Progress in Nuclear Energy 138 (2021) 103830

Contents lists available at ScienceDirect

Progress in Nuclear Energy

journal homepage: www.elsevier.com/locate/pnucene

Review

A survey of multi-objective optimization methods and their applications for nuclear scientists and engineers

Ryan H. Stewart *, Todd S. Palmer, Bryony DuPont

Oregon State University, 1500 SW Jefferson St., Corvallis, OR 97331, United States of America

ARTICLE INFO

Keywords:
Multi-objective problem
Optimization
Nuclear engineering

ABSTRACT

Problems in nuclear engineering – such as reactor core design – involve a multitude of design variables including fuel or assembly configurations; all of which require careful consideration when constrained by a set of objectives such as fuel temperature or assembly power density. Reactor design is one facet of nuclear engineering, where many nuclear engineers often face large multi-objective problems to solve. These types of problems can be solved by relying upon experts to aid in reducing the design space required for multi-objective optimization, however, computational optimization algorithms have been used to generate optimal solutions with reproducibility and quantitative evidence for designs. We present a review of multi-objective optimization literature including an introduction to optimization theory, commonly used multi-objective optimization algorithms, and current applications in nuclear science and engineering. From this review, researchers will glean an understanding of multi-objective optimization algorithms that are currently available, and gain a fundamental understanding of how to apply these techniques to a wide variety of problems in the fields of nuclear science and engineering.

1. Introduction

1.1. Background

Multi-objective optimization (MOO) algorithms have been around for over fifty years, and have flourished in recent decades due to continually increasing computational power. This increase in computational power has allowed many fields of engineering to explore how MOO algorithms can aid in designing better products, improving manufacturing techniques, and cutting costs. Many researchers have also leveraged these techniques to perform high-quality and cutting-edge research.

A variety of MOO algorithms have been developed, refined, and presented in literature. Fundamentally, multi-objective optimization problems have two major components: design variables, and objectives. Design variables are the independent variables that the designer has control over, whereas the objectives are the mathematical representation of the designer's optimal solution. Optimization methods are often divided into two categories: gradient and non-gradient. Gradient methods require continuous, well-behaved mathematical representations of the objectives being optimized. These properties are not always present for engineering problems, where discontinuities or poorly behaved relationships may exist between the design variables and the objectives. This can limit the usefulness of gradient methods for many engineering applications; as a result, this paper focuses instead on commonly used non-gradient MOO algorithms. Non-gradient algorithms are further broken down into two categories: classical and meta-heuristic (modern) techniques, which will be described in later sections of this paper.

Optimization algorithms allow designers to examine the affect design variables have on a set of objectives. For some problems, there is a single objective that dictates if a design is feasible; these are denoted as single-objective problems. MOO algorithms allow designers to examine competing objectives and determine what compromises are necessary to create a valid design. This allows the designer to evaluate a set of optimal designs and determine which design aligns with their preferences and goals.

Allowing multiple objectives to define a problem often forces compromise between these objectives. When a large number of objectives are considered it becomes impractical to optimize a problem without the aid of area experts, or a MOO algorithm. Previously, the task of optimizing a problem was left to experts in the field who have years of experience to guide their engineering judgment. While there can be inherent benefits to utilizing this knowledge, researchers who do not have access to this information often have to reinvent previous research. This fact is especially true for the field of nuclear engineering, where large volumes of work are proprietary, inhibiting the ability of researchers to build on previous work in a meaningful way.

* Corresponding author.
E-mail address: stewryan@oregonstate.edu (R.H. Stewart).

https://doi.org/10.1016/j.pnucene.2021.103830
Received 3 November 2020; Received in revised form 7 May 2021; Accepted 7 June 2021
Available online 17 June 2021

Fig. 1 shows the number peer reviewed journals containing 'multi-objective optimization' for various engineering professions over the past twenty years. The search was performed using ScienceDirect, where only journal article are examined and the top 25 journals are presented. For each engineering discipline (except nuclear) the search criteria followed the pattern: *(multi-objective optimization AND <engineering discipline>) AND NOT nuclear*. We exclude cross-listings for nuclear in other fields to prevent double counting. The search criteria for nuclear was *(multi-objective optimization AND nuclear)*. This search was meant to be as fair as possible between the various fields of engineering, however, there are other factors which could impact these statistics. Specifically, the number of engineers in each discipline would likely have a large impact on the number of publications presented. Despite these factors, we can make some general comments about these trends.

The nuclear field has steadily increased its use of MOO, however, the rate at which other fields have applied MOO exceeds the nuclear field over the past ten years. The task of making and justifying engineering decisions in many other fields appears to be transitioning towards computational optimization techniques. These techniques utilize data obtained from simulations and/or experiments to quantify a design decision, and select the best possible designs considering the multiple constraints placed on the system. This process allows a designer to have an optimal set of solutions to select from which are both reproducible and provide quantitative evidence to support their decision making. In this paper, we introduce the fundamental concepts of MOO and introduce a set of commonly used MOO algorithms. Following this, current research utilizing MOO algorithms in nuclear science and engineering is presented to illustrate the insights MOO algorithms currently provide.

1.2. Multi-objective optimization theory

In a multi-objective optimization problem, a designer attempts to find the 'best' solution(s) to a problem by maximizing or minimizing a set of objectives. To express this mathematically, a few definitions are required. Consider an optimization problem with n design variables, which a designer has direct control over. These design variables are allowed to be continuous (i.e. fuel height) or discrete (i.e. number of fuel pins) and are used to create a unique design. The design variables are designated as $x_1, x_2, \dots, x_n$; where $\mathbf{x}$ is the vector containing all n design variables. These variables create a design/parameter space, denoted as D, where D is in some real coordinate space ($D \in R^n$).

There is a corresponding objective space O, where O is also in some real coordinate space ($O \in R^k$). The objective space is spanned by a set of k objective functions ($f_k(\mathbf{x})$), which are a function of our design variables $\mathbf{x}$. Similar to the design variables, the objective functions can be continuous (i.e. k_{eff}) or discrete (i.e. total number of assemblies). The objective functions are dependent variables, which the algorithm can influence via manipulation of the design variables. A designer has influence over the form of the 'optimal' solution in the selection of the objective functions. The collection of objective functions ($f_k(\mathbf{x})$) for an instance of the design variables ($\mathbf{x}$) is called the objective vector ($\mathbf{F}$), as seen in Eq. (1).

$$\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))^T \tag{1}$$

The objective vector acts as a mapping function between the design space (D) and the objective space O ($\mathbf{F} : D \rightarrow O$) (Chiandussi et al., 2012). A set of design variables determine the values of the objective functions, which creates a solution in the form of an objective vector. The relationship between the design and objective space is the fundamental aspect optimization algorithms attempt to explore. For an optimization problem, the designer seeks a set of solutions (objective vectors) which are optimal within the objective space. The standard form of a multi-objective optimization problem is given by Eq. (2) (Arora, 2013). In the literature, this is described as a minimization problem, where we seek a set of design variables which minimizes the objective vector. Along with our objective functions, additional constraints may be placed on the problem in the form of m inequality constraints (g_j), and e equality constraints (h_l). These constraints can augment the existing objective functions (i.e. bound an objective function), or introduce a new constraining variable.

$$\begin{aligned} &min\ \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))^T \\ &\text{subject to } g_j(\mathbf{x}) \le 0, \quad j = 1, 2, \dots, m \\ &\qquad\qquad h_l(\mathbf{x}) = 0, \quad l = 1, 2, \dots, e \end{aligned} \tag{2}$$

Each objective vector ($\mathbf{F}$) is a solution to the MOO problem, where an optimal design ($\hat{\mathbf{x}}$) consists of an objective vector with minimized values for each of the objective functions. After the optimization process, a set of objective vectors are obtained, and the designer may select the most appropriate design based on this information.

Fig. 2 shows a graphical representation of the design and objective space. On the left hand side, we present a design space ($D \in R^n$) comprised of two variables x_1 and x_2, where we have selected four designs denoted **a**, **b**, **c**, and **d**. The right hand side represents the objective space ($O \in R^k$), which is comprised of two objectives f_1 and f_2. To denote the minimum for each objective function within the objective space, the nomenclature f_k^* is used. The objective vectors for each of the four designs are denoted as $\mathbf{F}_a$, $\mathbf{F}_b$, $\mathbf{F}_c$, and $\mathbf{F}_d$ and the Utopian solution is given by $\mathbf{F}^*$. The Utopian solution represents a vector where each objective function is at its minima ($\mathbf{F}^* = (f_1^*, f_2^*, \dots, f_n^*)^T$).

In an optimization problem, solutions are sought which are close to the Utopian solution, where closeness can be described in multiple ways depending on the algorithm. For the example in Fig. 2, we describe the thick black line as solutions which are within our objective space, and close to the Utopian solution. This line is referred to as the Pareto front (P), which is used to designate non-dominated solutions within the objective space. A non-dominated solution can be described by looking at two separate designs, $\mathbf{x}$ and $\mathbf{y}$. If $\mathbf{F}(\mathbf{x})$ is equal to or less than all the objective functions in solution $\mathbf{F}(\mathbf{y})$ (i.e. $f_i(\mathbf{x}) \le f_i(\mathbf{y})$), and strictly less in at least one objective function (i.e. $f_i(\mathbf{x}) < f_i(\mathbf{y})$), $\mathbf{y}$ is said to be dominated by solution $\mathbf{x}$.

The locations of designs **a**, **b**, **c**, and **d** in R^n are displayed in Fig. 2, along with their corresponding objective vectors in R^k. $\mathbf{F}_a$ is found outside of the acceptable objective space O, which denotes that the $\mathbf{F}_a$ does not meet the requirements set by the objective functions and is not a suitable design. The other solutions are all within the objective space, and require a more detailed discussion. We first examine $\mathbf{F}_c$ in its relation to $\mathbf{F}_b$. $\mathbf{F}_b$ is equal to $\mathbf{F}_c$ in attribute f_1 and closer to $\mathbf{F}^*$ in attribute f_2, thus $\mathbf{F}_c$ is dominated by $\mathbf{F}_b$. While $\mathbf{F}_c$ is a viable solution within the objective space, there are other solutions simultaneously increase an objective without degrading the others.

A point of interest lies in the difference between $\mathbf{F}_b$ and $\mathbf{F}_d$, where both points are on the Pareto front. A question naturally arises; which solution is more optimal? For a MOO problem there is no single optimal solution, instead the Pareto front provides a set of non-dominated solutions. These solutions are all considered optimal, where they contain compromises between the multiple objectives. It is up to the designer to determine what the 'best' solution is for their problem. Typically this will be determined by examining the Pareto front and determining which of these solutions fits the designer's needs. This can be done in a multitude of different ways, and may require further optimization if too many solutions are presented as optimal.

For many optimization problems there is no global optimal solution (i.e. a solution which is able to simultaneously optimize all of the objectives). While there is typically no global optima, there can be local minima. Local minima are solutions which appear optimal, but do not actually provide the most optimal solutions once additional portions of the objective space are discovered. This concept is important to differentiate from the Pareto front, as some optimization algorithms can converge prematurely to local optima if proper care is not taken.

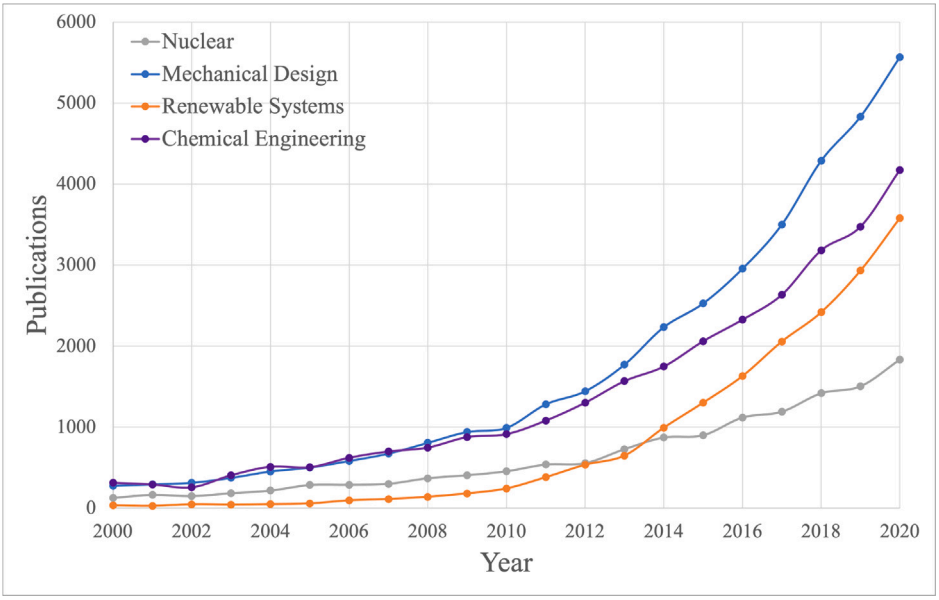

Fig. 1. Publications in Elsevier journals containing 'Multi-Objective Optimization' for various engineering disciplines.

