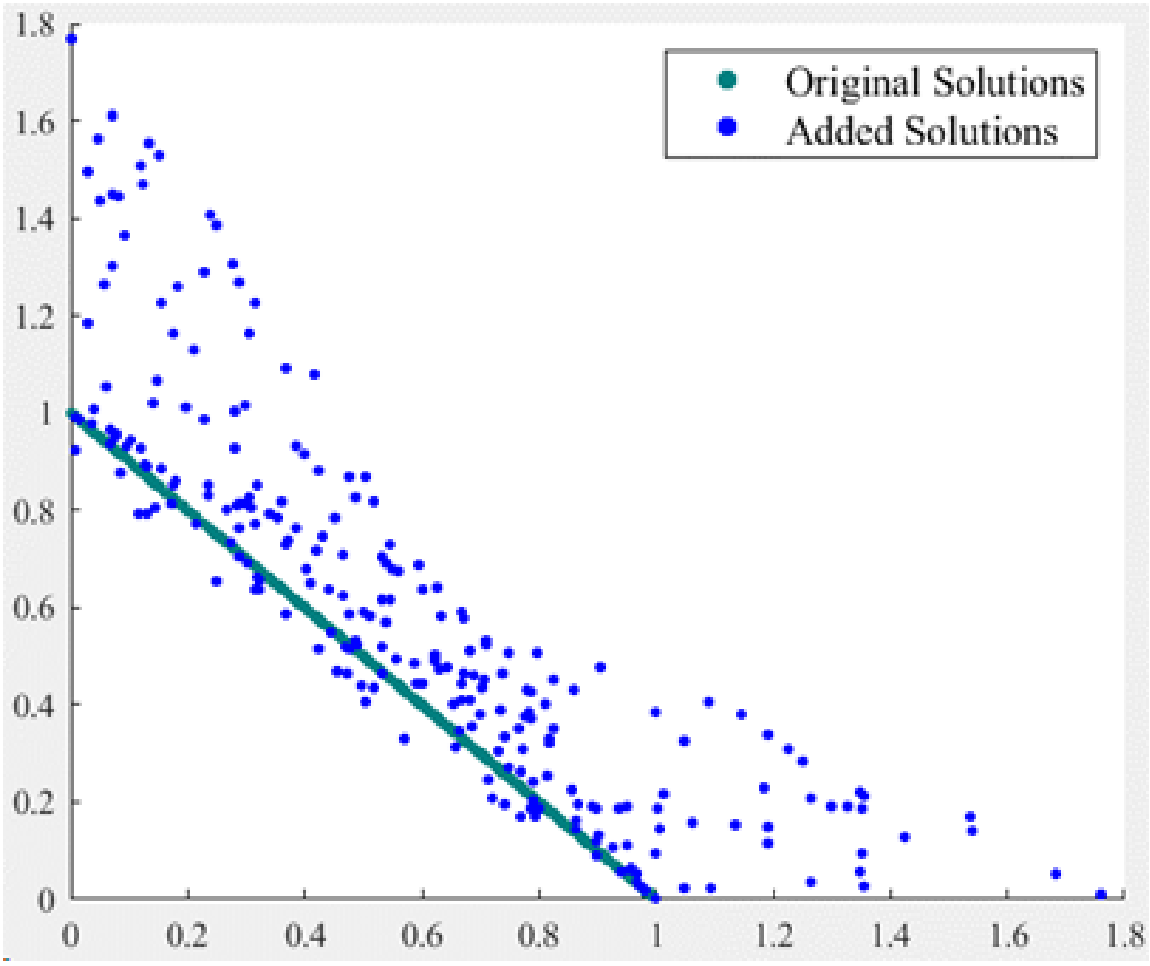
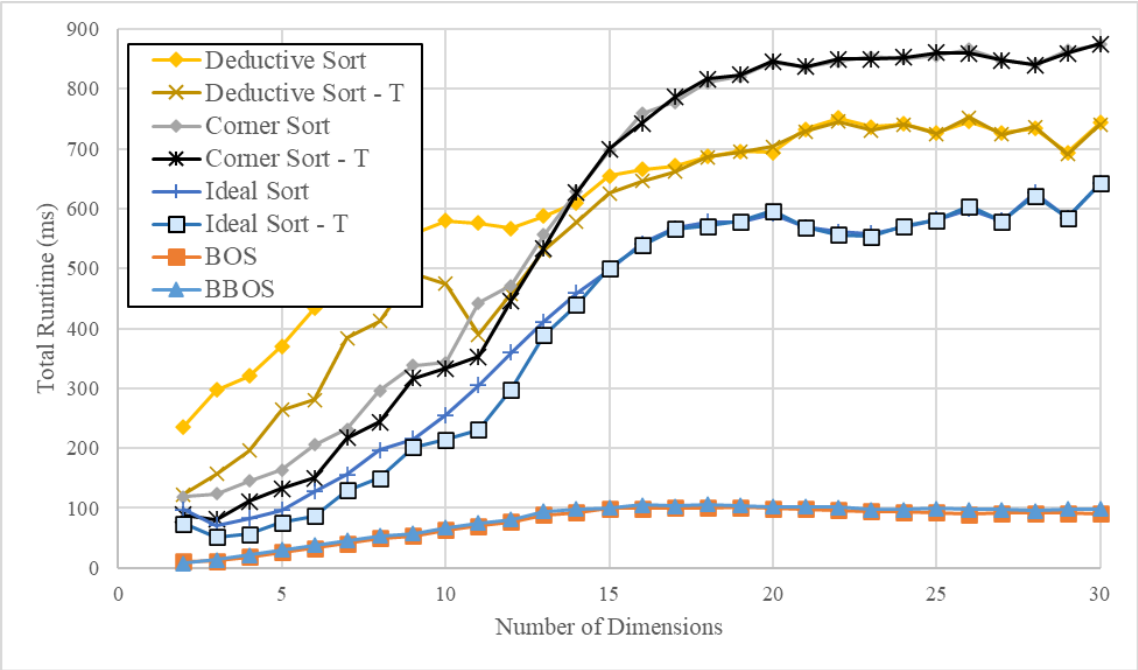
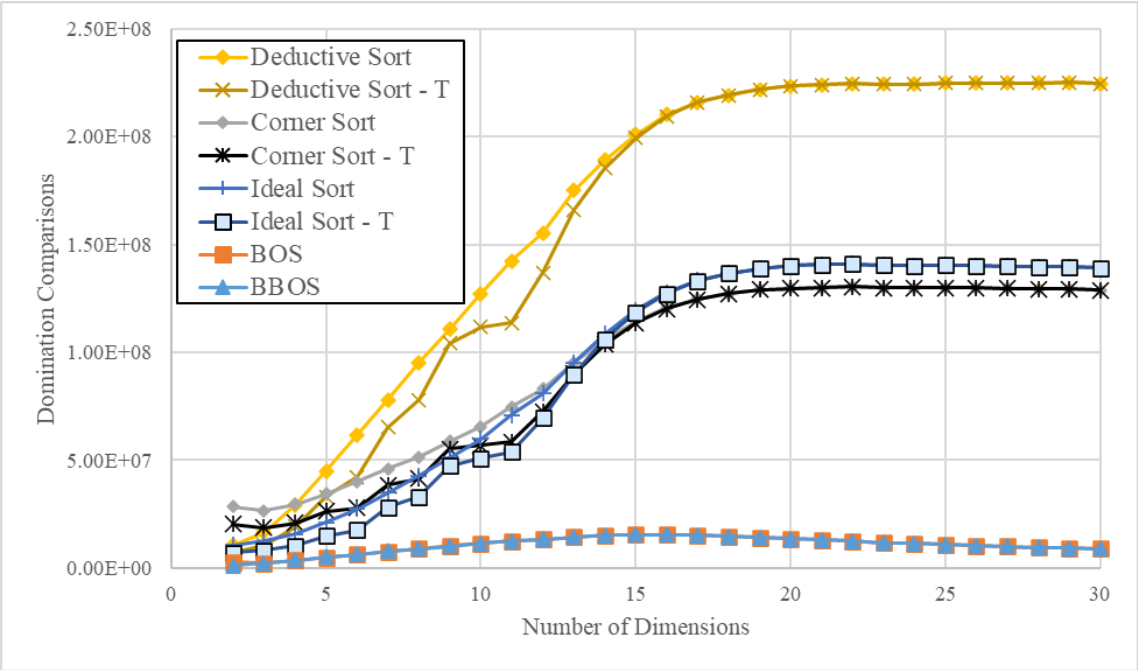


Figure 4. Example evolved dataset with 1,000 original solutions and 250 added solutions.

6.2. EXPERIMENTAL RESULTS

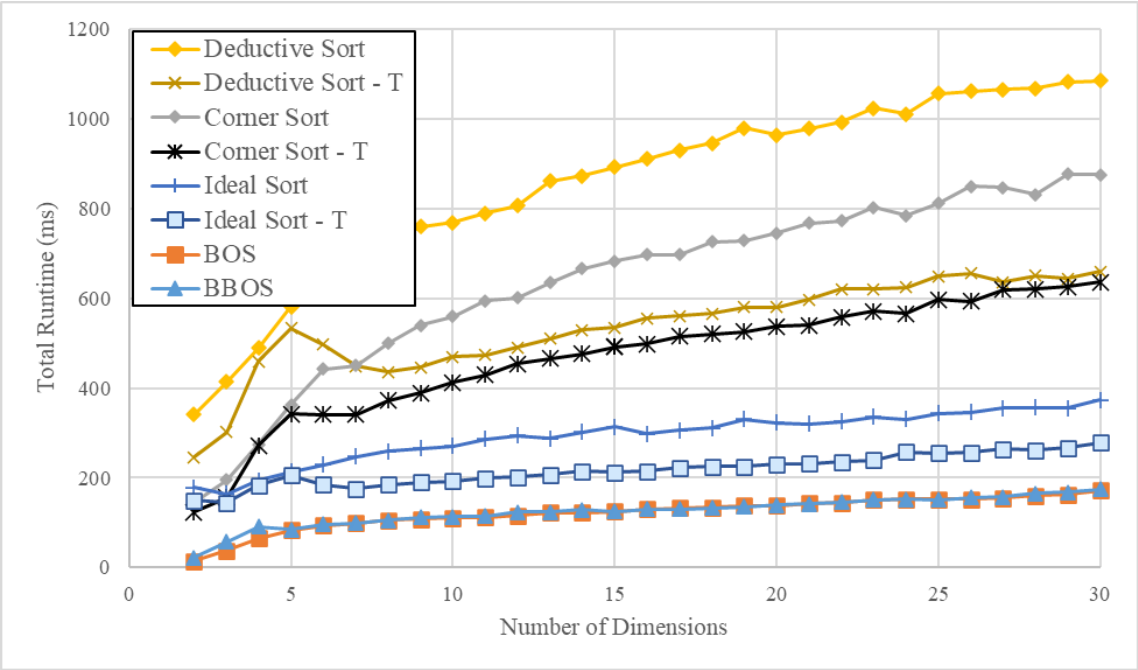


(a)

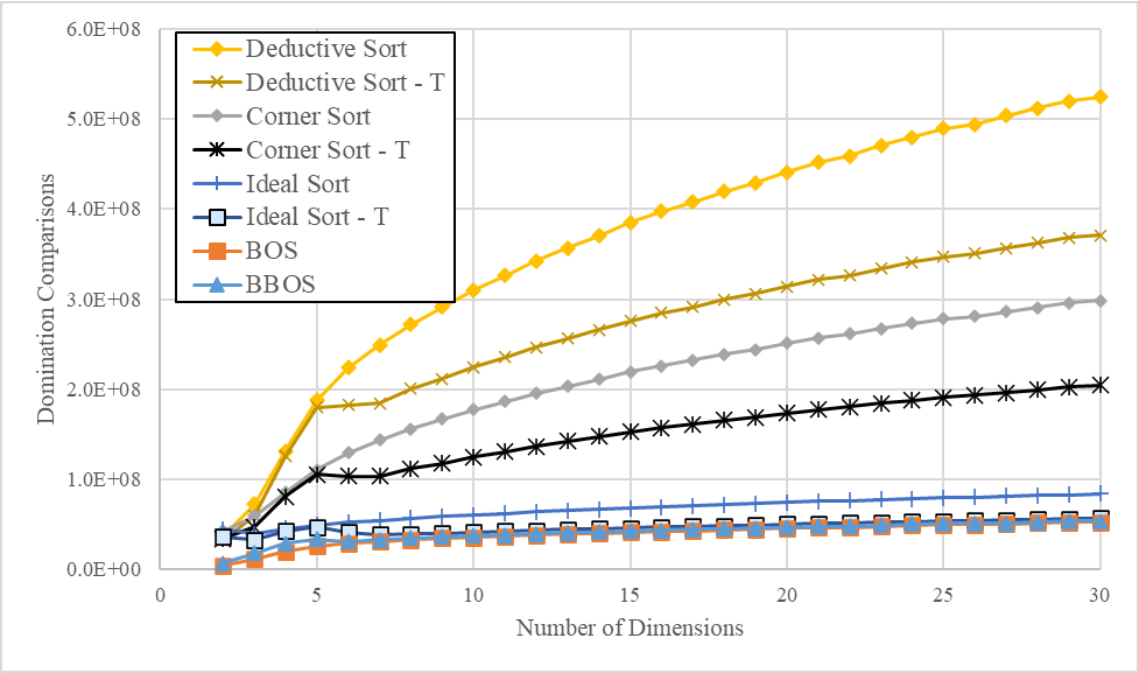


(b)

Figure 5. Computational performance for cloud datasets with 10,000 solutions and an incrementing number of objectives. a) Required runtime. b) Required number of dominance comparisons.

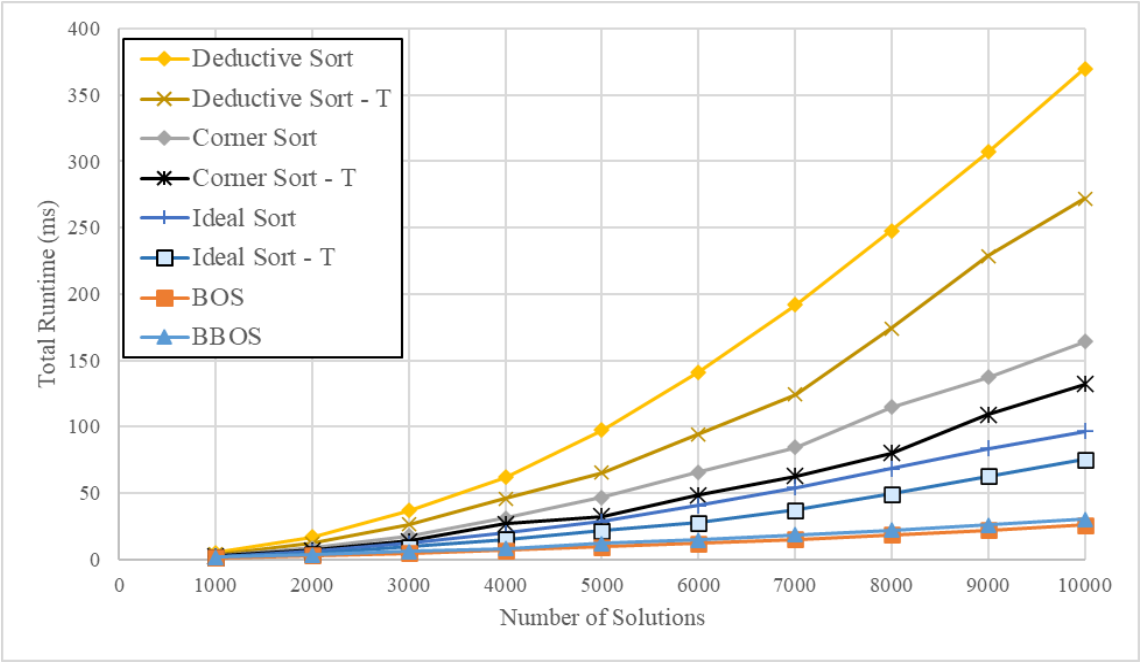


(a)

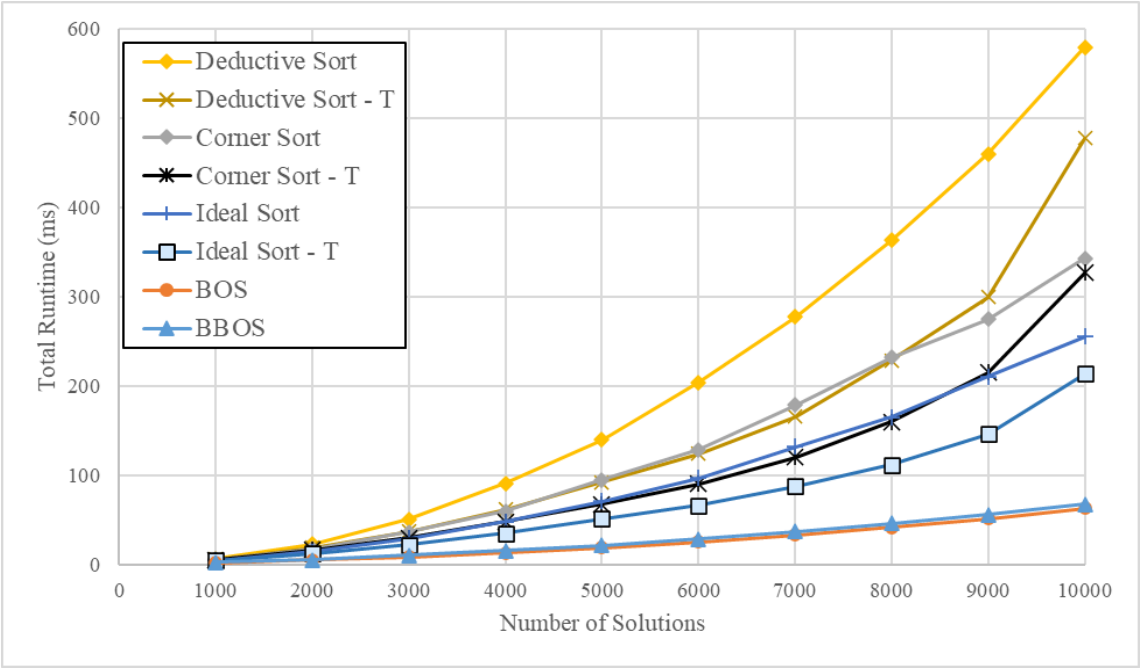


(b)

Figure 6. Computational performance for evolved datasets with 10,000 solutions and an incrementing number of objectives. a) Required runtime. b) Required number of dominance comparisons.

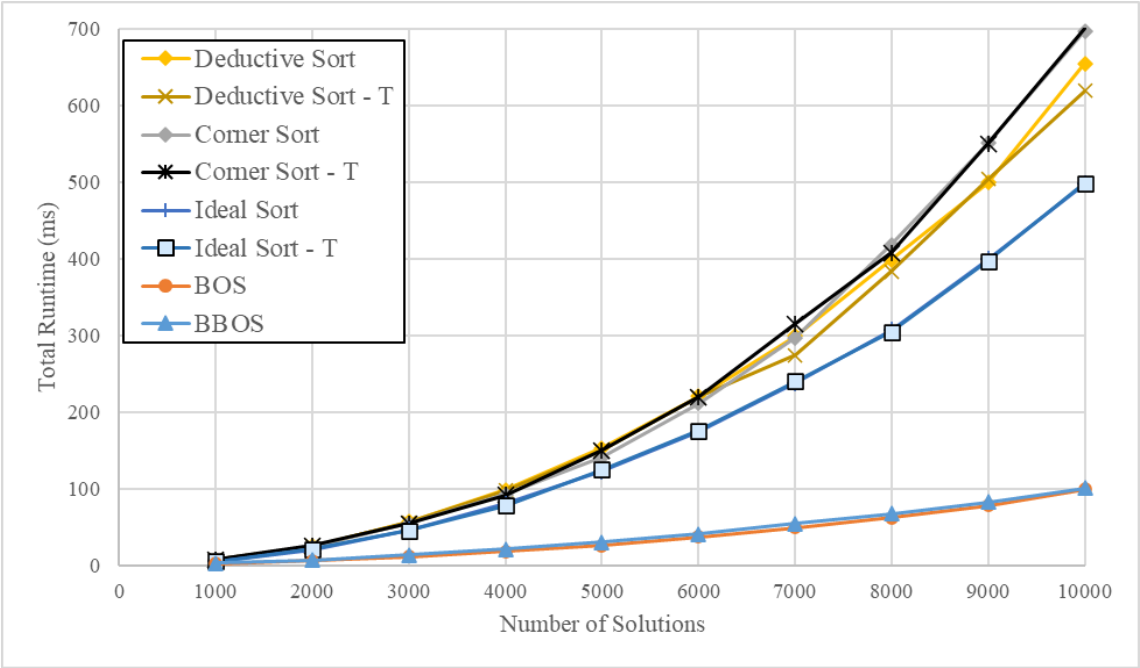


(a)

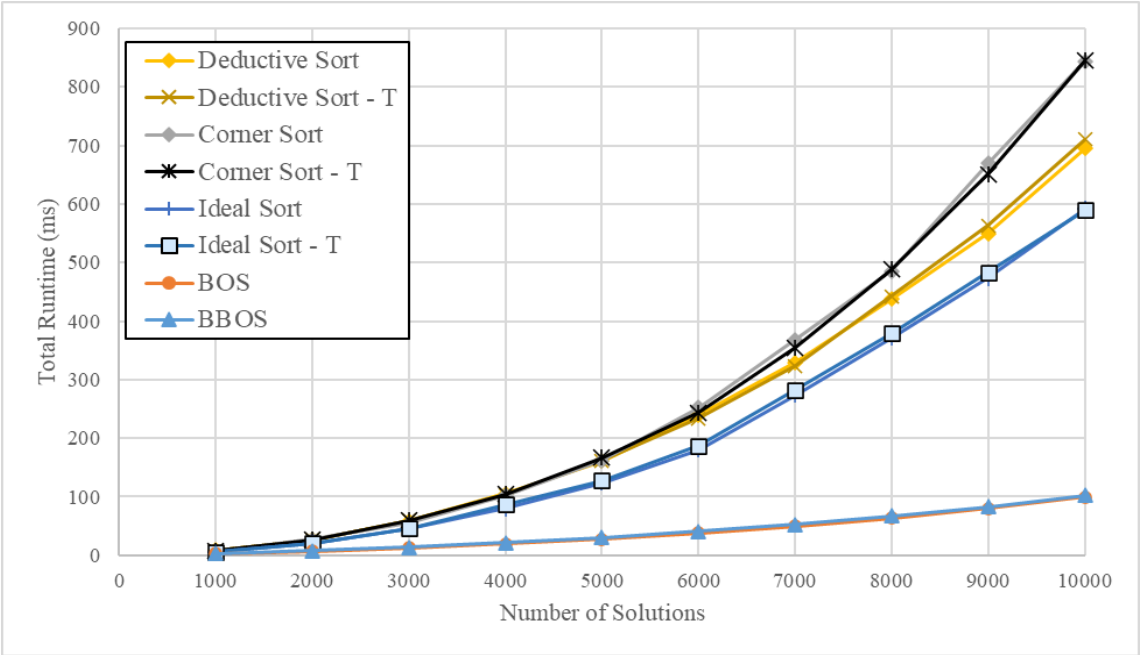


(b)

Figure 7. Computational performance for cloud datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives.

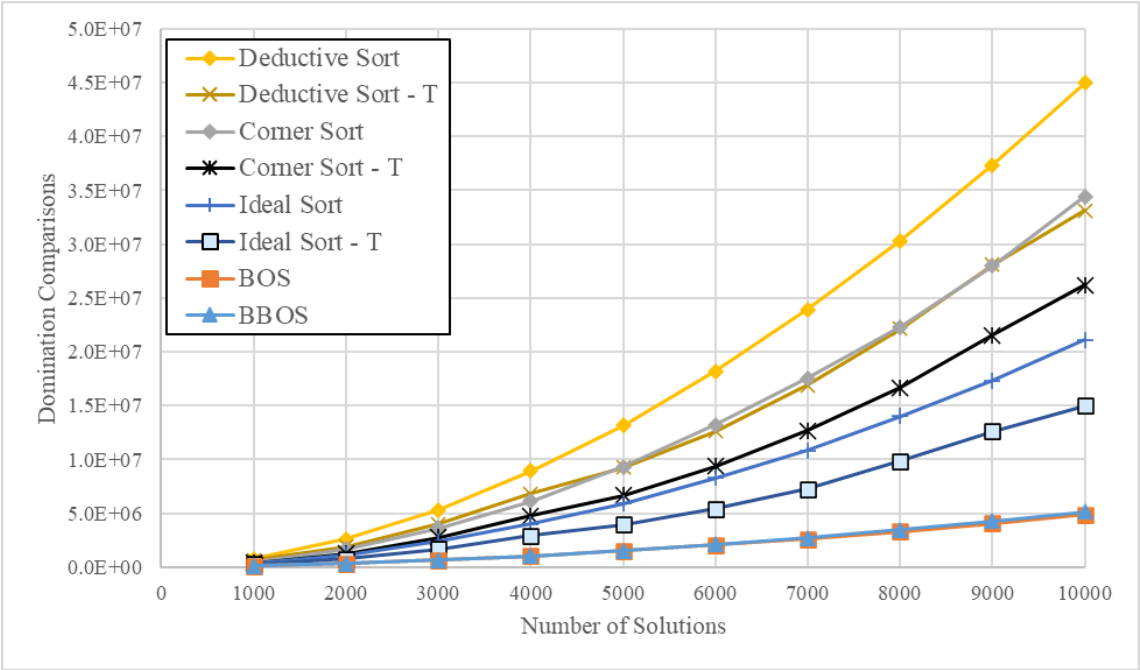


(c)

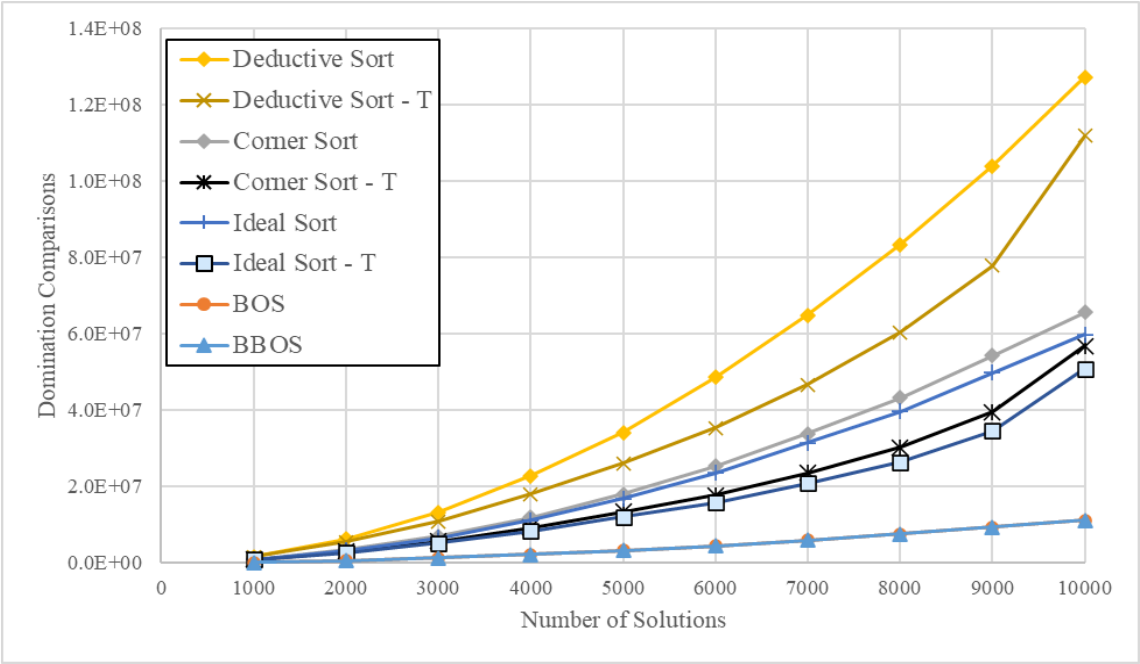


(d)

Figure 7. Computational performance for cloud datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

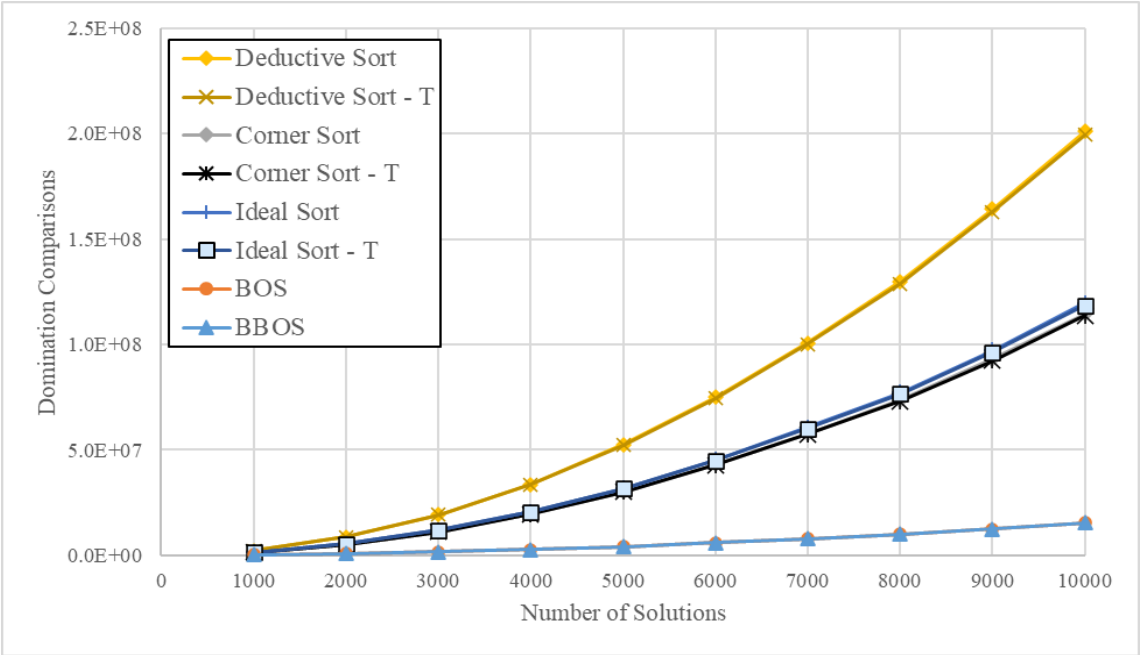


(e)

