Struct Multidisc Optim (2013) 48:763–775
DOI 10.1007/s00158-013-0925-6

RESEARCH PAPER

The smart normal constraint method for directly generating a smart Pareto set

B. J. Hancock · C. A. Mattson

Received: 18 October 2012 / Revised: 7 February 2013 / Accepted: 7 March 2013 / Published online: 19 June 2013

Abstract In design situations where a single solution must be selected, it is often desirable to present the designer with a smart Pareto set of solutions—a minimal set of non-dominated solutions that sufficiently represents the tradeoff characteristics of the design space. These sets are generally created by finding many well-distributed solutions and then either filtering out the excess ones or searching more closely in those regions that appear to have significant trade-off. Such methods suffer from the inherent inefficiency of creating numerous solutions that will never be presented to the designer. This paper introduces the Smart Normal Constraint (SNC) method—a Pareto set generation method capable of *directly generating* a smart Pareto set. Direct generation is achieved by iteratively updating an approximation of the design space geometry and searching only in those regions capable of yielding new smart Pareto solutions. This process is made possible through the use of a new, computationally benign calculation for identifying regions of high tradeoff in a design space. Examples are provided that show the SNC method performing significantly more efficiently than the predominant existing method for generating smart Pareto sets.

Keywords Multiobjective optimization · Minimal Pareto set · Smart Pareto filter · Normal constraint method · Direct generation

B. J. Hancock · C. A. Mattson (✉)
Department of Mechanical Engineering,
Brigham Young University,
Provo, UT 84602, USA
e-mail: mattson@byu.edu

1 Introduction and literature survey

Engineering design often involves making decisions between two or more conflicting objectives. When the designer faces such decisions, more than one solution may exist that will meet the design goals. In these multiobjective problems, knowledge about the Pareto frontier of the design space can help designers sift through the potentially large body of feasible solutions. The Pareto frontier is the set of all solutions—known as Pareto optimal solutions—for which no other solution is better in all objectives (Pareto 1964). Given a set of Pareto optimal solutions, a designer can recognize what the various optimal solutions might be, depending on the value associated with each objective in the problem. Pareto sets are therefore especially useful in understanding the tradeoff relationships between particular objectives in a multiobjective problem.

1.1 Evolution of Pareto set generation algorithms

Many algorithms exist for generating Pareto sets. However, not all algorithms are equally efficient or effective at representing the design space. The strategies used to generate Pareto sets have evolved through a number of phases as the discipline of multiobjective optimization has matured (Mattson et al. 2004). In phase one, attempts were made to obtain any set of Pareto solutions, from which the final design could be selected. In phase two, specific sets of Pareto optimal solutions were sought that were equally distributed along the Pareto frontier. These equally distributed sets could be viewed as superior to others because they guarantee that no significantly large portion of the Pareto frontier is unrepresented. In the most recent phase of the evolution of Pareto set generation strategies, a push has been made to obtain minimal sets of Pareto optimal solutions that adequately represent the tradeoff properties of the entire Pareto frontier. Because a single design is eventually selected from the entire design space in most design scenarios, it is desirable to somehow provide the designer with as few designs as possible, while still informing him or her of all the possible potential regions of high tradeoff in that space. This minimal set that simplifies the choice of a final design without significant loss of information has been termed a *smart Pareto set* (Mattson et al. 2004).

Under the current state of the art, this smart Pareto set is generally created by producing a well-distributed set, and then filtering out all solutions that do not represent a significant amount of tradeoff between objectives with respect to any other solution already in the set. It is built on the assumption that a designer is often willing to sacrifice a small amount in one objective if a large benefit in another objective could be gained. Thus, fewer designs are needed in these regions of relatively insignificant tradeoff in order to provide the designer with a sufficient amount of information about what combinations of objective values are obtainable.

In this paper, the authors propose a significant improvement to this third phase of Pareto set development—direct generation of smart Pareto sets, in contrast to using a smart Pareto filter. The smart Pareto filter approach is to first generate many solutions, then reduce the set by removing solutions that are deemed insignificantly different from other Pareto solutions. While this approach is beneficial in many instances, at least one potential drawback is the computational inefficiency of generating a large number of designs, only to remove a significant portion of them from consideration. Therefore, a desirable goal would be to generate only those solutions that will be of interest to the designer; in other words, to directly generate the smart Pareto set. In this paper, an algorithm is proposed that directly generates a smart Pareto set of solutions through the use of a new scalar term known as the *smart distance* between solutions. In nearly all cases, this results in a significant decrease in the amount of required function calls to produce a smart Pareto set.