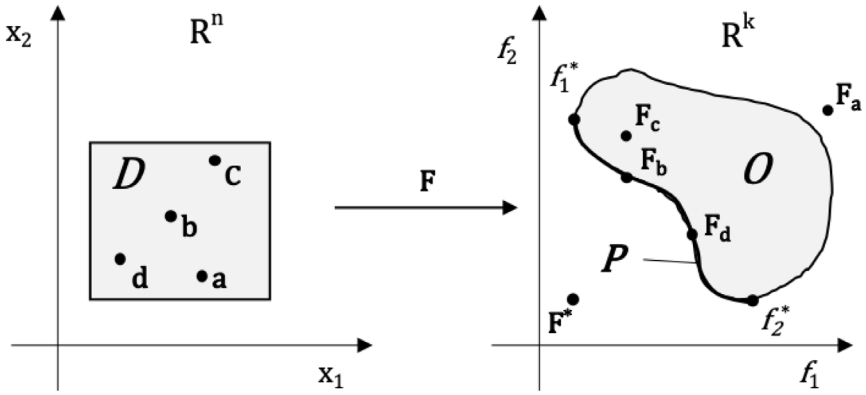

Fig. 2. Design and objective space for a two dimensional optimization problem.

1.3. Construction of a multi-objective optimization problem

To provide a basis of understanding for the remainder of the paper, we present how MOO problems are typically formulated. We break up the construction into three major components: generating a problem statement, selecting and setting up an optimization algorithm, and determining how objectives/constraints are calculated.

Formulating a problem statement involves selecting the design variables, objective functions, and constraints. For design variables we are interested in how many variables are present, what ranges or options are placed on them, and what type of variable they are. The number of variables represent an aspect of the degrees of freedom in the problem. Increasing the number of variables likely provides greater diversity in the number of designs that will be present on the PF. This typically comes at the cost of increased difficulties in finding the true PF. However, understanding the effect that each design variable has on the design can allow users to reduce ranges for each design variable or remove design variables that have little to no effect on the design. This can help maximize the diversity of designs, while minimizing the number of design variables. Similar to this, determining an appropriate range for design variables can have a substantive impact on the number of designs present on the Pareto front. Typically, this will be bounded by physical restrictions for each design variable. The last aspect to consider for design variables is their form; namely, if the design variables are discrete or continuous. The form that design variables take can have important implications on algorithm selection, as some algorithms cannot handle discrete design variables or require specific implementations.

Increasing the number of objective functions will increase the dimensions of the PF, as we are increasing the degrees of freedom for the problem. This can strain optimization algorithms as we are attempting to manage multiple objective functions at one time and finding an optimal solution may be computationally inefficient. However, if we can intelligently reduce the number of objective functions in the problem then we can still provide a diverse set of designs, while minimize the resources required to solve the problem. For example, if we place an objective function on the problem whose values do not change much within the design space then we could likely remove this objective function or simply place it as a constraint instead.

For constraints, we are interested in the number of constraints, type of constraint (equality or inequality), ranges of these constraints, and the implications of violating a constraint. The number and ranges of constraints can have a significant impact on feasibility of finding designs. While constraints can help narrow down to more desirable designs, it can also constrict the design space and return non-optimal designs if too strict of constraints are placed on the problem. If little is known about the objective space or how the constraints will affect the problem, it is typically advantageous to break the problem into two

parts. For the first part, practitioners could use a sensitivity study or broad optimization algorithm to determine the size of the objective space and the effects of constraints on this space. Once this is determined, a full optimization study could be performed with adequately selected objectives and constraints. The final aspect to consider is how the constraints will affect the optimization algorithm. Depending on the implementation of an optimization algorithm, some will not consider solutions with constraint violations, while others my apply a penalty to the quality of the solution. Preventing designs with constraint violations can significantly reduce the ability of an optimization algorithm to find a PF with truly optimal solutions. Instead, the optimization algorithm may end up selecting less optimal designs with no constraint violations. We also note that constraints can be used to limit ranges of the objective functions if specific ranges are not desirable.

The next step is to select and set up the optimization algorithm. In the following sections we describe both approaches to multi-objective optimization and various optimization algorithms. The simplest multi-objective optimization formulation involves a single mathematical formula, where each objective is weighted and added together to create a cumulative sum. The weighting scheme can have major implications on the designs present in the PF, and adjusting these weighting schemes can reveal different portions of the PF. This process is often repeated multiple times, where the user is essentially trying to optimize the weights of the objectives to best represent their desires. Depending on the computational resources required, this may not be feasible. We also note that it is typically good practice to scale all of the objective functions when they are being examined for some type of summative fitness evaluation. This is especially important when objectives have different orders of magnitude, which would unfairly bias an objective to be more important. While meta-heuristic algorithms to not rely on a strict weighting scheme, they will likely have multiple hyper-parameters to tune the algorithm. It is typically best to look at how other researchers have set up a similar optimization algorithm, and if utilizing a pre-built optimization algorithm, typically the default options are adequate.

The last aspect to consider when constructing an optimization problem is how the objective functions/constraints will be calculated. Depending on the problem space, a high-fidelity physics model may be required to evaluate the objective functions/constraints. For high-fidelity problems, it is important to realize the time requirements for performing an iterative optimization approach, where hundreds or thousands of designs may be analyzed. However, users can generate a surrogate model to reduce the number of function evaluations necessary to identify optimal solutions. To do this, users can evaluate a costly or high-fidelity objective function or function(s) for a reduced set of designs, and extrapolate between these designs to create a multidimensional model that can predict – to a reasonable degree of accuracy – an objective function value based only on the value of the input design variables. While using a surrogate model can reduce the computational time required, it is important to ensure that the model is accurate and represents the underlying physics adequately.

The initial generation of a MOO problem involves selecting the design variables, objective functions, and constraints; selecting the optimization algorithm; and determining how the objective functions/constraints will be evaluated. The goal of this paper is to present the various approaches and optimization algorithms that are commonly used in the field of multi-objective optimization. Through this, the reader will gain an understanding of how to set up and deploy optimization algorithms that meet their specific needs.

2. Approaches to multi-objective optimization

In additional to the optimization algorithm, the designer also has purview over the approach used to apply the optimization algorithm. There are four major approaches that a designer can utilize when applying optimization algorithms; each of which provides unique aspects which may be beneficial for a particular optimization process. The four approaches are *A Priori*, Progressive, *A Posteriori*, and No Preference. Following this, a we provide a brief description on the use of hybrid approaches. The various approaches determine when the problem will be solved, and are used in conjunction with a particular optimization algorithm. Each of these approaches will be given a brief discussion below to facilitate their integration into a MOO algorithm.

2.1. A Priori

A Priori methods typically involve an iterative process of assigning weights to each objective function in accordance with its importance or preference to the designer. An objective function that is deemed important would be given a higher weight than a function which contributes little to the design. The weighted objective functions are then combined in some manner and the resulting function is minimized. This requires the designer to have reasonable insight into the problem before they begin the optimization process, as the resulting Pareto front is sensitive to the weighting scheme. If the weights do not accurately represent the designer's preferences, new weights must be selected. Furthermore, if the designer cannot obtain an adequate solution, they may be required to update the objectives or the design space. The general process for *A Priori* optimization is shown in Fig. 3.

Not every *A Priori* method fits within the structure shown in Fig. 3. However, this illustration provides insight into the flow for typical *A Priori* optimization techniques. *A Priori* methods vary in both scope and complexity. Several methods are presented in Sections 3–4. The interested reader can examine other review papers for more in depth discussions and examples (Chiandussi et al., 2012; Cui et al., 2017).

2.2. Progressive preference

Progressive methods (or interactive methods) rely on continual input from the designer to influence the types of solutions found in the objective space (Andersson, 2000). The designer provides updated preferences for the objectives to continually refine the Pareto front. After each iteration, the designer adds information, typically in the form of new or updated objective functions/constraints. This process is continued until there is an adequate number of acceptable designs. Progressive methods are used less frequently to solve MOO problems due to a reliance on the designer to articulate their needs in a succinct and successive fashion, which can be difficult for large problems. This can be accentuated by a lack of knowledge in the objective space, where updating an objective function/constraint incorrectly can result in the failure to produce a solution, or produce an unintended solution. Each update to the objective functions/constraints is unique, and will affect the resulting solutions. If the resulting objective space does not meet the designer's needs, they are forced to start anew from the beginning of the optimization process. Finally, the constant inputs provided by the designer can significantly slow down the search, as they must be physically present to proceed (Andersson, 2000). Due to these disadvantages, specific methods will not be described; however, additional resources for these methods can be found in Steuer and Choo (1983) and Benayoun et al. (1971).

While we do not explore the use of progressive preference in this paper, it is prudent to mention recent research into this approach. Since user input is required to further the MOO problem, many researchers are looking at the applications of machine learning in conjunction with the progressive preference approach. Current literature has examined using machine learning for objective reduction to reduce the objective space required (Yuan et al., 2018; Saxena et al., 2013). Machine learning has also been employed to learn how to minimize the number of conflicting objectives or reduce the number of objectives to a minimal set (Duro et al., 2014). These methods have begun to explore the uses of the progressive preference and their integration with machine learning, however, they are not prevalent enough in literature to provide relevant examples in the fields of nuclear science and engineering.

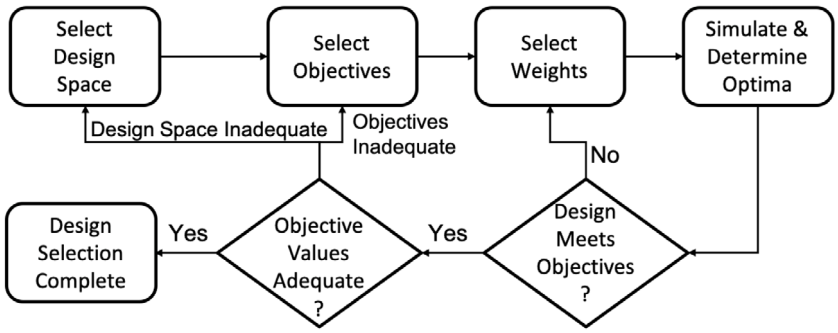

Fig. 3. *A Priori* optimization design flowchart.

2.3. A Posteriori

A Posteriori methods generate sets of solutions first, and analyzes them for optimality later. The design space is adequately sampled before any optimization process is performed, and these results can be used to explore for optimal solutions. Once the model results are obtained, the user can define a list of weights to explore the Pareto front. Once the Pareto front has been discovered, the designer can examine these solutions and determine if they are optimal. If they are not, they can update the list of weights to explore other areas on the Pareto front, or they can update the design space again and run additional models. This idea is presented in Fig. 4, where generating and evaluating models is completed before any weights or optimal configurations are determined.

This approach is often used when the designer has little or no knowledge of the objective space. *A Posteriori* methods give an estimate of the Pareto front, rather than explicit points (Marler and Arora, 2004). Depending on the problem and number of constraints, the number of solutions obtained can be on the order of tens to thousands. If applied to problem formulations within a large design space, *A Posteriori* approaches can be computationally intensive. This is due to the fact that neglecting to prioritize any particular constraints means that a lot of simulations need to be run (Marler and Arora, 2004). Conversely, *A Posteriori* methods can be used with smaller sets of data if some type of reduced-order (surrogate, meta) model is used to determine the objective functions.

2.4. No preference

No preference optimization is used by designers who cannot decisively define their preferences. This is often the case if the designer has limited knowledge of the objective space and wants to perform an initial analysis. This approach instead attempts to find solutions without a user defined weighting scheme; many of the members of this family of algorithms are generalizations of *A Priori*/*A Posteriori* algorithms. Many MOO problems have no global optima, and as such, the no preference method tends to find local optima. While these algorithms may not find fully optimal solutions, they can provide insight into the objective space.