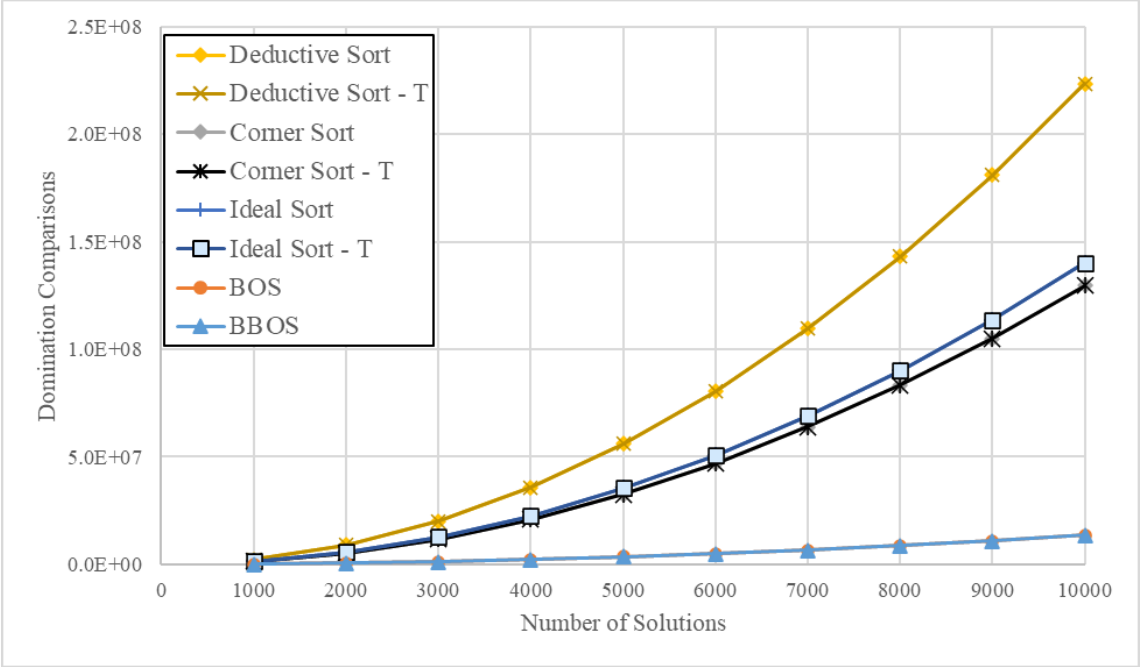


(f)

Figure 7. Computational performance for cloud datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

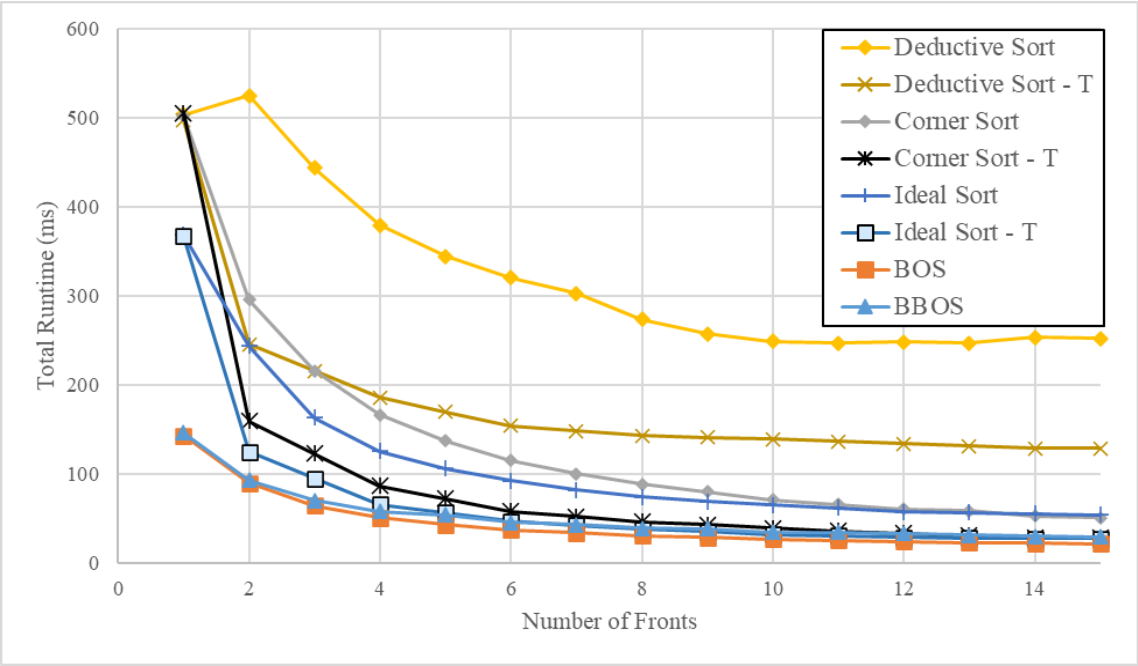


(g)

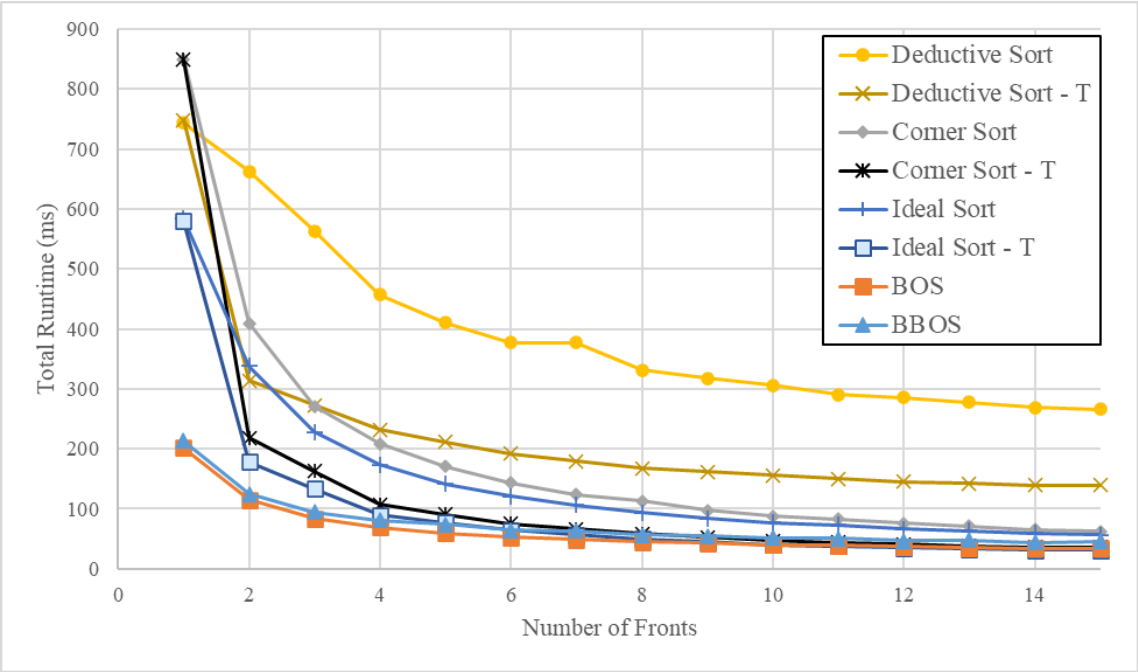


(h)

Figure 7. Computational performance for cloud datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

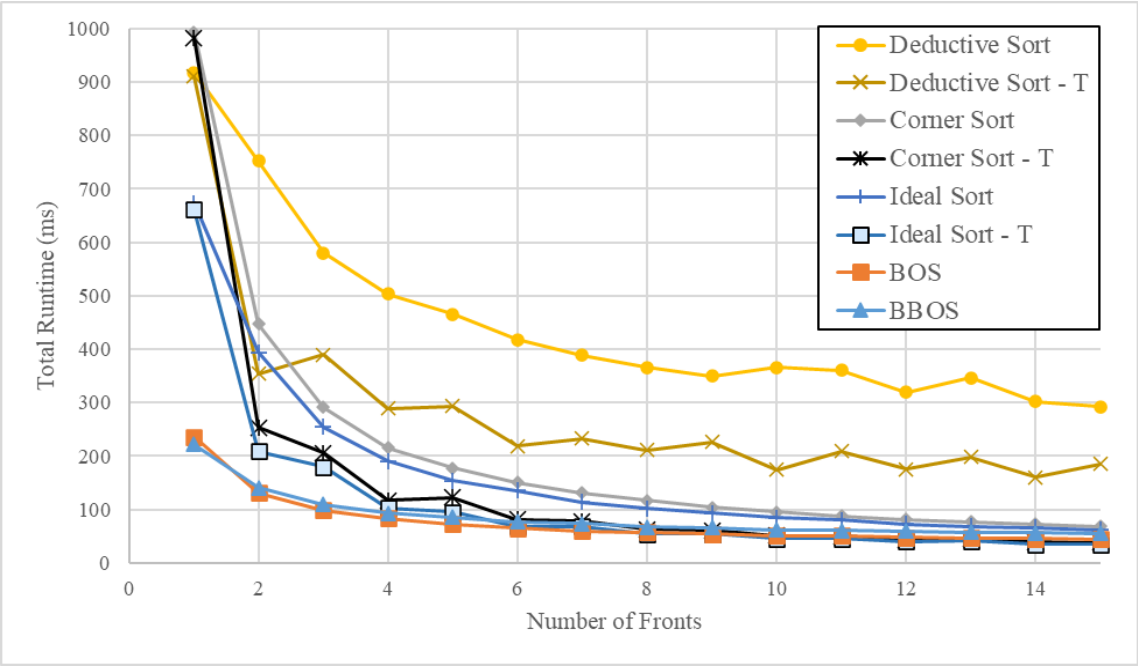


(a)

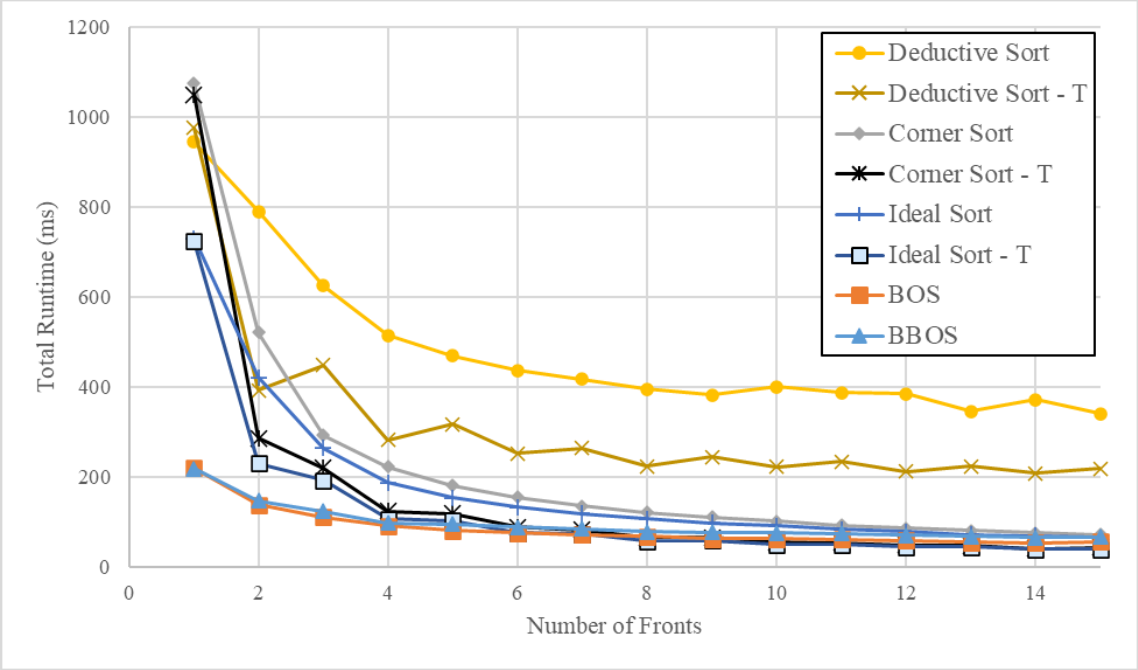


(b)

Figure 8. Computational performance for fixed front datasets with an incrementing number of fronts and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives.

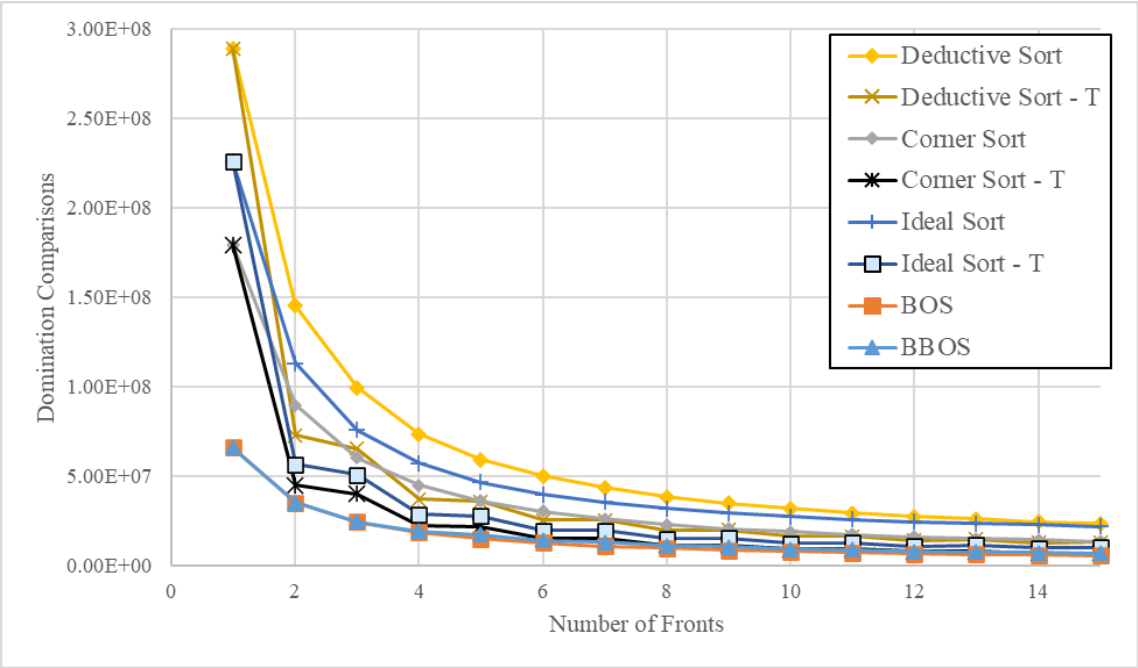


(c)

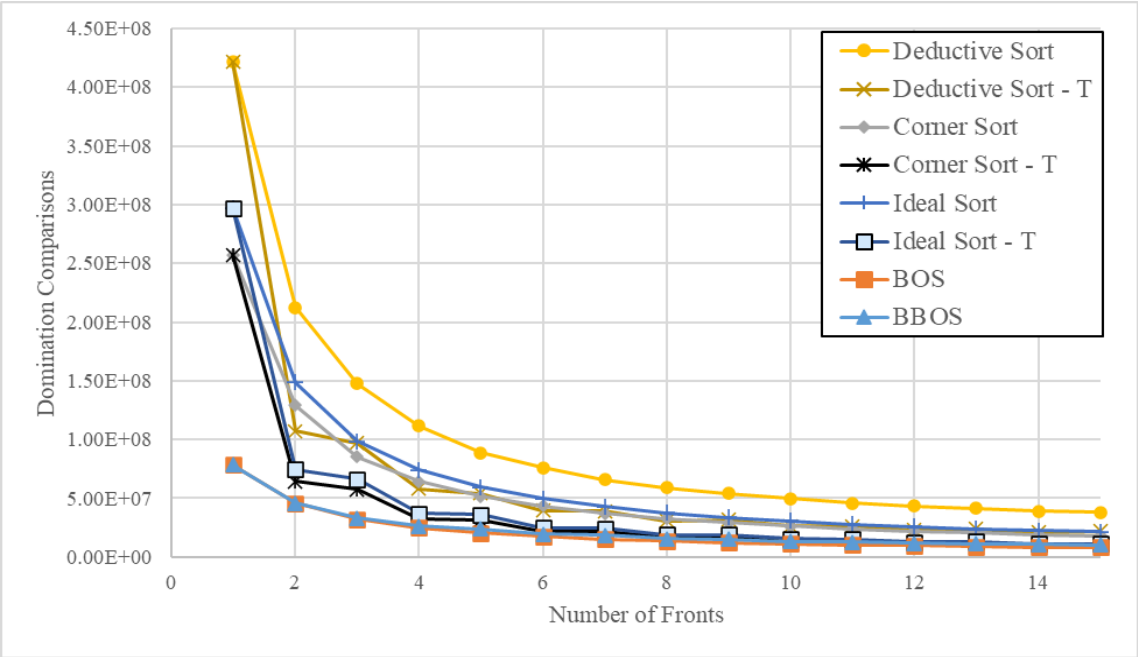


(d)

Figure 8. Computational performance for fixed front datasets with an incrementing number of fronts and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

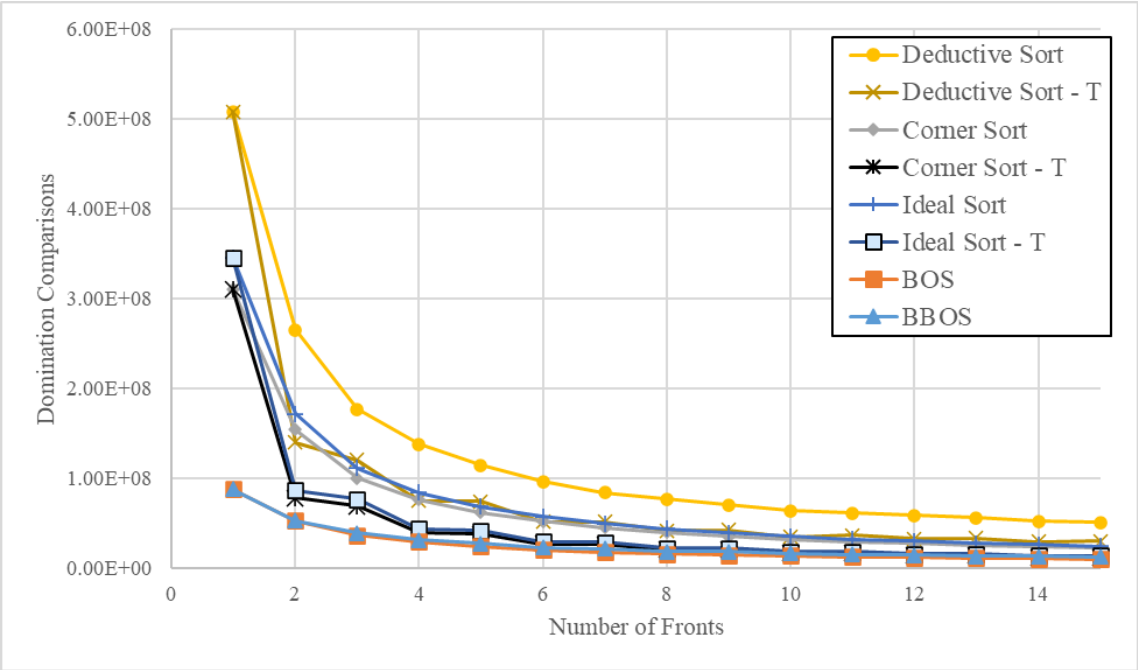


(e)

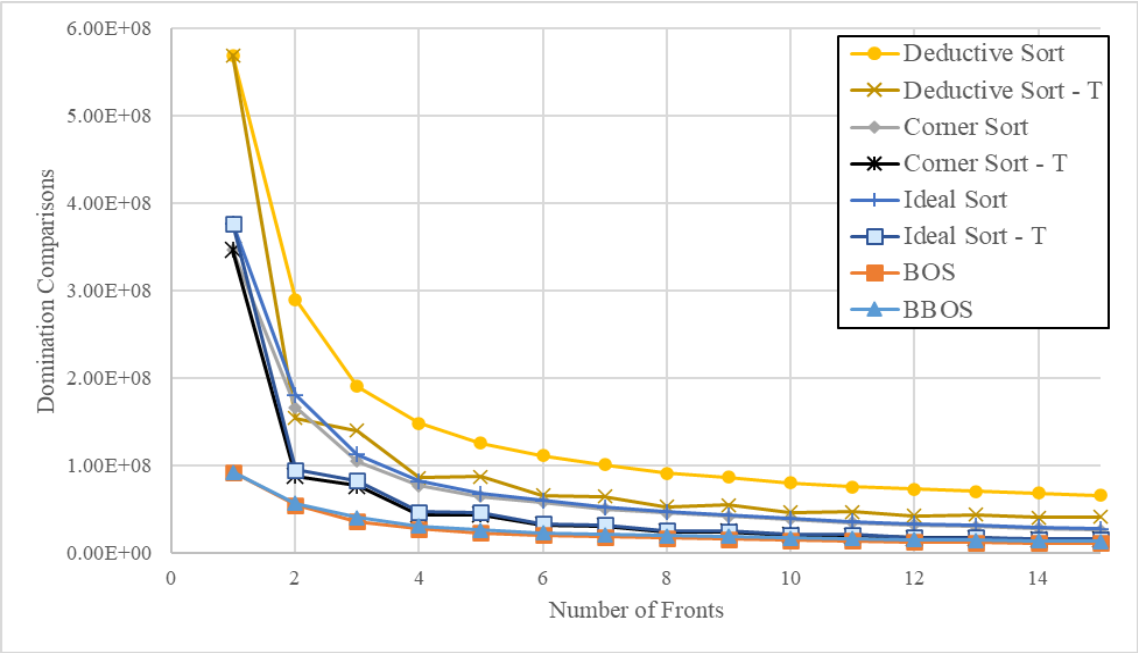


(f)

Figure 8. Computational performance for fixed front datasets with an incrementing number of fronts and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

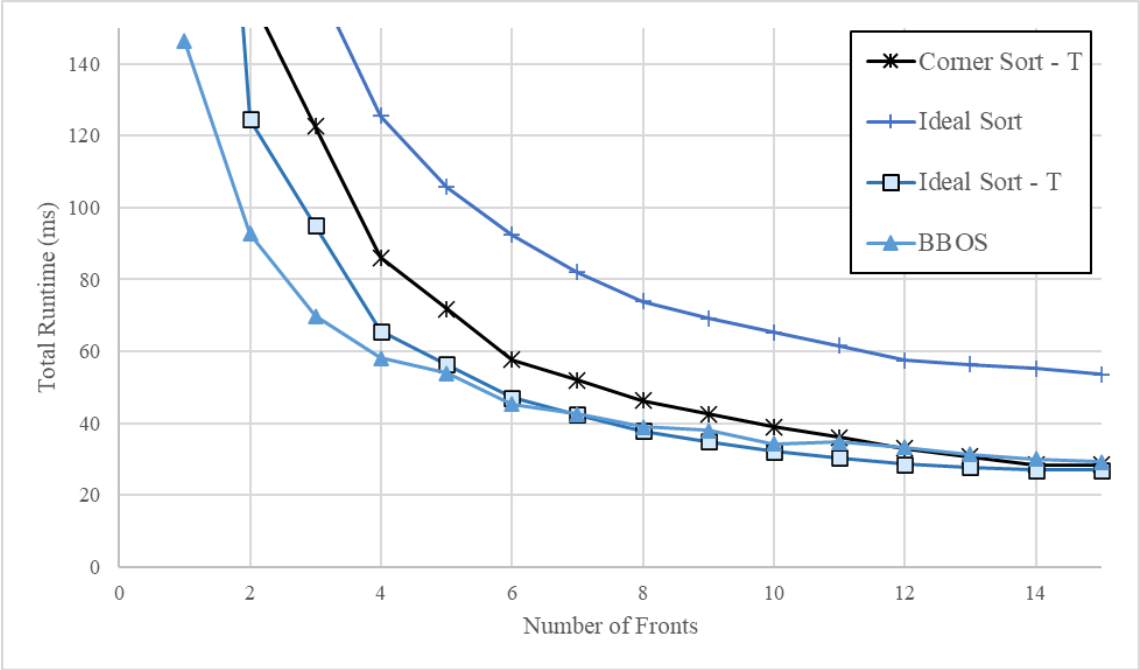


(g)

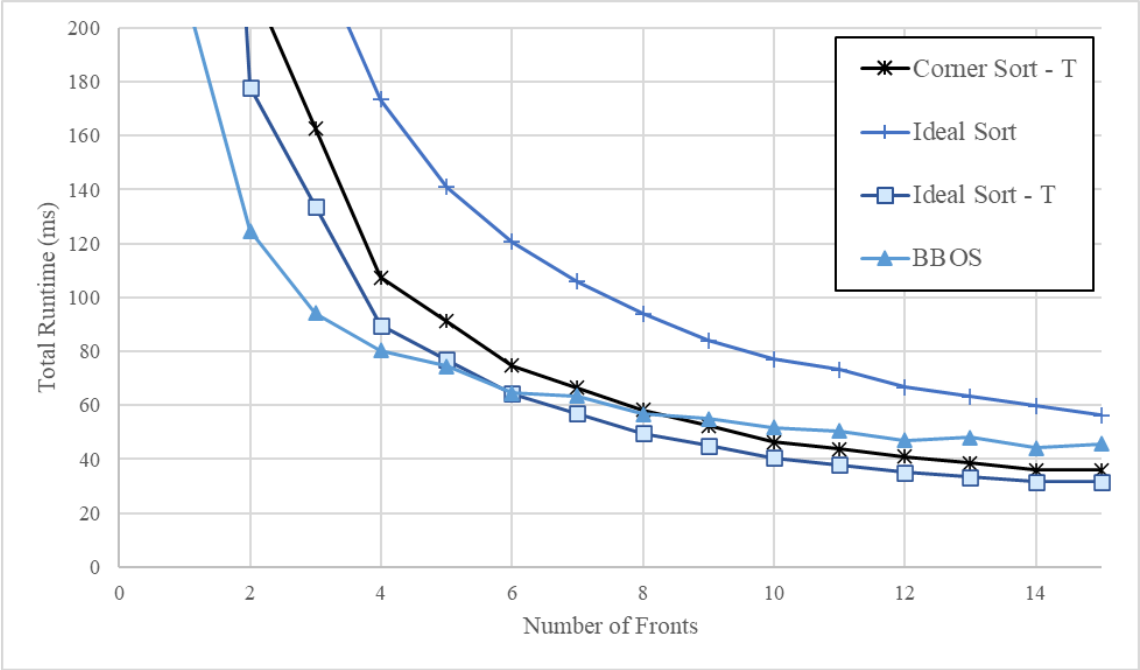


(h)

Figure 8. Computational performance for fixed front datasets with an incrementing number of fronts and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

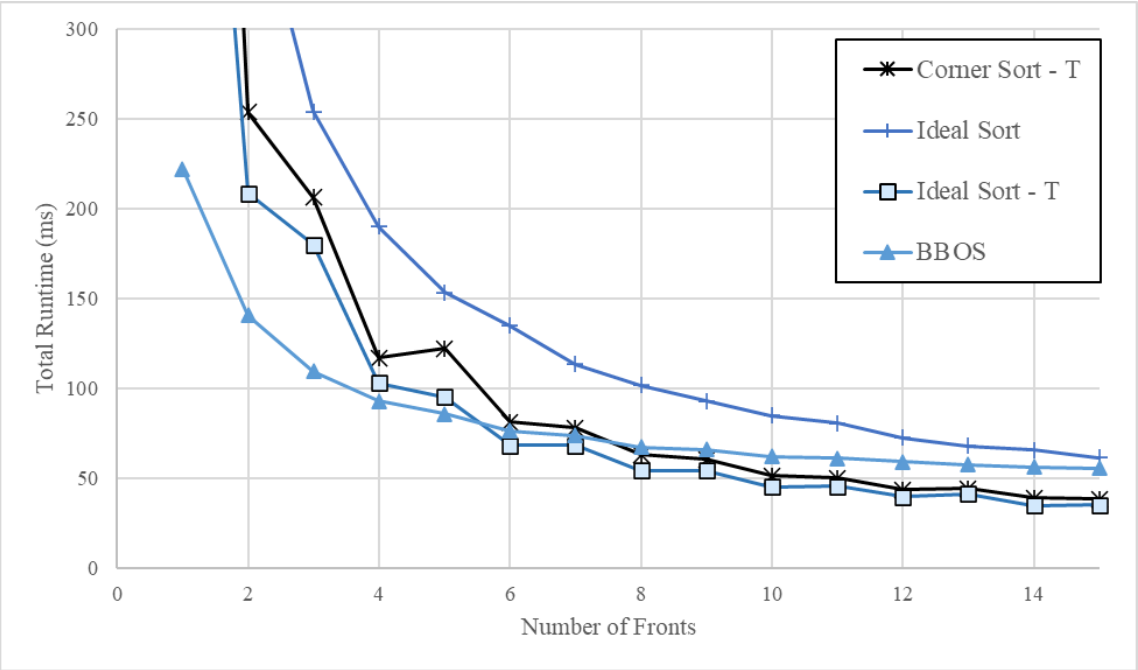


(a)

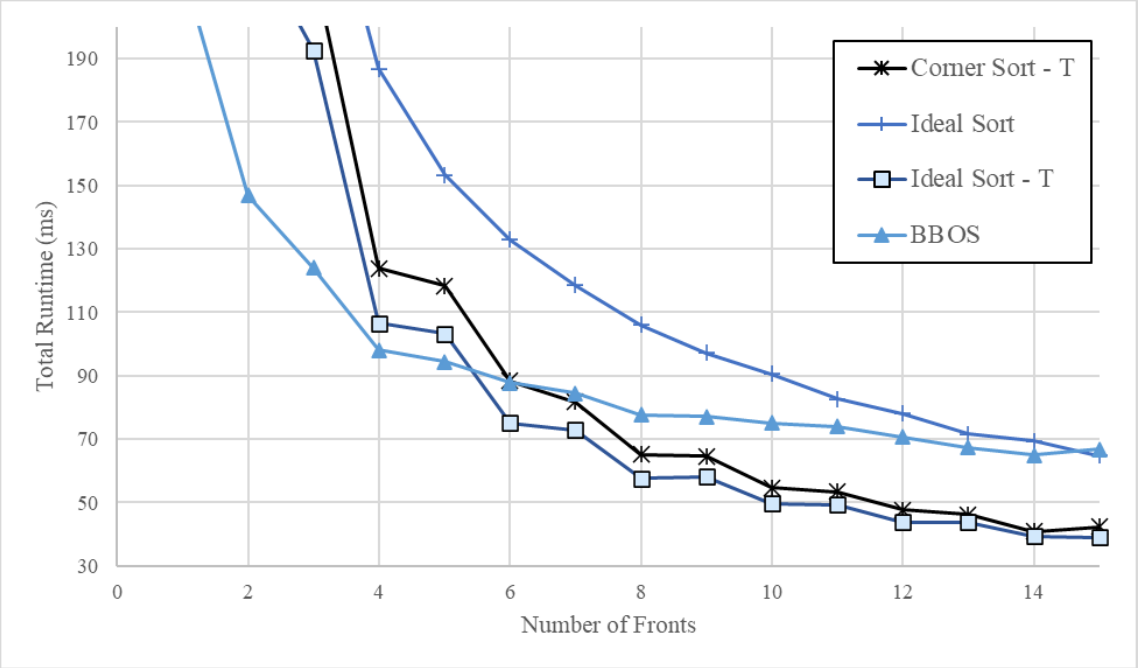


(b)

Figure 9. Zoomed region of required runtime for fixed front datasets with an incrementing number of fronts and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives.