1.2 Survey of minimal representation algorithms

In situations where the design variable space includes continuous variables, the Pareto frontier can include an infinite number of potential solutions. Even in situations where it is theoretically possible to obtain all solutions, this is often prohibitively computationally expensive (Ruzika and Wiecek 2005) and impractical to portray graphically, particularly in problems with many objectives (Aittokoski et al. 2009). Furthermore, such a large amount of data may be difficult for the designer to analyze. As a result, many methods have been created with the intent of abbreviating or consolidating the set of Pareto solutions to be presented to the designer. In-depth surveys of multiobjective optimization methods as a whole can be found in Ruzika and Wiecek (2005); Marler and Arora (2004).

Many algorithms exist for identifying solutions located on regions of the Pareto frontier known as "knee points," which occur where an improvement in one objective results in significant worsening of at least one other objective (Bechikh et al. 2010; Schutze and Laumanns 2008; Deb and Tiwari 2006). These solutions can be identified and provided to the designer after the full Pareto frontier has been found, or optimization can be performed specifically in the region of a knee point after developing an approximation of the Pareto frontier (Rachmawati and Srinivasan 2009). Another method for minimizing the number of final solutions in a Pareto set is known as data clustering. It consists of grouping multiple points into clusters based upon their compactness, connectedness, or spatial separation from other clusters, in order to select a few solutions that represent each of the relatively unique regions of the Pareto frontier (Aittokoski et al. 2009; Handl and Knowles 2007). Evolutionary algorithms have been used in many of these instances to locate knees and identify clusters (Rachmawati and Srinivasan 2009; Zitzler and Thiele 1998). Also, as mentioned before, a filtering method may be used to identify regions of practically insignificant tradeoff (PIT) and remove solutions that provide little unique information to the designer (Mattson et al. 2004).

Apart from other strengths and weaknesses that these various methods may have, they all have in common the inefficiency that comes with generating many solutions that will never be considered by the designer. All of these methods present the designer with a subset of the solutions that they have generated. Currently unaddressed in the literature is the development of a method capable of producing primarily just those solutions that will be a part of the minimal set provided to the designer. The method proposed in this paper to achieve that purpose is partially based upon the Normal Constraint (NC) method (Messac et al. 2003). The literature concerning the NC method and its applicable variations is reviewed in Section 2.3.

The remainder of this paper is organized as follows: we begin in Section 2 by reviewing the traditional multiobjective optimization problem formulation and the normal constraint method of generating an evenly distributed Pareto set. In Section 3 we introduce the primary mechanism that enables direct generation of smart Pareto sets. In Section 4 we introduce and discuss the SNC method analytically and mathematically. In Section 5 we demonstrate the efficiency of the SNC method with three popular example problems and observe its utility. Finally, in Section 6 we offer concluding remarks.

2 Technical preliminaries

This section provides the traditional mathematical definition of a multiobjective optimization problem (MOP) and identifies the important features that are required for understanding the concepts presented throughout this paper. The NC method is then introduced along with two of its variations, which together serve as a foundation for the SNC method.

2.1 The multiobjective optimization problem

The generic MOP can be stated as Problem 1 (*P1*):

$$\min_{\mathbf{x}} \; \{\mu_1(\mathbf{x}),\; \mu_2(\mathbf{x}),\; \ldots,\; \mu_n(\mathbf{x})\} \qquad (n \geq 2) \tag{1}$$

subject to

$$\mathbf{g}(\mathbf{x}) \leq 0 \tag{2}$$

$$\mathbf{h}(\mathbf{x}) = 0 \tag{3}$$

$$\mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \tag{4}$$

where $\mathbf{x}$ is a vector of design variables, $\boldsymbol{\mu}$ is a vector of design objectives, $\mathbf{g}$ and $\mathbf{h}$ are inequality and equality constraint vectors, respectively, and $\mathbf{x}_l$ and $\mathbf{x}_u$ are vectors containing the lower and upper bounds of the design variables. Here and throughout this paper, the variable n refers to the number of objectives in the problem. In this form, *P1* produces a set of Pareto optimal solutions. In Section 2.2, a common Pareto set generation algorithm is presented whose primary strategy for generating individual solutions serves as the basis for the SNC method proposed in this paper.

Each point in the design space of an MOP represents a feasible design solution. Therefore, for the remainder of this paper, these two terms (point and solution) will be used interchangeably. Two important types of reference points which exist in the design space of every MOP are defined below.

Anchor points are specific points in the feasible design space that correspond to the *minimum* values of the respective individual objectives. The anchor point for the i-th objective is expressed as

$$\mu^{i*} = [\mu_1(\mathbf{x}^{i*}) \;\; \mu_2(\mathbf{x}^{i*}) \;\; \ldots \;\; \mu_n(\mathbf{x}^{i*})]^T \tag{5}$$

where $\mathbf{x}^{i*}$ is defined as $\mathbf{x}^{i*} = \arg \min_{\mathbf{x}} \; \mu_i(\mathbf{x})$ subject to the constraints of the *P1*, given by (2) and (4).