2.5. Hybrid approaches

In addition to the four major preferences provided in this section, there are also hybrid methods which utilize more than one preference. A common approach uses a no-preference or *a posteriori* approach to gain insight into the relationship between the design variables and the objective functions. Similar to this, a sensitivity analysis can be used to determining relationships between the design and objective spaces. Both methods can lead to a reduction of design variables, objective functions, or constraints if they are not valuable. Other options can include providing reference designs, such that the algorithm attempts to optimize the solution in this vicinity (Deb and Sundar, 2006; Vargas et al., 2021). This allows the designer to focus the optimization algorithm on specific areas of the objective space.

Other hybrid approaches utilize multiple meta-heuristic algorithms or meta-heuristic algorithms in conjunction with classical methods. Meta-heuristic methods are typically used an initial examination of the PF. After this, a classical method or more localized meta-heuristic method can be used to narrow down the resulting PF. Additionally, with the use of agent-based software, multiple optimization algorithms can be used simultaneously with data sharing to help accelerate the discovery of the Pareto front (Gebreslassie and Diwekar, 2018). Finally, tiered optimization problems can also be explored to help break up an optimization problem. A tiered problem can use multiple types of optimization algorithms in multiple tiers where each tier either refines the optimization problem or updates the objective functions/constraints. This process can either solve each tier independently or in-tandem, where solutions from one tier are pushed to another tier and vice-versa (Buckhorst and Schmitt, 2020). Given the diversity of optimization algorithms and approaches, there will likely be a continued hybridization of these approaches which will help solve a variety of science and engineering problems.

3. Classical multi-objective optimization techniques

Classical optimization techniques are used to find optimal solutions by searching the design space via stochastic sampling or sweeping methods. Classical methods can have long execution times and potentially fail to converge on an optimal solution depending on the objective space. Despite these features, many of these techniques can be incorporated into meta-heuristic algorithms as fitness functions to help guide the search. A selection of these techniques are presented below, where variations on these algorithms can typically be applied to each of the approaches presented in Section 2.

3.1. Weighted sum

The weighted sum method (Andersson, 2000) for *A priori* optimization utilizes a linear combination of weights and optimization functions to find optimal solutions, as seen in Eq. (3),

$$\mathbf{F}(\mathbf{x}) = min \sum_{i=1}^{m} w_i f_i(\mathbf{x}), \tag{3}$$

where w_i is the weight for ith objective function, $f_i(\mathbf{x})$, and $w_i > 0$ for all i. The larger the w_i, the more important f_i is to the problem; this means f_i will have a larger effect on the set of solutions that are obtained. The fact that $w_i > 0$ substantiates the fact that the objective function f_i must play some role in determining an optimal solution.

This method removes much of the complexity in MOO and is fairly easy to implement for most problems. Consequently, the burden of assigning the weights falls on the designer. The weighting scheme is extremely important in determining the sections of the Pareto front that

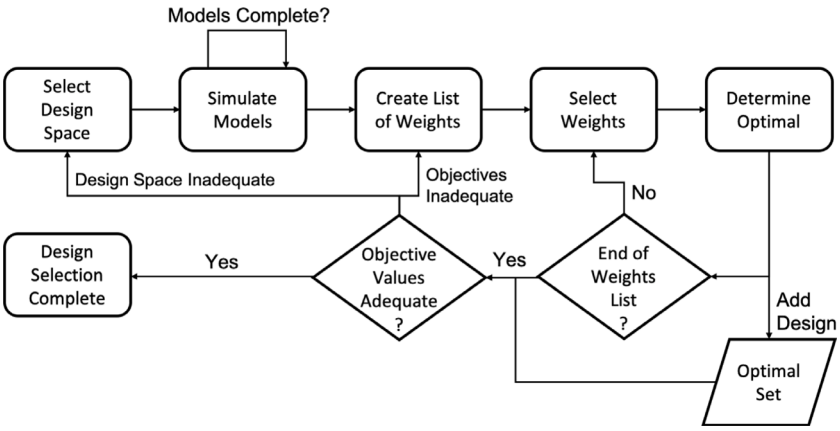

Fig. 4. A Posteriori optimization design flowchart.

are explored. Assigning inappropriate weights can lead to a Pareto front that fails to meet the designer's goals, and may require the designer to iterate on the weights.

The weighted sum approach for *A Posteriori* optimization utilizes Eq. (3), but rather than adjusting the weights on the fly, a list of weighting vectors are created. Each set of weighting vectors ($\mathbf{W}_j = [[w_{1,0}, w_{2,0}, \dots, w_{i,0}], \dots [w_{1,j}, w_{2,j}, \dots, w_{i,j}]]$) are the employed to determine the shape and position of the Pareto front. From a given set of weighting vectors, a set of solutions is obtained. If these do not fit the user's preferences, new weight sets can be generated.

The weighted sum approach can have complications stemming from scaling the objective functions and the complexity of the Pareto front (Das and Dennis, 1997). These issues may either cause sections of the Pareto front to be missed entirely or yield a poorly distributed representation.

Due to the simplistic nature of the approach, the weighted sum method is computationally cheap and easy to apply for most problems. However, it is often difficult to determine an appropriate weighting scheme to accurately capture the designer's goals, which can diminish the apparent benefits. Despite this, the weighted sum is a common methodology for constituting fitness functions for meta-heuristic algorithms, as described in Section 4.

3.2. Lexicographic

The lexicographic method requires the user to arrange the objective functions in order of importance. Once the order is set, each optimization function is solved such that each subsequent problem does not degrade any of the previous optimization function's solutions (Andersson, 2000). This process is described in Eq. (4):

$$\begin{aligned} &min\ \mathbf{F}_i(\mathbf{x}) \\ &\text{given that } \mathbf{F}_j(\mathbf{x}) \le \mathbf{F}_j(\mathbf{x}_j^*),\ j = 1, 2, \dots, i-1;\ i > 1 \\ &\qquad i = 1, 2, \dots, m\ , \end{aligned} \tag{4}$$

where i represents the order of the optimization functions, and $\mathbf{F}_j(\mathbf{x}^*)$ is the optimal solution of the jth objective function from the jth iteration. Each subsequent optimization problem is allowed to find new solutions, however, we add the constraint that any new solutions cannot degrade previous objective functions. The lexicographic method is relatively easy to set up, and requires less initial guess work than the weighted sum method. The solutions obtained from the lexicographic method are highly sensitive to the order of the optimization sequence, and adjusting the sequence will explore different sections of the Pareto front. Due to the strict inequality in Eq. (4), there may be sections of the Pareto front which are unobtainable.

The lexicographic method does not allow for the degradation of any previous solved objective functions. This has lead to a variation known as the hierarchical method (Marler and Arora, 2004). The hierarchical method relaxes some of the constraints by introducing a *relaxation constraint* (δ_i) as seen in Eq. (5):

$$\begin{aligned} &min\ \mathbf{F}_i(\mathbf{x}) \\ &\text{given that } \mathbf{F}_j(\mathbf{x}) \le \left(1 + \frac{\delta_i}{100}\right) \mathbf{F}_j(\mathbf{x}_j^*), \\ &\qquad j = 1, 2, \dots, i-1;\ i > 1 \\ &\qquad i = 1, 2, \dots, m. \end{aligned} \tag{5}$$

The relaxation constraint δ_i allows the jth objective to increase the right hand side of Eq. (5) by some percentage when attempting to find an optimal solution for $\mathbf{F}_j(\mathbf{x})$. Introducing this term adds complexity to the problem, as the designer is then required to determine an acceptable δ_i for each objective function. This allows additional sections of the Pareto front to be explored that may have been unobtainable using the stricter lexicographic method.

3.3. ϵ-Constraint

The ϵ-Constraint method optimizes the problem in accordance with the single most important objective function. All remaining objective functions are collapsed into a set of constraints. This takes the form of a single optimization function f_i, where the remaining f_j objective functions become ϵ_j constraints, as shown in Eq. (6) (Chiandussi et al., 2012):

$$\begin{aligned} &min\ f_i(\mathbf{x}) \\ &\text{given that } f_j(\mathbf{x}) \le \epsilon_j,\ j = 1, 2, \dots, k;\ j \ne i. \end{aligned} \tag{6}$$

There are two important aspects of utilizing the ϵ-Constraint method; which f_i is the most important optimization function, and how to build the constraints out of the objective functions. It is possible to imagine a situation where each constraint is extremely restrictive, and no feasible solutions are found. Equally likely is the scenario in which the defined constraints are too broad, such that non-optimal solutions are retained and presented as optimal. Due to this, knowledge of the objective space is essential to ensure optimal solutions are presented. Similar to the Lexicographic method, the selection of f_i strongly impacts which sections of the Pareto front are explored.

3.4. Physical programming

Physical programming is the most complex of the classical methods described in this section. It utilizes a system of physical goals placed on each objective function to determine acceptable solutions (Messac, 1996). These goals are formulated based on the designer's intimate knowledge of the problem. Each optimization function is given an

Table 1
Objectives and preferences for physical programming.

Common objectives	Preferences
Greater than or equal to	Desirable
Less than or equal to	Tolerable
Equal	Undesirable
In the range of	Unacceptable

objective to achieve, and preference ranges for what is deemed acceptable. Table 1 contains an example of common objectives and preferences, where the objectives represent the type of solutions being sought, and preferences illustrate the designer's understanding of the range of solutions.

Physical programming does not require weights for each optimization function, which can help limit the number of iterations to find the Pareto front. In contrast to Fig. 3, Fig. 5 shows the removal of an iteration step for setting weights and determining if the design captures the users preferences. This is due to the fact that physical programming inherently describes a designer's preference without the need of a weighting scheme. Fig. 5 corresponds to an *A Priori* approach, however an *A Posteriori* formulation is also attainable for physical programming. For *A Posteriori* physical programming, the constraint values used to describe the preference ranges are systematically modified in the context of a search algorithm (Messac et al., 2001). As these constraints are varied, different sections of the Pareto front can be discovered and analyzed. While the updating of preferences is very similar to updating a weighting scheme, it is paramount to recall that in updating the preferences, we are actually updating the range of desired solutions for each objective function.

Physical programming requires the designer to have intimate knowledge of the problem to determine acceptable objectives and preferences for each objective function. Depending on the type of problem and number of objective functions, this type of optimization can be overwhelming. Despite this, physical programming is extremely versatile due to its ability to incorporate problem constraints (requirements) and objectives (preferences). Physical programming can also be used to create a better fitness function for meta-heuristic optimization algorithms (Reynoso-Meza et al., 2014; Stewart and Palmer, 2021).

3.5. Global criterion

The global criterion method employs the ideas presented in Section 3.1 (Weighted Sum). Each objective function is weighted to generate a single function. This method utilizes a weighted exponential sum, which is given in Eq. (7), in conjunction with a variety of schemes to obtain the Pareto front.

$$U = \left[\sum_{i=1}^{m} w_i (f_i(x) - f_i^*)^p\right]^{\frac{1}{p}}, \; F_i(\mathbf{x}) > 0 \tag{7}$$

Two methods which utilize a variation of Eq. (7) are the *Objective Sum Method* and *Min–Max Method*. The Objective Sum Method sets all values in the weighting vector to 1, and p is set to 1. This reduces the equation to be similar to Eq. (3), where each $w_i = 1$. This method provides a singular Pareto optimal solution, which can be useful if little is known about the problem set initially. Once an initial solution is found, additional optimization can be performed to gain a deeper understanding of the problem.