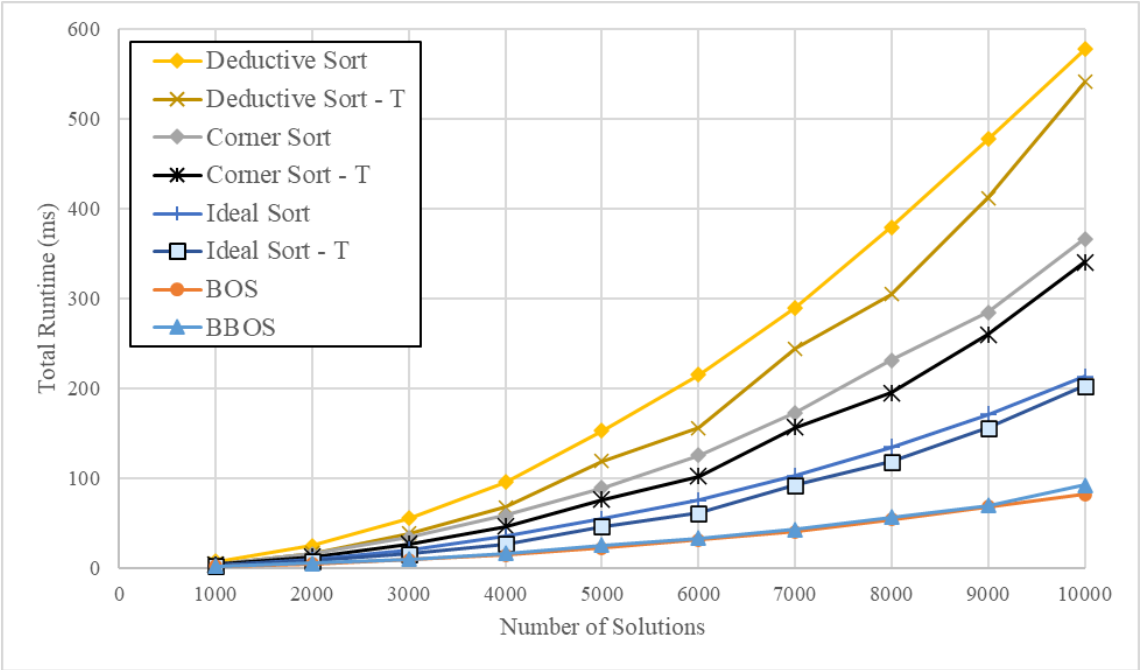


(c)

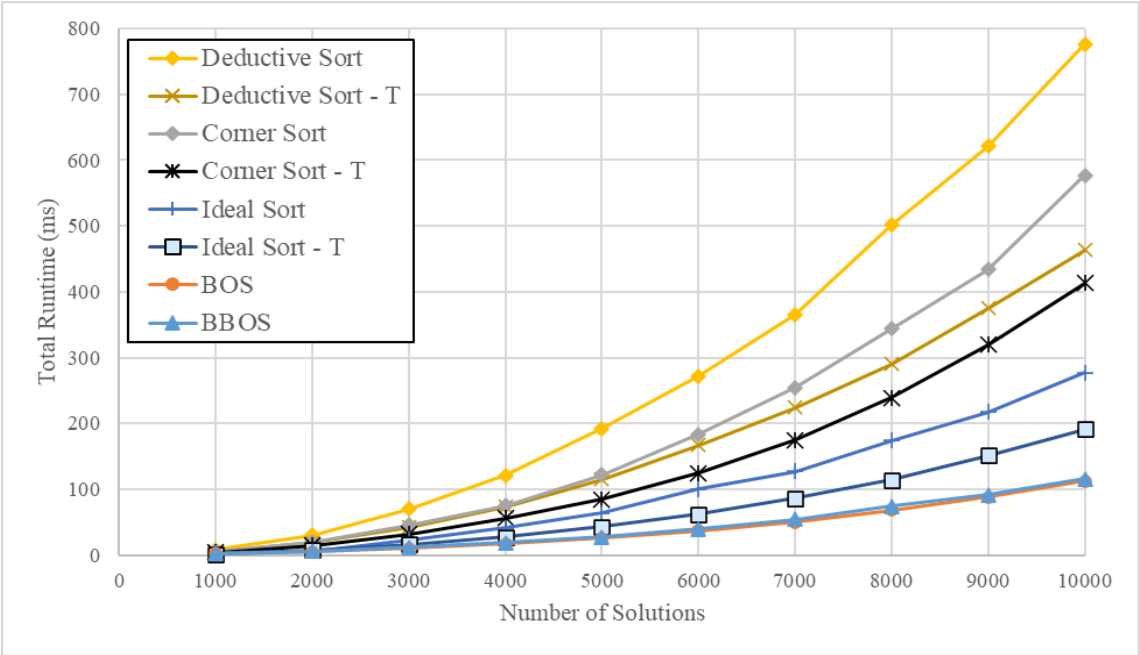


(d)

Figure 9. Zoomed region of required runtime for fixed front datasets with an incrementing number of fronts and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. (cont.)

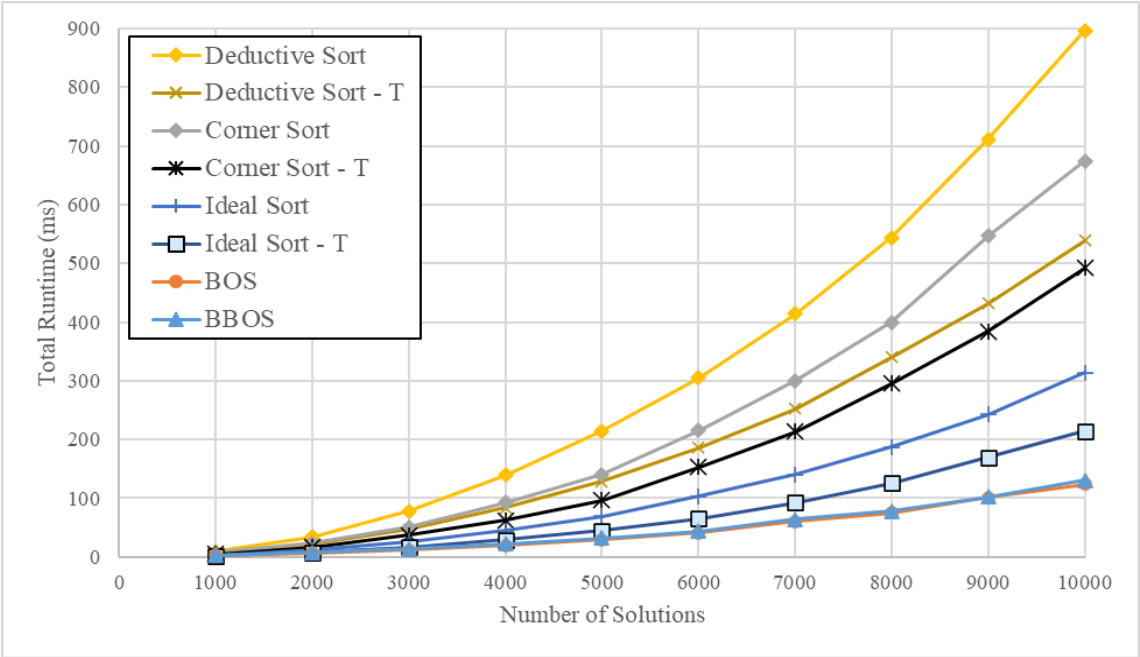


(a)

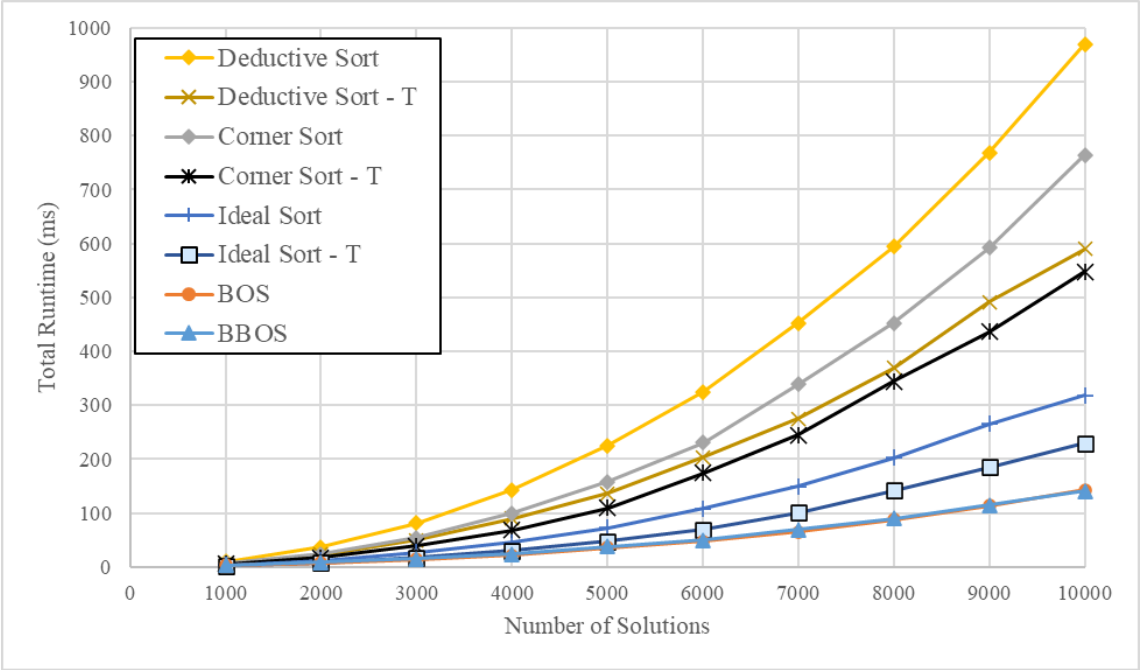


(b)

Figure 10. Computational performance for evolved datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives.

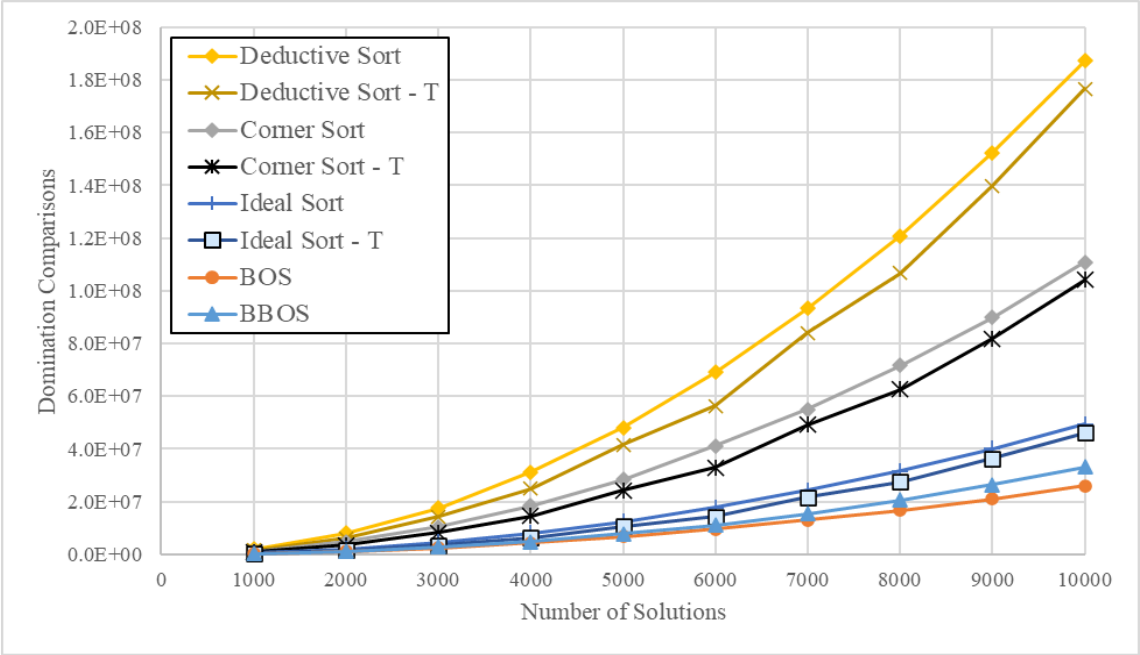


(c)

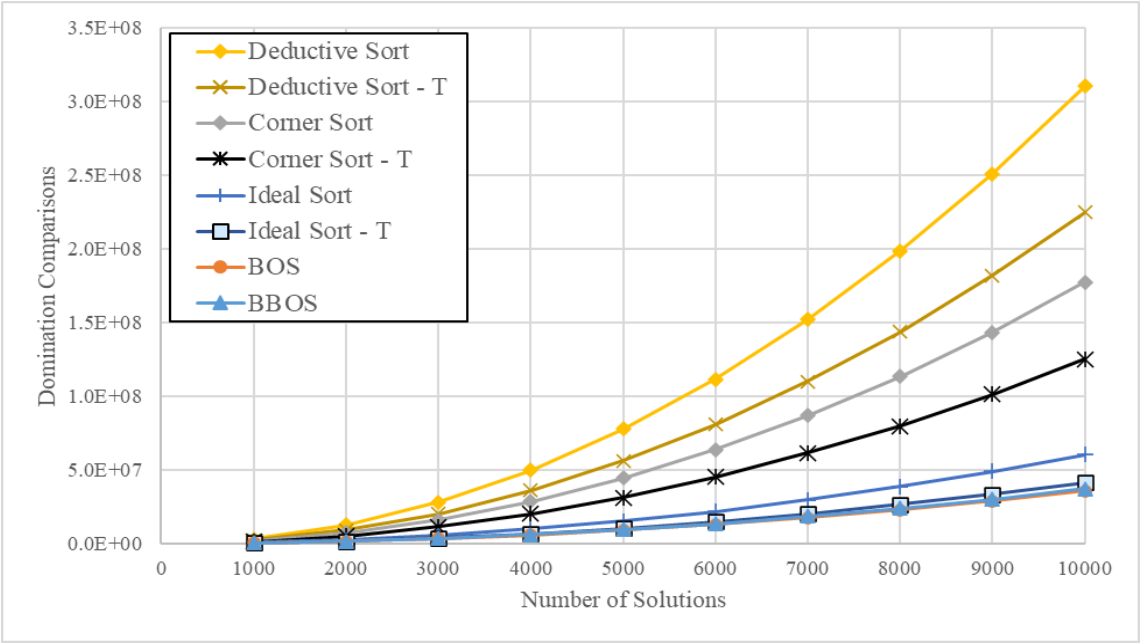


(d)

Figure 10. Computational performance for evolved datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

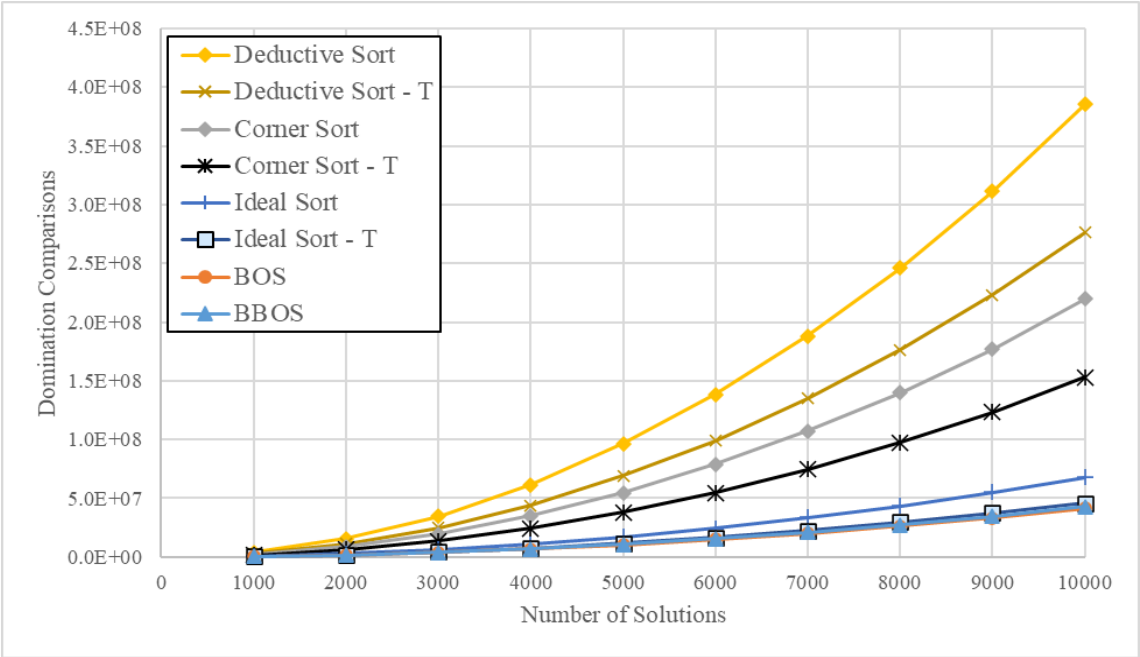


(e)

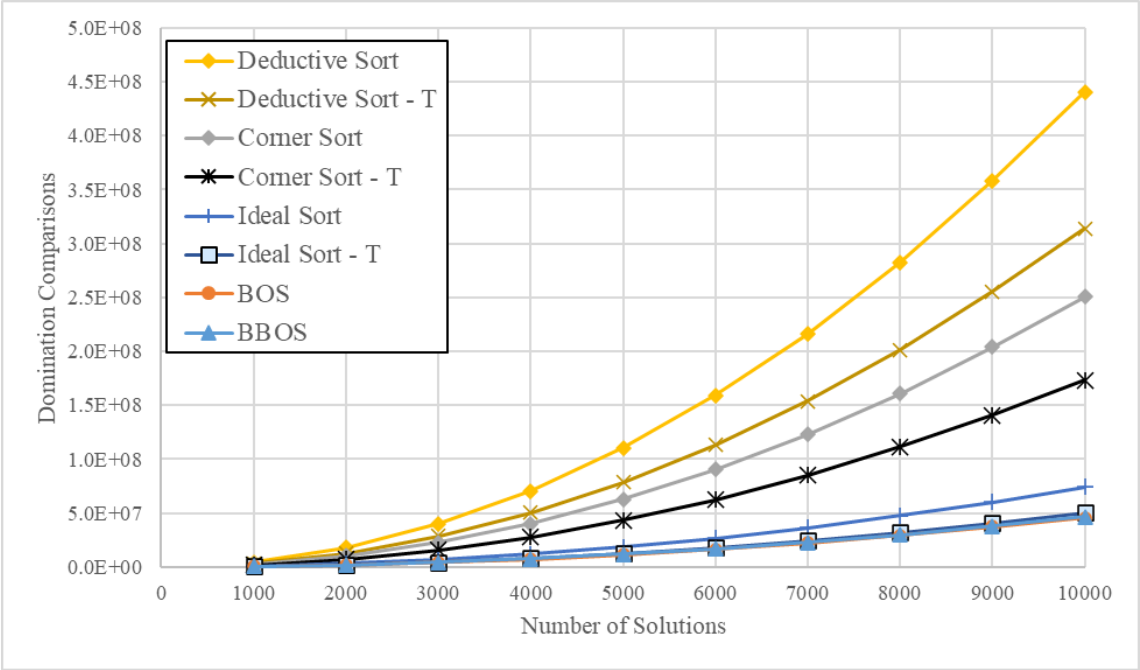


(f)

Figure 10. Computational performance for evolved datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)



(g)



(h)

Figure 10. Computational performance for evolved datasets with an incrementing number of solutions and a set number of objectives. a) Required runtime with 5 objectives. b) Required runtime with 10 objectives. c) Required runtime with 15 objectives. d) Required runtime with 20 objectives. e) Required number of dominance comparisons with 5 objectives. f) Required number of dominance comparisons with 10 objectives. g) Required number of dominance comparisons with 15 objectives. h) Required number of dominance comparisons with 20 objectives. (cont.)

7. DISCUSSION

The average total runtime (a) and the average number of dominance comparisons required (b) for each of the examined algorithms while sorting the cloud datasets with an ascending number of objectives is detailed in Figure 5. The figure first describes the two constructive front methods (BOS and BBOS) as the most efficient according to both metrics. Of the inferred dominance methods, Ideal Sort, and its terminable form (Ideal Sort – T), consistently outperform their counterparts with respect to runtime. Further, it is demonstrated that just the vanilla form of Ideal Sort is more efficient than even the terminable forms of Deductive Sort and Corner Sort (Deductive Sort – T and Corner Sort – T, respectively). Examining the number of dominance comparisons required, the inferred dominance methods are again secondary to their constructive front counterparts. Ideal Sort and Corner Sort grapple for the best efficiency, a contest that appears to be based on the number of objectives. While the Corner Sort algorithms sometimes require less comparisons, they are hampered by the complexity of repeatedly finding a corner solution [16], allowing the Ideal Sort algorithms to consistently outperform the former with respect to runtime. Additionally, note the tendency of terminability to lose its superiority over the vanilla methods as the number of objectives becomes large in cloud populations. This occurs as a greater proportion of the population becomes nondominated [16], reducing the number of fronts whose sorting can be avoided by early termination.

These findings are mirrored by the results of the two-axis parameter variation of cloud datasets (Figure 7). In relatively low dimensionality ((a), (b)), the Ideal Sort methods universally outperform their inferred dominance competitors for all population

sizes. Further, they are much more competitive here to the constructive front methods than in high-dimensional space. Terminable variations clearly outperform their associated vanilla algorithms at this level of low dimensionality.

As the number of objectives is increased, a growing gap emerges between the inferred dominance methods and the efficient constructive front methods. While Corner Sort again eclipses Ideal Sort in terms of required comparisons as the number of objectives becomes high, the latter maintains a consistent advantage in terms of algorithmic runtime. As expected, the benefits of terminability diminish as the number of objectives increases.

Performance measures corresponding to the tests on fixed front data sets are shown in Figure 8. As other studies have demonstrated [12], the number of required comparisons and runtime generally decrease, for a given population size and dimensionality, as the number of fronts increases. As the number of fronts becomes larger, the superiority of the constructive front methods over their inferred dominance counterparts additionally becomes less pronounced. The benefits of terminability are well-defined in this figure, as the terminable implementations outperform their vanilla variations across all instances. While the Corner Sort algorithms again jockey with their Ideal Sort counterparts for superiority in terms of the dominance comparisons required, Ideal Sort and Ideal Sort – T, as before, consistently require less runtime than Corner Sort and Corner Sort – T, respectively. The non-smooth descent of the terminable methods is an interesting behavior highlighted; a characteristic most clearly demonstrated by Deductive Sort – T in the higher dimension sets of Figure 8 ((c), (d)). This occurs as scenarios with an even number of equivalently sized fronts allows termination

immediately after the $F/2^{th}$ front is determined, where F is the number of fronts. When an odd number of fronts are present, the $F/2 + 1^{th}$ front must also be sorted. Populations with an even number of like-sized fronts thus demonstrate greater improvement when moving from vanilla to terminable implementations. While this behavior may not be obvious from the busier regions of Figure 8, this behavior is noted for each of the terminable algorithms examined.

Figure 9 provides a focused look at a busy region of the runtime figures of Figure 8. Eliminating the poorer performing inferred dominance methods, the best performing (Corner Sort – T, Ideal Sort, and Ideal Sort – T) are compared to the state-of-the-art constructive front method, BBOS. At each level of dimensionality, Ideal Sort – T is shown to outperform BBOS beyond a certain number of fronts. Further, Corner Sort – T is shown to outperform BBOS in many cases, though, itself, being outperformed by Ideal Sort – T in each case. The vanilla version of Ideal Sort is also shown to approach the runtime performance of BBOS at the very highest number of dimensions and fronts examined. These findings support the notion of Ideal Sort as a meaningful contributor to the arsenal of NDS algorithms. Perhaps more importantly, these findings cement the benefits of terminability, as an algorithm (Corner Sort), never before shown to outperform a member of the constructive front class, demonstrates superiority via the inclusion of a termination operation. Combining the aptitudes of Ideal Sort and terminability, Figure 9 defines perhaps the first instance of an inferred dominance method (Ideal Sort – T) regularly outperforming a state-of-the-art constructive front procedure.

The performance characteristics for each method, when applied to the described evolved datasets, are shown in Figure 6 and Figure 10. As with the cloud dataset

experiments, the two constructive front methods are shown to outperform the inferred dominance methods in each instance. Ideal Sort and Ideal Sort – T are shown to sizably outperform the other inferred dominance methods and exhibit only a slight disadvantage to BOS and BBOS in terms of the number of comparisons required. This disadvantage is manifested in the runtime requirements as well, but unlike in the cloud experiments, the Ideal Sort algorithms become more competitive as the number of objectives increases. This occurs as the evolved datasets do not create scenarios where most solutions are nondominated when the number of objectives is high.

In analyzing the results of these tests, attention should be paid to the significance of each test class. While cloud datasets are easy to implement and provide some feedback on NDS algorithm performance, their occurrence within true multiobjective optimization scenarios is limited. Specifically, they likely only occur at the onset of the optimization procedure when preliminary solutions are initialized using a random generation procedure [7]. Subsequent iterations are then likely characterized by populations with multiple fronts, more closely resembling the fixed front populations examined. As the multiobjective EA continues and reduces the number of Pareto fronts, a scenario akin to the evolved datasets examined is likely to exist until the algorithm is concluded [20]. Frequently outnumbering cloud dataset scenarios within the operations of a multiobjective EA, performance on fixed front and evolved datasets may be considered more important and demonstrative of an NDS algorithm's computational efficiency.

Conveniently for Ideal Sort, cloud scenarios are the only dataset class heralding the constructive front methods as far and away superior. Under the more critical and prevalent scenarios, Ideal Sort becomes much more competitive and can even exceed the

performance of the constructive front methods. These arguments have parallels to the argument for terminability, as well; the benefits of which subside in high-dimensional cloud space but are evident and meaningful while operating on the more critical dataset scenarios.

8. CONCLUSION

Improving the efficiency of NDS algorithms has been a topic of considerable interest since the introduction of the first NDS multiobjective EAs. Comprising a majority of the time required by an employing EA, NDS procedures boasting increased efficiency broaden the scale of problems that can be addressed using these popular evolutionary procedures. Herein was introduced a novel, terminable NDS algorithm, shown capable of competing with and outperforming other state-of-the-art NDS procedures. Further, this algorithm is easier to understand and implement than many state-of-the-art methods. Perhaps more importantly, the benefits of terminability were introduced and demonstrated. This concept has been shown to enable considerable efficiency improvements to NDS algorithms retaining this property. The utilization of terminability was even shown to grant Corner Sort superiority over BBOS in some instances, a feat not previously demonstrated by the vanilla Corner Sort algorithm. While the namesake algorithm of this investigation, Ideal Sort, does add a novel (and in some instances, computationally state-of-the-art) algorithm to the literature of NDS procedures, the discussion of terminability it enticed beseeches the integration of this property into the otherwise superior constructive front methods. While this integration may require a

reasonable exertion due to the structure of current constructive front algorithms, the computational benefits such an integration may induce could be well worth the effort.

REFERENCES

Fonseca, C. M., & Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Proc. International Conference of Gen. Alg.*, San Mateo, California, 1993, pp. 416-423.

K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: Wiley, 2001.

J.D. Schaffer, "Some experiments in machine learning using vector evaluated genetic algorithms", Ph.D. dissertation, Vanderbilt University, Nashville, TN, 1984.

J. Horn, N. Nafpliotis, and D. E. Goldberg, “A niched Pareto genetic algorithm for multiobjective optimization,” in *Proc. 1st IEEE Conf. Evol. Comput. World Congr. Comput. Intell.*, vol. 1. Orlando, FL, USA, Jun. 1994, pp. 82–87.

T. Murata and H. Ishibuchi, “MOGA: Multi-objective genetic algorithms,” in *Proc. IEEE Int. Conf. Evol. Comput.*, vol. 1. Perth, WA, Australia, Nov. 1995, pp. 289–294.

J. Knowles and D. Corne, “The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation,” in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1. Washington, DC, USA, 1999, p. 105.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002

E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization,” in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Problems* (EUROGEN), 2002, pp. 95–100.