Anti-anchor points are specific points in the feasible design space that correspond to the *maximum* values of the respective individual objectives. The anti-anchor point for the i-th objective is expressed as

$$\mu^{i\circ} = [\mu_1(\mathbf{x}^{i\circ}) \;\; \mu_2(\mathbf{x}^{i\circ}) \;\; \ldots \;\; \mu_n(\mathbf{x}^{i\circ})]^T \tag{6}$$

where $\mathbf{x}^{i\circ}$ is defined as $\mathbf{x}^{i\circ} = \arg \max_{\mathbf{x}} \; \mu_i(\mathbf{x})$ subject to the constraints of the *P1*, given by (2) and (4).

2.2 Review of the normal constraint method

Under the Normal Constraint (NC) method, the MOP is converted into a series of single-objective optimization (SOO) problems, each with a different set of additional linear constraints calculated to produce a Pareto solution in a particular region of the design space. The NC method consists of 5 steps, which we will outline in this section in the context of a bi-objective sample problem, shown in Fig. 1. In problems where $n > 2$, the lines described in these steps are replaced by their higher dimensional counterparts, planes or hyperplanes. Further details on the method may be found in Ismail-Yahaya and Messac (2002).

Step 1: Generation of reference points
Use (5) to locate the anchor points. In Fig. 1, the anchor points have been represented by stars.

Step 2: Construction of utopia line vector(s)
The line connecting the anchor points is known as the utopia line. Define the utopia line vector N_j using the equation

$$N_j = \mu^{j*} - \mu^{n*} \qquad \forall j \in (1, 2, \ldots, n-1) \tag{7}$$

Thus, in the case of $n > 2$, $n - 1$ utopia line vectors are defined, all of which point to μ^{n*}, the anchor point corresponding to dimension n.

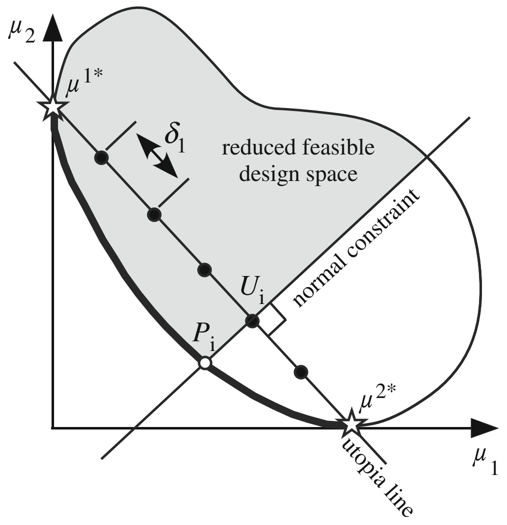


Fig. 1 Graphical representation of the NC method for a bi-objective problem

Step 3: Calculation of utopia line increments
Based upon the number of utopia line points m_j that the user desires in each utopia line direction N_j, an increment δ_j is created, using the equation

$$\delta_j = \frac{1}{m_j - 1} \qquad \forall j \in (1, 2, ..., n-1) \tag{8}$$

Step 4: Generation of utopia line points
Generate each utopia line point using equation

$$U_i = \sum_{j=1}^{n} \alpha_i^j \mu^{j*} \tag{9}$$

where the non-dimensional parameter α_i^j satisfies

$$0 \leq \alpha_i^j \leq 1 \tag{10}$$

and

$$\sum_{j=1}^{n} \alpha_i^j = 1 \tag{11}$$

Note that by incrementing α^j by δ_j between 0 and 1, an even distribution of points is generated between the provided utopia line points.

Step 5: Single-objective optimization
For each utopia plane point U_i solve Problem 2 (*P2*):

$$\min_{\mathbf{x}} \mu_n(\mathbf{x}) \tag{12}$$

subject to (2) and (4) as well as

$$N_j(\mu(\mathbf{x}) - U_i)^T \leq 0 \qquad \forall j \in (1, 2, ..., n-1) \tag{13}$$

This additional linear constraint given by (13) excludes all points found below the line that intersects the utopia line point and is orthogonal to the utopia line. Thus, from each utopia line point is produced a corresponding point on the Pareto frontier. Figure 1 shows the Pareto point P_i that was produced by *P2* using utopia line point U_i. When *P2* is solved using a gradient-based approach, locally Pareto points may be found by the NC method where the Pareto frontier is disjointed. For this reason, users are encouraged to use the NC method in conjunction with a global Pareto filter, as described in Messac et al. (2003). In applications difficult for gradient-based algorithms, other SOO algorithms may be used, such as genetic algorithms.