Another method for utilizing global criterion is the Min–Max Method which seeks to minimize the distance to the Utopian solution $\mathbf{F}^*$. The Min–Max Method neglects the weighting functions and utilizes p as a measure for the distance to the Utopian solution. The Min–Max Method uses Eq. (8) to find an optimal solution. For $p = 1$, the Equation reverts to a varied form of Eq. (3). At $p = \infty$, all types of distances are weighted equally, and thus a single solution will result (Marler and Arora, 2004).

$$min \left[\sum_{i=1}^{k} \left(\frac{f_i(x) - f^*}{f^*}\right)^p\right]^{\frac{1}{p}} \tag{8}$$

4. Meta-heuristic (modern) multi-objective optimization techniques

Meta-heuristic optimization techniques attempt to leverage different algorithms rather than explicit mathematical formulations. The hope is that regions of interest can be determined more effectively, and less time will be wasted in calculating results which will not yield optimal solutions. Within meta-heuristic optimization, there are algorithms that are borrowed from multiple fields of study, including biology, physics, and geography. Fig. 6 shows a high-level categorization of these MOO algorithms associated with each field of study. Each algorithm presented in Fig. 6 will be briefly described to provide a general understanding of the methodology. Each algorithm description also includes the types of problems that are typically solved and some of the inherent consequences in implementing each algorithm. This is not an exhaustive list of MOO algorithms; however, they are common in many engineering applications and are frequently available in software packages. There are often variations on each type of algorithm that are used to solve specific problems or amalgamations of these algorithms to perform multi-level optimization.

For meta-heuristic algorithms, there typically needs to exist a method to assess the objective vector. The objective functions are typically combined to create a quantitative understanding of the solution's applicability to the MOO problem. This concept is implemented through a *fitness function*, and is determined by the user to assess the viability of an objective vector by giving it a score based on the objective functions that the user has identified as important. The fitness function should quantitatively rank the individuals to ensure that optimal solutions are found. Often, the fitness function will utilize some of the methods from classical optimization algorithms, such as the weighted sum method, to determine the importance of the objectives.

4.1. Biology inspired

4.1.1. Genetic algorithms

Genetic algorithms (GAs) hinge on Darwin's theory that physically superior organisms tend to pass on their genetic traits. In MOO, the genetic traits are solutions that are passed to a future generation and tend to be non-dominated. The process of finding objective functions is stochastic in nature which leads to a large section of the Pareto front being discovered. GAs have a long history in optimization work, and have branched out to be versatile in both the algorithm features and the types of problems they can solve.

For GAs, the first step is encoding the physical data into a usable form. To perform the encoding process, some nomenclature is necessary. A *chromosome*, also called an *individual*, is a set of independent variables in the design space that correspond to a solution in objective space. Within a chromosome are *genes*, which represent a physical independent variable. Each gene has multiple *alleles* which are a particular value of the dependent variable. A set of individuals is called a *population*, the population is examined to determine which individuals should survive and pass on their genes. To encode an independent variable, it is typically mapped to a binary representation (Brownlee, 2011). For example, imagine a two dimensional optimization problem which is looking at varying the height and uranium enrichment of a fuel rod. In this example, suppose the fuel height can take on four values, and the fuel enrichment can take on six values. For the fuel height, a bit string of two would be required to represent all the possible values; a bit string of three would be required for the enrichment. Table 2 shows how each height and weight percent is encoded into a binary string.

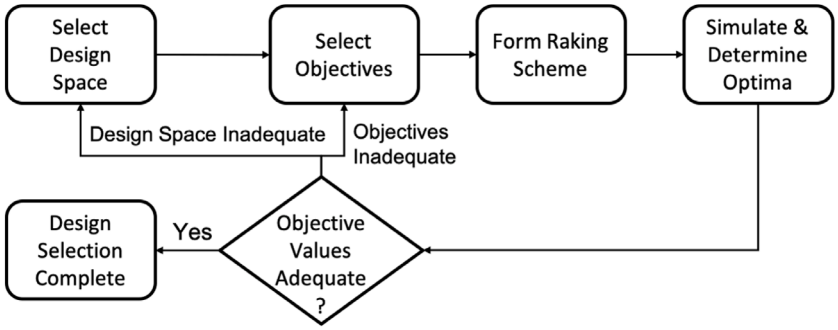

Fig. 5. Physical programming optimization design flowchart (Messac, 1996).

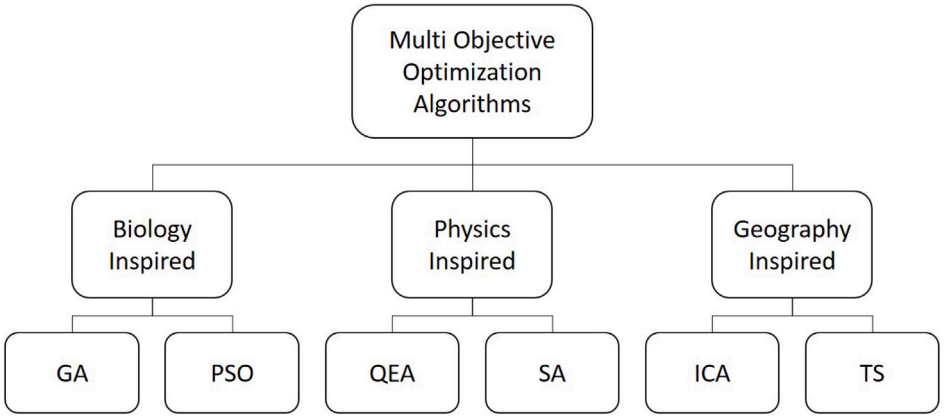

Fig. 6. MOO algorithm breakdown by field of study.

Table 2
Binary encoding for example fuel pin design.

Fuel height (cm)	50	60	70	80		
Encoded value	00	01	10	11		
Fuel enrichment (wt%)	1.0	1.5	2.0	2.5	3.0	3.5
Encoded value	000	001	010	011	100	101

Fig. 7 shows how these two independent variables can be combined to create an individual, with a set of individuals creating the population. The example on the left hand side of Fig. 7 is the individual **10 011** which would correspond to a fuel height of 70 cm and an enrichment of 2.5 wt%. If additional variables were added, the chromosome would increase in length to incorporate the additional features.

Once the encoding process has taken place, each individual must be assessed based on a fitness function. Using the example from above, if the user's objective is a fuel pin which maximizes fuel mass, they might use the fitness function seen in Eq. (9),

$$\mathbf{F}(x) = f(x_1, x_2) = h * wt\%, \tag{9}$$

where $f(x_1, x_2)$ is the objective vector for the fuel mass. In this example, the fuel diameter and density are assumed constant, which creates a simple fitness function. The optimal solution would be the tallest fuel pin with the highest enrichment. The fitness function is often not as simple as the example above, and may require a combination of normalized and/or weighted parameters. Normalizing and weighting these attributes provides a fitness function which can express the user's goals while maintaining the flexibility to span various units and orders of magnitudes.

Once a population of individuals has been created and their fitness functions have been evaluated, a specified number of individuals will mate (typically individuals with a high fitness function) to create offspring. The offspring will gain traits from both parents, in hopes that this will create a more fit individual. This process involves selecting a crossover point and then generating one or two offspring to represent the combination of these genes. Fig. 8 shows the methodology for creating offspring based on the parent chromosomes.

Generating offspring is one way to introduce variety into a population, and encourage the production of diverse solutions. Another method is to introduce mutations at a low rate. This involves taking a gene, or set of genes and flipping the bits to create a new offspring. Fig. 9 shows this process for two genes being flipped.

The general GA process encodes the dependent variables into acceptable individuals, and generates an initial random population. Each individual's fitness is then determined and ranked. The fittest individuals will tend to mate and produce offspring, some of which will mutate. Some of the individuals with low fitness will be removed from the population to create room for the offspring. This process is one generation. As the generations continue, individuals tend towards the Pareto front.

A subset of GA's were created to solve MOO problems, denoted as Multi Objective Genetic Algorithms (MOGAs). There are two specific MOGAs that are used widely in research: Non-dominated Sorting Genetic Algorithm (NSGA/NSGA-II) and the Strengthened Pareto Evolutionary Algorithm (SPEA) (Deb et al., 2002; Zitzler et al., 2001). NSGA groups populations into an ordering system of Pareto dominance, and these groups are examined to ensure that non-similar (neighbor) members survive. This means that the Pareto front found is both well surveyed and diverse in nature (Brownlee, 2011). SPEA selects solutions which are determined to be non-dominated and are in a relatively low density of non-dominated solutions. Non-dominated solutions ensure that the solution is either on the Pareto front or nearby, and does not have a multitude of neighbors nearby. If the solution is not on the Pareto front, or has a lot of neighbors nearby, it has a lower probability of being passed to the next generation. These solutions are then archived to create a separate population, which

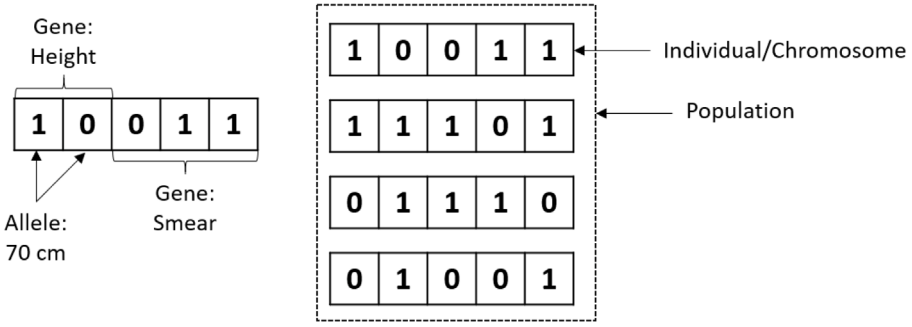

Fig. 7. Terminology for gene, allele, chromosome, and population for genetic algorithms.

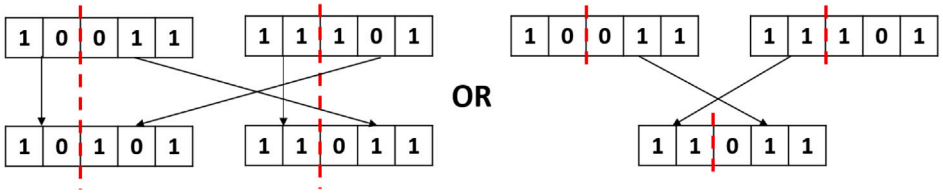

Fig. 8. Two forms of genetic cross-over of chromosomes.

Fig. 9. Mutation of two genes in a chromosome.

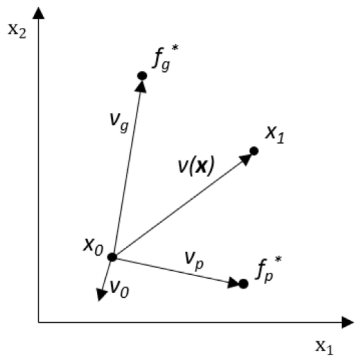

Fig. 10. Movement of a particle in a PSO algorithm.

tends to be on the Pareto front (Brownlee, 2011). This solution set is continually updated throughout the generations, and if a solution in the set becomes dominated, it is replaced.

Genetic algorithms stochastically search the design space, which can lead to two major consequences. The convergence rate can be slow depending on the number of objectives and the size of the objective space, which can in turn lead to an increase in computational time. Stochastic searching also implies that the global optima, if it exists, may be lost. Despite these disadvantages, genetic algorithms do provide a well-defined and diverse Pareto front because they use neighbor or archival methods. Because most genetic algorithms maintain a subset of solutions from generation to generation, there is inherent parallelism that can be exploited (Fonseca and Fleming, 1993).

4.1.2. Particle swarm optimization

Particle swarm optimization (PSO) algorithms attempt to optimize a design space by describing how flocks or swarms of organisms (such as birds, bees, or ants) tend to interact with each other to find areas of interest (Kennedy and Eberhart, 1995). A key feature of swarm optimization algorithms is the concept of collective intelligence, where each swarm member knows the optimal solution of the swarm in addition to their individual optimal position. A PSO algorithm starts by defining a population of particles, each representing a set of variables in the design space, which corresponds to a position $\mathbf{x}$. Similar to GAs, each particle has a fitness function which defines how optimal the solution is. Its position in the design space is then updated by introducing a velocity term. The velocity term, seen in Eq. (10), takes into consideration two aspects: the best solution obtained by any particle (f_g^*), and the best solution of the particle in question (f_p^*):

$$v(\mathbf{x}) = v_0 + c_1 * \texttt{rand}() * (f_p^* - f(\mathbf{x})) + c_2 * \texttt{rand}() * (f_g^* - f(\mathbf{x})), \qquad (10)$$

where, c_1 and c_2 are both learning factors, assigned at the beginning of the problem and used to weight the importance of the global and particular solution. Likewise, v_0 is the previous velocity and `rand()` is a random number between 0 and 1 used to encourage the particle to explore areas towards the optimal solution, but not the optimal solution itself. Fig. 10 shows an example of this for a two dimensional problem, where the global and particular terms from Eq. (10) are denoted as v_g and v_p.