M. L. Wong, “Parallel multi-objective evolutionary algorithms on graphics processing units,” in *Proc. 11th Annu. Conf. Companion Genet. Evol. Comput. Conf. Late Breaking Papers (GECCO)*, Montreal, QC, Canada, 2009, pp. 2515–2522.

K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints,” *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

P. C. Roy, M. M. Islam, and K. Deb, “Best order sort: A new algorithm to non-dominated sorting for evolutionary multi-objective optimization,” in *Proc. Genet. Evol. Comput. Conf. Companion (GECCO)*, Denver, CO, USA, 2016, pp. 1113–1120.

P. C. Roy, K. Deb, and M. M. Islam, “An efficient nondominated sorting algorithm for large number of fronts,” *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 859-869, March 2019.

N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Sep. 1994

S. Tang, Z. Cai, and J. Zheng, “A fast method of constructing the nondominated set: Arena’s principle,” in *Proc. 4th Int. Conf. Nat. Comput. (ICNC)*, vol. 1. Jinan, China, 2008, pp. 391–395.

K. McClymont and E. Keedwell, “Deductive sort and climbing sort: New methods for non-dominated sorting,” *Evol. Comput.*, vol. 20, no. 1, pp. 1–26, Mar. 2012.

H. Wang and X. Yao, “Corner sort for Pareto-based many-objective optimization,” *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 92–102, Jan. 2014.

X. Zhang, Y. Tian, R. Cheng, and Y. Jin, “An efficient approach to nondominated sorting for evolutionary multiobjective optimization,” *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, Apr. 2015.

S. Mishra, S. Mondal, S. Saha, and C.A.C. Coello, "Generalized best order sort algorithm for nondominated sorting," *Swarm and Evol. Comput.*, vol. 43, pp. 244-264, 2018.

J. W. J. Williams, “Algorithm 232 – heapsort,” *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, 1964.

Y. Tian, H. Wang, X. Zhang, and Y. Jin, "Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization," *Complex and Intell. Systems*, vol. 3, no. 4, pp. 246-263, 2017.

III. DISASTER RECOVERY STRATEGY GENERATION VIA MULTIOBJECTIVE HEURISTIC OPTIMIZATION

Samuel Vanfossan, Benjamin Kwasa, and Suzanna Long

Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, MO 65409

ABSTRACT

In the wake of extreme events, the response efforts conducted to restore an affected area comprise two strategy horizons. Initially, short-term emergency procedures are conducted, providing search and rescue facilities along with temporary relief and medical aid. These primary measures are followed by an extended restoration period during which the area's infrastructure must be reinstated, returning the area to operating capacity. It is of paramount importance that these second-stage restoration efforts be completed as quickly and affordably as possible; reestablishing the affected area's internal infrastructure and external connectivity while mitigating the financial burden these efforts assume. Determining the schedule of operations that must be conducted to achieve this restoration is a difficult task with substantial ramifications on these time and cost considerations. It is, therefore, critical that tools be developed to assist decision-makers in the discovery and selection of *optimal* recovery strategies. Herein a methodology is proposed leveraging agent-based simulation and multiobjective heuristic optimization to generate a set of Pareto efficient recovery strategies. This methodology provides decision-makers with an instrument to automatically generate well-performing solutions, enabling the expedient and cost-effective reinstatement of disaster-affected

areas. Through the improved restorative efforts facilitated by these allowances, the impact of extreme events can be effectively reduced.

Keywords: Disaster recovery scheduling, multiobjective optimization, evolutionary algorithm, genetic algorithm, agent-based modeling, resource constrained heuristic optimization

1. INTRODUCTION

Following a disaster event, the response efforts conducted to restore an affected area comprise two strategy horizons. Initially, short-term emergency procedures are conducted, providing search and rescue facilities along with temporary relief and medical aid. These primary measures are followed by an extended restoration period during which the area's infrastructure must be reinstated, returning the area to operating capacity (Ramachandran et al., 2015). While many efforts have sought to establish decision-making procedures that guide recovery agencies' short-term emergency procedures, the need for methods advising the extended restoration period remains (Holguín-Veras & Jaller, 2012; Hale & Moberg, 2005; Horner & Widener, 2011; Altay & Green, 2006; Galindo & Batta, 2013). These methods should seek to assist decision-makers in intelligently scheduling the activities required for restoration, enabling the re-entry of the affected area into the broader supply chain as expediently and affordably as possible.

Differences in the monetary and temporal expenses incurred to restore a disaster-affected area can have substantial and lasting ramifications on the communities impacted. A primary concern is the return of the affected area to the broader supply chain as

affordably as possible (Çağnan & Davidson, 2003; Galindo & Batta, 2013). Disaster events are shown to cause economic hardship both immediately after the event and well into the future (Ojha et al., 2018). It is therefore of great importance that the activities performed to restore the affected area be carried out in way that does not unduly compound these economic difficulties.

The direct monetary considerations of recovery efforts (labor, materials, fuel, etc.) must also be coupled with the time-sensitive nature of other economic concerns. While an area's infrastructure is nonfunctioning, similarly positioned businesses may not be able to operate. This not only results in lost income for the businesses themselves, but diminished tax revenues and tributary commerce for the encompassing municipality. Consider an extreme example, where an estimated $2.9 billion in gross regional product was lost during the two-month period following the Loma Prieta earthquake that hit the San Francisco area in October of 1989 (Brady & Perkins, 1991). Additionally, the restoration efforts individual businesses must assume often depend on the functionality of surrounding infrastructure and can extend beyond the horizon of public recovery (Masoomi et al., 2018). Consequently, an elongation of the timeline required for infrastructure restoration can prolong the period during which affected businesses are unable to operate.

Community outmigration can further exacerbate socioeconomic losses over time. This permanent population dislocation can stem from several factors related to the duration of time required to reestablish community operation: namely, infrastructure disruption and loss of employment (Masoomi et al., 2018). As these influences persuade residents and organizations to relocate from the affected community, the businesses

opting to remain, and the municipality itself, can be negatively affected by the loss of the outmigrant's usual revenue. In this way, the length of time required to reinstitute a disaster area's infrastructure supply chain can impact a community long after restoration efforts have been completed. While some studies demonstrate that the economic depressions immediately after a natural disaster can be followed by a flurry of economic activity during the restoration period, it's important to note the jobs created during this period may be transient. That is, they are largely based on contract labor and may move to a new location once recovery efforts have been completed (Jiménez Martinez et al., 2020).

While a strictly monetary impact may be derived for each of the previous considerations, some tribulations related to disaster restoration time have more abstract implications. Chief concerns among the literature are the relationships between disaster event trauma and post-traumatic stress disorder (PTSD) and depression (Steinglass & Gerrity, 1990; Madakasira & O'Brien, 1987; Neria et al., 2008; Qu et al., 2014, Houston, 2015). Capable of gravely impacting the lives of those they affect; the prevalence and severity of these disorders may be related to the speed at which restoration is completed. This relationship exists as symptoms may be triggered or worsened by reminders of a traumatic event, e.g., an affected area not yet returned to its pre-restoration functionality (Houston et al., 2015; Carlson and Ruzek, 2003). Just as an individual's perceived trauma may extend beyond the destructive event into the post-event chaos and recovery, there is evidence that the duration of traumatic exposure may impact the severity and commonality of PTSD symptoms (Neria et al., 2008; Adams, 2014; Houston et al., 2015). Stress and depression may also manifest from disaster-induced factors such as reduced

business and/or unemployment during infrastructure recovery and the subsequent personal/organizational rebuilding period (Qu et al., 2014). Intensifying these concerns is the diminished availability of many support resources prior to restoration achievement (Houston et al., 2015). While research has shown the importance of early and frequent counselling following a major trauma or stressor, these services may not be widely available until a state of normalcy is resumed (Torres-Mendoza et al., 2021). Previous studies further acknowledge the need for long-term (multi-year) community mental health monitoring, assessment, outreach, and services following major disasters (Houston et al., 2015). It may then be beneficial, with respect to community health and economics, to take steps to mitigate stress and anxiety to the extent, and with the rapidity, possible. Expedient restoration of an impacted area's damaged infrastructure is one critical step in this mitigation process.

Those affected by a disaster event are also subject to many other impacts such as time lost in school or the ability to participate in social and extracurricular activities. Incalculable losses may be subsumed in the form of missed opportunities and experiences for individuals instead focusing time and resources on rebuilding their damaged ecosystems. As the true economic impact of many of these time-sensitive disaster effects may be difficult to quantify (and, in many cases, can be of incomparable value), it is important that both economic and temporal considerations be made in developing recovery strategies for disaster-affected areas. Further, efforts to place a monetary value on ramifications transcending economic consideration inherently introduce bias and distort the problem to be addressed. This linearization of time and cost also disallows associated tradeoff analysis which may be quite valuable to decision-makers when

comparing recovery strategies. Considering the crucial objectives of restoration time and cost independently may, therefore, be more appropriate, informative, and dispassionate.

In the following section, some fundamental observations about disaster recovery are first discussed. Current methods addressing disaster recovery are then critically analyzed and the characteristics of a desirable strategy generation model are determined and disclosed. Section 3 details the workings of a proposed model, demonstrating the desireabilities previously presented. In the subsequent section, a simulated disaster region is established and presented to the model, prompting the model's production of suggested recovery strategies. The recommended strategies of the model are then shared and discussed, comparing the performance of those produced with that of intuitive alternatives. A final section makes concluding remarks and offers some suggestions for future work.

2. OBSERVATIONS, CURRENT METHODS, AND THE DESIRED MODEL

2.1. PRELIMINARY DISASTER RECOVERY OBSERVATIONS

To facilitate a critical analysis of current recovery literature and methods, some observations about post-disaster scenarios are first presented. Foundational is the mechanism through which recovery is completed. After short-term emergency procedures are completed, teams of restoration professionals are called to the affected area to begin repairing the damaged infrastructure (Ramachandran, 2015). These *vehicles of restoration* include teams of varyingly skilled laborers —such as electricians, linemen, machine operators, plumbers, and engineers, to name a few— and the tools, resources,

and literal vehicles they need to complete their restoration tasks. Teams responsible for reinstating roadway infrastructure, for example, may include physical laborers and equipment operators, bulldozers and other debris-clearing vehicles, and the heavy equipment used in the repair of damaged roads and bridges. Teams dedicated to electrical repair may include electricians and linemen, transport vehicles loaded with replacement electrical components, and the specialty vehicles needed to access and repair overhead and underground electrical elements. The contents of these inexhaustive lists and other required assets move through physical space and often themselves rely on infrastructure elements to perform repairs. They can be blocked physically by debris and other impediments and may not be able to conduct operations in damaged locations while required infrastructures are unavailable. These vehicles perform their operations following a schedule of repair activities that should be organized to restore the impacted area as quickly and affordably as possible. Creating this schedule of repairs, however, is mired by a few complexities that can substantially impact these two objectives.

Consider the scene of Figure 1, depicting the types of damage frequently caused by natural disasters events (Myint et al., 2008). Therein a damaged storefront sits along a street littered with downed trees and foreign debris. Mangled electrical and communication lines cross the roadway, close to the site where an impacted hydrant spews water. Damage scenarios of the kind commonly arise following disaster events and can be thought of as a complex overlay of damaged infrastructure systems. For example, this scene may be thought of as a location in space simultaneously occupied by respective portions of larger transportation, electrical, communications, and water systems. This spatial overlay makes reasonable the assumption that *precedence* considerations be made

when attempting to restore the damaged infrastructure. An intuitive example of this is the need for the roadway to be cleared before the appropriate equipment can be brought in to repair the downed electrical and communication lines. The existence of this, and other precedence relationships, means the amount of time required to restore the damaged infrastructure shown in the scene is not trivial to determine. For instance, the total time cannot be assumed to be the mere maximum of the times required to restore individual infrastructure systems. If the maximum amount of restoration time needed for any one system is owned by the electrical infrastructure, the total time needed will be no less than that required for the roadway clearing plus that required for electrical repair. Estimating the total time required as the sum of that necessitated by each infrastructure is also not veracious as some systems may be repaired concurrently. For example, it may be reasonable that the electrical and water infrastructure can be simultaneously restored once the roadways have been reinstated.

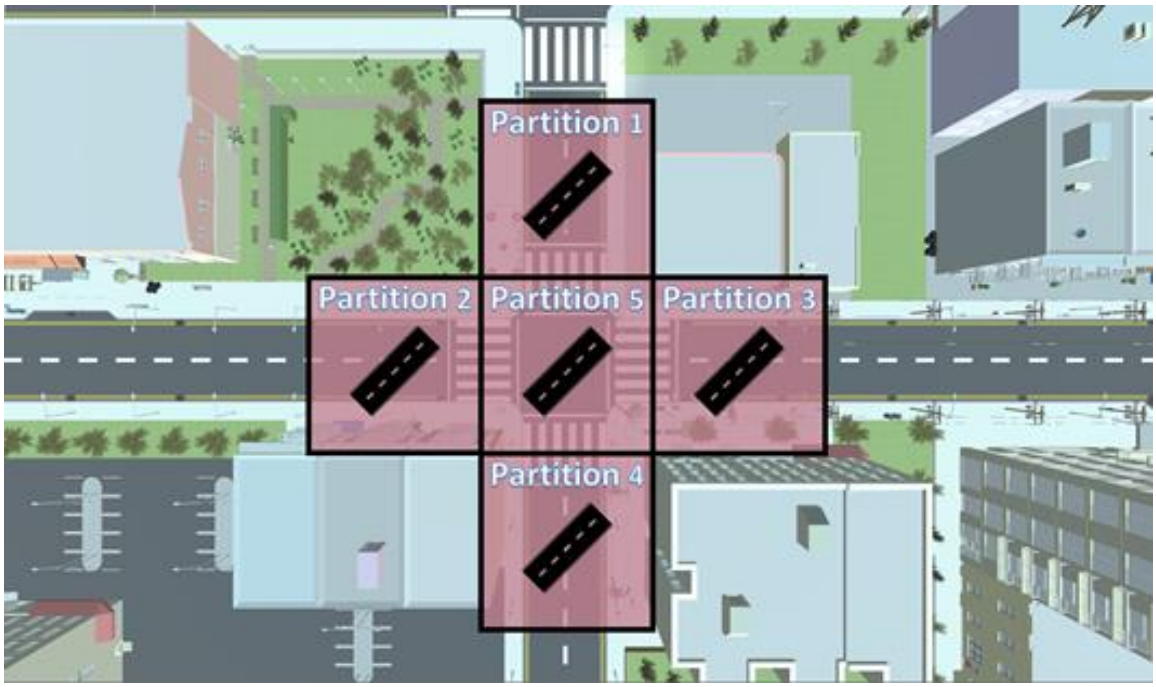
The cost required to complete these restorations is similarly complicated to determine as repair efforts will undoubtedly incur both time-dependent (*variable*) and *fixed* expenses (Ojha, 2019). Differences in the lengths of time needed to complete repairs, for example, require the employment of restoration vehicles for varying time horizons. These variations correspond to disparate expenditures for things such as labor wages, fuel and hospitality costs, and resource storage expenses, to name a few (Ojha et al., 2021). The length of time each utilized resource is employed, then, impacts the overall restoration cost objective. Fixed costs associated with employing a resource similarly factor into the total restoration cost. Commonly as some form of hiring, acquisition, or other overhead expense, a fixed cost may accompany each resource

employed and is not related to the length of time the resource is utilized (Ojha, 2019). When more resources are employed, the restoration may be completed sooner; however, more fixed costs are assumed. Thus, the amount of resources utilized similarly impacts both restoration time and cost.

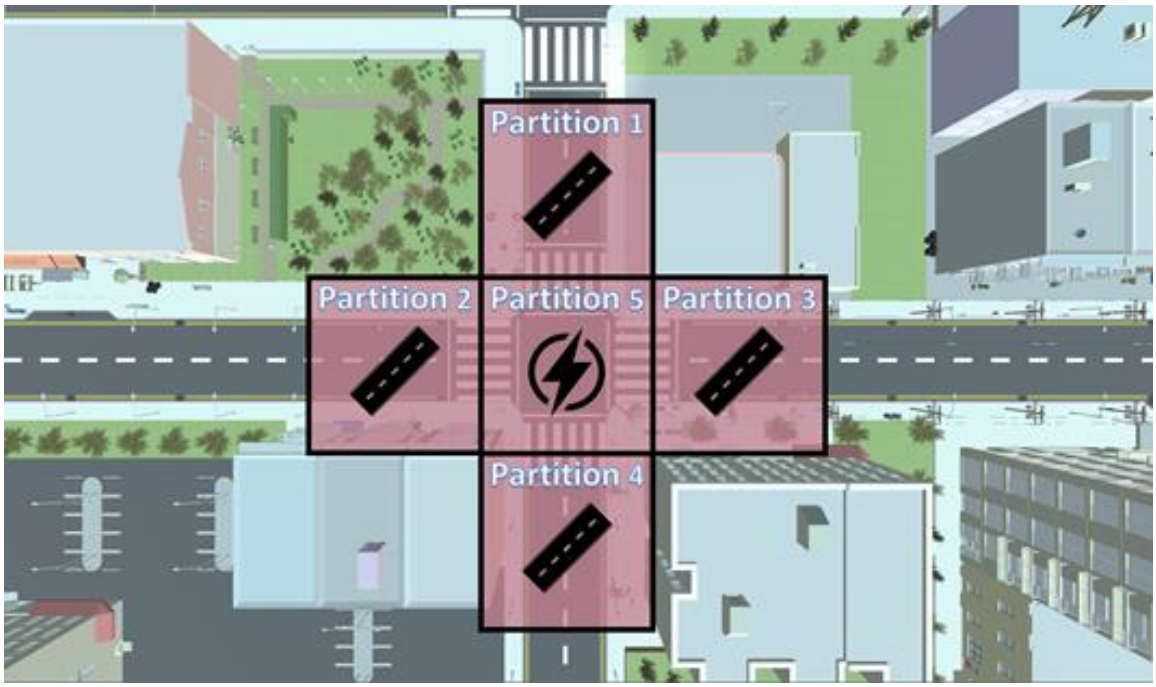
Figure 1. Simulated disaster scene with damage to multiple critical infrastructure systems.

Another factor that needs to be explored is the accessibility of damaged infrastructure with respect to space and time. Consider the two simplified post-disaster scenarios represented in Figure 2; each subfigure depicts a partitioned network with damage indicated by a red tinting of the partition and a symbol indicating the affected infrastructure element. In each case, the impacted partitions are surrounded by a functional region to which no damage has occurred, from which the vehicles of restoration enter to make repairs. The first scenario, Figure 2 (a), includes five partitions

requiring restoration to associated roadway network sections. This damage is nearly mirrored in Figure 2 (b), the only difference being the type of damage to the central partition (Partition 5), now to the electrical infrastructure instead of the roadway. As access to Partition 5 from the functional surrounding is denied by damage to the roadways of Partitions 1-4 (in both subfigures), efforts to repair the affected infrastructure of the central partition may be temporarily delayed. Access to this central partition is eventually enabled by a restoration of one of the other damaged partitions. Figure 2 (b) reveals that even if the roadway infrastructure for Partition 5 is not damaged, efforts to restore the partition's impacted electrical systems are similarly dependent on the restoration proceedings of the neighboring roadway partitions. Therefore, the current feasibilities of a location's required restoration efforts are not only dependent on the damage to the specific location, but also on that sustained by surrounding locations. In short, a damaged partition may have to wait until another damaged partition is restored to begin receiving its own infrastructure repair. Recounting the scenario of Figure 2 (b), an additional tier of complexity is realized as the feasibility of repairing Partition 5's electrical infrastructure is dependent on prerequisite repairs that are not only in another partition but are also to a completely different infrastructure system. These insights emphasize the spatial, temporal, and interconnective dependencies of restoration efforts and the necessity of accounting for them when scheduling post-disaster recovery efforts.



(a)



(b)

Figure 2. Simplified disaster scenarios with partitioned infrastructure regions. a) Disaster scenario with five infrastructure partitions, each exhibiting damage to the region's roadway infrastructure. b) Disaster scenario with five infrastructure partitions, four exhibiting damage to the region's roadway infrastructure and one with damage to the region's electrical infrastructure.

The order in which repairs occur can also impact the total time and cost required to restore a disaster-affected area. This is demonstrated even in the simple example of Figure 3, where damage has befallen two partitions of a community's infrastructure. Partition 1 has experienced damage to both its roadway and electrical infrastructure systems while only the roadway of Partition 2 is impacted. Each instance of damage is also known to require one day to be restored. Both partitions may be accessed immediately to begin repairs but recall that a partition's roadway infrastructure should first be functional before other systems may be repaired. Finally assume that the example community only has the resources to repair one roadway partition at a time. If the roadway of Partition 2 is restored first, then three days will be required to complete all repairs. If, however, the damage to Partition 1's roadway infrastructure is restored first, the entire area can be reinstated in two days. In the latter scenario, damage to Partition 1's electrical infrastructure and Partition 2's roadway system can be simultaneously repaired during the second day. These schedules are summarized in Table 1.

The circumstances of different overlaid-system scenarios will undoubtedly necessitate the making of differing precedence and feasibility rules. The presence of these relationships can yield situations where identical sets of repairs may require different amounts of time to complete depending on the order in which they are slated. The simple examples disclosed demonstrate the need for any model to make considerations of the kind when scheduling and optimizing the processes of disaster-affected infrastructure recovery.

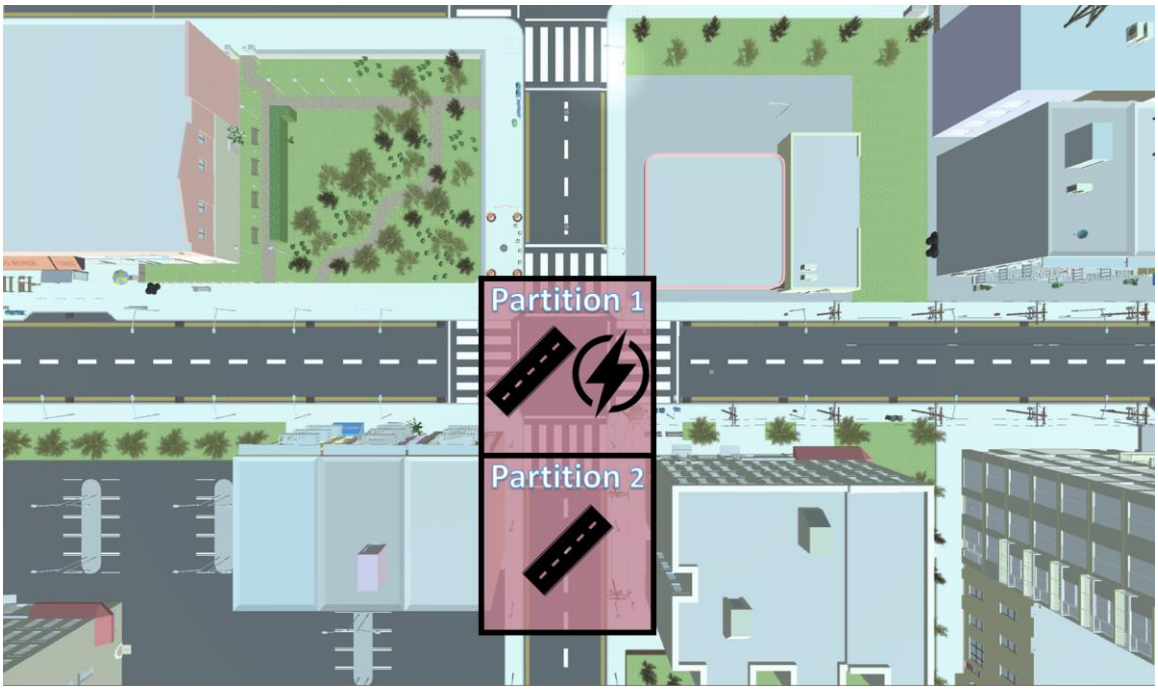


Figure 3. Disaster scenario with a partition exhibiting damage to multiple infrastructure systems (roadway and electrical).

Table 1. Alternative restoration schedules and required restoration time for damage exhibited in Figure 3.

Schedule	**Partition 1 Roadway First**	**Partition 2 Roadway First**
Day 1 Repairs	Partition 1 Roadway	Partition 2 Roadway
Day 2 Repairs	Partition 2 Roadway, Partition 1 Electrical	Partition 1 Roadway
Day 3 Repairs	-	Partition 1 Electrical
Total Time Required	2 Days	3 Days

2.2. CURRENT METHODS ADDRESSING DISASTER RECOVERY

Several general mechanisms have been utilized by the limited literature addressing post-disaster recovery planning. In many approaches, restoration curves derived from historical disaster events are used to estimate restoration times (Applied Technology Council, 1992; Chang, Seligson, and Eguchi, 1996; Nojima et al., 2001). While these may provide some insight and a reasonable expectation of total recovery time, they do little to inform decision-makers regarding recovery task scheduling and cannot be used to optimize operations. Further, these models do not account for the spatial and temporal variability existent between different disaster locations and scenarios (Çağnan & Davidson, 2003). Alternatively, some studies have sought to estimate the total restoration costs required to reinstate disaster-damaged infrastructure systems (Ojha, 2019; Ojha et al., 2021). While these are useful in providing some estimation of expected restoration costs, they are not geared toward the optimization of restoration operations or minimizing the costs incurred.

Numerous models have proposed the use of a resource constraint model, incorporating available resources and the amount of damage incurred to determine restoration time (Isumi, Nomura, and Shibuya, 1985; Balantyne et al., 1990; Chang et al., 2000). While an appropriate choice for modeling and optimization, these employments have typically focused on a single type of infrastructure, not incorporating other infrastructure systems and their interdependencies. Moreover, reasonable effort may be needed to construct these formal optimization models.

Linked systems are examined in some cases, seeking to incorporate the relationships between infrastructure systems and their effects on restoration time. A

linked system approach is endorsed by Zhang (1992), utilizing Markov chains to model infrastructure restoration proceedings. Other mathematical models using a variety of techniques have also been developed to optimize recovery efforts (Wang et al., 2005; Minas, Simpson, and Tacheva, 2020). However, these models often lack the specificity to provide a detailed schedule of restoration activities. Additionally, they can be highly scenario- and disaster-specific and require extensive efforts to formulate. A more robust procedure capable of providing a granular schedule of recovery operations while being situationally flexible and easy to utilize would be much more valuable.

Ramachandran et al. (2015) created a framework in which publicly available geospatial data was used to formulate a combinatorial graph of linked infrastructure components. Incorporating an infrastructure component precedence scheme, the expected recovery time for a given area was estimated using damage totals and the critical path method (CPM). This method is admirable in its ability to discern the required information from readily available data sources and its reconciliation of multiple infrastructure components.

However, the approach may be improved by incorporating spatial and precedence considerations into its determination of restoration time. Consider the scenario of Figure 4 where a disaster has impacted the infrastructure systems of two partitions. Note that Partition 2 requires repairs to both the roadway and water systems within its boundaries. Partition 1 blocks immediate admittance to its neighbor as the roadway infrastructure of the former first needs to be restored. Assuming one day is required to restore each damaged infrastructure system within each node, a naïve CPM assessment with no precedence considerations may conclude that only two days are needed to recover the

disclosed partitions. Alternatively, consider the assessment made while adhering to spatial and precedence relationships. Just as discussed in Section 2.1, a reasonable assumption is that the damaged roadway of an impacted partition needs to first be reinstated before the appropriate vehicles can be brought in to repair other infrastructure(s). The intricacies of the scenario thus dictate that three days are needed to repair the damaged infrastructures: one day to repair the roadway of Partition 1, one day to repair the roadway of Partition 2, and a final day to restore Partition 2's water infrastructure.

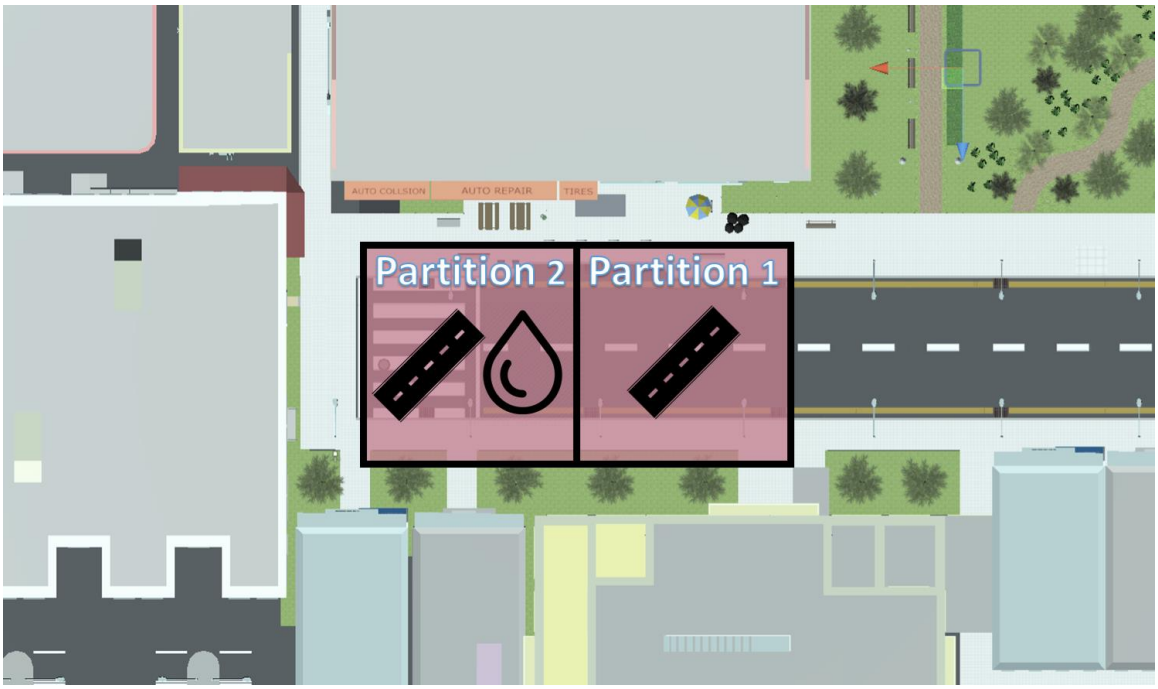


Figure 4. Disaster scenario with a partition exhibiting damage to multiple infrastructure systems (roadway and water).