2.3 NC method improvements

Since the inception of the NC method in 2002, this algorithm has been widely used and researched. Consequently, many variants and improvements of the NC method have been proposed. A year after introducing the NC method, the original authors proposed the normalized normal constraint (NNC) method, which mitigates objective scaling issues by normalizing the design objective space before carrying out the steps of the NC method (Messac et al. 2003). Means of improving the distribution of Pareto solutions by modifying or replacing the utopia plane have been suggested by Martinez et al. (2007); Sanchis et al. (2008); Motta et al. (2012). Hybrid algorithms combining the NNC method with evolutionary algorithms have been proposed in order to avoid local optima (Martinez et al. 2007, 2009). Martinez et al. (2007) also recently proposed the uniform normalized normal constraint method, which uses the distribution of known Pareto solutions to guide it in searching for a new set of Pareto solutions that are more uniformly distributed along the Pareto frontier.

While the many variants reviewed here have improved the effectiveness and flexibility of the NC method, they focus on the second phase of Pareto set generation algorithms—the generation of well-distributed sets. Still lacking in the literature are modifications to the NC method that will carry it into the third phase—the generation of smart Pareto sets.

Two other variants in particular are significant for the purposes of this paper because the fundamental principles behind them are integral to the SNC method. They are briefly explained here, but for a full understanding of these methods and how to implement them, see Messac and Mattson (2004) and Boyce and Mattson (2008), respectively.

In order to guarantee even representation of the entire Pareto frontier, Messac and Mattson propose that for problems where $n > 2$, the utopia plane be extended to include not only the region bounded by the anchor points, but rather all regions of the utopia plane that could produce a Pareto point in the design space upon the evaluation of *P2* (Messac and Mattson 2004). This extended region of the utopia plane is bounded by the anchor points as well as the perpendicular projections of the anti-anchor points. Without an extended utopia plane, there is no guarantee that the generated set will represent the complete Pareto frontier for problems where $n > 2$.

Because a design space with a disjointed Pareto set is capable of performing multiple single objective optimizations that produce the same Pareto point, Boyce and Mattson propose a method of recognizing which utopia plane points will reduce redundant Pareto points and avoiding these SOOs (Boyce and Mattson 2008). This is done by recognizing when at least one of the normal linear constraints used in generating a point is not active. When this is the case, one can remove all utopia plane points that lie in the region between the normal constraints that actually generated the given point (but are separated from it) and the parallel normal constraints that would be generated directly through the given point such that all normal constraints would be active. Figure 7c in Section 4.3.3 shows a situation in which this improvement could potentially eliminate a number of redundant SOOs.

3 Mechanisms for smart Pareto set generation

Mattson et al. (2004) first introduced the concept of a smart Pareto set, based upon the assumption that "when the tradeoff is significant… a designer is willing to give up an insignificant amount in one objective to gain significantly in another." In this paper, direct generation of a smart Pareto set is achievable through the use of a scalar value that can be assigned to any point in the design space based upon its distance and direction from all other Pareto solutions. To fully appreciate the value of this new technique, we must first consider the two mechanisms which have enabled past attempts at producing smart Pareto sets of points—the smart Pareto filter (Mattson et al. 2004) and smart constraints (Haddock et al. 2008).

3.1 The smart Pareto filter

The fundamental concept of the smart Pareto filter is that there is a user-defined shape—known as the Practically Insignificant Tradeoff (PIT) region—surrounding each Pareto solution, inside of which no other Pareto solution may reside. This PIT region is depicted in Fig. 2a for a bi-objective case. The user defines the PIT region by providing values for two control parameters $\Delta\mathbf{t}$ and $\Delta\mathbf{r}$. The smart Pareto filter operates by arbitrarily selecting a point in the Pareto set and removing all points that lie within the PIT region surrounding it. This process is then repeated for all remaining points in the set. One strength of this approach for creating a smart Pareto set is that it may be used in conjunction with any algorithm capable of producing a well-distributed Pareto set. A weakness, however, is that with this approach, the designer could potentially spend valuable resources generating solutions in areas of insignificant tradeoff that will be discarded without providing any valuable information to the designer.

3.2 Direct generation by smart constraints

Haddock et al. (2008) suggested a method of producing a smart Pareto set with a new type of PIT region formed by additional linear constraints, known as *smart constraints*. User-provided values for parameters $\Delta\mathbf{b}$ and $\Delta\mathbf{s}$ define this region, as shown in Fig. 2b. For details on how the smart constraints are formulated, see Haddock et al. (2008). The developers of this method found it had significant drawbacks. Because each Pareto solution discovered introduces a new constrained PIT region that is retained in subsequent SOOs, the reduced feasible design space becomes highly multimodal, causing difficulties for gradient-based algorithms. They concluded about their own method, "it will nearly always take more function evaluations to directly generate smart Pareto sets, than it would be to simply generate numerous solutions, and remove unwanted solutions by smart filtering, as proposed by Mattson et al," (Haddock et al. 2008).