A common variation of the PSO algorithm used in optimization is the Bees Algorithm (BA). The BA focuses on exploring optimal sites thoroughly and randomly sampling (scouting) other portions of the objective space to determine if additional optimal sites are present (Brownlee, 2011). When an optimal site is found, the next generation of particles explores the area around this point in an attempt to find the optimal solution. To prevent premature convergence, additional particles are sent out to randomly sample other areas of the solution space. If one of these particles finds a more optimal solution, additional particles will be sent to that area. This type of behavior can be seen in Fig. 11, where scouting particles find a new minima in the middle figure, and the remaining bees start to move towards this area.

PSO and BA have both been outfitted to solve MOO problems for both continuous and discrete optimization problems. However, most PSO/BA algorithms tend to focus on solving continuous problems due to the randomness in the velocity term. The velocity term in PSO algorithms enables a swarm member to over shoot a known 'best' solution. This has the advantage of an individual member being able

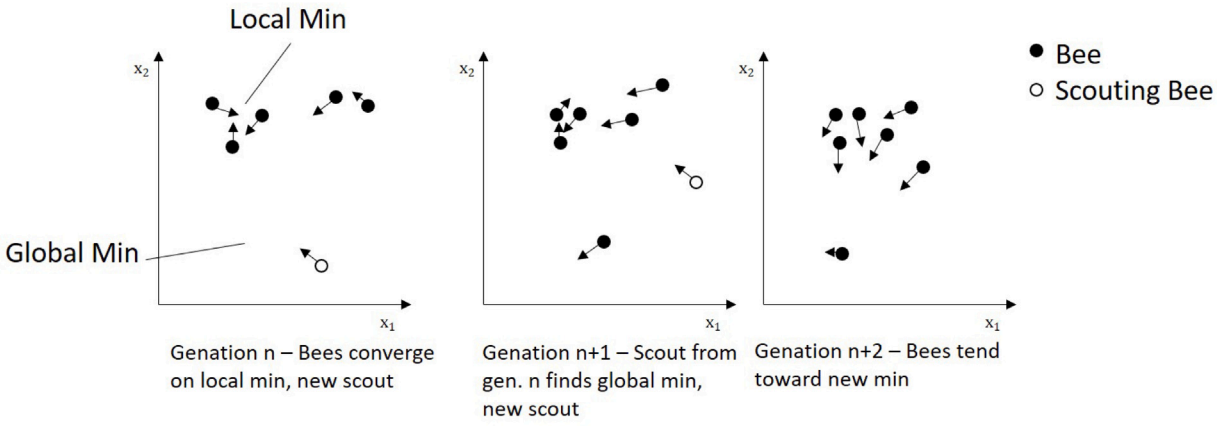

Fig. 11. Convergence of BA on a globally optimal solution.

to explore additional areas in the objective space without becoming trapped in a local minima/maxima. Despite this, in exceptionally large MOO problems, if too few swarm members are selected, they tend to converge on local optima without finding an optimal solution. Sending out additional scouts each iteration plays a parallel role to the velocity term in PSO algorithms in preventing the algorithm from only converging on a limited number of solutions. Swarm optimization tends to be relatively fast and easy to implement for most problems.

4.2. Physics inspired

4.2.1. Quantum Evolutionary Algorithm

Quantum Evolutionary Algorithms (QEA) are a blend of quantum computing and evolutionary (genetic) algorithms, which rely on quantum bits (Q-bits) of information and quantum gates (Q-gates) to perform optimization analysis (Han and Kim, 2002). Q-bits replace the binary bit system used in GAs, as seen in Section 4.1.1, where Q-gates take the place of evolutionary operators such as crossover and mutation.

To understand the benefits of utilizing a QEA, some quantum computing nomenclature is required. A Q-bit is similar to a bit in traditional computing, but can take on additional information rather than 0 or 1 explicitly. This is similar to a gene in a GA, and contains information on a given independent variable. The most common expression for a Q-bit can be seen in Eq. (11).

$$q = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \quad |\alpha|^2 + |\beta|^2 = 1, \tag{11}$$

For a Q-bit, α and β are the probability amplitudes of observing the Q-bit in either a 0 or 1 state, respectively. The probability amplitudes are typically complex numbers and the Q-gates applied are in the complex plane. This property allows a Q-bit to be in the 1 state, 0 state, or any linear combination of the two states. Q-bits are combined to make Q-individuals (similar to individuals in GAs), which are a string of n Q-bits, as seen in Eq. (12). Once the Q-bit individual has been created, it goes through a Q-gate, which will adjust α and β in some way to yield a different state. One common example of a Q-gate is the rotation gate (U), which is defined in Eq. (13):

$$Q = \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \beta_1 & \beta_2 & \dots & \beta_n \end{bmatrix}, \quad |\alpha_i|^2 + |\beta_i|^2 = 1 \;\; i = 1, 2, \dots, n. \tag{12}$$

$$U(\Delta\theta_i) = \begin{bmatrix} cos(\Delta\theta_i) & -sin(\Delta\theta_i) \\ sin(\Delta\theta_i) & cos(\Delta\theta_i) \end{bmatrix}. \quad i = i, 2, \dots, m. \tag{13}$$

In Eq. (13), $\Delta\theta_i$ is a rotation angle for the mth Q-bit, which increases the probability of either a 1 or 0 state, depending on the angle. One of the more difficult tasks in QEA is selecting $\Delta\theta_i$ for the problem, however a value between -0.05π and 0.05π is typically deemed acceptable (Wei and Fujimura, 2010). Other types of Q-gates are more straightforward, such as the NOT gate. The NOT gate is similar to a mutation in GAs, where it will flip the state from a 0 to a 1, or vice versa (Han and Kim, 2002).

After the Q-individual undergoes a Q-gate, the probabilities for α_i and β_i are manipulated, and the Q-individuals are observed. Each Q-bit in the Q-individual is observed based on the probabilities of α and β, which yields either a 0 or 1 for that Q-bit. The Q-individuals in the population are then observed and assessed based on their fitness function, and some subset of non-dominated solutions are selected and stored. After each Q-gate, the diversity of the solutions will decrease as each Q-bit will tend towards either 0 or 1 (Wu et al., 2016). After the completion of the QEA, the stored set will compose the Pareto front.

The Multi-Objective Quantum Evolutionary Algorithm (MOQEA) was developed to apply concepts of QEA to multi-objective problems by decomposing them into multiple single objective sub-problems (Wei and Fujimura, 2010). For MOQEA, the decomposition of the MOO problem allows for each sub-problem to be computed in parallel. After the sub-problem undergoes a generation, the most optimal solution is sent to an External Population (EP) and checked to determine if it is the most optimal solution found for the sub-problem. The EP and most optimal solution for each sub-problem are then used to inform the selection of Q-bit individuals for the sub-problem's next generation. Multi-Chain Quantum Inspired Evolutionary Algorithms (MCQEA) and Bloch Quantum-Inspired Evolutionary Algorithm (BQEA) were developed to search the solution space of a problem without decomposition (Zhang et al., 2014; Li and Li, 2007). MCQEA/BQEA utilize a higher dimensional Q-bit to encode additional information and use common EA operations, such as mutation, to expedite convergence (Li and Li, 2007).

QEAs have successfully been applied to MOO problems via MOQEA/MCQEA/BQEA to leverage the advantages of quantum computing. MOQEA provides an algorithm which can rapidly converge on a set of solutions because it is highly parallelizable and information can be easily shared between generations. However, MOQEA assumes the problem is able to be decomposed which may not be applicable to all MOO problems. MCQEA/BQEA challenge this assumption by utilizing a higher dimensional Q-bit to rapidly explore the solution space for optimal sets.

4.2.2. Simulated Annealing

Simulated Annealing (SA) exploits statistical mechanics to find the lowest energy states for a large number of atoms at low temperatures,

Table 3
Fitness evaluation for fuel mass.

y (height/wt%)	y' (height/wt%)	Fitness (y/y')	ΔE	P ($y' \rightarrow y$)	T
70/3.5	60/3.5	210/175	−35	0.859	230
70/3.5	60/3.5	210/175	−35	0.497	50
60/1.0	70/1.0	60/70	10	1.000	230

where each energy state represents a solution to the problem (Kirkpatrick et al., 1983). The SA process examines energy states over multiple generations to help determine an optimal solution. Initially the acceptance criteria for keeping a state between generations is very loose; this process is similar to raising the temperature (heating up) of a material in metallurgic annealing. The acceptance criteria is constantly refined to focus on optimal solutions; similar to decreasing the temperature (cooling down) in the annealing process. Once the problem has finally "cooled", the result will be an optimal solution.

The process of SA is stochastic in nature, and utilizes a hill-climbing algorithm to find an optimal solution. In design space, a point y is selected and its solution is assessed via its fitness function. The point is then moved in a random direction to a new point (y') and the fitness function is again examined. If the fitness function for y' is better than y, then y' becomes y ($y' \rightarrow y$), and the process will continue. If the fitness function of y' is worse than y, there will be some probability, expressed in Eq. (14), of accepting the new solution:

$$P(y' \rightarrow y) = 1 - \exp\left(\frac{\Delta E}{T_N}\right). \tag{14}$$

The function $\Delta E = F(y') - F(y)$, and T_N is the temperature during the Nth generation, which is used to determine the probability of acceptance. There are multiple approaches to selecting the initial temperature, but an initial estimate typically takes the form $T = \Delta E_{max}$. Higher temperatures allow for a larger selection of results to occur, as there is a higher probability of moving in a non-optimal direction. As the temperature is decreased the probability of accepting non-optimized solutions decreases, where the solution(s) found are highly dependent on the cooling scheme selected. Typically the temperature for each subsequent generation ($N + 1$) is determined by $T_{N+1} = cT_N$, where c is 0.01–0.2. This rate of temperature decrease produces a slow cool-down which ensures that the solution does not converge on local minima.

Utilizing the example portrayed in Section 4.1.1, an initial point (y) is selected at random with a height of 70 cm and an enrichment of 3.5 wt%. The point then moves a random direction and a new point (y') is selected. Table 3 shows the impact of temperature on the probability of selecting y'.

Archived Multiobject Simulated Annealing (AMOSA) utilizes dominance ordering in the application of SA to a multi-objective problem (Bandyopadhyay et al., 2008). This is accomplished by creating an archive of non-dominated solutions that the algorithm has experienced thus far. As the algorithm continues, the archive will continue to grow, which may require constraints to be placed on the archive to maintain usability — removing solutions that have become dominated in the archive, or by removing solutions that are clustered together. These two methods ensure that the archived solutions are both non-dominated solutions and they are representative of the Pareto front. As the temperature is reduced, these archived solutions are examined further to produce an optimal set.

Given a sufficient cooling time, the global optimum can almost always be found, which indicates that SA is extremely valuable when trying to find a single solution. This ability to converge on a global optimum is accompanied by a slow convergence rate, and a more time-consuming algorithm. Similar to GA, SA can be executed in parallel which helps it overcome this setback. SA algorithms tend to be sensitive to initial conditions and the cool down rate placed on the problem, which can make the algorithm challenging for novice users.

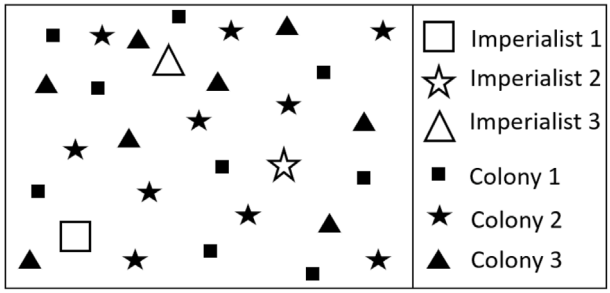

Fig. 12. ICA example for imperialists and colonies.

4.3. Geography inspired

4.3.1. Imperialistic Competition Algorithm

The Imperialistic Competition Algorithm (ICA) is derived from colonialism, where nations compete for control and dominance over resources and other nations. The imperialists in ICA are non-dominated solutions, and the colonies are solutions which are used to explore the design space. Powerful imperialists take possession of weaker colonies and draw the colonies closer to explore the surrounding areas. After multiple generations there is one imperialist nation left, with colonies nearby to determine the most optimal solution (Atashpaz-Gargari and Lucas, 2007).