Without knowledge of the precedence and spatial relationships between the damaged partitions, the real time required to restore this network cannot be reliably

determined. While Ramachandran et al. (2015) makes concessions for intricacies of this kind by incorporating lags into their CPM formulation, the estimated delays they utilize cannot adequately capture the complex interactive and scenario-dependent variabilities of coupled infrastructure systems. Further, the methodology proposed by Ramachandran et al. (2015) does not culminate in the creation of a schedule of activities needed for restoration; instead, an estimate for the total time required for recovery is precipitated.

Masoomi and van de Lindt (2017) proposed a system wherein an area's infrastructure was modeled as a series of nodes and de facto arcs, overlaid on a grid-like division of the region. In addition to the basic infrastructure elements of power and water systems, their investigation included aggregate residential and business building data, and the location of schools. Spatial considerations were included in their modeling efforts, allowing this information to be used in the determination of realistic recovery strategies. The approach employed determined a priority level for the damaged components following a simulated disaster. Addressing the highest priority component first, the shortest path along the spatial infrastructure network enabling this component's restoration is slated for completion. This is then repeated with the next highest priority component until the area is completely restored. While this allows the determination of a relatively utilitarian restoration schedule, a minimum total restoration time is not sought or generally achieved.

Masoomi and van de Lindt (2017) represents one of the most complete strategies within the literature for the creation of informed recovery schedules. This model, however, does not consider the expense associated with different recovery strategies, focusing instead on time required, exclusively. Additionally, restoration times are

calculated using fixed (or in some cases, infinite) resource amounts; a convention that does not translate well to real-world restoration scenarios where resources are both limited and adjustable (Almoghathawi et al., 2019). Perhaps the biggest shortcoming of this methodology is the careful data gathering and modeling required to utilize the tool. This represents a barrier to its use by municipalities and administrations not equipped with the erudite knowledge needed for implementation and utilization.

While the examined models provide some utility and demonstrate many valuable characteristics, the need for more generalizable, informative, and considerate models remains. This assessment of the current literature is supported by many survey works, citing the deficit of disaster recovery methods in comparison to the wealth of work addressing emergency response tactics and disaster preparedness and mitigation stratagems (Altay & Green, 2006; Wright, Liberatore, & Nydick, 2006; Lettieri, Masella, & Radaelli, 2009; Simpson & Hancock, 2009; Richey et al., 2009; Galindo & Batta, 2013).

2.3. THE DESIRED MODEL

Examining the features and capabilities of models within the literature, the characteristics of a desired model are discerned. A satisfactory model shall:

- Require only readily available data without the need for significant manual collection or processing
- Incorporate multiple critical infrastructure component systems into a single combinatorial methodology

- Consider both temporal and monetary ramifications in recommending recovery strategies
- Maintain the ability to scale and generalize to a variety of situations and scenarios, namely:
    - Differing infrastructure component systems
    - Differing regions and disaster types
    - Differing area and granularity requirements
- Enable the incorporation of precedence and spatial considerations in developing a recovery schedule
- Culminate in a granular schedule of restoration activities to be performed, along with the expected costs and times required
- Automatically perform the processes necessary to translate from readily available data to actionable recommendations, allowing utilization by non-erudite operators
- Allow multiple runs to be conducted easily, enabling the consideration of multiple scenarios

3. AN EVOLUTIONARY METHODOLOGY

3.1. REASONING ABOUT INFRASTRUCTURE SYSTEMS

To allow the determination of well-performing restoration schedules, a means of reasoning about infrastructure systems systematically is first required. Per the observations of the previous section, the mechanism selected should allow for the consideration of spatial and temporal characteristics, along with the discernment and

utilization of precedence information. The selection of a mechanism that enables objective and standardized performance evaluation is also desirable to easily compare competing schedules. Combinatorial graphs, such as those used by Ramachandran et al. (2015) and Masoomi and van de Lindt (2017), are a great structure that can easily be made to meet these criteria.

While any number of representation schemes may be used to encode the required information within graph structures, the approach utilized by the proposed method is here presented. Prerequisite to this discussion is a very basic understanding of graph theory; wherein a graph is described as a set of abstractions, known as vertices (or *nodes*), sharing some set of relationships as defined by associated *arcs*. A simple graph with four nodes and five arcs is shown in Figure 5. In the proposed methodology, nodes will be representative of partitions to a disaster-affected area such as those made in Figure 2, Figure 3, and Figure 4. Arcs then describe the connection of these infrastructure systems between partitions. For example, if a roadway travels from one partition to another, an arc will connect the nodes associated with these partitions. Partitions not containing the selected infrastructure are not assigned a node and are not connected by any arcs. As illustration, the roadway infrastructure system of Figure 2 (a) is translated from partitioned map (Figure 6 (a)) into network graph (Figure 6 (b)), below. Damage to partitions can then be described in differences between the normal and post-disaster network graphs. Note the difference between Figure 6 (b) and Figure 6 (c), detailing the roadway infrastructure before and after damage to Partition 3, respectively.

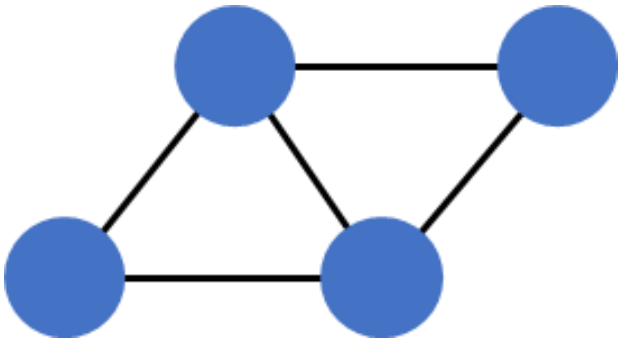
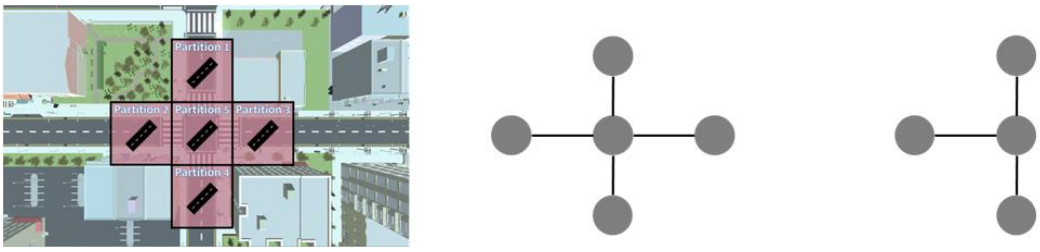


Figure 5. Simple network graph with four nodes and five arcs.



(a) (b) (c)

Figure 6. Disaster scenario with five roadway partitions translated to a network graph. a) Five partitions of a region's roadway infrastructure system. b) A network graph of the partitioned infrastructure system of Figure 6 (a). c) A network graph of the partitioned infrastructure system of Figure 6 (a) with damage to Partition 3.

Individual graphs can then be created to describe the status of different infrastructure systems. Using a common partition strategy, the overlay of infrastructures within partition boundaries is easily represented. The adoption of an informative labeling convention allows for the straight-forward and systematic integration and utilization of spatial relationships. This approach uses a Cartesian approach to define a partition's (and thus, a node's) relational geographic location. The sample region of Figure 2 is encoded using this scheme for four infrastructure systems (roadway (a), electric (b), water (c), and communications (d)) in Figure 7. While the first two coordinate of each node describe

their location within the partitioned region, the third coordinate indexes the type of infrastructure the node represents. These individual infrastructure graphs are additionally combined along this third coordinate axis to enable the standardized encoding of precedence information between different infrastructure systems. For example, in Figure 7, an edge placed between the central node of the roadway graph (*Node (1, 1, 0)*) and that of the electric graph (*Node (1, 1, 1)*) can be used to describe the previously discussed precedence relationship between these two system partitions. Here, *Node (1, 1, 0)* is a *precedence node* to *Node (1, 1, 1)*. This strategy of translating an area's overlayed infrastructure systems into a single combinatorial graph is critical as it allows the use of established network optimization algorithms in reasoning about and evaluating these connected systems.

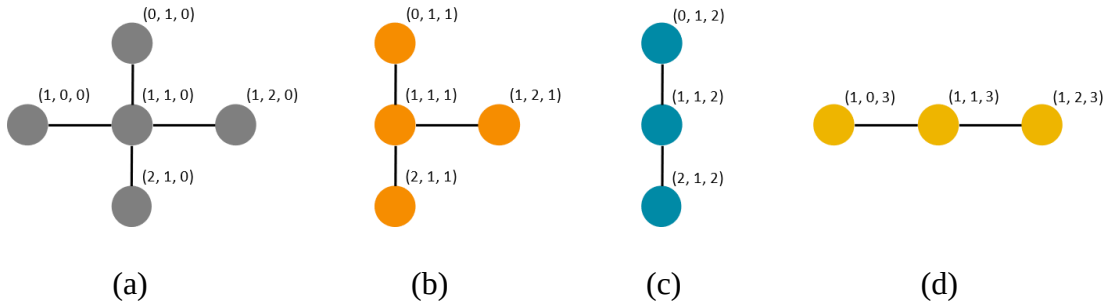


Figure 7. Example network diagrams for four infrastructure systems. a) Roadway infrastructure network diagram. b) Electrical infrastructure network diagram. c) Water infrastructure network diagram. d) Communications infrastructure network diagram.

3.2. AUTOMATED NETWORK GRAPH FORMULATION

Creating these networks graphs manually, however, would be quite tedious and time consuming. Fortunately, the standardized nature of these combinatorial representations allows them to be constructed and updated automatically. While it is safe

to assume that many municipalities are equipped with graphical information systems that would make easy work of such a task, a barebones approach to creating these graphs from minimal information is next presented. This mechanism can be used in the worst-case scenario when all that is possessed is a rudimentary map of a region's infrastructure systems. Recall that these maps should generally be available as Ramachandran et al. (2015) demonstrated the ability to model a region's critical infrastructure systems from publicly available data sources.

Using Python's OpenCV computer vision library (Bradski, 2000), a selected infrastructure map is first divided into a specified number of partitions. In Figure 8 (b) the sample infrastructure map of the Figure 8 (a) is divided into 6 equally sized partitions. Counting the number of pixels matching the infrastructure defining color (in Figure 8, a nice blue), it is determined which partitions contain elements of the selected infrastructure. Those partitions with a positive pixel count are then assigned a node via Python's networkx package for graphical network modeling (Hagberg et al., 2008). Partitions with a pixel count of zero do not contain the selected infrastructure and are not assigned a node. These pixel counts can also be used to describe the *prevalence* of an infrastructure system within a partition. Partitions with a greater prevalence for a particular infrastructure system contain more elements of that system than partitions with lower associated prevalence values. Here, *elements* is a general term and may describe things like yards of wire, feet of pipe, or square feet of pavement area, dependent on the associated infrastructure system. In Figure 8 (b), only the top-right partition fails to receive a node. Using the Cartesian labeling convention discussed in the previous subsection, spatial relationships are preserved and discernable. Arcs connecting these

nodes are next needed to complete the network graph. OpenCV is again used to automate this process. Here, the boundaries of each partition are examined; that is, the outermost pixel layer of each of the partition rectangles. In Figure 8 (b), the boundaries of the two right-most partitions are highlighted yellow. The characteristics of adjoining boundary sections are then compared, exemplified for two partitions by the two red highlighted boundary lines in Figure 8 (b). If each of these boundary sections contains a pixel representative of the selected infrastructure, then the two nodes associated with these partitions are joined by an arc. With respect to this example, *Node (0, 0)* and *Node (0, 1)* are connected in Figure 8 (c). Completing this analysis for all adjoining boundary sections, the complete network graph of the infrastructure system is produced. This procedure can be applied to any infrastructure system of interest and is scalable to regions of any size.

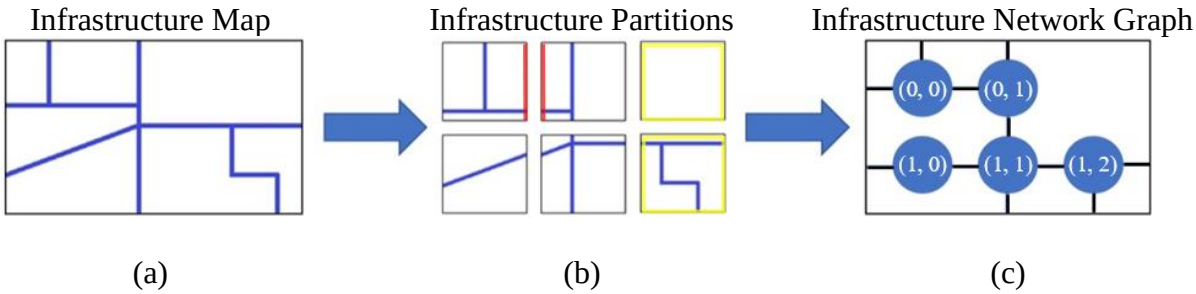
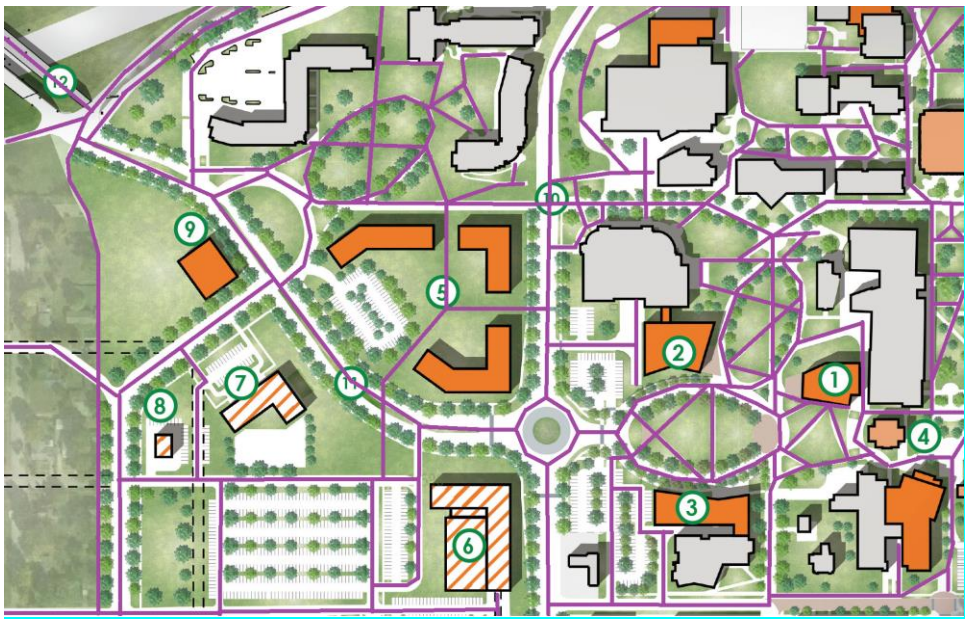


Figure 8. Translation of a sample infrastructure map into a representative infrastructure network graph. a) Sample infrastructure map. b) Sample infrastructure map divided into 6 infrastructure partitions. c) Network graph of sample infrastructure map.

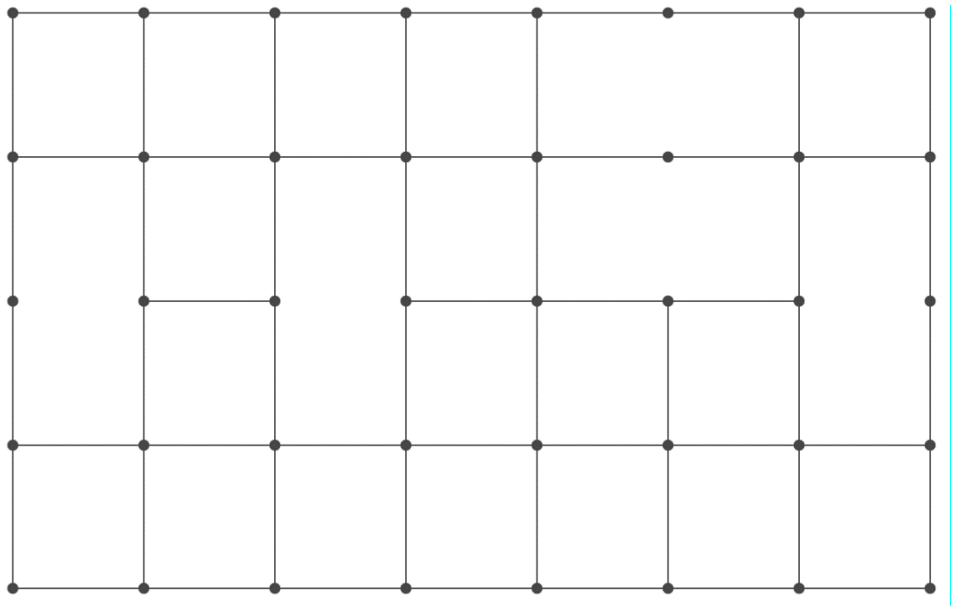
While this network generation approach may seem rudimentary, its granularity and veracity to actual infrastructure maps can become quite credible as the number of partitions increases. Note how the network representation of a region's sidewalk system

(Missouri University of Science and Technology, 2022) becomes more analogous to the actual map (Figure 9 (a)) as the number of partitions increases (Figure 9 (b – e)). When determining the number of partitions to utilize, decision-makers should consider the sophistication of the network to be represented, the amount of effort required to determine damage at varying granularities, and the increased algorithmic time needed to reason about networks with a greater number of nodes.

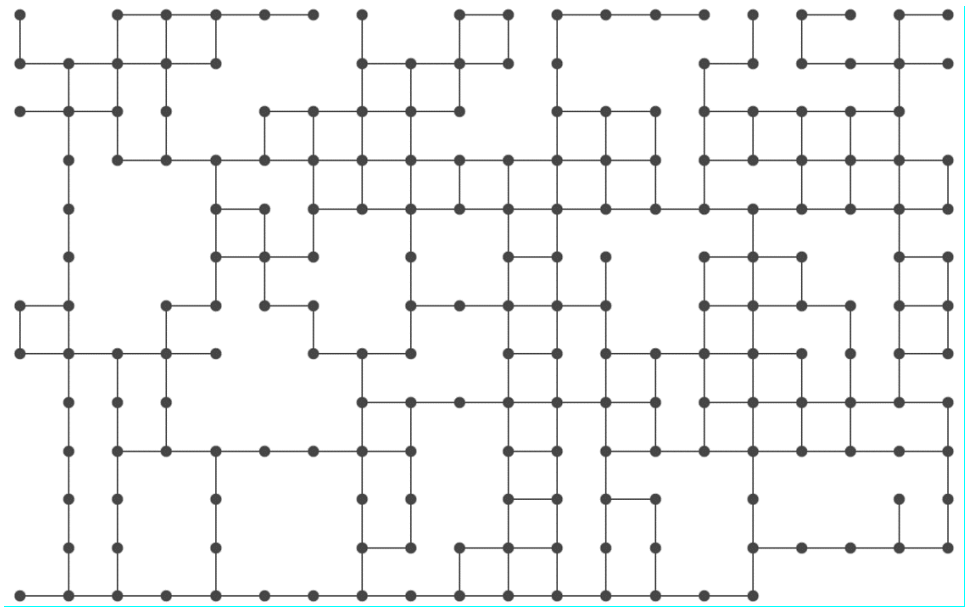


(a)

Figure 9. Translation of a region's sidewalk system into representative network graphs at varying granularities. a) Sidewalk infrastructure map of a selected region. b) Automatically generated network graph of sidewalk infrastructure with a 5-by-8 partition granularity. c) Automatically generated network graph of sidewalk infrastructure with a 13-by-20 partition granularity. d) Automatically generated network graph of sidewalk infrastructure with a 25-by-39 partition granularity. e) Automatically generated network graph of sidewalk infrastructure with a 50-by-78 partition granularity.

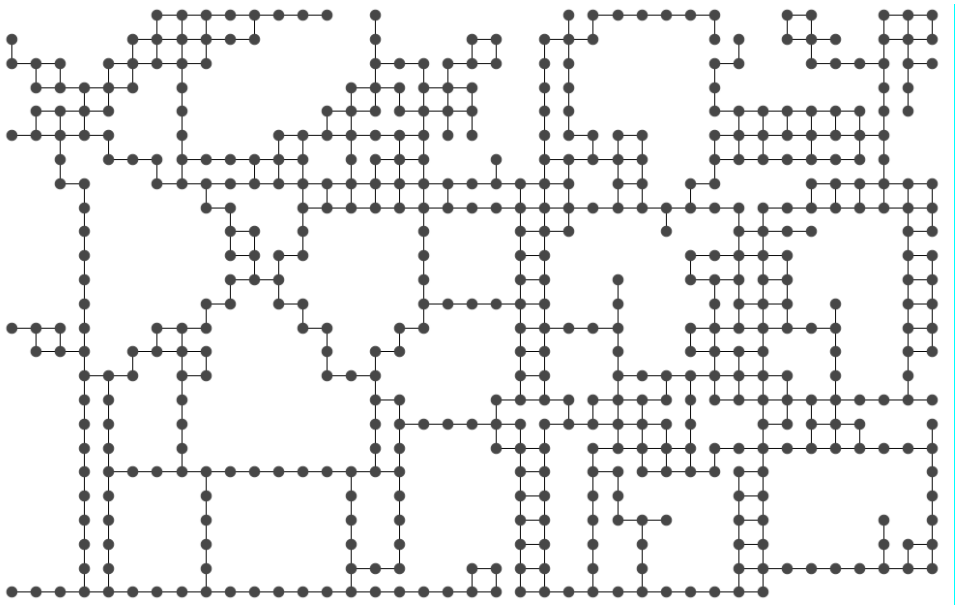


(b)

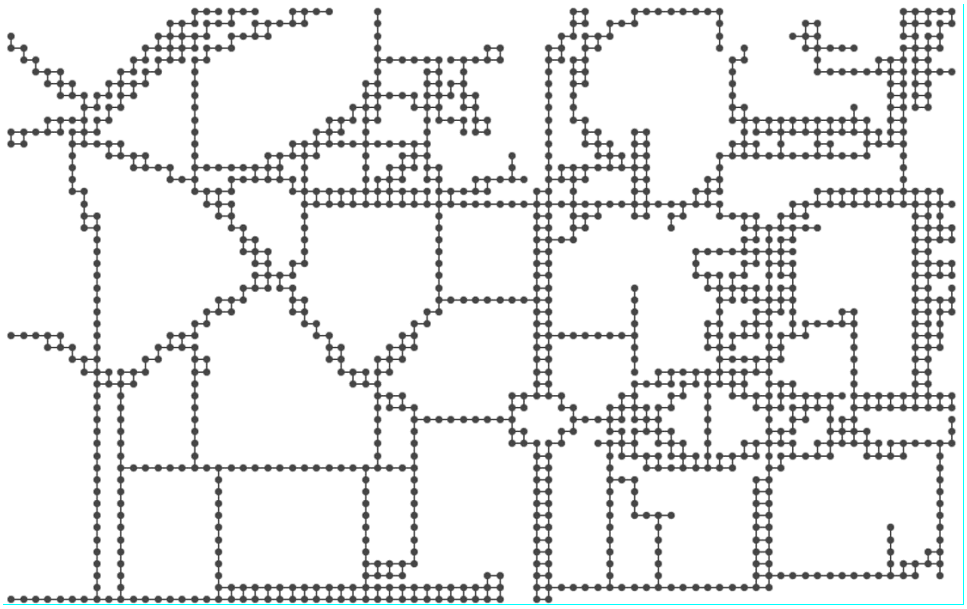


(c)

Figure 9. Translation of a region's sidewalk system into representative network graphs at varying granularities. a) Sidewalk infrastructure map of a selected region. b) Automatically generated network graph of sidewalk infrastructure with a 5-by-8 partition granularity. c) Automatically generated network graph of sidewalk infrastructure with a 13-by-20 partition granularity. d) Automatically generated network graph of sidewalk infrastructure with a 25-by-39 partition granularity. e) Automatically generated network graph of sidewalk infrastructure with a 50-by-78 partition granularity. (cont.)



(d)



(e)

Figure 9. Translation of a region's sidewalk system into representative network graphs at varying granularities. a) Sidewalk infrastructure map of a selected region. b) Automatically generated network graph of sidewalk infrastructure with a 5-by-8 partition granularity. c) Automatically generated network graph of sidewalk infrastructure with a 13-by-20 partition granularity. d) Automatically generated network graph of sidewalk infrastructure with a 25-by-39 partition granularity. e) Automatically generated network graph of sidewalk infrastructure with a 50-by-78 partition granularity. (cont.)

3.3. OPTIMIZATION BY SIMULATION

Equipped with a means to systematically represent dependent infrastructure systems, a method using this allowance to intelligently restore damaged systems of the kind should be developed. As discussed, arranging the required repairs in a manner that restores the damaged area as quickly and affordably as possible is critical, but no easy task. Represented by damaged nodes, differently ordering these required repairs has been demonstrated to yield different restoration costs and times. Seeking to simultaneously minimize these two metrics comprises a multiobjective optimization problem with a complex decision space. In fact, preferably ordering the repairs for each infrastructure system may be thought of as a permutation problem similar in complexity to the classical traveling salesman or vehicle routing problems (Bellmore & Nemhauser, 1968; Dantzig & Ramser, 1959). Herein, the required repairs for each infrastructure system may be arranged in $n!$ ways, where n is the number of damaged nodes. This number becomes incredibly large for even a relatively small number of nodes, exceeding 1 trillion possible permutations for just 15 damaged nodes. Already a set of NP-Hard problems, these permutations are further complicated by the intricacies they have been demonstrated to share and the challenge of selecting appropriate resource levels to repair each infrastructure (Karp, 1972). These considerations mean that finding feasible schedules may be difficult and that a holistic approach must be assumed in creating a combinatorial permutation of all required repairs.

While all possible arrangements may be assessed when given enough time, this is impractical not only because of the quickly exploding number of permutations, but also because of the time required to determine the feasibility of, and assess, each alternative.

Permutation problems of the kind have been addressed using a number of heuristics including genetic algorithms, simulated annealing, ant colony optimization, nearest neighbor algorithms, pairwise exchange, and variable-opt approaches, to name a few (Razali & Geraghty, 2011; Skiscim & Golden, 1983; Manfrin et al., 2006; Monnot & Toulouse, 2014; Kizilateş & Nuriyeva, 2013; Verhoeven et al., 1995; Bentley, 1990). These inexact methods have been adopted to find very-good solutions in a reasonable amount of time and have enjoyed considerable success. Recognizing this efficacy, a multiobjective genetic algorithm is proposed to generate well-performing restoration schedules for post-disaster infrastructure systems. Inspired by the processes of natural selection and evolution, multiobjective genetic algorithms look to evolve a population of solutions toward the Pareto frontier of an objective space (Schaffer, 1985; Srinivas & Deb, 1994). This Pareto frontier is comprised by a set of *Pareto efficient* solutions from which no solution can be selected that is better than another member of the set by all objectives. They exist in contrast to a set of dominated solutions which are not better than any member of the Pareto efficient set by any objective and are worse than a member of the Pareto efficient set by at least one objective (Steuer, 1986). An illustration of this relationship is shown in Figure 10. Unless it is comprised by only one solution, no solution of the Pareto efficient set can be considered objectively best. Therefore, some subjectivity must be introduced in making a final selection. However, determining a set of Pareto efficient solutions is incredibly useful as it provides decision-makers with a set of alternatives that are strictly not worse than any known solution and disclose useful tradeoff information between the competing objectives.

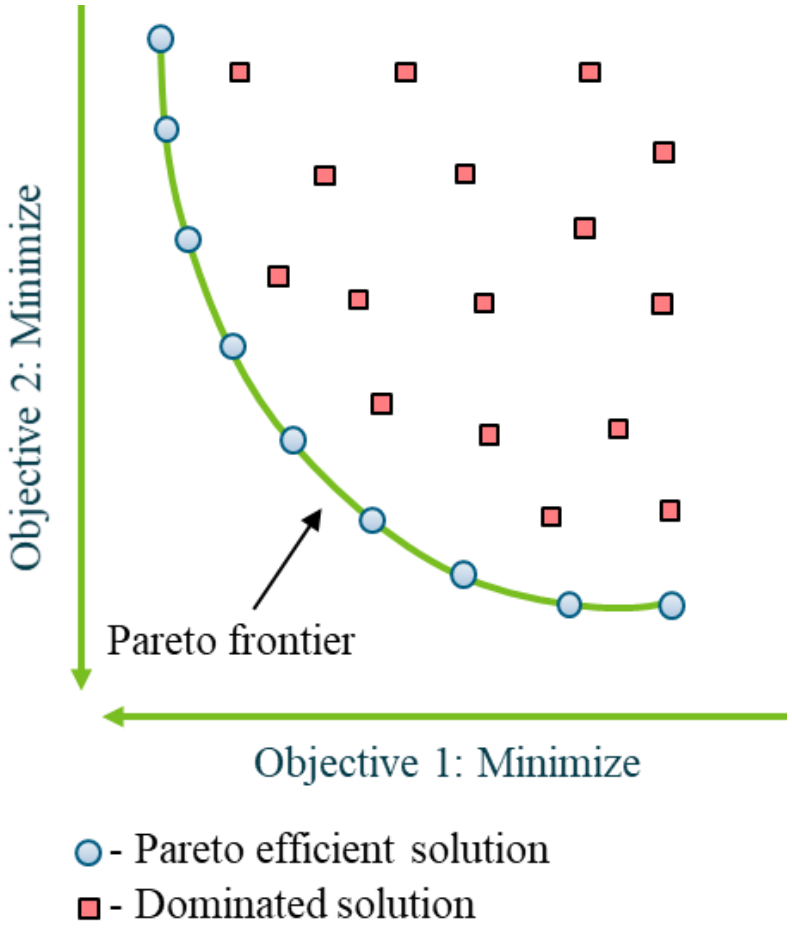


Figure 10. Relationships between Pareto efficient solutions, dominated solutions, and the Pareto frontier for a bi-objective optimization scenario.