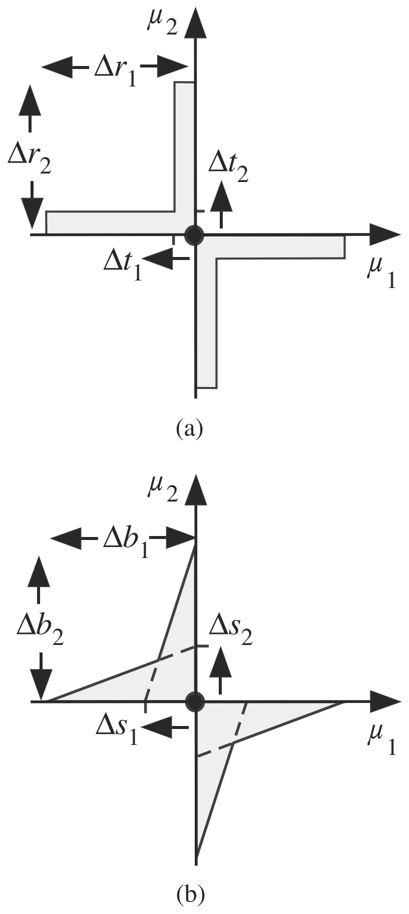


Fig. 2 The user-defined PIT region (*shaded*) surrounding a point when using (**a**) the smart Pareto filter (Mattson et al. 2004) and (**b**) smart constriants (Haddock et al. 2008)

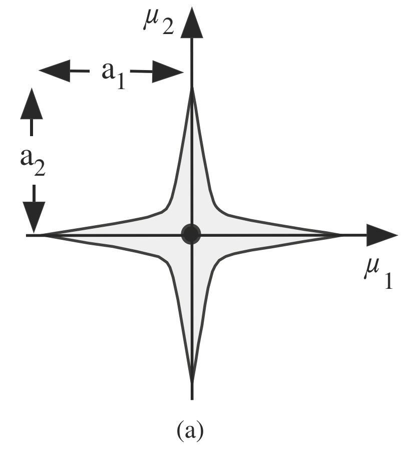
3.3 Direct generation by smart distance

As will be demonstrated in Section 4 and 5, direct generation of a smart Pareto set is possible with the assistance of a scalar term—the *smart distance* between points in the design space. For this mechanism, the shape of the PIT region around a point is called a Lamé curve in 2D or a hyper-Lamé curve in nD (see Fig. 3). The PIT region consists of all points that lie on or within the curve. These points all have a smart distance $s \leq 1$ from the center point. Because all members of a smart Pareto set do not lie within the PIT regions of any other member, this means that each will have a smart distance of $s \leq 1$ with respect to all other members in the set. The formula for the smart distance between two points is given by the equation

$$s = \|\mathbf{Ad}\|_p \qquad (0 < p \leq 2) \tag{14}$$

where

$$\mathbf{A} = \begin{bmatrix} \frac{1}{\mathrm{a}_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\mathrm{a}_n} \end{bmatrix} \qquad (\mathbf{a} > 0) \tag{15}$$



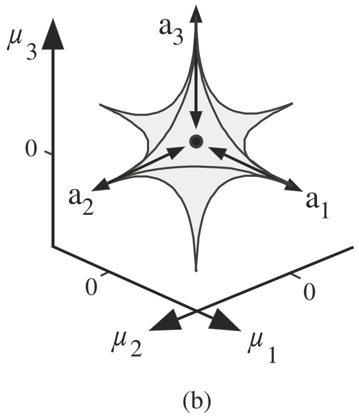


Fig. 3 The user-defined PIT region (*shaded*) surrounding a point when using smart distance for a (**a**) 2D and (**b**) 3D case. Mathematically, these regions contain all points with a smart distance $s \leq 1$

d is a vector between the two points in the design space, and $\|\mathbf{Ad}\|_p$ follows the accepted formula for calculating the p-norm of a vector (Rynne 2007), which in this case is given by

$$\|\mathbf{Ad}\|_p = (\sum_{i=1}^{n} |A_{i,i}d_i|^p)^{\frac{1}{p}} \tag{16}$$

The variables **a** and p are user-defined values that allow the user to determine the distribution of the smart Pareto points that will be generated for that particular problem. Each value a_i corresponds to objective i in the problem and may be interpreted as the amount of change in that objective that would constitute a significant difference between two points in the user's mind if all other objectives remain practically unchanged. As shown in Fig. 3, any Pareto point that lies within the distance a_i of another Pareto point without significant tradeoff in one or more other objectives will fall within the PIT region and be discarded. Thus, larger values for the elements of **A** will result in fewer points in a smart Pareto set. The parameter p affects the curvature of the PIT region and therefore controls the extent to which high tradeoff between objectives is required in order for two points nearby each other to both remain in the smart Pareto set. The effect of p on the shape of the PIT region is illustrated in Fig. 4. While the method will work for any value of p between 0 and 2, it is assumed that for most purposes, the user will select a value between 0 and 1, resulting in a shape that resembles the PIT regions of other mechanisms.