ICA begins by creating a population of countries which are vectors of length m, where each m_i represents a solution. The cost of each country is then found using a fitness function, where the highest cost countries (N_{imp}) form the imperialists, and the remainder form colonies. The colonies are divided among the imperialists, based on the cost of each empire — the combination of an imperialist and their colonies. Empires with a higher cost will be awarded more colonies. An illustration of the colonies, imperialists, and empires can be seen in Fig. 12. Squares are members of empire 1, stars are members of empire 2, and triangles are members of empire 3. In this example, the star imperialist has the highest cost and is awarded 10 colonies, the square is awarded 9 colonies, and the triangle imperialist is the weakest empire and is awarded 8 colonies.

Each generation, the colonies are moved towards their respective imperialist, where colonies move to a new position c' according to Eq. (15). Here `rand()` is a random number between 0 and 1, d is the distance between the colony and the imperialist, θ is an angle in radians between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$, and β is a user defined variable (typically set to 2) to ensure that the colony moves towards the imperialist. In ICA, colonies do not move in a straight line towards the imperialist, but are shifted by θ. This allows colonies to explore areas around the imperialist.

$$c' = c + \texttt{rand}() * d * \theta * \beta \tag{15}$$

Once the colonies have all been moved, the colony's cost is computed. If a colony in an empire has a cost that is greater than the imperialist, their roles are switched. The colony becomes the imperialist and the imperialist is now a colony. After each generation, the empire's cost is determined, and the colonies of the weakest empires have a probability of being given to an empire with a higher cost. If there are empires with no colonies, they are removed from the simulation. This process will continue until a single empire remains, where the imperialist in this empire would indicate an optimal solution.

The Multi-Objective Imperialistic Competitive Algorithm (MOICA) was created to solve global MOO problems (Sherinov and Unveren, 2017). This algorithm retains the same premise as ICA, but adds a few important concepts to improve its ability to optimize based on multiple objectives. The most important aspects of MOICA are two sets of non-dominated solutions; local non-dominated solution (LNDS) sets

and global non-dominated solution (GNDS) sets. The LNDS set represents the most optimal solutions for each imperialist nation, and the GNDS contains the best imperialists among all of the empires present. This allows for several non-dominated solution sets to be present and explored at a single time, which increases the probability of finding a global dominant solution. MOICA also introduces additional algorithms to improve the local search capacity ("economic changes") and ensures a diverse solution set by crossing the solution sets of two imperialist nations ("revolution").

ICAs tend to survey optima thoroughly by pulling colonies towards imperialists, which allows for a greater probability of a global optima to be found. Setting up an ICA optimization problem has limited parameters to adjust making it a relatively simple technique.

4.3.2. Tabu Search

Tabu Search (TS) algorithms find optimal solutions by constraining future solution selection, based on previous solutions. This requires retaining memory (a tabu list) about where the algorithm has explored previously to ensure that the problem does not find the same solution multiple times. Intermediate and long term memory can be used to introduces biases towards areas of promising solutions (intensification) or towards unexplored areas in the solution space (diversification).

The TS algorithm starts by selecting a point in design space and evaluating its fitness function. It then generates a list of possible movements and evaluates the fitness function at each of these sites in order. Each site will undergo a series of tests to ensure the move is acceptable. The first test determines whether the move increases the fitness function more than any other move tested thus far (Glover, 1990). If the fitness function is better, the algorithm retains this information and examines the next solution in the list. If the fitness function is worse, it must go through an additional series of tests to determine if the information will be retained. The first test for a failed fitness function checks if the move is Tabu (if the move has been previously explored). If the move is Tabu, it must undergo an additional check to determine if it will be retained. The set of tests for a Tabu move is called the aspiration criteria. The aspiration criteria allows the TS to escape local minima, while retaining the ability to make the best move possible (Glover, 1990). If the move is determined to either pass the aspiration criteria or is not Tabu, it is designated as a possible best solution. Each solution in the list of moves is tested, and the best solution is then selected as the move. This process is shown in Fig. 13.

The Multi-Objective Tabu Search (MOTS) method was developed to generalize the TS method to a MOO problem (Hansen, 1997). MOTS utilizes a set of solutions, each with their own Tabu list. Each solution moves towards the Pareto front while simultaneously attempting to move away from neighboring solutions. This produces solutions along the Pareto front which are unique and diverse and employs a weighting vector to take into account where other solutions are in the problem and ensures that the solution is moving towards a more optimal area. MOTS can be sensitive to the user-specified number of solutions which explore the design space. If too few solutions are examined, the Pareto front may not be thoroughly explored.

TS optimization tends to find global optima, and efficiently escape local optima. These methods are well equipped to solve combinatorial and discrete problems. However, TS optimization requires knowledge about the surrounding solutions, which makes it difficult to be executed efficiently on parallel computing architectures. It is also highly dependent on the initiating variables used to set up the problem. As a result, TS algorithms can be difficult for novice users.

5. Optimization applications for nuclear science and engineering

5.1. Fuel reload patterns

Fuel reloading for nuclear power plants consists of removing, shuffling, and adding new fuel assemblies to the reactor core. During this time, typically a quarter to a third of the assemblies are removed and replaced with fresh fuel. The remaining and new assemblies must then be configured in the core to maintain a desired cycle length, while trying to limit poison concentration and power peaking. Fuel loading patterns are notoriously difficult to optimize as the number of fuel loading patterns for n assemblies approaches $n!$ (Akbari et al., 2018).

Akbari et al. (2018) utilize ICA to examine fuel loading patterns for the VVER-1000 Bushehr Nuclear Power Plant (BNPP). The cost function for ICA incorporates k_{eff} and the minimum/maximum power peaking factors, which penalizes high radial peaking and configurations which are not critical. 1/12 and 1/6 symmetry core were examined, where the fuel assembly types at the outermost edge were held constant. The design space consisted of six different assembly types, with variable enrichments and burnable poisons. They were able to reduce the cost function of the current BNPP by 18% for the 1/12 symmetry core and by 53% for the 1/6 symmetry core. This reduction was due to a decrease in the maximum and minimum power peaking factor, an increase in the cycle length, and an increase in total burnup for the two cores. Overall, the ICA method presented two unique solutions to the fuel loading pattern which were deemed to be better than the current design for BNPP.

Wu et al. (2016) also examined pressurized water reactor (PWR) loading pattern designs for the Maanshan Nuclear Power Plant using a hybrid method involving a QEA and TS. QEA was utilized for its ability to explore the design space (i.e. place assemblies into the core) and generate an initial set of solutions by allowing objectives and constraints to be slightly violated. Once this initial set was found, TS swaps individual assemblies to find solutions which fulfill the designers' preferences. The authors focused their fitness function on fuel utilization, the hot channel factor, and the moderator temperature coefficient. Each factor had an associated weight, and were linearly combined. The fuel loading patterns examined both 1/4 and 1/8 core symmetry, where each assembly in the 1/4 core was described by a four Q-bit string, which required a Q-bit individual with of length 100. The authors imposed limits on where each assembly was allowed to be placed to reduce the size of the problem. Once a plausible loading pattern was found using QEA, TS swapped and rotated individual fuel assemblies in an attempt to improve the three objectives. If this did not yield a desired solution, the QEA algorithm continued searching the design space for additional plausible loading patterns. The authors found that combining QEA with TS increased the success rate of finding loading patterns which met their objectives, and reduced the time compared to utilizing QEA alone. Using the hybrid method increased the success rate of finding a viable core from 20% to 70% and reduced the time requirements from 14.42 to 11.10 h.

Park et al. (2014) utilized a multi-objective SA process with an adaptively constrained discontinuous penalty factor to improve the search efficiency. Domingos et al. (2006) compared PSO with other common evolutionary-based methods on simplified core geometries found in literature. They found PSO yields easier modeling and less computational effort to produce the same results with evolutionary-based methods. Yadav and Gupta (2011) utilized PSO to determine optimal PWR core configurations to attain reasonably low peaking factors, while using a traditional neutron transport solver rather than a surrogate model based approach. Safarzadeh et al. (2014) examined a hybrid bee colony algorithm to reduce the time requirements for solving core reloading patterns for a VVER-1000 core in comparison to a typical PSO.

5.2. Core design

Reactor core design involves multiple characteristics for both design variables and objective functions. This is done by exploring variations on core geometry and material composition while maintaining appropriate power levels, cycle lengths, feedback mechanisms, etc. The number of variables to optimize is often daunting and multiple

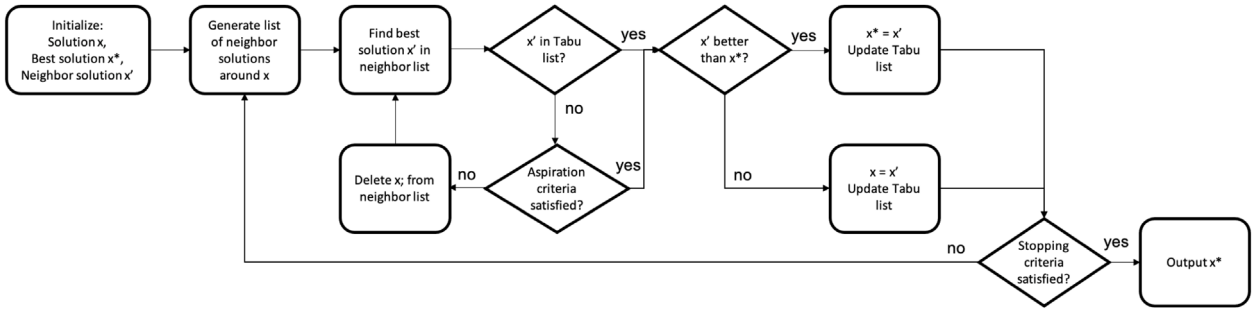

Fig. 13. Tabu Search algorithm.

papers examine only optimizing a few key variables to expedite the optimization process.

Yilmaz et al. (2006) examined optimizing the placement of burnable poisons for the Three Mile Island-1 plant. This search entailed design variables such as the number of poison pins and their concentrations. Optimal solutions reduced the total amount of poison in the core and maintained a strict set of constraints. These constraints were the beginning of cycle (BOC) boron concentration and maximum peak pin power. This optimization process used GA with a fitness function which varied depending on the maximum peak power, poison amount, and BOC boron concentration. For example, if the solution violated the maximum peak pin power and the BOC boron concentration, the fitness function would receive a high negative score as a penalty, which would likely result in the solution being removed from the population. If the solution only violated the pin power, the fitness function was altered to minimize the pin power. Finally, if the pin power objective was satisfied, the revised fitness function became minimizing the poison concentration. This method found a set of solutions which would allow the designer to minimize the burnable poison at the expense of increasing of BOC boron concentration. From there it was determined that a cost analysis could be performed to determine the most appropriate solution.

Significant work has been performed at Texas A&M for reactor design which combines regression analysis and genetic algorithms (Kumar and Tsvetkov, 2015). This work uses multivariate regression to generate surrogate models for predicting various objectives associated with core design. This framework was originally utilized to optimize high temperature gas reactors (HTRGs), where they focused on burnup rate, peaking factors, and a desired k_{eff}. A genetic algorithm was used to optimize a single fuel pin cell, full core neutronics, thermal hydraulics, and the energy conversion for a HTGR. This facilitated the generation of correlations between the various objectives and yielded an optimal solution based on the fuel pin radius, inlet temperature, enrichment, and flow rate. Kajihara and Tsvetkov (2019) expanded on this work by examining high-level transuranic (TRU) waste destruction rates.

Multiple authors have examined optimizing the design space for sodium fast reactors. Qvist and Greenspan (2014) built a code framework (ADOPT) which optimizes the core design based on a set of given objective functions such as coolant velocity, fuel maximum centerline temperature, and cladding peak inner wall temperature. Iterations between multiple physics modules including neutronics, thermal hydraulics, and structural mechanics were performed to provide a more realistic core design. They examined a small modular sodium fast reactor where there objectives were to maintain a 25 year core life, minimize the reactivity swing, and minimize radial peaking. While the problem was inherently multi-objective, they present one solution which addresses all of the constraints and yields a viable core. Zeng et al. (2020) and Adam et al. (2019) have recently developed a framework which couples the Argonne Reactor Computation (ARC) suite with the sensitivity analysis and optimization toolkit DAKOTA within the NEAMS workbench. This work is similar to previously described work and relies on a GA to drive the Pareto front discovery, while surrogate modeling is used to reduce the computation time associated with full core modeling.