A general overview of the proposed method is shown in Figure 11. This approach follows a backbone resembling that of NSGA-II (likely the most widely employed multiobjective genetic algorithm), with several modifications that allow it to handle the task of optimizing disaster recovery schedules (Deb et al., 2002). The high-level workings of this method are described briefly here, with intricacies discussed in greater detail in the following subsections. After determining the required repairs and representing this damage on the combinatorial infrastructure graph, an initial population

of restoration schedules (solutions) is first created. Each member of this population is simply a randomly generated permutation of the required repairs along with randomly assigned resource levels to address each infrastructure type. The nature and representation of these population members is further discussed in Section 3.3.1. This initial, or *parent*, population is then used to create a population of *offspring* via crossover (Section 3.3.2) and mutation (Section 3.3.3) operators. The combined parent and offspring populations are then passed to an agent-based simulation (Section 3.3.4) used to determine the repair time and cost associated with each schedule. This procedure also augments the schedules into feasible solutions, making sure that precedence and accessibility constraints are not violated. This is necessary as the randomly generated solutions (and those manufactured by offspring creation procedures) will likely exhibit several infeasibilities without modification. These augmented solutions, along with their elicited performance metrics are passed back to the genetic algorithm for selection (Section 3.3.5). This procedure determines a subset of the total population which will be used as parents in creating the next set of offspring, completing the first *iteration* of the genetic algorithm. The selected solutions and the offspring they generate are then passed to the simulation, as before, and the iterative process (the loop formed by black arrows in Figure 11) is repeated until some stopping criteria are met. Following several iterations and termination, the proposed method culminates in a set of feasible, automatically generated restoration strategies for decision-maker consideration and comparison.

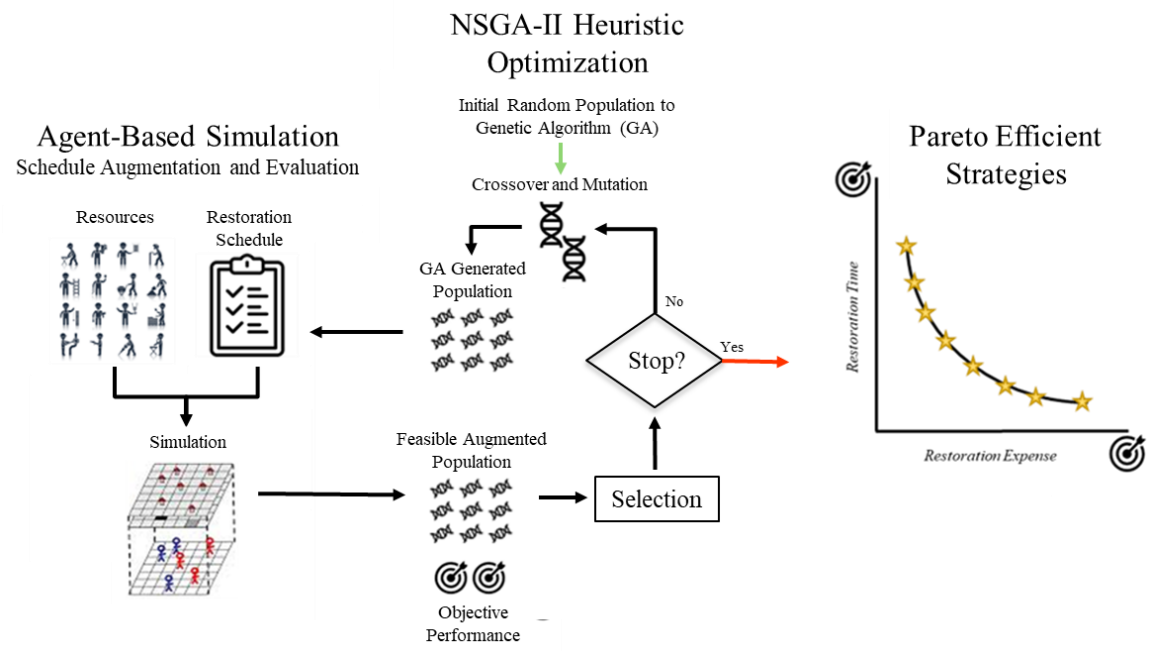


Figure 11. High-level overview of proposed method operations.

3.3.1. Solution Representation. To encode resource and schedule information for use by the genetic algorithm and agent-based simulation, a representation scheme is necessary. Maintaining the terminology of its biologic inspiration, this representation is known as a *chromosome*. Each chromosome is comprised of a sequence of *bits* which encode pieces of information about the solution. Here, a compound representation is used. The first part of the chromosome describes the resource levels selected to restore each type of infrastructure. Any unit may be used to describe resource levels in this initial section. Further, multiple bits may be dedicated to address the same infrastructure system; e.g., a bit to describe the number of workers capable of repairing low-voltage electrical lines and a bit dictating the number of lineman able to work on high-voltage components. The remainder of this investigation uses units of *number of repair teams* assigned to each infrastructure for convenience and ease of understanding and discussion.

In the example of Figure 12, three infrastructure systems need repair: roadway, water, and electricity. The first three bits of this chromosome define the number of restoration teams assigned to repair these respective infrastructures: two teams capable of roadway restoration, one to address the electric system, and two for water infrastructure repair. The second portion of the chromosome has one bit for each required repair to an infrastructure system. The order of these bits then describes the order that these repairs are set to begin. In Figure 12, six damaged nodes are scheduled for repair in the order shown. Note, for any application of the proposed method, this second section is a permutation of all damaged infrastructure nodes.

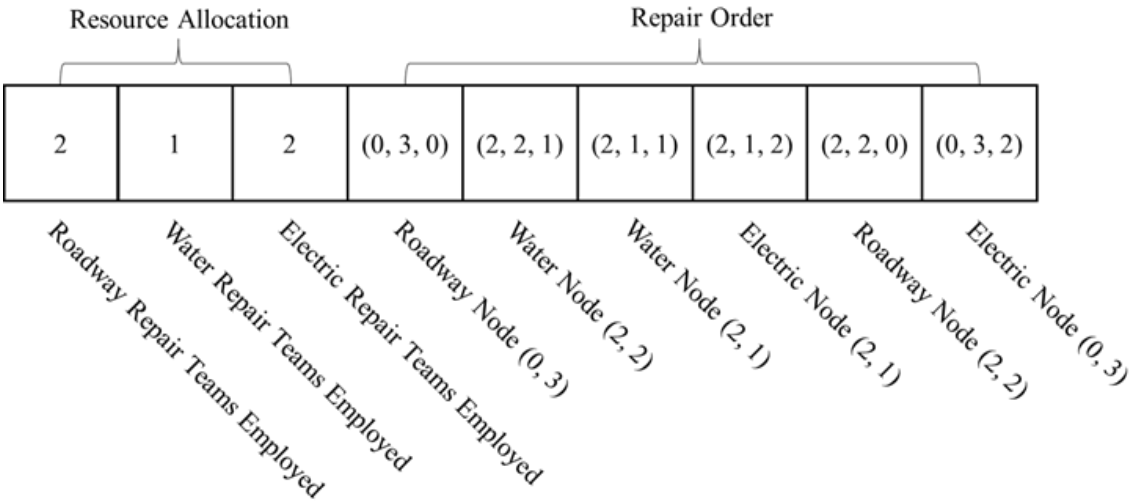


Figure 12. Sample compound solution representation for proposed method.

3.3.2. Crossover Operators. Crossover is an operation within genetic algorithms that (generally) uses two parent chromosomes to create an offspring exhibiting some characteristics of each parent. This mimics the combination of parental genes within the

offspring of many biologic organisms (Holland, 1992). Because the chromosomes used in this work have some special characteristics, as discussed in Section 3.3.1, a custom crossover operator is be adopted. Detailed in Pseudocode 1, the crossover procedure first selects two random chromosomes from the population of parent solutions. Two bit-positions along the first parent chromosome are also randomly selected. If the earliest selected bit along this sequence, call it position *y*, is within the range of bits dictating resource levels (Pseudocode 1 – Line 6), a simple single-point crossover is utilized (Deb & Agrawal, 1995). This procedure creates an offspring by combining the bits of the first parent up to and including position *y*, then appends the bits of the second parent occurring after position *y*. Alternatively, if the earliest selected bit occurs within the range dictating restoration order (Pseudocode 1 – Line 10), a more sophisticated crossover protocol is required; crossover procedures applied to this range must take care to maintain a permutation of all required repairs. Classical approaches such as one-point, two-point, and uniform crossover do not ensure the preservation of this property (Oliver, 1987). Here, a procedure known as *OX Crossover* is utilized. Described in Pseudocode 1, this operator is discussed in greater detail in Davis (1985).

At each generation, the crossover method of Pseudocode 1 is used to produce a population of offspring equal in number to the parent population used in their creation. It is hoped, through these operations, that offspring are bred which improve upon their parent chromosomes by one or more objectives.

Pseudocode 1: **Crossover**

Functions:
firstX(*M*, *N*): returns the first item in *N* not already in *M*

Indexing:
list[*C* : *D*]: the Cth through D^{th} items of *list*, inclusive

Inputs:
population: population of solution chromosomes (parents)

Outputs:
offspring: chromosome sharing characteristics of parents

Pseudocode:

```
set r as the number of bits dictating resource levels
set end as the length of the chromosome representation used
randomly select parent1 and parent2 from population
randomly select two bit positions: a and b
set y as the minimum of a and b
set z as the maximum of a and b
if y ≤ r:
    parent1Contribution = parent1[1 : y]
    parent2Contribution = parent1[y+1 : end]
    offspring = parent1Contribution + parent2Contribution
else:
    offspring = parent1[0 : r]
    parent1Contribution = parent1[y : z]
    for i in range(y-r):
        append firstX(parent1Contribution, parent2[r+1 : end]) to
        offspring
    append parent1Contribution to offspring
    for j in range(z+1 : end):
        append firstX(offspring, parent2[r+1 : end]) to offspring
return offspring
```

3.3.3. Mutation Operators. Mutation is a process within genetic algorithms used to achieve and maintain solution diversity from one generation to the next. Analogous to biologic mutation within living organisms, this procedure stochastically modifies chromosomes to avoid local optima and keep the population from becoming too similar (Holland, 1992). Mutation further allows characteristics not exhibited by any member of the population to be injected into a chromosome, potentially allowing a new region of the decision space to be explored.

As with the crossover operator, the proposed model employs a specialized mutation procedure to handle the used representation's unique properties. Disclosed in Pseudocode 2, this mutation operator can introduce diversity into both the resource level and restoration schedule sections of a population's chromosomes. If a randomly generated number from the uniform distribution between 0 and 1 is less than or equal to a supplied mutation rate (*mutationRate*), the procedure will mutate the selected chromosome. If it is greater than *mutationRate*, no mutation will occur. When mutating, if a second randomly generated number from the same distribution is less than or equal to a supplied threshold (*resWeight*) the operator will mutate the resource level portion of the chromosome. In such a case, a random bit dictating resource level will be replaced with a random integer between one and a user-defined cap for the associated resource (*resCap*). Supplied as a list, a scenario employing three resource types with a *resCap* sequence of [3, 8, 6] could utilize at most 3 units of Resource 1, 8 units of Resource 2, and 6 units of Resource 3. If, however, the second randomly generated number is greater than *resWeight*, the restoration schedule portion of the chromosome is mutated by swapping the values of two randomly selected bits from this range. For simplicity, only this swap operator is used when mutating the restoration schedule section of a chromosome. Other operators, such as inversions or slides, may be used, as long as they preserve the permutation nature of this section.

Note the utility of *resWeight*, allowing the user to dictate the general distribution of mutation between the resource level and restoration schedule portions of a population's chromosomes. This control is useful as some scenarios may underserve one of these equally important sections if mutation is left to totally random bit selection. This

can particularly occur when one of these chromosome portions is much longer than another, wherein the shorter portion may be left receiving very few mutations.

Pseudocode 2: **Mutation**

Functions:

swap(*U*, *V*, W): swaps the bit-values of position *U* and position *V* within chromosome *W*

random(): produces a random value from the uniform distribution between 0 and 1

Indexing:

list[*E*]: the Eth item of *list*

Inputs:

population: population of solution chromosomes (parents)

mutationRate: proportional likelihood a solution will be mutated

resWeight: the proportional likelihood a given mutation will occur to the resource level section of the respective chromosome

resCap: a list defining the maximum level of each potential resource

Outputs:

mutant: chromosome which may have been mutated

Pseudocode:

```
set r as the number of bits dictating resource levels
set end as the length of the chromosome representation used
set mutant as the chromosome which may be mutated
if random() ≤ mutationRate:
    if random() ≤ resWeight:
        randomly select a bit position from 1 to r, inclusive: k
        replace the kth bit-value of mutant with a random integer from
          1 to resCap[k]
    else:
        randomly select a bit position from r+1 to end, inclusive: g
        randomly select a bit position from r+1 to end, inclusive: h
        swap(g, h, mutant)
return mutant
```

At each generation, Pseudocode 2 is applied to each chromosome within the combined population of parents and crossover-born offspring. As a measure of preserving elitism and diversity, Pareto efficient solutions that occupy a unique point in the objective space are not mutated traditionally. Instead, a copy of the original version is appended to

the population before the original is mutated as usual. Note that this individuality is not determined by the uniqueness of a chromosome, but by the objective performance that chromosome achieves. This measure looks to maintain the benefits of mutation while avoiding the loss of solutions with unmatched performance to potentially detrimental stochastic modification.

3.3.4. Evaluation by Simulation. Agent-based simulation serves a dual purpose within the proposed method. When presented with a chromosome, this procedure simultaneously augments (if required) the solution to have a feasible restoration schedule and serves as a performance evaluator, determining the time and cost required by the chromosome to restore the disaster area. This strategy side-steps the need to manually craft feasibility constraints which may be arduous to formulate and are variable from one area or situation to another. Similarly, the establishment of formal objective functions is not required, with objective values derived instead from the proceedings of the simulation.

To facilitate a discussion of this mechanism (Pseudocode 3), a few definitions are first required. Within a combinatorial graph representing an area's infrastructure systems, a *damaged node* describes a partition containing damage to the infrastructure system the node defines. Alternatively, an *undamaged node* represents a partition containing the associated infrastructure while exhibiting no damage. Each node also possesses a set of *neighbors*, which are the nodes of the same infrastructure type to which it is normally connected by an arc. An *accessible node* has at least one undamaged neighbor that is linked to the functional surroundings by arcs connecting undamaged nodes of the same infrastructure type. A *feasible node*, finally, is one which may be restored given the current

state of the combinatorial graph. For a node to be feasible, it must be accessible and each of its precedence nodes must be accessible and undamaged. While these general feasibility rules are adopted for the current investigation, any desired ruleset may be integrated into this flexible framework.

These statuses are easily encoded within combinatorial graphs and discerned by network optimization techniques. To describe the impacts of a disaster event, all damaged nodes and the arcs connected to them are removed, such as demonstrated in Figure 6. Whether a node is accessible can be determined by a simple tree search along all arc paths branching out from the node in question; if one of these branches reaches the functional surroundings, the node is known to be accessible. This tree search can also be used to determine if an accessible node is feasible by checking if each of the node's precedence nodes are accessible, in addition to being undamaged. When a damaged feasible node is restored, it is readded to the combinatorial graph and arcs are drawn between it and each of its undamaged neighbors. This reinstitution moves the overall network closer to its pre-disaster condition and may affect the statuses of other nodes. Through these procedures, the systematic assessment of a proposed restoration plan can be completed by the agent-based scheme of Pseudocode 3.

Pseudocode 3: **Agent-Based Simulation Evaluation**

Functions:

restore(O, Q): adds node O to graph Q and connects O to undamaged neighbors

Indexing:

list[E]: the Eth item of *list*

list[C : D]: the Cth through D^{th} items of *list*, inclusive

Inputs:

chromosome: restoration solution from genetic algorithm

G: area combinatorial graph with damaged nodes (and associated arcs) removed
timeReq: matrix of time required to restore each node
fixedCosts: list of fixed costs associated with each resource type
variableCosts: list of costs per unit time associated with each resource type

Outputs:

feasChromosome: augmented chromosome with feasible restoration schedule
resCost: cost to restore disaster area by implementing *chromosome*
resTime: time to restore disaster area by implementing *chromosome*

Pseudocode:

```
set r as the number of bits dictating resource levels
set end as the length of the chromosome representation used
set restorationLog as empty
resources = [ ]
for h in range(r):
    append r(h) agent(s) of type h to resources
mark each agent in resources as available
set jobs as the ordered list of restorations (chromosome[r+1 : end])
mark each job in jobs as notAssigned
clock = 0
while any job in jobs is not completed:
    mark each idle agent as available
    for each available agent in resources:
        assign agent to first feasible job in jobs of appropriate type
        mark agent as unavailable for timeReq[job] steps of clock
        mark job as assigned
        record job assignment to agent in restorationLog
        if no job is currently feasible for agent:
            mark agent as idle
    clock += 1
    for any job just completed by an agent:
        restore(job, G)
        mark job as completed
        record job completion by agent in restorationLog
        mark agent as available
resTime = clock
resCost = chromosome[1 : r] • fixedCosts
for each agent in resources:
    find timeUtilized by examining restorationLog
    resCost = resCost + (timeUtilized * variableCosts[agent.type])
set feasChromosome as the order of job assignment in restorationLog
return feasChromosome, resCost, resTime
```

The simulation begins by creating a set of *resources* which are responsible for restoring the damaged infrastructure systems. Each *agent* within *resources* has an attribute, *type*, defining the type of infrastructure system it may restore. The number of each *type* of *agent* created is dictated by the resource level portion of the *chromosome* passed to the simulation. Each *agent* is then marked as available as none have yet been assigned a restoration task. The ordered set of *jobs*, or nodes requiring repair, is next gleaned from the restoration schedule portion of the *chromosome*. Each *job* within *jobs* is then marked as notAssigned, indicating that no *agent* has yet been dispatched to restore it.

The time regulator of the simulation, *clock*, is then initialized and the task of restoring the damaged infrastructure systems is begun. While any *job* remains not completed, the procedure will look to find an available *agent* of the appropriate *type* to complete the required repairs. Assignment occurs following the order of *jobs* (inherited from the *chromosome*), with an available *agent* being assigned the first feasible *job* (related to a feasible node) it is qualified to complete. Upon this assignment, the agent is marked as unavailable for the amount of time required to complete the repairs of the assigned *job*, as prescribed by *timeReq*. The *job* is then marked as assigned and the assignment is recorded in *restorationLog*. If no feasible *job* can be found for an agent, the *agent* enters an idle state until the next *clock* step. This assignment search is conducted for each available *agent*, resulting in all *resources* being either unavailable or idle before the next *clock* step is initiated.

When no available *resources* remain, *clock* is stepped to the next value, signifying the passing of one time-unit within the simulation. Upon each step of *clock*, an assigned *job* may be completed by an *agent*. Upon this occurrence, the node of the combinatorial

graph (*G*) associated with the *job* is restored and the *job* is marked as completed. This completion is also recorded in *restorationLog*. The *agent* completing the *job* is next marked as available and is eligible for assignment to a new *job*.

When every *job* is completed, the simulation is ended, and the performance metrics of the *chromosome* can be determined. The amount of time required by the solution to restore the area, *resTime*, is set equal to *clock*. The cost required by the solution to complete the overall restoration, *resCost*, is then determined as the sum of all fixed and variable costs incurred. *resCost* is first set equal to the dot product of the resource level portion of *chromosome* and a list of the fixed costs associated with employing one unit of each resource, *fixedCosts*. For instance, a scenario with *chromosome*-defined resource levels of [2, 5, 1] and a *fixedCosts* list of [$300, $200, $250] would accumulate $1,850 of fixed costs: 2 units * $300/unit for resource 1, 5 units * $200/unit for resource 2, and 1 units * $250/unit for resource 3. The variable costs associated with each individual *agent* of *resources* is then added to *resCost* to finalize the metric. Examining *restorationLog*, the time (in *clock* steps) between the first assignment of a *job* and the last instance of completing a *job* is determined for each *agent*. This value, *timeUtilized*, is then multiplied by the cost per unit time of employing the *type* of *agent*, as prescribed by *variableCosts*. The sum of this product for every *agent* and the fixed costs previously determined comprise the total *resCost* mandated by the examined solution.

The agent-based model finally examines *restorationLog* to determine the actual order in which each *job* was assigned. This order is the final output, *feasChromosome*, of the procedure. This augmented chromosome is free of any precedence or feasibility violations, which may have been present in the input *chromosome*. Returning these

augmented solutions, all alternatives output by the simulation are perfectly viable restoration strategies. These feasible chromosomes are then passed back to the genetic algorithm for selection based on their associated time and cost objective values.

3.3.5. Multiobjective Selection. To maintain a fixed population size, a subset of all solutions returned from the agent-based model needs to be selected to serve as the parent population of the genetic algorithm's next generation. A popular selection method within multiobjective genetic algorithms is to use nondominated sorting to assign an overall *fitness* value to each solution, then retain those deemed most fit (Deb et al., 2002). This fitness is determined by sorting the set of solutions into successive *Pareto fronts* based on their objective values. The first Pareto front of a set of solutions are all those solutions that are Pareto efficient. If the first Pareto front were removed from the set, those solutions becoming Pareto efficient comprise the second Pareto front. This is done repeatedly until all solutions are assigned a front. Solutions with the lowest front number, or *rank*, are then considered by the selection mechanism as most fit.

Starting with the first front, the selection procedure adds successive fronts of solutions to the population to be retained until a minimum specified number of solutions has been added. If necessary, solutions from the worst included rank are then removed by a crowding distance operator until the exact number of specified solutions is retained. Following these operations, the population passed to each generation of the genetic algorithm is exactly the same size, regardless of the number of new solutions created by crossover and mutation.

The selection mechanism of this approach follows that of NSGA-II (Deb et al., 2002), with the exception of the nondominated sorting method utilized. While capable,

the Fast Nondominated Sort procedure used in NSGA-II is much less efficient than contemporary methods. Instead, the proposed method uses Ideal Sort (Vanfossan & Kwasa, 2022) to expedite nondominated sorting and generate the final set of restoration strategies more quickly.

4. DISASTER SIMULATION AND PROPOSED METHOD APPLICATION

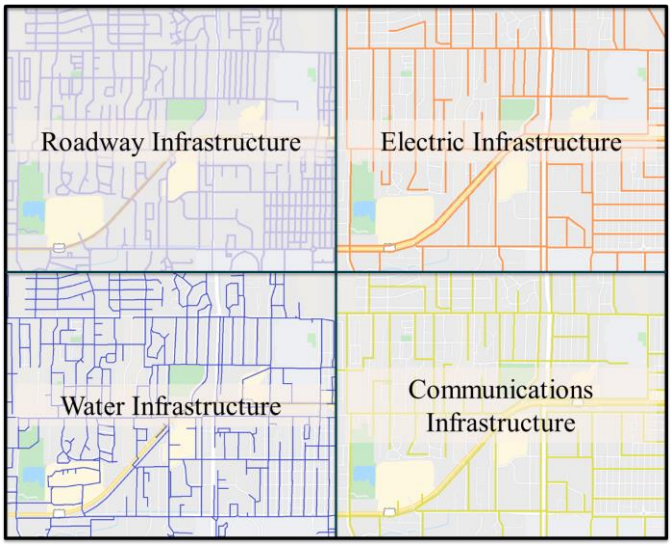
4.1. A SIMULATED DISASTER

To test the proposed method, a simulated disaster scenario was created using the actual infrastructure maps of the nearly 2-square-mile sample region shown in Figure 13 (a). Here, four basic infrastructure systems are included: roadway, electric, water, and communications. Following the presented procedures, the infrastructure maps (Figure 13 (b)) were partitioned by a common 13-by-16 grid, dividing each system into 208 rectangular partitions. Each partition, then, relates to an area roughly measuring 465 feet-by-575 feet. The automated process disclosed was then used to generate the network graph of each infrastructure system, as shown in Figure 13 (c). Finally, the networks are combined into a single combinatorial graph through the connections prompted by the precedence schedule of Table 2. Specifically, an arc is drawn from each node to any precedence nodes the former relies on. As demonstrated, this precedence relationship is made considering both infrastructure type and locality. For example, an electric infrastructure node existing at graph location *(0, 0, 1)* will be connected by an arc to the similarly located roadway infrastructure node, *Node (0, 0, 0)*. Correspondingly, a communications infrastructure node at this spatial location, *Node (0, 0, 3)*, would be

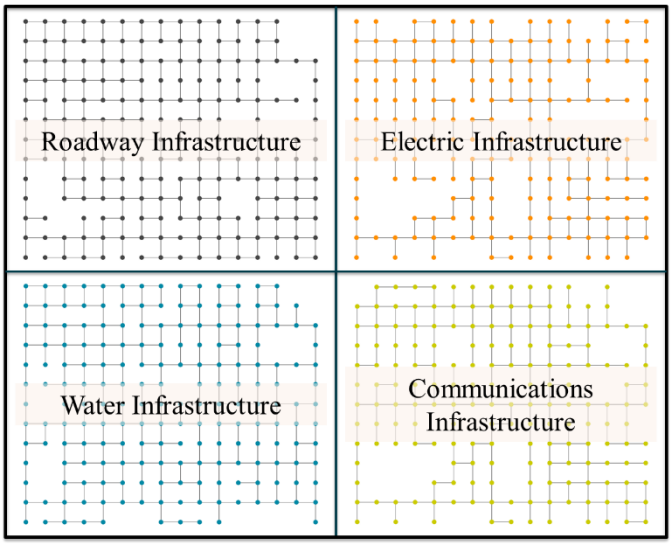
connected to both *Node (0, 0, 0)* and *Node (0, 0, 1)*. If a node is of an infrastructure type having a precedence requirement, but the required node does not exist in the same locality, the geographically closest node of the required precedence infrastructure is linked as the precedence node. If a tie exists for closest required precedence node, the candidate node defining the partition with the greatest infrastructure prevalence is chosen. Recall that this prevalence is automatically recorded by the automated network generation procedure of Section 3.2. Each of these processes can be performed systematically, requiring no effort from the end-user to encode these precedence relationships.

(a)

Figure 13. Identification of infrastructure systems from sample geographic region and translation to representative network graphs. a) Sample region comprised of co-located infrastructure systems. b) Infrastructure maps describing selected systems present within the sample region of Figure 13 (a). c) Network graphs representative of the infrastructure maps of Figure 13 (b).



(b)



(c)

Figure 13. Identification of infrastructure systems from sample geographic region and translation to representative network graphs. a) Sample region comprised of co-located infrastructure systems. b) Infrastructure maps describing selected systems present within the sample region of Figure 13 (a). c) Network graphs representative of the infrastructure maps of Figure 13 (b). (cont.)

Table 2. Precedence relationships between four selected infrastructure systems.

Infrastructure Type	Precedence Infrastructures
Roadway	-
Electric	Roadway
Water	Roadway
Communications	Roadway, Electric

After constructing the combinatorial graph, the effects of a simulated disaster were introduced. Seeking to emulate the impacts of a tornado, the damage schedule of Table 3 and Figure 14 was used. Here, nodes of the combinatorial graph are set as damaged according to the type of infrastructure they describe, their geographic location, and the associated damage probabilities of Table 3. For instance, the electric infrastructure node corresponding to Partition (7, 7) has an 80% chance of being damaged as it exists within the *Red Region* of Figure 14. For each infrastructure system, the probability of damage is greatest for nodes within the *Red Region*, less for those within the *Yellow Region*, and even less for those in the *Green Region*. This is adopted to mimic the reduced damage that may be experienced by areas an increasing distance from the touchdown path of a tornado (Roueche & Prevatt, 2013).

Table 3. Likelihood of node damage by type of infrastructure system and region classification, as defined by Figure 14.

Infrastructure Type	Likelihood of Node Damage		
	Red Region	Yellow Region	Green Region
Roadway	90%	75%	50%
Electric	80%	60%	40%
Water	30%	15%	10%
Communications	70%	50%	30%

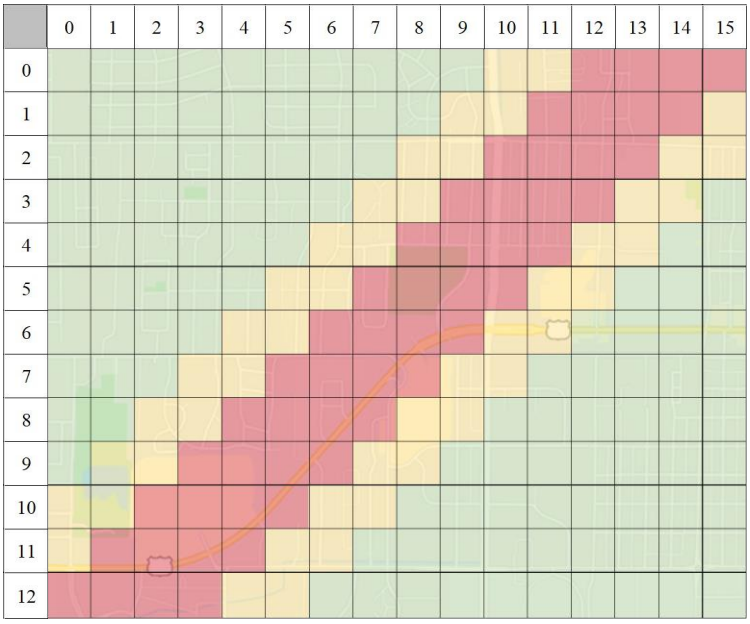


Figure 14. Tornado damage schedule, describing relative damage expectations by map partition.

In a real disaster scenario, the amount of damage to each node may be estimated by observing the affected region or the application of assessment models (Spedheger et al., 2002; Marshall, 2002; Hashemi & Alesheikh, 2011; Myint et al., 2008; Wu & Cui, 2018; Erdik et. al, 2011; Kryvasheyeu et al., 2016; Gong, 2013; Foresti, 2015). These estimates can then be used in conjunction with resource repair rates to determine the length of time required to repair each node (Ramachandran et al., 2015). Here, the time required by an appropriate resource team to repair each node is simply set as a random integer between 1 and 10. At most, then, a node will take 10 clock steps of the simulation to restore. While a more sophisticated damage simulation method may have been used, this basic approach easily incorporates the unpredictable and nonuniform damage that may accompany varying types of disasters (Masoomi et al., 2018; Lu & Guan, 2017).