As with the other mechanisms, once these user-defined values have been given, the algorithm can run autonomously until a complete smart Pareto set has been generated. The lack of dependence on real-time input from a user allows the algorithm to work quickly. Because the user has stated explicitly what differences in tradeoff he or she considers to be significant enough to merit representation in the final smart Pareto set, it is the user's preferences that ultimately determine the distribution of that set.

The mechanism of smart distance is unique in that it defines a PIT region by a single scalar value (smart distance) rather than by the region bounded by multiple lines with differing equations. As will be seen in Section 4, this allows for an algorithm to identify not just whether or not a new point is a smart Pareto point, but also to what extent the point is "smart." This ability is what enables the SNC method to more efficiently search a design space for a full smart Pareto set than existing methods for identifying smart Pareto sets.

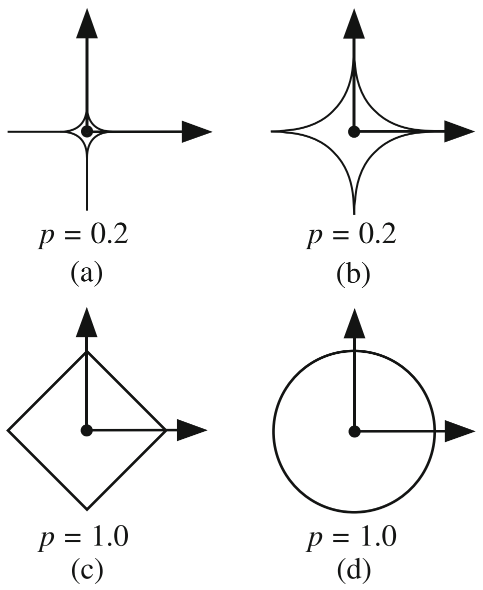


Fig. 4 The effect of the user-defined parameter p on the shape of the smart distance PIT region where all values of (**a**) are equal

4 The smart normal constraint method

This section introduces and discusses the Smart Normal Constraint (SNC) method. The method and its advantages over existing methods will be first discussed analytically, then described mathematically for an n-objective case. Numerical examples for a 2D, 3D, and 5D case are provided in Section 5.

4.1 An analytical description of the SNC method

The purpose of the SNC method is to directly generate smart Pareto sets in a computationally efficient way. The process for generating this smart Pareto set using the SNC method is illustrated for two dimensions in Fig. 5. Figure 5a shows that the anchor points have been identified and a line has been drawn between them that is divided up by a series of points. At this stage in the process, the SNC method is identical to the NC method. The main theoretical concept underlying the SNC method is that the constructed line is an approximation of the Pareto frontier. Clearly, for this first iteration, it is a low fidelity approximation. The approximation is improved, however, as each new Pareto point is found during the course of the optimization. The approximation has visibly improved in Fig. 5b, for example, as we now have two segments of piece-wise linearly distributed points forming the approximation instead of one. Having an updated approximation of the Pareto frontier provides the algorithm with information about where new smart Pareto points are

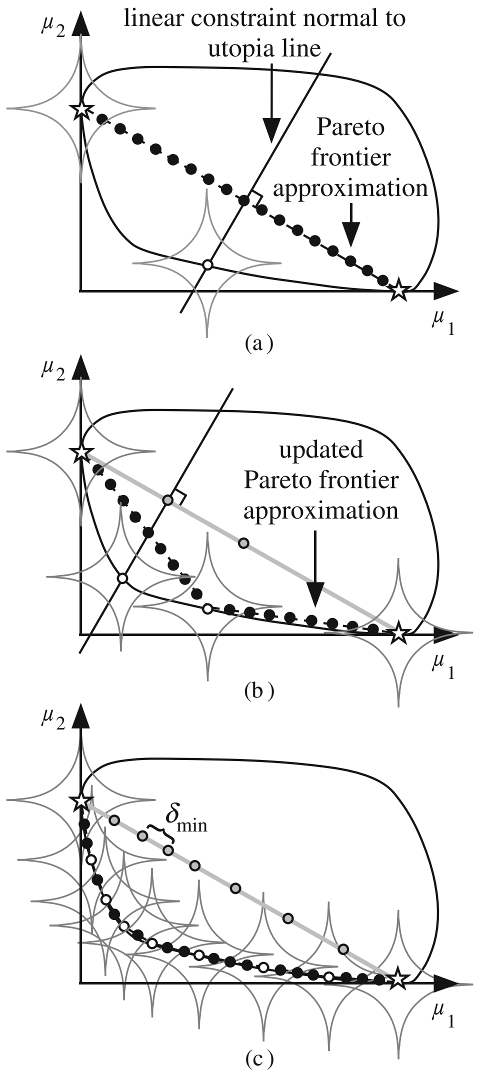

μ_2
linear constraint normal to utopia line
Pareto frontier approximation
μ_1
(a)
μ_2
updated Pareto frontier approximation
μ_1
(b)
μ_2
$\delta_{\min}$
μ_1
(c)