Hourcade et al. (2013) created the framework TRIAD for optimizing sodium fast reactors. Ingremeau et al. (2011) utilized the optimization code FARM to develop an optimal core design for high temperature gas reactors. Anderson et al. (2019) examined boiling water reactor fuel bundle optimization using the Multi-Objective Optimization utilizing Genetic algorithms for Lattice Enhancement (MOOGLE) methodology, which reduced the number of pin varieties in a traditionally optimized fuel assembly. Gougar et al. (2010) examined the use of a real-coded genetic algorithm to perform preliminary design analyses of pebble bed reactors.

5.3. Nuclear data

Nuclear data is essential in the design of advanced reactor concepts and criticality experiments, as it is used in high-fidelity simulations to model safety and operational constraints. The accuracy of a given simulation depends on the ability to model the physics correctly; incorrect nuclear data can lead to false or inaccurate conclusions. The process used to generate nuclear cross-sections is very important for reactor physics, detection, and criticality experiments. Often users want to limit the number of cross-section sets and computational time required, while retaining the appropriate physics.

Yi and Sjoden (2013) and Yi et al. (2016) examined approaches for collapsing a fine group library to a coarse group library using a PSO algorithm. Their approach generated a fitness function which minimized the error from the response functions and the relative forward/adjoint neutron flux errors. This allowed them to compare coarse group transport results with the fine group to determine the accuracy of the coarse group's solution. They examined a detector model with a goal of collapsing a fine (30) group library to a coarse (3) group library. Utilizing a PSO algorithm they found that they were able to generate a coarse group library that generated a detector response within 5% of the fine-group library. This method was then extended to the modeling of a fuel pin, where they collapsed a 47 group library to a 5 group library. Their new fitness function examined the pcm (per cent milli) difference in k_{eff} from the reference model to the collapsed model, where they found that the optimized solutions for the 5 group library were within 14 pcm of the fine group results. When the number of groups was increased to 7, the difference was reduced to 2 pcm. This method could be applied to additional problems to generate cross section libraries with a minimal number of energy groups, while retaining the desired essential physics.

To help assess the quality in nuclear data measurements, critical assemblies are often created to quantify small inaccuracies in specific cross-sections. Ney et al. (2018) utilized a GA to optimize the design of a criticality experiment to measure the sensitivity of k_{eff} to molybdenum cross sections in the intermediate energy ranges. In the experiment design, the authors attempted to maximize the sensitivity coefficient associated with an intermediate energy range. The design variables

included the number of unit cells, the moderator plate thickness, and the molybdenum plate thickness. The criticality experiment was performed with four different materials, and for each material they were able to generate a configuration with additional constraints on the system such as the weight, and the criticality condition. They found that Teflon provided the highest sensitivity coefficient for the desired energy ranges. This study found that the time required to develop a critical experiment was drastically reduced by applying a GA rather than attempting to find a configuration of differing materials by hand.

Other aspects of cross-section generation examine adjusting nuclear data evaluation measurements applied to reduce the uncertainty in criticality benchmark experiments (Arthur et al., 2019). Along with this, Stover and Turinsky (2012) examined how to optimize experimental setups to reduce nuclear data uncertainty associated with fast reactors.

5.4. Detectors

The ability to detect a neutron's energy is a notoriously difficult task, and often requires the use of spectrum unfolding algorithms. These algorithms estimate the response of neutrons through various reactions to help reconstruct the neutron spectrum. To create a more effective neutron unfolding algorithm, various forms of prior information are considered to create an appropriate spectrum. Woo et al. (2019) applied a GA which incorporated multiple types of prior information in an attempt to find a neutron spectrum on the Pareto front. The authors utilized two separate fitness functions: one to minimize the error of the spectrum from measured values and the second to maximize the Shannon entropy. The fitness functions represent the prior knowledge of the problem and can be used to estimate the neutron spectrum. The authors compared the solutions from this technique to a known spectrum, where they found that incorporating a genetic algorithm provided a neutron spectrum which was comparable to the known spectrum.

Creating specific neutron spectra plays an important role in detection, nuclear forensics, and medical applications. This can involve placing a series of moderators between a neutron source and the detector to yield a desired spectrum. Bogetic et al. (2018) developed a software package for generating an optimized neutron beam given a predetermined set of constraints. The design constraints which play a role in the neutron energy include the energy tuning assembly's dimensions/materials, number of cells, and the order of the cells. The optimization process was performed on a weighted flux describing the difference between the desired spectrum and the spectrum associated with a specific design iteration, where constraints on both the weight and the efficiency of the design are employed to ensure non-realistic solutions are excluded. This method was compared with known sources for nuclear forensics and boron neutron capture therapy. For nuclear forensics, the goals were to obtain neutron spectra which represent thermonuclear and prompt fission neutrons. Comparing nine different expected isotopes, the energy tuning assembly yielded a spectrum within the uncertainty of the reference spectrum. Similarly, for boron neutron capture therapy the algorithm was able to find an appropriate design which reproduced the reference spectrum while simultaneously reducing the size of the energy tuning assembly. Given the success in these initial examples, further research is ongoing to generalize this system for additional problems.

5.5. Thermal hydraulics

Pressurized water reactors utilize a high pressure system to preserve liquid water in the core during normal operations, where the pressure is maintained by an electrically-heated pressurizer. The design of this system has recently been examined to determine an optimal configuration using a hybrid non-dominated sorting algorithm to minimize the system's size and weight (Wang et al., 2016). The design variables were the pressurizer inner diameter, pressurizer spray coefficient, primary pressure, and outlet temperature, and the objectives were to simultaneously minimize the volume and weight of the system. A set of physical constraints were also placed on the problem to ensure the pressurizer was able to perform its intended functionality. The authors found solutions on the Pareto front which could reduce the volume or weight by 18% and 16% individually, or reduce both simultaneously by more than 10% compared to the current design. Their results concluded that a large height to diameter ratio, spray coefficient, and relatively low reactor coolant temperature can help reduce the weight and volume of the pressurizer in a PWR system.

Similar work has been performed to improve the design process of nuclear power plants. Wilding et al. (2020) utilized the Optimization Preference Tool for the Improvement of Nuclear Systems (OPTIONS) for optimizing two different nuclear system designs: the flash-Rankine power conversion system (PCS) and the passive reaction cooling system PERCS. For the PCS optimization, there were seven optimization parameters for the system which were used to maximize the thermodynamic efficiency and minimize the capital equipment cost of three specific PCS designs. The PCS system was examined in greater detail by optimizing the PCS superstructure utilizing a mixed-integer GA to determine a design based on 22 different combinations of independent variables to examine multiple superstructures, where the objectives (thermodynamics efficiency and capital cost) remained the same. This sought to generate a more defined Pareto front than simply optimizing one or two versions of the PCS system, as specific components are required to build each. The PERCS system contained six optimization parameters to minimize the equipment capital cost, minimize the deviation of the core outlet temperature from steady-steady state, and to maximize the cooling duration of the PERCS system. For the PCS system, utilizing a GA allowed for the optimization of the three specific PCS designs and generated a Pareto front for each design respectively. The Pareto front presented a significant increase in the capital cost when increasing the thermodynamics efficiency. However, when the authors applied the mixed-integer GA, they found a Pareto front which dominated the original three designs, and also found designs with a new highest achievable efficiency of 35.63%. Optimization of the PERCS system found a Pareto front with 50 design options, all of which dominated the base design proposed for the system. This work found that both the PCS and PERCS systems in their current design were not optimal, and found that MOO was able to significantly improve these systems.

Thermal hydraulic optimization has also been applied on a smaller scale to the optimization of wire-wrapped fuel assemblies by Raza and Kim (2007, 2009). They examine combining a GA along with a local search method in conjunction with a surrogate model to reduce computational time. Their work examined enhancing the heat transfer capabilities for a liquid metal fast reactor fuel assembly. The design variables were the ratio of the wire-wrap diameter to fuel rod diameter and wire-wrap pitch to fuel diameter. To determine the acceptability of the assembly, the two objective functions focused on optimizing the Nusselt number (to maximize heat transfer) and friction loss, where these two objectives naturally compete. Initially, they examined multiple surrogate modeling techniques to determine the most appropriate model which retained the accuracy of the high-fidelity model used. Utilizing the hybrid GA/local search, the authors were able to find the Pareto front between the Nusselt number and the friction loss to examine the relationship between the two when designing a fast reactor assembly. An increase in the wire-wrap diameter caused an corresponding increase in the heat transfer and the friction loss in the assembly, where increasing the wire-wrap pitch reduced the friction loss at the cost of reducing the heat transfer. The Pareto front was able to provide context to these changes as it was found that the change in heat transfer rate is much higher than the change in the friction loss. Finally, this study showed that designs which tended to maximize heat transfer showed lower overall temperatures in the assembly and a more even temperature distribution.

6. Discussion and conclusions

Many applications of optimization in real world nuclear science and engineering problems are inherently multi-objective. Often, the functions to optimize are in stark conflict with each other, and optimizing one function will negatively effect the others. Finding solutions to this type of problem often requires balancing a set of objectives and determining what type of solutions are most appropriate. Multi-objective optimization algorithms provide a framework for solving these large difficult problems. Many of the algorithms presented in this paper search the user's design space for a set of optimal solutions, where these solutions are typically non-dominated and evenly distributed on the Pareto front. This gives the designer a set of solutions from which to select, where each solution has no alternative that would simultaneously improve one objective function without degrading another.

We have presented two different categories of methods (classical and meta-heuristic) for multi-objective optimization. We further subdivided these categories into four different approaches (*A Priori*, Progressive, *A Posteriori*, and No Preference) used to determine optimal solutions. The methods presented each have strengths and weaknesses depending on the amount of user input, strength of solution sets, and implementation cost. Current literature tends to favor meta-heuristic techniques to solve MOO problems due to their robust nature and typical ease of implementation (Ney et al., 2018; Gougar et al., 2010; Akbari et al., 2018; Yilmaz et al., 2006). As these methods continue to mature and become more readily available, their use in solving MOO problems in nuclear engineering will continue to grow. This will provide nuclear engineers and scientists with tools to solve large multi-physics problems and find highly desirable solutions to expedite nuclear research.

CRediT authorship contribution statement

Ryan H. Stewart: Conceptualization, Writing - review & editing, Writing - original draft. **Todd S. Palmer:** Writing - review & editing, Writing - original draft. **Bryony DuPont:** Writing - review & editing, Writing - original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Adam, B.M., et al., 2019. Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.10 User's Manual. Technical Report SAND2014-4253, Sandia National Laboratory.

Akbari, R., Abbasi, M., Faghihi, F., Mirvakili, S., Mokhtari, J., 2018. A novel multi-objective optimization method, imperialistic competitive algorithm, for fuel loading pattern of nuclear reactors. Prog. Nucl. Energy 108.

Anderson, B., Kropaczek, D., Hou, J., 2019. BWR fuel bundle optimization based on three-dimensional fuel rods. Trans. Am. Nucl. Soc. 374–377.

Andersson, J., 2000. A Survey of Multiobjective Optimization in Engineering Design. Technical Report LiTH-IKP-R-1097, Department of Mechanical Engineering, Linkoping University.

Arora, J., 2013. Introduction to Optimum Design, third ed. Elsevier.

Arthur, J., Bahran, R., Hutchinson, J., Pozzi, S.A., 2019. Genetic algorithm for nuclear data evaluation applied to subcritical neutron multiplication inference benchmark experiments. Ann. Nucl. Energy 133, 853–862. http://dx.doi.org/10.1016/j.anucene.2019.07.024, URL: http://www.sciencedirect.com/science/article/pii/S0306454919304098.

Atashpaz-Gargari, E., Lucas, C., 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In: 2007 IEEE Congress on Evolutionary Computation. pp. 4661–4667. http://dx.doi.org/10.1109/CEC.2007.4425083.

Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K., 2008. A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Trans. Evol. Comput. 12 (3), 269–283.

Benayoun, R., de Montgolfier, J., Tergny, J., Laritchev, O., 1971. Linear programming with multiple objective functions: Step method (stem). Math. Program. 1 (1), 366–375.

Bogetic, S., Bevins, J., Bernstein, L., Slaybaugh, R., Vujic, J., 2018. Metaheuristic optimization method for neutron spectra shaping. Trans. Am. Nucl. Soc. 118.