The following parameters and scenario inputs were finally supplied to the model, prompting its generation of a set of Pareto efficient restoration strategies:

- Population Size: 100 Solutions
- Maximum Generations: 500
- Clock Step: 1 Day
- *mutationRate*: 0.1
- *resWeight*: 0.25
- *resCap*: [10, 10, 10, 10]
- Resources: Roadway Repair Team(s), Electric Repair Team(s), Water Repair Team(s), Communications Repair Team(s)
- Resource Costs (*fixedCosts* and *variableCosts*) defined by Table 4

Table 4. Fixed and variable resource cost schedule by resource type.

Resource Type	***fixedCosts***	***variableCosts***
Roadway Repair Team	$6,500 / Team	$480 / Day / Team
Electric Repair Team	$8,250 / Team	$1,120 / Day / Team
Water Repair Team	$6,200 / Team	$800 / Day / Team
Communications Repair Team	$9,000 / Team	$960 / Day / Team

Note, here, that arbitrary *fixedCosts* and *variableCosts* are assigned for the example resource types. These can be easily modified to reflect the real costs incumbent of the scenario to which the method is being applied.

4.2. METHOD RECOMMENDATIONS AND ALTERNATIVE RESTORATION STRATEGIES

Following 500 generations, the model returned solutions with 11 unique objective pairings. These comprised a set of Pareto efficient solutions spanning from 65 to 457 days to complete, while incurring expenses between $1,351,150 and $1,519,950. The objective values owned by each of these alternatives are listed here:

Format: (Restoration Time, Restoration Expense)

- (65 Days, $1,519,950)
- (67 Days, $1,512,110)
- (70 Days, $1,511,950)
- (78 Days, $1,484,350)
- (92 Days, $1,457,150)
- (93 Days, $1,452,350)
- (115 Days, $1,430,100)
- (127 Days, $1,409,400)
- (153 Days, $1,400,400)
- (229 Days, $1,370,700)
- (457 Days, $1,351,150)

While 100 unique solutions were generated, only the 11 unique objective function pairs were precipitated. This demonstrates another complexity of the scenario, in which two recovery schedules may produce identical objective function results. This could serve problematic for recovery planners attempting to manually produce alternative solution

strategies. This potential frustration is sidestepped by the automated strategy generation of the proposed method.

To provide some context toward the efficacy of the proposed method, a few intuitive alternative strategies are conceived and evaluated. Each alternative is introduced here, with performance disclosed and discussed in Section 4.3. When generating these alternatives, it is assumed that a mechanism is possessed to ensure that only feasible strategies are produced. A further assumption is that some procedure is available to assess each alternative in terms of time and cost required. While easily achieved by the agent-based simulation of the proposed method, these capabilities may not be available to decision-makers using traditional approaches to generate recovery schedules. Nonetheless, these facilities will be used in creating and assessing the alternative strategies to isolate and highlight the benefits of the proposed model's evolutionary generation strategy.

4.2.1. Random Generation Strategy. A very basic approach is to generate several random solutions and then pick a desirable one. This is akin to the random population generation initiating the proposed method. There, 100 solutions were created.

4.2.2. Maximum Resource Strategy. Alternatively, decision-makers may look to develop strategies that restore the affected area as quickly or affordably as possible. Attempting to repair all damage as quickly as possible, the maximum number of each resource is employed. Here, this implies the availability of 10 units of each resource (a *chromosome*[1 : *r*] sequence of [10, 10, 10, 10]). However, the order in which nodes are repaired will still impact overall restoration time and cost. Thus, 100 randomly generated

restorations schedules are followed using these maximum resource levels to get some idea of this strategy's performance range.

4.2.3. Minimum Resource Strategy. Seeking to complete the required restorations as affordably as possible, the minimum number of each resource is employed: a *chromosome*[1 : *r*] sequence of [1, 1, 1, 1]. As before, 100 random restoration schedules were followed using this minimalist strategy to approximate its range of performance.

4.2.4. Most Damaged First Strategy. A seemingly intuitive strategy may be to restore those nodes sustaining the most damage (that is, those that will take the longest to repair), first. In this way, the most burdensome repairs are completed first, allowing resources to become available with more frequency during latter repair stages. This may help avoid scenarios where the long restoration times of nodes with many dependent counterparts can cause resources to sit idle, substantially extending overall completion time. Using this front-loaded restoration order, 100 solutions with randomly generated resource levels are created and assessed.

4.2.5. Least Damaged First Strategy. Instead, a strategy restoring the least damaged nodes first could also be used. Ordering restorations by increasing repair time, more nodes are restored earlier, allowing the network to resemble its undamaged state earlier in the restoration process. This may have the beneficial property of reducing the number of nodes that are not feasible quicker, helping avoid scenarios where resources sit idly with no feasible repairs. Here again, 100 solutions with random resource levels were created and assessed following this scheduling strategy.

4.3. STRATEGY COMPARISON AND DISCUSSION

The multiobjective performance of the solutions generated by each strategy is shown in Figure 15. This all-solution view is presented to give an idea of the scope and distribution of the solutions generated by each strategy. A zoomed view of a busy region of the solution space (Restoration Time: 60 Days – 160 Days; Restoration Cost: $1,400,000 - $2,000,00) is then shown in Figure 16, highlighting the objective superiority of the solutions produced by the proposed method. When all generated solutions are considered, each of those created by the proposed method are Pareto efficient. Further, all 500 solutions generated by the alternative strategies are dominated by at least one of these Pareto efficient options. Finally, the Pareto efficient set associated with each strategy is shown in Figure 17, demonstrating the proposed method's ability to cover the entire spectrum of solution performance achieved by other methods while simultaneously producing objectively better alternatives.

While the dominance enjoyed by the proposed method's solutions describes objective superiority, a further investigation is made to describe the degree of this superiority. Some summary statistics are presented, describing the multiobjective performance and distribution of the solutions generated by each examined strategy (Table 5). The yellow highlighted values within Table 5 denote the strategy performing the best with respect to respective summary statistics. For example, the best minimum restoration time achieved by any strategy is owned by the proposed method, requiring 65 days.

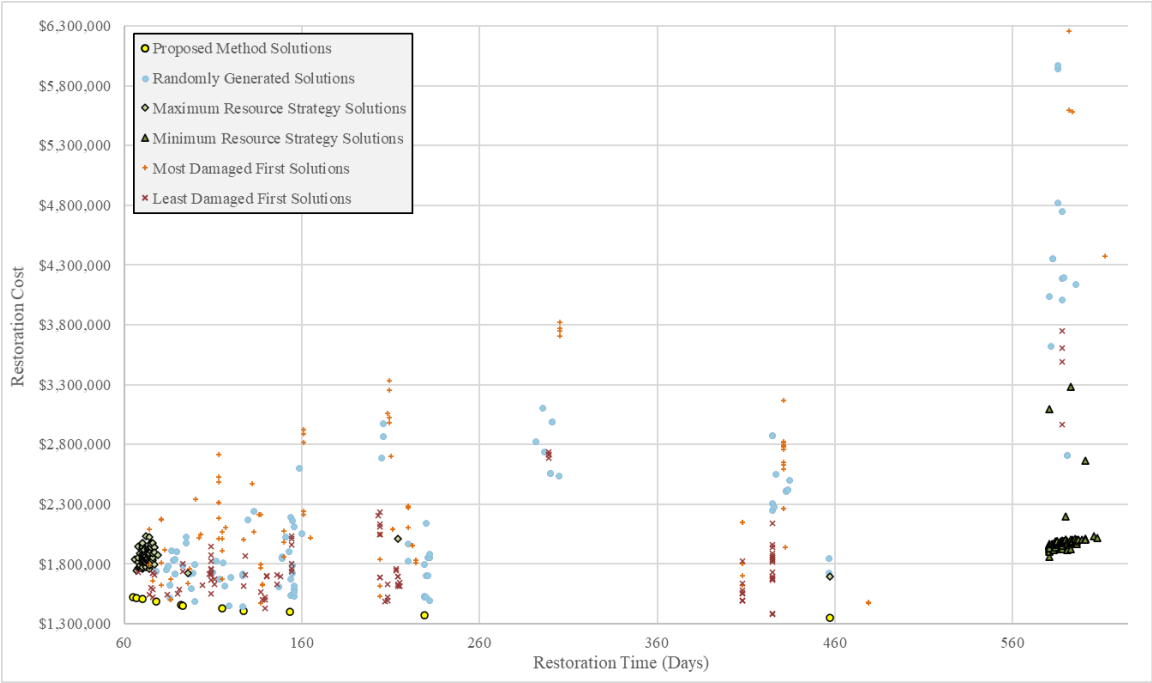


Figure 15. Multiobjective performance of recovery strategies created by proposed and alternative solution generation methods.

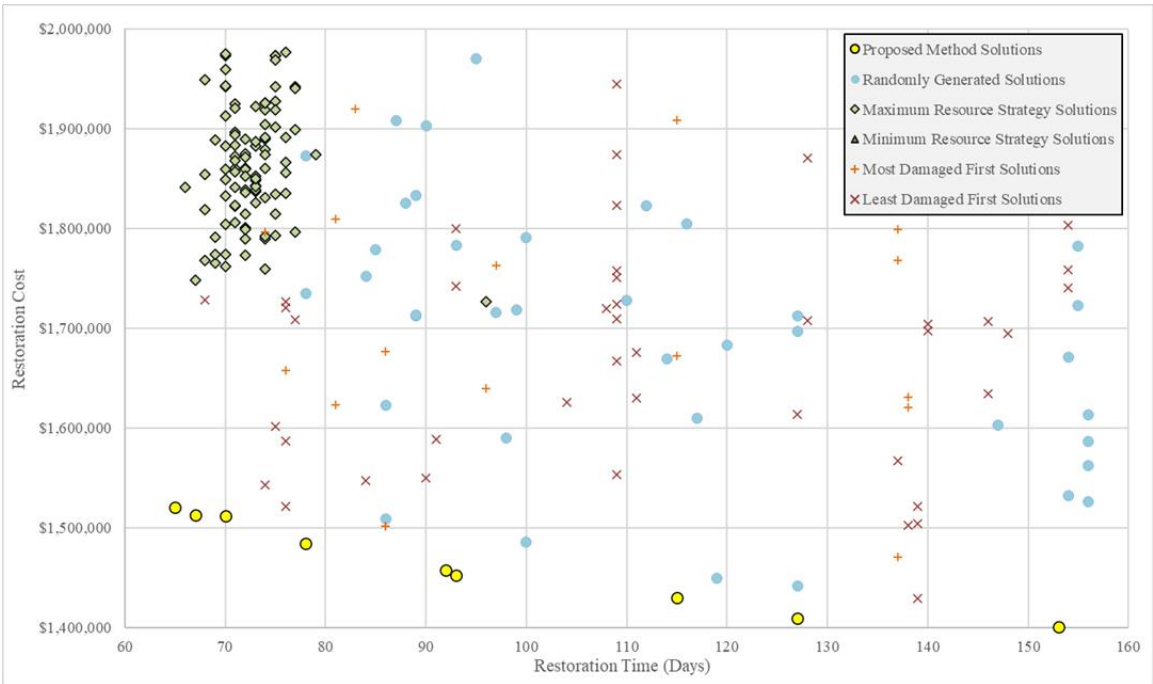


Figure 16. Selected region of multiobjective performance of recovery strategies created by proposed and alternative solution generation methods.

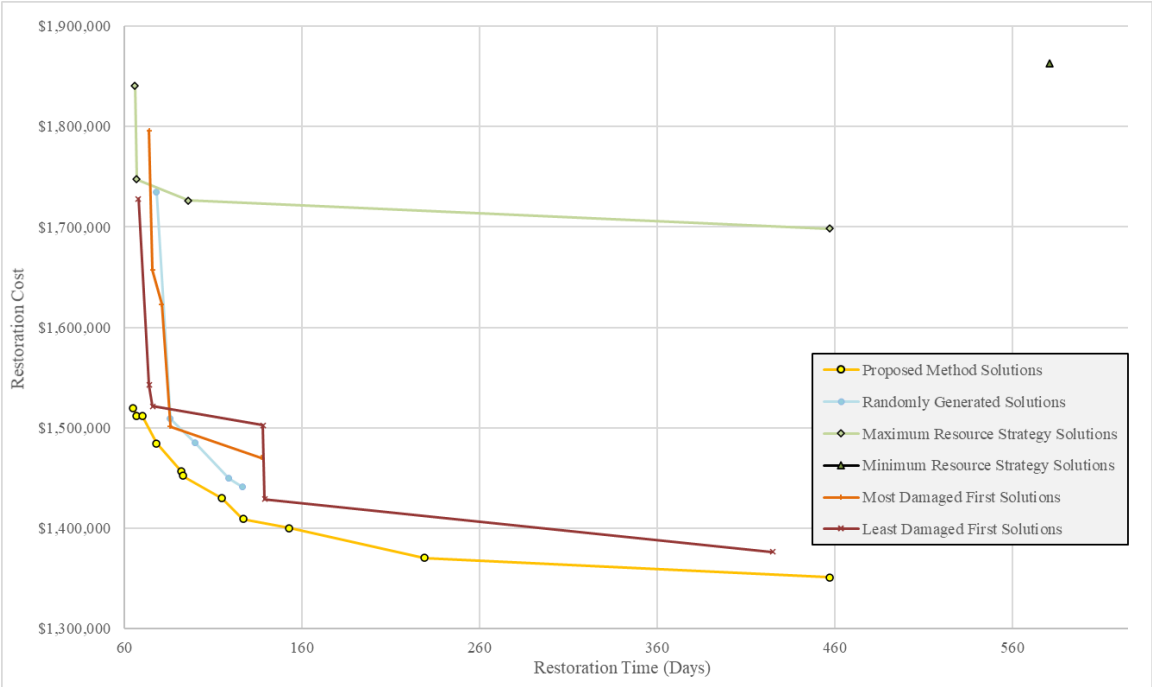


Figure 17. Pareto frontiers of solutions created by proposed and alternative strategy generation methods.

Table 5. Multiobjective performance statistics of proposed and alternative strategy generation methods.

Strategy	Proposed Method		Randomly Generated		Maximum Resource		Minimum Resource		Most Damaged First		Least Damaged First	
Objective	Res. Time (Days)	Res. Cost ($)	Res. Time (Days)	Res. Cost ($)	Res. Time (Days)	Res. Cost ($)	Res. Time (Days)	Res. Cost ($)	Res. Time (Days)	Res. Cost ($)	Res. Time (Days)	Res. Cost ($)
Mean	140.55	1,445,419.09	254.37	2,312,579.60	77.89	1,865,913.60	588.47	2,005,698.90	238.82	2,355,428.90	236.57	1,828,620.10
Minimum	65	1,351,150	78	1,440,980	66	1,698,500	581	1,862,910	74	1,470,280	68	1,376,710
1st Quartile	74	1,404,900	119	1,712,813	71	1,824,740	584	1,962,150	115	1,809,970	111	1,613,330
2nd Quartile	93	1,452,350	183	1,889,955	72	1,860,060	588	1,979,070	204	2,104,440	204	1,706,800
3rd Quartile	140	1,498,150	425	2,564,675	74	1,906,340	592	1,991,910	408	2,703,910	408	1,863,825
Maximum	457	1,519,950	596	5,968,890	457	2,033,580	608	3,281,150	612	6,258,050	588	3,749,570
Hypervolume Indicator	0.9937		0.9568		0.9232		0.0508		0.9582		0.9794	

These summary statistics further describe the proposed method as the best performing of all examined strategies. In fact, the proposed method boasts the best performance by each summary statistic with respect to solution restoration cost. These summary statistics can be misleading, however. When describing restoration time for each examined strategy, a casual observer may deduce that the Maximum Resource

Strategy competes with the proposed method with respect to restoration time. Indeed, four of the included summary statistics for restoration time exhibit best values achieved by the Maximum Resource Strategy. However, these values are affected by the tight distribution of this strategy's solutions. Discernable from the tight cluster of green diamonds in Figure 16, this dense distribution yields quartile and mean values that are favorable, while each of these Maximum Resource Strategy solutions are in fact inferior to some subset of proposed method solutions.

A powerful method for describing the multiobjective performance of populations of solutions is the hypervolume indicator or *S-metric* (Zitzler & Thiele, 1998; Beume, 2009). This value describes the multiobjective space dominated by at least one member of a population of solutions, bound by some universally dominated reference point. The hypervolume indicator is perhaps the most widely adopted evaluation metric for multiobjective population quality as it is a unary, pareto dominance-compliant performance measure. That is, whenever a population of solutions dominates another, the hypervolume indicator of the former is always larger. Being the case, the hypervolume indicator values disclosed in Table 5 describe the population of the proposed method as the best performing. Note that the reference point used in determining these hypervolume indicators was the combination of the worst objective values observed for any solution of any method during the investigation.

A final exercise seeks to compare the solutions found by the proposed method to a set of hypothetical bounds for restoration time and restoration cost. In this idealized (and assuredly unrealistic) scenario, imagine that all constraints are removed from the model. Then, any node could be repaired at any time, without worrying about its accessibility or

if its precedence considerations have been met. In a scenario of the kind, the total restoration cost could be minimized by utilizing one unit of each resource type. Therein, fixed costs and variable costs are minimized as no constraints are present to force a unit to sit idle. Similarly, the total restoration time could be minimized by utilizing as many resources as possible and applying a parallel resource makespan minimization strategy (Graham et al., 1979; Dessouky et al., 1990). Here, a *resCap* of [10, 10, 10, 10] was used as this was the maximum resource allotment at the disposal of the proposed method.

These methods yielded a best-case bound for the unrestricted case of the disaster scenario at $1,325,150 and 59 days for restoration cost and restoration time, respectively. The most optimal restoration cost value yielded by the proposed method ($1,351,150) is roughly 2% worse than this idealized cost bound. Examining restoration time, the best generated value of the proposed method (65 days) is about 10% worse than its idealized counterpart. As comparing these objectives independently is underinformative, a multiobjective assessment is also made. Relating to the hypervolume indicator, it is of interest to see how much of the multiobjective space between this idealized bound and some reference point is dominated by a set of solutions. Here, again, the reference point is the combination of the worst observed objective values conjured by any method during the investigation is used ($6,258,050, 612 Days). Because this is a completely bounded space, the region dominated by a set of solutions can be represented as a proportion or percentage.

The population of solutions produced by the proposed method is shown to dominate around 97.8% of this space (Figure 18). This high coverage proportion and the proximity of found solutions to the idealized bounds is impressive, considering the varied constraints the proposed method must satisfy. Further, the region left undominated by the proposed

method's solutions may not represent feasible space when the mentioned constraints are considered. While the constraint cognizant bounds of the examined scenario cannot be easily determined, the proposed method's ability to approach and dominate the objective space of the unrealistic bounds established is impressive.

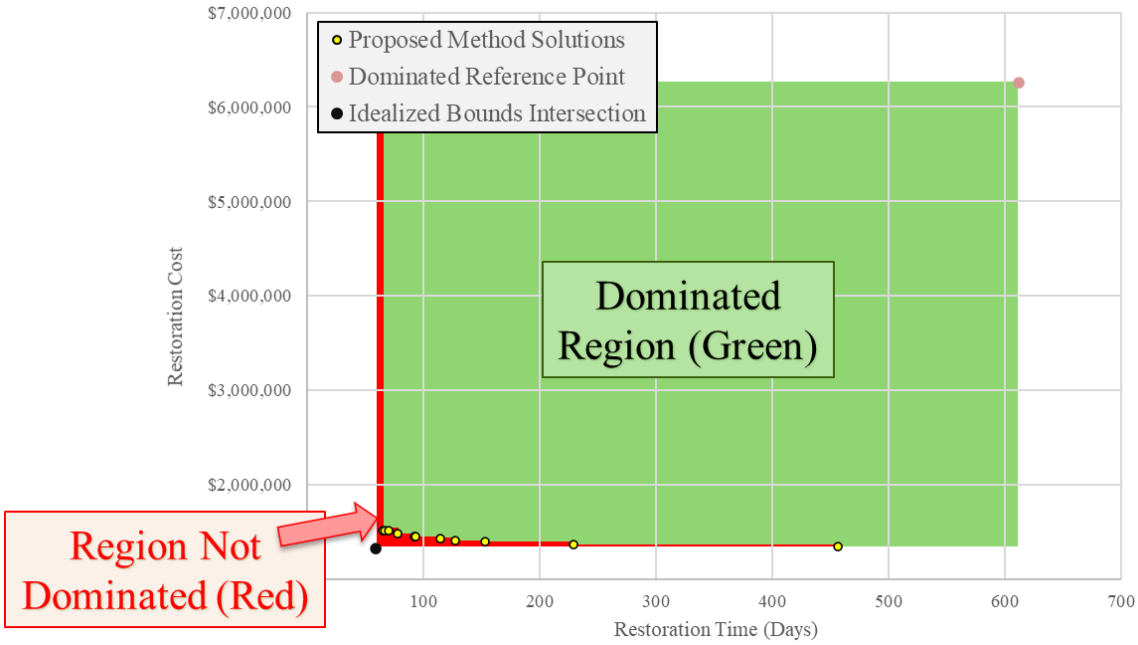


Figure 18. Hypervolume of proposed method's Pareto frontier between unrestricted hypothetical bounds and a universally dominated reference point.

4.4. SCALING TO A REAL-WORLD DISASTER SCENARIO

With the merits of the proposed method demonstrated on a simulated restoration scenario, it is of interest to investigate how the benefits realized scale to real disaster situations. While each disaster scenario will have an intractable number of nuances and intricacies -certainly beyond what is captured by the simple scaling procedure used here-,

this exercise may provide some indication of the magnitude of temporal and monetary savings enabled by the proposed method.

An EF5-Rated, multi-vortex tornado struck the city of Joplin, MO in May of 2011, causing catastrophic damage to buildings and infrastructure. One of the deadliest and most expensive natural disasters in recent U.S. history, the State of Missouri reported the requiring of 25 days to restore the region's critical infrastructure (Ramachandran et al., 2015. The municipality of Joplin additionally reported the utilization of $150 million in completing this restoration (Onstot, 2013).

To scale this real-world situation to the simulated scenario introduced previously, the former's restoration cost and time ($150,000,000, 25 Days) are first mapped to the multiobjective solution space of the simulation. Suppose the procedure followed to restore Joplin's critical infrastructure maps to some point in the simulation space that is in the top 10% of known solutions (i.e., all solutions generated by any method of this investigation) for both restoration cost and time. The average restoration cost and time of solutions within this well-performing region is $1,528,576 and 71 days, respectively. Note that this performance assumption is generous as conflicting objectives often discourage solutions that are universally well-performing. In fact, if we look at the best 10% of known solutions according to restoration cost, their average objective performance is $1,528,576 and 201 days. If we look at the best 10% of known solutions according to restoration time, these objective values are $1,853,667 and 71 days. By assuming the strategy followed in remedying Joplin's damaged infrastructure achieved the best combination of these independent top-decile averages, the competence of models able to find even better solutions is strongly supported.

Assuming this placement of the real-world solution in the simulated objective space, the proposed method is shown capable of producing solutions that can improve both objectives. In fact, multiple solutions generated by the proposed method Pareto dominate this mapped point. The best improvement, along each objective, from this mapped point to a solution generated by the proposed method is also determined. With respect to restoration cost, an improvement of 11.61% to $1,351,150 is discerned. This improvement is 8.45% to 65 days when considering the objective of restoration time. When these percent savings are scaled back to the real-world Joplin disaster, they describe potential savings of roughly $17,415,000 and 2.11 days, respectively. This simple scaling method, though very primitive, gives some indication of the scope of real-world benefits enabled by the utilization of the proposed method.

The noted value for restoration cost savings assumes the entire $150,000,000 dedicated to restoring Joplin's damaged infrastructure was discretionary in nature. Here, *discretionary* is meant to describe those costs that can be controlled, such as the fixed and variable costs considered by the simulation model. These exist in contrast to *unavoidable* expenses: things attached to costs that must be assumed, such as construction materials. This all-discretionary assumption, of course, is not veracious to actual expenditures. While this research does not seek to determine the proportions of Joplin's mentioned cost total that are inevitable versus those that may be impacted by strategy decisions, Figure 19 describes the potential savings possible at different points along this discretionary-unavoidable spectrum.

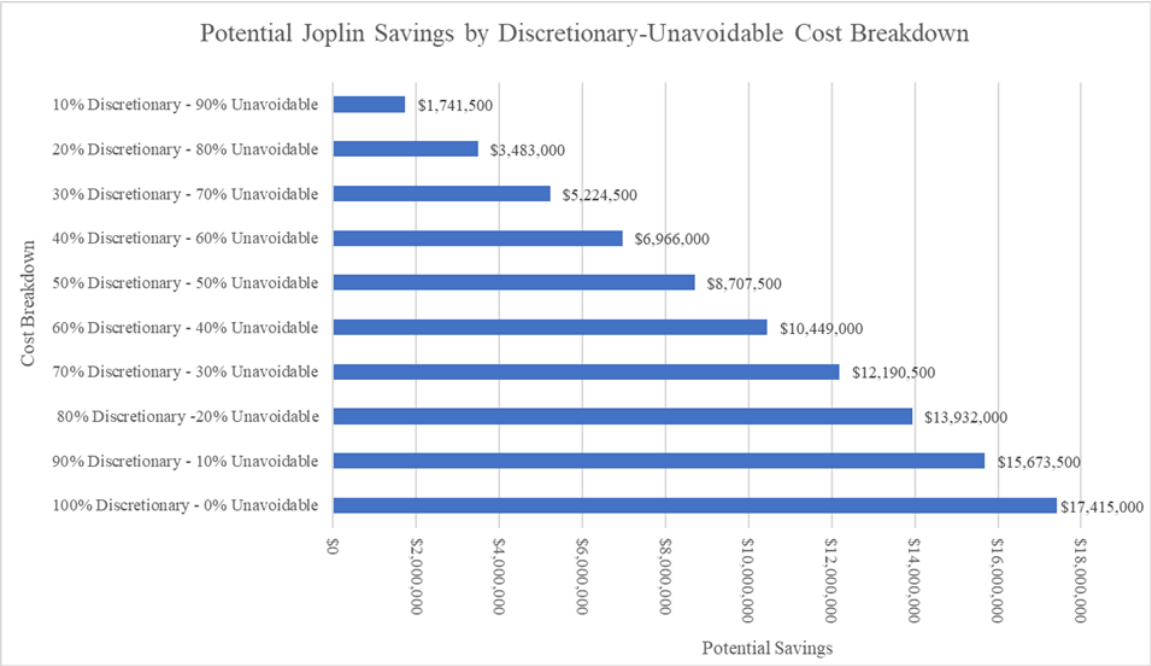


Figure 19. Potential restoration cost savings of the Joplin tornado recovery effort when simulated scenario results are scaled using different discretionary versus unavoidable cost breakdowns.

5. CONCLUSIONS AND FUTURE WORK

Following disaster events, it is crucial that a region's critical infrastructure systems be restored as quickly and affordably as possible. Failure to do so can have serious and lasting negative effects on the well-being of impacted communities and the residents that comprise them. Scheduling the granular set of recovery activities needed to complete this restoration is a challenging task, made difficult by a variety of spatial and precedence relationships incumbent of co-located and interdependent infrastructure systems. Further, differences in the order that these recovery activities are completed can have drastic impact on the length of time and cost of disaster restoration.

This work proposed a method to generate granular and well-performing restoration strategies, seeking to repair an impacted region's disaster damaged infrastructure as affordably and quickly as possible. The multiobjective genetic algorithm with agent-based simulation yielded fitness functions was shown to outperform other strategy generation methods when applied to a simulated disaster. In fact, strategies generated by the proposed method were shown to Pareto dominate all solutions produced by any of the study's competing methods with respect to restoration cost and time. This is preliminary evidence of the utility of the proposed method and hints at the real-world benefits its application could deliver. This method warrants further investigation as enhancing the restoration processes of disaster-impacted areas is a meaningful and consequential endeavor.

While the application here presented is focused on finding well-performing restoration strategies for a given disaster scenario, the proposed methodology could also be used in a resource planning capacity. For a certain region, decision-makers may easily simulate multiple disaster scenarios and determine the performance metrics achieved by different resource allocations. In this way, advance information about the resource levels needed to satisfactorily address differing disaster situations may be assumed. Further, experimentation may be conducted to see how altering specific resource allocations impacts a strategy's position along the cost-time tradeoff curve.

A few obvious extensions and opportunities for further study are also noted. First, this study used entirely deterministic data. The restoration times for damaged infrastructure elements were known with certainty, whereas there is assuredly some stochasticity to these values in real scenarios. Modeling this stochasticity may enable more robust objective value estimates if multiple simulations of each produced strategy

can be run. Further, this model does not allow resources to sit idle if there is an available job for them to complete. While it may seem counterintuitive, the ability to have a resource do nothing (even when feasible jobs exist) may yield reduced overall restoration cost and/or time. Allowing this optional idle state may improve the objective function values that can be attained by a modified model. Lastly, the proposed method is not tied solely to applications of disaster recovery. Its premises and procedures can be applied to any situation where interdependent systems are being constructed, modified, or repaired.

REFERENCES

Adams, Z. W., Danielson, C. K., Sumner, J. A., McCauley, J. L., Cohen, J. R., & Ruggiero, K. J. (2015). Comorbidity of PTSD, major depression, and substance use disorder among adolescent victims of the spring 2011 tornadoes in Alabama and Joplin, Missouri. *Psychiatry*, *78*(2), 170-185.

Almoghathawi, Y., Barker, K., & Albert, L. A. (2019). Resilience-driven restoration model for interdependent infrastructure networks. *Reliability Engineering & System Safety*, *185*, 12-23.

Altay, N., & Green III, W. G. (2006). OR/MS research in disaster operations management. *European journal of operational research*, *175*(1), 475-493.

Applied Technology Council. (1992). A model methodology for assessment of seismic vulnerability and impact distribution of water supply systems. Applied technology council.