Fig. 5 The SNC method in progress for a bi-objective case: (**a**) after the first SOO, (**b**) after the second SOO, using an updated Pareto frontier approximation, and (**c**) upon completion, with a smart Pareto set. Pareto points are hollow, utopia line points are shaded, and approximation points are filled. Note that the PIT Lamé curves around each Pareto point are not used as constraints. They simply illustrate those regions that are within one smart distance of any Pareto point

most likely to be found. Before each SOO, benign smart distance calculations are performed between the approximation points and all known (currently existing) Pareto points. For each approximation point, the nearest known Pareto point (in terms of smart distance) is identified. The approximation point with the largest smart distance to its nearest known Pareto point is selected as the point that is most likely to yield a new smart Pareto point. A normal constraint is then constructed through that point and an SOO is performed. Repeating this process multiple times results in an increasingly more accurate Pareto frontier approximation. Fig. 5c illustrates the smart Pareto set (hollow points) that remains once no approximation point (filled points) is significantly different from all discovered Pareto points. In other words, no approximation point has a minimum smart distance greater than 1 with respect to the existing set of Pareto points. At this time the algorithm terminates.

It is worth noting that the primary strategy for creating individual Pareto solutions is the same for both the NC and SNC methods—linear constraints perpendicular to the utopia line are constructed and SOOs are performed. However, in the SNC method, these constraints are constructed through iteratively updated approximation points instead of through utopia line points. Thus, instead of requiring the user to predefine the number and locations of SOOs before any information about the true shape of the Pareto frontier is known, the SNC method allows the user to describe the type of distribution which he or she would like to have in the final set of solutions (through parameters **a** and p), and the algorithm dynamically adjusts the spacing between the constructed normal constraints accordingly. This allows for a higher resolution of points in regions with large curvature, and fewer function calls in nearly every case. In Fig. 5c, δ_{min} identifies the spacing of points on the utopia plane (shaded points) that would be required for the NC method to produce a smart Pareto set with the same resolution as the SNC method in this example. In the Appendix, an insightful flowchart highlights the similarities and differences between the flow of the NC and SNC methods.

4.2 A mathematical description of the SNC method

The SNC method can be divided into 7 simple steps. Steps 2-7 repeat until there are no more regions of the Pareto surface approximation that appear capable of yielding a smart Pareto point. Once again, the terms line, plane, and hyperplane can be interchanged to match the dimension of the problem being solved.

Step 1: Generation of reference points
Use (5) and (6) to locate the anchor points and anti-anchor points. While the anti-anchor points are often not Pareto points, including them as vertices on the edges of the Pareto frontier approximation guarantees coverage of the entire Pareto frontier, similar to using an extended utopia plane in the NC method (see Messac and Mattson (2004) in Section 2.3).

Step 2: Connectivity of approximation vertices
Determine how to divide up the approximation of the Pareto frontier into approximation segments or approximation planes. For bi-objective cases, connect each approximation vertex point to the neighboring vertices on either side of it, as was done in Fig. 5. When $n > 2$, find the connectivity of approximation vertices by linearly projecting them onto the utopia plane and finding the Delaunay triangulation of the projected set. Delaunay triangulation subdivides a geometric object into contiguous simplices such that their minimum angles are maximized. Figure 6 shows a Delaunay triangulation of the projections of approximation vertices onto the utopia plane. To perform this step, the authors use the built-in Matlab function *delaunayn*. For more information on how Delaunay triangulation is carried out, see Barber et al. (1996).

Step 3: Approximation of Pareto frontier
Generate evenly spaced approximation points on each approximation plane using (17), which is similar to (9) for distributing points on the utopia plane in the NC method. The anchor points, μ^{j*} are simply replaced with the approximation vertices, P_k, that define each approximation plane, according to the results of Step 2.

$$S_i = \sum_{k=1}^{n} \alpha_i^k P_k \tag{17}$$

where once again, the non-dimensional parameter α_i^k satisfies constraints on α (see (10) and (11)), and α^k is varied from 0 to 1 with a fixed increment of δ_k to result in an even distribution of approximation points over the entire Pareto frontier. The value for δ_k in this equation is once again arbitrary, depending on how close to each other the designer would like the approximation points to be. In practice, the authors have found it simple and effective to set δ_k equal to the shortest Euclidean distance between a center point and