Brownlee, J., 2011. Clever Algorithms: Nature-Inspired Programming Recipes. Jason Brownlee.

Buckhorst, A.F., Schmitt, R.H., 2020. Multi-staged, multi-objective optimization for operation management in line-less mobile assembly systems (lmas). Proc. CIRP (ISSN: 2212-8271) 93, 1121–1126. http://dx.doi.org/10.1016/j.procir.2020.04.046, https://www.sciencedirect.com/science/article/pii/S2212827120306119, 53rd CIRP Conference on Manufacturing Systems 2020.

Chiandussi, G., Codegon, M., Ferrero, S., Varesio, F., 2012. Comparison of multi-objective optimization methodologies for engineering applications. Comput. Math. Appl. 63 (5), 912–942.

Cui, Y., Geng, Z., Zhu, Q., Han, Y., 2017. Review: Multi-objective optimization methods and application in energy saving. Energy 125, 681–704, URL: http://search.proquest.com/docview/1932185331/.

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. Struct. Optim. 14 (1), 63–69. http://dx.doi.org/10.1007/BF01197559.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.

Deb, K., Sundar, J., 2006. Reference point based multi-objective optimization using evolutionary algorithms. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. In: GECCO '06, Association for Computing Machinery, New York, NY, USA, pp. 635–642. http://dx.doi.org/10.1145/1143997.1144112.

Domingos, R.P., Schirru, R., Pereira, C.M.N.A., 2006. Particle swarm optimization in reactor core design. Nucl. Sci. Eng. 152 (2), 197–203. http://dx.doi.org/10.13182/NSE06-A2575, arXiv:https://doi.org/10.13182/NSE06-A2575.

Duro, J.A., Kumar Saxena, D., Deb, K., Zhang, Q., 2014. Machine learning based decision support for many-objective optimization problems. Neurocomputing 146, 30–47. http://dx.doi.org/10.1016/j.neucom.2014.06.076, URL: https://www.sciencedirect.com/science/article/pii/S0925231214008753. Bridging Machine learning and Evolutionary Computation (BMLEC) Computational Collective Intelligence.

Fonseca, C., Fleming, P., 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Genetic Algorithms: Proceedings of the Fifth International Conference.

Gebreslassie, B.H., Diwekar, U.M., 2018. Heterogeneous multi-agent optimization framework with application to synthesizing optimal nuclear waste blends. Clean Technol. Environ. Policy 20, 137–157. http://dx.doi.org/10.1007/s10098-017-1464-4.

Glover, F., 1990. Tabu search: A tutorial. Interfaces 20 (4), 74–94.

Gougar, H., Ougouag, A., Terry, W., Ivanov, K., 2010. Automated design and optimization of pebble-bed reactor cores. Nucl. Sci. Eng. 165 (3), 245–269. http://dx.doi.org/10.13182/NSE08-89.

Han, K.-H., Kim, J.-H., 2002. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. IEEE Trans. Evol. Comput..

Hansen, M.P., 1997. Tabu search for multiobjective optimization: MOTS. In: Proceedings of the 13th International Conference on Multiple Criteria Decision Making. Springer-Verlag.

Hourcade, E., Jasserand, F., Ammar, K., Patricot, C., 2013. SFR core design: a system-drive multi-criteria core optimization exercise with TRIAD. In: Status of ASTRID Nuclear Island Pre-Conceptual Design.

Ingremeau, X., Rimpault, G., Dumaz, P., David, S., Plancq, D., Dardour, S., Zabiégo, M., 2011. FARM: A new tool for optimizing the core performance and safety characteristics of Gas Cooled Fast Reactor cores.

Kajihara, T., Tsvetkov, P., 2019. Parametric optimization of TRU destruction rates in HTR cores using genetic algorithms and regression methods. In: Proceeding of the American Nuclear Society - Winter 2019. pp. 371–373.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4. pp. 1942–1948. http://dx.doi.org/10.1109/ICNN.1995.488968.

Kirkpatrick, S., Gelatt, Jr., C.D., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.

Kumar, A., Tsvetkov, P.V., 2015. A new approach to nuclear reactor design optimization using genetic algorithms and regression analysis. Ann. Nucl. Energy 85, 27–35. http://dx.doi.org/10.1016/j.anucene.2015.04.028, URL: http://www.sciencedirect.com/science/article/pii/S0306454915002285.

Li, P., Li, S., 2007. Quantum-inspired evolutionary algorithm for continuous space optimization based on Bloch coordinates of qubits. Neurocomputing 72 (1), 581–591. http://dx.doi.org/10.1016/j.neucom.2007.11.017, URL: http://www.sciencedirect.com/science/article/pii/S0925231207003839.

Marler, R., Arora, J., 2004. Survey of multi-objective optimization methods for engineering. Struct. Multidiscip. Optim. 26 (6), 369–395. http://dx.doi.org/10.1007/s00158-003-0368-6.

Messac, A., 1996. Physical programming: Effective optimization for computational design. AIAA J. 34 (1), 149–158.

Messac, A., Sukam, C., Melachrinoudis, E., 2001. Mathematical and pragmatic perspectives of physical programming. AIAA J. 39 (5), 855–893.

Ney, A., Fritz, D., Singh, S., Langlitz, N., 2018. Genetic algorithm-based optimization for nuclear criticality experiment design. Trans. Am. Nucl. Soc. 119.

Park, T.K., Joo, H.G., Kim, C.H., 2014. Multicycle fuel loading pattern optimization by multiobjective simulated annealing employing adaptively constrained discontinuous penalty function. Nucl. Sci. Eng. 176 (2), 226–239. http://dx.doi.org/10.13182/NSE12-41, arXiv:https://doi.org/10.13182/NSE12-41.

Qvist, S., Greenspan, E., 2014. The ADOPT code for automated fast reactor core design. Ann. Nucl. Energy 71, 23–36. http://dx.doi.org/10.1016/j.anucene.2014.03.013, URL: http://www.sciencedirect.com/science/article/pii/S030645491400125X.

Raza, W., Kim, K.-Y., 2007. Evaluation of surrogate models in optimization of wire-wrapped fuel assembly. J. Nucl. Sci. Technol. 44 (6), 819–822. http://dx.doi.org/10.1080/18811248.2007.9711319, URL: https://www.tandfonline.com/doi/abs/10.1080/18811248.2007.9711319, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/18811248.2007.9711319.

Raza, W., Kim, K.-Y., 2009. Shape optimization of 19-pin wire-wrapped fuel assembly of LMR using multiobjective evolutionary algorithm. Nucl. Sci. Eng. 161 (2), 245–254. http://dx.doi.org/10.13182/NSE161-245, arXiv:https://doi.org/10.13182/NSE161-245.

Reynoso-Meza, G., Sanchis, J., Blasco, X., García-Nieto, S., 2014. Physical programming for preference driven evolutionary multi-objective optimization. Appl. Soft Comput. 24, 341–362. http://dx.doi.org/10.1016/j.asoc.2014.07.009, URL: https://www.sciencedirect.com/science/article/pii/S1568494614003391.

Safarzadeh, O., Zolfaghari, A., Zangian, M., Noori-kalkhoran, O., 2014. Pattern optimization of PWR reactor using hybrid parallel artificial bee colony. Ann. Nucl. Energy 63, 295–301. http://dx.doi.org/10.1016/j.anucene.2013.08.011, URL: http://www.sciencedirect.com/science/article/pii/S0306454913004192.

Saxena, D.K., Duro, J.A., Tiwari, A., Deb, K., Zhang, Q., 2013. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. IEEE Trans. Evol. Comput. 17 (1), 77–99. http://dx.doi.org/10.1109/TEVC.2012.2185847.

Sherinov, Z., Unveren, A., 2017. Multi-objective imperialistic competitive algorithm with multiple non-dominated sets for the solution of global optimization problems. Soft Comput. 22, 8273–8288. http://dx.doi.org/10.1007/s00500-017-2773-6.

Steuer, R., Choo, E.-U., 1983. An interactive weighted tchebycheff procedure for multiple objective programming. Math. Program. 26 (3), 326–344.

Stewart, R., Palmer, T.S., 2021. Utilizing a reduced-order model and physical programming for preliminary reactor design optimization. EPJ Web Conf. 247, 06049. http://dx.doi.org/10.1051/epjconf/202124706049.

Stover, T.E., Turinsky, P.J., 2012. Experiment optimization to reduce nuclear data uncertainties in support of reactor design. Nucl. Technol. 180 (2), 216–230.

Vargas, D.E., Lemonge, A.C., Barbosa, H.J., Bernardino, H.S., 2021. Solving multi-objective structural optimization problems using GDE3 and NSGA-II with reference points. Eng. Struct. 239, 112187. http://dx.doi.org/10.1016/j.engstruct.2021.112187, URL: https://www.sciencedirect.com/science/article/pii/S0141029621003370.

Wang, C., Yan, C.-Q., Wang, J.-J., Li, F.-F., 2016. Multi-objective optimization of electric heating pressurizer in nuclear power plant. In: ICONE, International Conference on Nuclear Engineering, 24.

Wei, X., Fujimura, S., 2010. Multi-objective quantum evolutionary algorithm for discrete multi-objective combinational problem. In: 2010 International Conference on Technologies and Applications of Artificial Intelligence.

Wilding, P.R., Murray, N.R., Memmott, M.J., 2020. The use of multi-objective optimization to improve the design process of nuclear power plant systems. Ann. Nucl. Energy 137, 107079. http://dx.doi.org/10.1016/j.anucene.2019.107079, URL: http://www.sciencedirect.com/science/article/pii/S0306454919305870.

Woo, M.H., Kim, J.H., Kim, J.W., Yim, C.W., Lee, J.Y., Kim, D.H., Khuat, Q.H., Seo, B.K., Shin, C.H., Kim, J.K., 2019. An application of genetic multi-objective optimization algorithm to neutron spectrum unfolding problem. Prog. Nucl. Sci. Technol. 6, 230–233.

Wu, S.-C., Chan, T.-H., Hsieh, M.-S., Lin, C., 2016. Quantum evolutionary algorithm and tabu search in pressurized water reactor loading pattern design. Ann. Nucl. Energy 94, 773–782. http://dx.doi.org/10.1016/j.anucene.2016.04.039, URL: http://www.sciencedirect.com/science/article/pii/S030645491630202X.

Yadav, R., Gupta, H., 2011. Optimization studies of fuel loading pattern for a typical pressurized water reactor (PWR) using particle swarm method. Ann. Nucl. Energy 38 (9), 2086–2095. http://dx.doi.org/10.1016/j.anucene.2011.05.019, URL: http://www.sciencedirect.com/science/article/pii/S030645491100209X.

Yi, C., Sjoden, G., 2013. Energy group structure determination using particle swarm optimization. Ann. Nucl. Energy 56, 53–56.

Yi, C., Sjoden, G., Edgar, C., 2016. Group structure optimization using the PyGroup code. Trans. Am. Nucl. Soc. 114.

Yilmaz, S., Ivanov, K., Levine, S., Mahgerefteh, M., 2006. Application of genetic algorithms to optimize burnable poison placement in pressurized water reactors. Ann. Nucl. Energy 33 (5), 446–456. http://dx.doi.org/10.1016/j.anucene.2005.11.012, URL: http://www.sciencedirect.com/science/article/pii/S030645490500277X.

Yuan, Y., Ong, Y., Gupta, A., Xu, H., 2018. Objective reduction in many-objective optimization: Evolutionary multiobjective approaches and comprehensive analysis. IEEE Trans. Evol. Comput. 22 (2), 189–210. http://dx.doi.org/10.1109/TEVC.2017.2672668.

Zeng, K., Stauff, N.E., Hou, J., Kim, T.K., 2020. Development of multi-objective core optimization framework and application to sodium-cooled fast test reactors. Prog. Nucl. Energy 120, 103184. http://dx.doi.org/10.1016/j.pnucene.2019.103184, URL: http://www.sciencedirect.com/science/article/pii/S0149197019302938.

Zhang, R., Wang, Z., Zhang, H., 2014. Quantum-inspired evolutionary algorithm for continuous space optimization based on multiple chains encoding method of quantum bits. Math. Probl. Eng. http://dx.doi.org/10.1155/2014/620325.

Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report, Swiss Federal Institute of Technology Zurich.