Ballantyne, D. B. (1990). Earthquake loss estimation modeling of the Seattle water system. Kennedy/Jenks/Chilton.

Bellmore, M., & Nemhauser, G. L. (1968). The traveling salesman problem: a survey. *Operations Research*, *16*(3), 538-558.

Bentley, J. L. (1990). Experiments on traveling salesman heuristics. In *Proceedings of the first annual ACM-SIAM symposium on discrete algorithms* (pp. 91-99).

Beume, N., Fonseca, C. M., Lopez-Ibanez, M., Paquete, L., & Vahrenhold, J. (2009). On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, *13*(5), 1075-1082.

Bradski, G. (2000). The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(11), 120-123.

Brady, R. J., & Perkins, J. B. (1991). *Macroeconomic effects of the Loma Prieta earthquake*. Oakland, Calif.: Association of Bay Area Governments.

Çağnan, Z., & Davidson, R. (2003). Post-earthquake lifeline service restoration modeling. In *Advancing mitigation technologies and disaster response for lifeline systems* (pp. 255-264).

Carlson, E. B., & Ruzek, J. (2003). Effects of traumatic experiences. *National Center for PTSD, Department of veterans affairs*. (Full Text).

Chang, S. E., Rose, A. Z., Shinozuka, M., Svekla, W. D., & Tierney, K. J. (2000). Modeling earthquake impact on urban lifeline systems: *Advances and integration. Research progress and accomplishments*.

Chang, S. E., Seligson, H. A., & Eguchi, R. T. (1996). Estimation of the economic impact of multiple lifeline disruption: Memphis light, gas, and water division case study.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, *6*(1), 80-91.

Davis, L. (1985). Applying adaptive algorithms to epistatic domains. In *IJCAI* (Vol. 85, pp. 162-164).

Deb, K., & Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex systems*, *9*(2), 115-148.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.

Dessouky, M. I., Lageweg, B. J., Lenstra, J. K., & van de Velde, S. L. (1990). Scheduling identical jobs on uniform parallel machines. *Statistica Neerlandica*, *44*(3), 115-123.

Erdik, M., Şeşetyan, K., Demircioğlu, M. B., Hancılar, U., & Zülfikar, C. (2011). Rapid earthquake loss assessment after damaging earthquakes. *Soil Dynamics and Earthquake Engineering*, *31*(2), 247-266.

Foresti, G. L., Farinosi, M., & Vernier, M. (2015). Situational awareness in smart environments: socio-mobile and sensor data fusion for emergency response to disasters. *Journal of Ambient Intelligence and Humanized Computing*, *6*(2), 239-257.

Galindo, G., & Batta, R. (2013). Review of recent developments in OR/MS research in disaster operations management. *European journal of operational research*, *230*(2), 201-211.

Gong, J. (2013). Mobile lidar data collection and analysis for post-sandy disaster recovery. In *Computing in Civil Engineering (2013)* (pp. 677-684).

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference* (pp. 11–15).

Hale, T., & Moberg, C. R. (2005). Improving supply chain disaster preparedness: A decision process for secure site location. *International Journal of Physical Distribution & Logistics Management*.

Hashemi, M., & Alesheikh, A. A. (2011). A GIS-based earthquake damage assessment and settlement methodology. *Soil dynamics and earthquake engineering*, *31*(11), 1607-1617.

Holguín-Veras, J., & Jaller, M. (2012). Immediate resource requirements after hurricane Katrina. *Natural Hazards Review*, *13*(2), 117-131.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, *267*(1), 66-73.

Horner, M. W., & Widener, M. J. (2011). The effects of transportation network failure on people's accessibility to hurricane disaster relief goods: a modeling approach and application to a Florida case study. *Natural hazards*, *59*(3), 1619-1634.

Houston, J. B., Spialek, M. L., Stevens, J., First, J., Mieseler, V. L., & Pfefferbaum, B. (2015). 2011 Joplin, Missouri tornado experience, mental health reactions, and service utilization: Cross-sectional assessments at approximately 6 months and 2.5 years post-event. *PLoS currents*, *7*.

Isumi, M., Nomura, N., & Shibuya, T. (1985). Simulation of post-earthquake restoration of lifeline systems. *International journal of mass emergencies and disasters*, 3(1), 87-105.

Jiménez Martínez, M., Jiménez Martínez, M., & Romero-Jarén, R. (2020). How resilient is the labour market against natural disaster? Evaluating the effects from the 2010 earthquake in Chile. *Natural Hazards*, *104*(2), 1481-1533.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85-103). Springer, Boston, MA.

Kizilateş, G., & Nuriyeva, F. (2013). On the nearest neighbor algorithms for the traveling salesman problem. In *Advances in Computational Science, Engineering and Information Technology* (pp. 111-118). Springer, Heidelberg.

Kryvasheyeu, Y., Chen, H., Obradovich, N., Moro, E., Van Hentenryck, P., Fowler, J., & Cebrian, M. (2016). Rapid assessment of disaster damage using social media activity. *Science advances*, *2*(3), e1500779.

Lettieri, E., Masella, C., & Radaelli, G. (2009). Disaster management: findings from a systematic review. *Disaster Prevention and Management: An International Journal*.

Lu, X., & Guan, H. (2017). *Earthquake disaster simulation of civil infrastructures*. Beijing: Springer and Science Press.

Madakasira, S., & O'Brien, K. F. (1987). Acute posttraumatic stress disorder in victims of a natural disaster. *Journal of nervous and mental disease*.

Manfrin, M., Birattari, M., Stützle, T., & Dorigo, M. (2006). Parallel ant colony optimization for the traveling salesman problem. In *International workshop on ant colony optimization and swarm intelligence* (pp. 224-234). Springer, Berlin, Heidelberg.

Marshall, T. P. (2002). Tornado damage survey at Moore, Oklahoma. *Weather and forecasting*, *17*(3), 582-598.

Masoomi, H., & van de Lindt, J. W. (2017). Restoration and functionality assessment of a community subjected to tornado hazard. *Structure and Infrastructure Engineering*, 14(3), 275-291.

Masoomi, H., van de Lindt, J. W., & Peek, L. (2018). Quantifying socioeconomic impact of a tornado by estimating population outmigration as a resilience metric at the community level. *Journal of structural engineering*, *144*(5), 04018034.

Minas, J. P., Simpson, N. C., & Tacheva, Z. Y. (2020). Modeling Emergency Response Operations: A Theory Building Survey. *Computers & Operations Research*, 104921.

Missouri University of Science and Technology. (2022). Campus Master Plan. Missouri S&T. Retrieved March 14, 2022, from https://masterplan.mst.edu/

Monnot, J., & Toulouse, S. (2014). The traveling salesman problem and its variations. *Paradigms of combinatorial optimization: problems and new approaches*, 173-214.

Myint, S. W., Yuan, M., Cerveny, R. S., & Giri, C. (2008). Categorizing natural disaster damage assessment using satellite-based geospatial techniques. *Natural Hazards and Earth System Sciences*, *8*(4), 707-719.

Neria, Y., Nandi, A., & Galea, S. (2008). Post-traumatic stress disorder following disasters: a systematic review. *Psychological medicine*, *38*(4), 467-480.

Nojima, N., Ishikawa, Y., Okumura, T., & Sugito, M. (2001). Empirical estimation of lifeline outage time in seismic disaster. I*n Proc. of US-Japan Joint Workshop and Third Grantee Meeting, US-Japan Cooperative Research on Urban Earthquake Disaster Mitigation*, Seattle, WA, USA (pp. 516-527).

Ojha, A. (2019). *Quantifying restoration costs in the aftermath of an extreme event using system dynamics and dynamic mathematical modeling approaches*. Missouri University of Science and Technology. Rolla, MO (USA).

Ojha, A., Corns, S., Shoberg, T., Qin, R., & Long, S. (2018). Modeling and simulation of emergent behavior in transportation infrastructure restoration. *Emergent Behavior in Complex Systems Engineering: A Modeling Simulation Approach*, 349-368.

Ojha, A., Long, S., Shoberg, T., & Corns, S. (2021). Bottom-up resource and cost estimation for restoration of supply chain interdependent critical infrastructure. *Engineering Management Journal*, *33*(4), 272-282.

Oliver, I. M., Smith, D., & Holland, J. R. (1987). Study of permutation crossover operators on the traveling salesman problem. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987.

Onstot, L. I. (2013). Joplin, Missouri hit by EF-5 tornado on May 22, 2011. *Fact Sheet City of Joplin*. https://www.joplinmo.org/DocumentCenter/View/1985/Joplin_Tornado_factsheet

Qu, Z., Wang, C. W., Zhang, X., Ho, A. H., Wang, X., & Chan, C. L. (2014). Prevalence and determinants of depression among survivors 8 months after the Wenchuan earthquake. *The Journal of nervous and mental disease*, *202*(4), 275-279.

Ramachandran, V., Long, S. K., Shoberg, T., Corns, S., & Carlo, H. J. (2015). Framework for modeling urban restoration resilience time in the aftermath of an extreme event. *Natural Hazards Review*, *16*(4), 04015005.

Razali, N. M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering* (Vol. 2, No. 1, pp. 1-6). Hong Kong, China: International Association of Engineers.

Richey, R. G., Natarajarathinam, M., Capar, I., & Narayanan, A. (2009). Managing supply chains in times of crisis: a review of literature and insights. *International Journal of Physical Distribution & Logistics Management*.

Roueche, D. B., & Prevatt, D. O. (2013). Residential damage patterns following the 2011 Tuscaloosa, AL and Joplin, MO tornadoes. *J. Disaster Res*, *8*(6), 1061-1067.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications, 1985*. Lawrence Erlbaum Associates. Inc., Publishers.

Simpson, N. C., & Hancock, P. G. (2009). Fifty years of operational research and emergency response. *Journal of the Operational Research Society*, 60(sup1), S126-S139.

Skiscim, C. C., & Golden, B. L. (1983). *Optimization by simulated annealing: A preliminary computational study for the tsp*. Institute of Electrical and Electronics Engineers (IEEE).

Speheger, D. A., Doswell, C. A., & Stumpf, G. J. (2002). The tornadoes of 3 May 1999: Event verification in central Oklahoma and related issues. *Weather and forecasting*, *17*(3), 362-381.

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221-248.

Steinglass, P., & Gerrity, E. (1990). Natural Disasters and Post-traumatic Stress Disorder Short-Term versus Long-Term Recovery in Two Disaster-Affected Communities 1. *Journal of applied social psychology*, *20*(21), 1746-1765.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons.

Vanfossan, S., & Kwasa, B. (2022). *Ideal Sort: A Terminable, Efficient Nondominated Sorting Algorithm*. Manuscript in preparation.

Verhoeven, M. G. A., Aarts, E. H., & Swinkels, P. C. J. (1995). A parallel 2-opt algorithm for the traveling salesman problem. *Future Generation Computer Systems, 11*(2), 175-182.

Wang, K., Yin, Z., Yuan, F., & Zhou, L. (2005, November). A mathematical approach to disaster recovery planning. *In 2005 First International Conference on Semantics, Knowledge and Grid* (pp. 46-46). IEEE.

Wright, P. D., Liberatore, M. J., & Nydick, R. L. (2006). A survey of operations research models and applications in homeland security. *Interfaces*, 36(6), 514-529.

Wu, D., & Cui, Y. (2018). Disaster early warning and damage assessment analysis using social media data and geo-location information. *Decision support systems, 111*, 48-59.

Zhang, R. H. (1992). Lifeline interaction and post-earthquake urban system reconstruction. *In Proceedings of 10th World Conference on Earthquake Engineering* (pp. 5475-5480).

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature* (pp. 292-301). Springer, Berlin, Heidelberg.

SECTION

3. CONCLUSIONS AND FUTURE WORK

Multiobjective optimization is a powerful and accommodating tool, applicable to an incredible variety of real-world scenarios. As the simultaneous consideration of conflicting objectives usually disallows the existence of a globally optimum solution, several creative techniques have been developed to resolve multiobjective optimization problems. Perhaps the most widely applied is a class of a posteriori approaches utilizing evolutionary algorithms to generate a set of well-distributed Pareto efficient solutions. Specifically, evolutionary algorithms using a nondominated sorting ranking procedure have become standard approaches. In repeated investigations, these models have demonstrated admirable performance and garnered considerable practical, and academic, application.

While well-performing, a primary criticism of these methods is the computational complexity of the nondominated sorting procedure needed to evaluate solution alternatives. The burden of this resource intensive procedure has often limited the scope of scenarios to which nondominated sorting evolutionary algorithms can be applied in a reasonable time. Towards a remedy, considerable effort has been dedicated to improving the computational complexity of nondominated sorting algorithms. Success along this directive has seen the applicability of these evolutionary procedures grow considerably.

This scope enhancement lends to the primary research edicts of this work. First, this work has sought to improve the computational complexity of nondominated sorting

evolutionary algorithms. The included investigations have introduced new nondominated sorting mechanics, shown to achieve state-of-the-art runtime performance in some instances. With the improvements here introduced, the range of multiobjective optimization scenarios to which evolutionary algorithms may be applied is expanded. Through incremental works of the kind, more challenging and larger-scale problems may be tackled by more the computationally affordable optimization proceedings enabled.

Additionally, the second mandate of this work was to demonstrate the efficacy of multiobjective evolutionary algorithms when applied to challenging optimization scenarios. Here, an instance of this approach-class was tasked with the multiobjective consideration of a challenging permutation problem, further mired by numerous feasibility and resource constraints. Demonstrating objective success, beyond what was achieved by other intuitive strategies, another chapter has been added to the pedigree of evolutionary algorithm utility. Researches of this kind are important as the capabilities they describe sponsor additional efforts to improve the procedures of evolutionary algorithms, including nondominated sorting complexity.

These cooperative research directives serve well to further the study and utility of multiobjective evolutionary algorithms. Equipped with these powerful and innovative methods, the pervasive scenario of multiobjective optimization can be more intelligently and successfully addressed. Indeed, efforts improving the capabilities of multiobjective optimization work to improve the decision-making processes that impact and drive everyday life.

While several research extensions have been identified in individual contributions, the two reasoned to be most impactful are reiterated, here. A principal

contribution of PAPER II was an introduction of the concept of terminability, by which a nondominated sorting procedure may be stopped after a desired number of solutions have been assigned to their appropriate Pareto fronts. Allowing the procedure to sidestep the unnecessary sorting of highly dominated solutions that will not survive to the next generation of the evolutionary algorithm, the algorithmic runtime of both algorithms (nondominated sorting and the broader genetic algorithm) may be improved. This concept of terminability was shown to considerably improve runtime performance of the inferred dominance methods to which it was applied. This enhancement allowed methods of this inferred dominance class to outperform the otherwise superior class of constructive front nondominated sorting methods, in many instances. This sponsors the desire to modify strategies of the constructive front method class to utilize terminability, themselves. While some creativity may be required to achieve this objective, successful attempts may be very well worth the effort.

Additionally, PAPER III introduced an evolutionary approach to generate multidimensional network restoration solutions under a variety of precedence and other feasibility constraints. While the case study of infrastructure network recovery following a natural disaster was used, this approach may be applied to any scenario where dependent combinatorial networks are to be built or repaired. Applying the methods of the approach demonstrated may prove valuable in a variety of fields where scenarios of this kind arise.

BIBLIOGRAPHY

Athan, T. W., & Papalambros, P. Y. (1996). A note on weighted criteria methods for compromise solutions in multi-objective optimization. *Engineering optimization*, *27*(2), 155-176.

Aubry, J. F., Beaulieu, F., Sévigny, C., Beaulieu, L., & Tremblay, D. (2006). Multiobjective optimization with a modified simulated annealing algorithm for external beam radiotherapy treatment planning. *Medical physics*, *33*(12), 4718-4729.

Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE transactions on evolutionary computation*, *12*(3), 269-283.

Branke, J., Branke, J., Deb, K., Miettinen, K., & Slowiński, R. (Eds.). (2008). *Multiobjective optimization: Interactive and evolutionary approaches* (Vol. 5252). Springer Science & Business Media.

Bridgman, P. W. (1922). *Dimensional analysis*. Yale university press.

Carmichael, D. G. (1980). Computation of Pareto optima in structural design. *International Journal for Numerical Methods in Engineering*, *15*(6), 925-929.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, *45*(1), 41-51.

Chan, T. C., Craig, T., Lee, T., & Sharpe, M. B. (2014). Generalized inverse multiobjective optimization with application to cancer therapy. *Operations Research*, *62*(3), 680-695.

Charnes, A., & Cooper, W. W. (1961). *Management models and industrial applications of linear programming*. John Wiley & Sons.

Charnes, A., & Cooper, W. W. (1977). Goal programming and multiple objective optimizations: Part 1. *European journal of operational research*, *1*(1), 39-54.

Charnes, A., Clower, R. W., & Kortanek, K. O. (1967). Effective control through coherent decentralization with preemptive goals. *Econometrica: Journal of the Econometric Society*, 294-320.

Charnes, A., Cooper, W. W., & Ferguson, R. O. (1955). Optimal estimation of executive compensation by linear programming. *Management science*, *1*(2), 138-151.

Cheng, F. Y., & Li, D. (1996). Multiobjective optimization of structures with and without control. *Journal of Guidance, Control, and Dynamics*, *19*(2), 392-397.

Chiam, S. C., Al Mamun, A., & Low, Y. L. (2007, September). A realistic approach to evolutionary multiobjective portfolio optimization. In *2007 IEEE Congress on Evolutionary Computation* (pp. 204-211). IEEE.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on evolutionary computation*, *8*(3), 256-279.

Cohon, J. L. (2004). *Multiobjective programming and planning* (Vol. 140). Courier Corporation.

Corley, H. (1980). A new scalar equivalence for Pareto optimization. *IEEE Transactions on Automatic Control*, *25*(4), 829-830.

Corne, D. W., Jerram, N. R., Knowles, J. D., & Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd annual conference on genetic and evolutionary computation* (pp. 283-290).

Das, I., & Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural optimization*, *14*(1), 63-69.

Das, I., & Dennis, J. E. (1998). Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, *8*(3), 631-657.

Dauer, J. P., & Krueger, R. J. (1980). A multiobjective optimization model for water resources planning. *Applied Mathematical Modelling*, *4*(3), 171-175.

Dauert, J. (1992). Multicriteria optimization in engineering: a tutorial and survey. *Progress In Astronautics and Aeronautics: Structural Optimization: Status and Promise*, *150*, 209.

Deb, K., & Jain, H. (2013). 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*(4), 577-601.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, *6*(2), 182-197.

Dennis, R. (2002). *Inferring policy objectives from policy actions*. Federal Reserve Bank of San Francisco. Retrieved February 27, 2022, from https://www.frbsf.org/economic-research/publications/economic-letter/2002/april/inferring-policy-objectives-from-policy-actions/

Domenico, A., Nicola, G., Daniela, T., Fulvio, C., Nicola, A., & Orazio, N. (2020). De novo drug design of targeted chemical libraries based on artificial intelligence and pair-based multiobjective optimization. *Journal of Chemical Information and Modeling*, *60*(10), 4582-4593.

Federal Reserve Board. (2021). *Monetary Policy: What Are Its Goals? How Does It Work?* Monetary Policy Principles and Practice. Retrieved February 27, 2022, from https://www.federalreserve.gov/monetarypolicy/monetary-policy-what-are-its-goals-how-does-it-work.htm

Flores-Alsina, X., Rodríguez-Roda, I., Sin, G., & Gernaey, K. V. (2008). Multi-criteria evaluation of wastewater treatment plant control strategies under uncertainty. *Water research*, *42*(17), 4485-4497.

Gembicki, F. W. (1974). *Performance and Sensitivity Optimization: A Vector-Index Approach*. Case Western Reserve University. Cleveland, OH (USA).

Gerasimov, E. N., & Repko, V. N. (1978). Multicriterial optimization. *Soviet applied mechanics*, *14*(11), 1179-1184.

Goicoechea, A., Duckstein, L., & Fogel, M. M. (1976). Multiobjective programing in watershed management: A study of the charleston watershed. *Water Resources Research*, *12*(6), 1085-1092.

Haimes, Y. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics*, *1*(3), 296-297.

Hitch, C. (1953). Sub-optimization in operations problems. *Journal of the Operations Research Society of America*, *1*(3), 87-99.

Holdsworth, C., Kim, M., Liao, J., & Phillips, M. H. (2010). A hierarchical evolutionary algorithm for multiobjective optimization in IMRT. *Medical physics*, *37*(9), 4986-4997.

Holdsworth, C., Stewart, R. D., Kim, M., Liao, J., & Phillips, M. H. (2011). Investigation of effective decision criteria for multiobjective optimization in IMRT. *Medical Physics*, *38*(6Part1), 2964-2974.

Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994, June). A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence* (pp. 82-87). IEEE.

Hu, X., & Eberhart, R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1677-1681). IEEE.

Hwang, C. L., & Masud, A. S. M. (2012). *Multiple objective decision making methods and applications: a state-of-the-art survey* (Vol. 164). Springer Science & Business Media.

Hwang, C. L., Lai, Y. J., & Liu, T. Y. (1993). A new approach for multiple objective decision making. *Computers & operations research*, *20*(8), 889-899.

Ijiri, Y. (1965). *Management goals and accounting for control*. North Holland Publishing Company.

Janson, S., & Merkle, D. (2005, August). A new multi-objective particle swarm optimization algorithm using clustering applied to automated docking. In *International Workshop on Hybrid Metaheuristics* (pp. 128-141). Springer, Berlin, Heidelberg.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). IEEE.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. S*cience*, *220*(4598), 671-680.

Kong, D. S., Jang, Y. S., & Huh, J. H. (2015). Method and case study of multiobjective optimization-based energy system design to minimize the primary energy use and initial investment cost. *Energies*, *8*(6), 6114-6134.

Lin, J. (1976). Multiple-objective problems: Pareto-optimal solutions by method of proper equality constraints. *IEEE Transactions on Automatic Control*, *21*(5), 641-650.

Lundblad, C. (2007). The risk return tradeoff in the long run: 1836–2003. *Journal of Financial Economics*, *85*(1), 123-150.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, *26*(6), 369-395.

Mazumdar, R., Mason, L. G., & Douligeris, C. (1991). Fairness in network optimal flow control: optimality of product forms. *IEEE Transactions on communications*, *39*(5), 775-782.

McClymont, K., & Keedwell, E. (2012). Deductive sort and climbing sort: New methods for non-dominated sorting. *Evolutionary computation*, *20*(1), 1-26.

Messac, A., & Mattson, C. A. (2002). Generating well-distributed sets of Pareto points for engineering design using physical programming. *Optimization and Engineering*, *3*(4), 431-450.

Messac, A., Ismail-Yahaya, A., & Mattson, C. A. (2003). The normalized normal constraint method for generating the Pareto frontier. *Structural and multidisciplinary optimization*, *25*(2), 86-98.

Miettinen, K. (2012). *Nonlinear multiobjective optimization* (Vol. 12). Springer Science & Business Media.

Miettinen, K., & Mäkelä, M. M. (2006). Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research*, *170*(3), 909-922.

Miettinen, K., Ruiz, F., & Wierzbicki, A. P. (2008). Introduction to multiobjective optimization: interactive approaches. In *Multiobjective optimization* (pp. 27-57). Springer, Berlin, Heidelberg.

Mishra, S., Mondal, S., Saha, S., & Coello, C. A. C. (2018). GBOS: generalized best order sort algorithm for non-dominated sorting. *Swarm and Evolutionary Computation*, *43*, 244-264.

Mukerjee, A., Biswas, R., Deb, K., & Mathur, A. P. (2002). Multi–objective evolutionary algorithms for the risk–return trade–off in bank loan management. *International Transactions in operational research*, *9*(5), 583-597.

Murata, T., & Ishibuchi, H. (1995, November). MOGA: multi-objective genetic algorithms. In *IEEE international conference on evolutionary computation* (Vol. 1, pp. 289-294). Piscataway, NJ, USA: IEEE.

Nicolaou, C. A., & Brown, N. (2013). Multi-objective optimization methods in drug design. *Drug Discovery Today: Technologies*, *10*(3), e427-e435.

Nicolaou, C. A., Kannas, C., & Loizidou, E. (2012). Multi-objective optimization methods in de novo drug design. *Mini reviews in medicinal chemistry*, *12*(10), 979-987.

Nicolotti, O., Giangreco, I., Introcaso, A., Leonetti, F., Stefanachi, A., & Carotti, A. (2011). Strategies of multi-objective optimization in drug discovery and development. *Expert opinion on drug discovery*, *6*(9), 871-884.

Ogryczak, W. (1994). A goal programming model of the reference point method. *Annals of Operations Research*, *51*(1), 33-44.

Pulido, G. T., & Coello Coello, C. A. (2004). Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In *Genetic and Evolutionary Computation Conference* (pp. 225-237). Springer, Berlin, Heidelberg.

Rao, J. R., & Roy, N. (1989). Fuzzy set theoretic approach of assigning weights to objectives in multicriteria decision making. *International journal of systems science*, *20*(8), 1381-1386.

Rao, S. S. (1987). Game theory approach for multiobjective structural optimization. *Computers & Structures*, *25*(1), 119-127.

Rao, S. S., & Freiheit, T. I. (1991). A modified game theory approach to multiobjective optimization.

Roy, P. C., Deb, K., & Islam, M. M. (2018). An efficient nondominated sorting algorithm for large number of fronts. *IEEE transactions on cybernetics*, *49*(3), 859-869.

Roy, P. C., Islam, M. M., & Deb, K. (2016, July). Best order sort: a new algorithm to non-dominated sorting for evolutionary multi-objective optimization. In *Proceedings of the 2016 on genetic and evolutionary computation conference companion* (pp. 1113-1120).

Saaty, T. L. (1977). A scaling method for priorities in hierarchical structures. *Journal of mathematical psychology*, *15*(3), 234-281.

Saborido, R., Ruiz, A. B., Bermudez, J. D., Vercher, E., & Luque, M. (2016). Evolutionary multi-objective optimization algorithms for fuzzy portfolio selection. *Applied soft computing*, *39*, 48-63.

Schaffer, J. D. (1985). *Some experiments in machine learning using vector evaluated genetic algorithms*. Vanderbilt Univ., Nashville, TN (USA).

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221-248.

Stadler, W. (1988). Fundamentals of multicriteria optimization. In *Multicriteria Optimization in Engineering and in the Sciences* (pp. 1-25). Springer, Boston, MA.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley & Sons.

Subbu, R., Bonissone, P. P., Eklund, N., Bollapragada, S., & Chalermkraivuth, K. (2005). Multiobjective financial portfolio design: A hybrid evolutionary approach. In *2005 IEEE Congress on Evolutionary Computation* (Vol. 2, pp. 1722-1729). IEEE.

Suppapitnarm, A., Seffen, K. A., Parks, G. T., & Clarkson, P. J. (2000). A simulated annealing algorithm for multiobjective optimization. *Engineering optimization*, *33*(1), 59-85.

Sweetapple, C., Fu, G., & Butler, D. (2014). Multi-objective optimisation of wastewater treatment plant control to reduce greenhouse gas emissions. *Water Research*, *55*, 52-62.

Tang, S., Cai, Z., & Zheng, J. (2008, October). A fast method of constructing the non-dominated set: arena's principle. In *2008 Fourth International Conference on Natural Computation* (Vol. 1, pp. 391-395). IEEE.

Torres-Mendoza, Y., Kerr, A., Schnall, A. H., Blackmore, C., & Hartley, S. D. (2021). Community Assessment for Mental and Physical Health Effects After Hurricane Irma—Florida Keys, May 2019. *Morbidity and Mortality Weekly Report*, *70*(26), 937.

Wang, H., & Yao, X. (2013). Corner sort for Pareto-based many-objective optimization. *IEEE transactions on cybernetics*, *44*(1), 92-102.

Wang, Z., Ong, Y. S., Sun, J., Gupta, A., & Zhang, Q. (2018). A generator for multiobjective test problems with difficult-to-approximate pareto front boundaries. *IEEE Transactions on Evolutionary Computation*, *23*(4), 556-571.

Wendell, R. E., & Lee, D. N. (1977). Efficiency in multiple objective optimization problems. *Mathematical programming*, *12*(1), 406-414.

Wierzbicki, A. P. (1986). A methodological approach to comparing parametric characterizations of efficient solutions. In *Large-scale modelling and interactive decision analysis* (pp. 27-45). Springer, Berlin, Heidelberg.

Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K., & Liu, B. (2018). Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access*, *6*, 41256-41279.

Yoon, K. (1980). *Systems selection by multiple attribute decision making*. Kansas State University. Manhattan, KS (USA).

Yoon, K. P., & Hwang, C. L. (1995). *Multiple attribute decision making: an introduction*. Sage publications.

Yu, Y., Zhang, J. B., Cheng, G., Schell, M. C., & Okunieff, P. (2000). Multi-objective optimization in radiotherapy: applications to stereotactic radiosurgery and prostate brachytherapy. *Artificial Intelligence in Medicine*, *19*(1), 39-51.

Zadeh, L. (1963). Optimality and non-scalar-valued performance criteria. *IEEE transactions on Automatic Control*, *8*(1), 59-60.

Zhang, X., Tian, Y., Cheng, R., & Jin, Y. (2014). An efficient approach to nondominated sorting for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, *19*(2), 201-213.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, *3*(4), 257-271.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, *103*.

VITA

Samuel Alexander Vanfossan was raised in Excelsior Springs, MO, USA where he attended Excelsior Springs High School, graduating in May of 2013. He went on to graduate summa cum laude from Missouri University of Science and Technology in May of 2018 with a B.S. in Engineering Management: Industrial Engineering. Immediately following graduation, Samuel began Ph.D. studies at Missouri University of Science and Technology as a Chancellor's Distinguished Fellow, receiving a Ph.D. in Systems Engineering in May of 2022. During doctoral studies, his research interests included multiobjective and heuristic optimization, applied machine learning, and complex system modeling and simulation. Samuel served in a variety of capacities during his graduate career, including as a graduate research assistant within the University's Virtual and Augmented Systems Engineering Laboratory, and as the instructor of record for several graduate and undergraduate courses including Operations and Production Management, Operations Management Science, and Introduction to Operations Research.