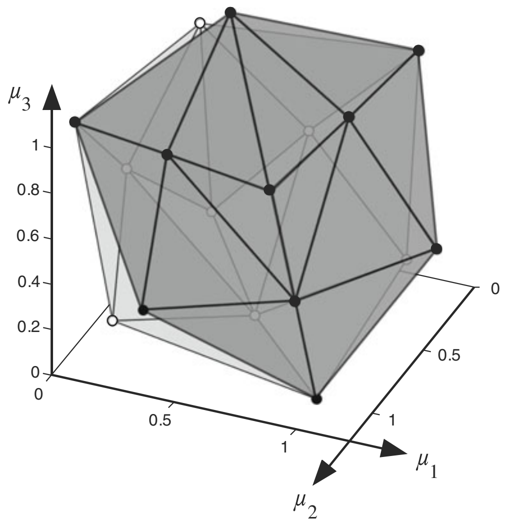


Fig. 6 Approximation vertices (*hollow points*) have been projected onto the utopia plane (*filled points*) and subdivided into triangles by Delaunay triangulation. This connectivity is used to make the planes that together approximate the Pareto frontier

the PIT region that defines one smart distance around it. This may be found using the equation

$$\delta_k = \left\| \min_{\mathbf{d}} \ \|\mathbf{d}\| \right\| \tag{18}$$

where $\mathbf{d}$ is a vector between the center point of the PIT region and any second point on the boundary of the PIT region, and (14) serves as an equality constraint on d with $s = 1$.

A smaller value of δ_k will result in a greater quantity of approximation points. Because all computations performed on approximation points are relatively benign (more approximation points will not result in more function calls of a designer's model, which are generally far more computationally expensive), the efficiency of this algorithm depends very little on selecting an ideal value for δ_k.

Step 4: Removal of restricted approximation points

Some SOOs provide information about certain regions of the Pareto frontier that will not produce smart Pareto points. Such regions exist when the Pareto frontier is discontinuous. In Step 7, these "restricted" regions are recorded. In this step, remove from further consideration any approximation points that lie in those restricted regions of the design space.

Step 5: Calculation of smart distances

The smart distance is calculated between each approximation point and all existing approximation vertices using (14).

Step 6: Generation of new Pareto point

Select the approximation point with the largest smart distance to its nearest known Pareto point and perform an SOO (Problem *P2*) using the standard normal constraints that intersect that point (given by (13)). In a problem where all objectives are independent variables, the initial values of $\mathbf{x}$ for *P2* can be set to the coordinates of the selected approximation point. In problems with dependent variables, the initial values of $\mathbf{x}$ can be set to the linear interpolation of the $\mathbf{x}$ vectors that produced each of the approximation vertices that generated the selected approximation point's plane. In most cases, these selected initial values are closer than those that could be obtained with the NC method using only information about the utopia plane. This results in generally fewer function calls per SOO for the SNC method compared the NC method.

Step 7: Addition of new restrictions

Check to see if the new Pareto point is a) dominated, b) redundant, or c) separated. If the point exhibits at least one of these three restriction characteristics, add a restriction for removing future approximation points in these regions that are now known to not be capable of producing smart Pareto points. These restrictions are explained in detail in Section 4.3.

4.3 Approximation point restrictions

Some points generated by an SOO have special traits that result in them being treated differently than other points and potentially providing additional information to the algorithm. This section provides descriptions of these traits and how they are handled (see Fig. 7 for visual examples). The restrictions corresponding to each trait can be applied in any order and are unaffected by the presence of more than one trait in a newly discovered point.

4.3.1 Dominated

A dominated point is typically produced by the SNC or NC methods when there are local minima or maxima in the design space that a gradient-based algorithm fails to recognize as such. By identifying any dominated points iteratively with a global Pareto filter (as described in Messac et al. 2003), the algorithm is able to avoid using those points as approximation vertices, which would decrease the accuracy of its approximation of the true Pareto frontier. Also, no approximation points that lie on the normal constraint line that produced a dominated solution will be considered for future SOOs.

4.3.2 Redundant

Because the true shape of the Pareto frontier is unknown and is only being approximated in the SNC algorithm, the SOO based upon an approximation point that lies outside all PIT regions may result in a Pareto point that actually does lie within a PIT region of another Pareto point. Where this is the case, the new Pareto point should not be kept in the smart Pareto set. It can, however, be used as an additional approximation vertex, which will improve the fidelity of the approximation for future SOOs. Once again, no approximation points that lie on the normal constraint line that produced a redundant solution will be considered for an SOO.

4.3.3 Separated

Sometimes a point is separated from the normal constraint lines or planes that were used in the SOO that created it. This separation indicates that there is a region of the design space in which all SOOs would yield the same solution (as described in Section 2.3)—in other words, the Pareto

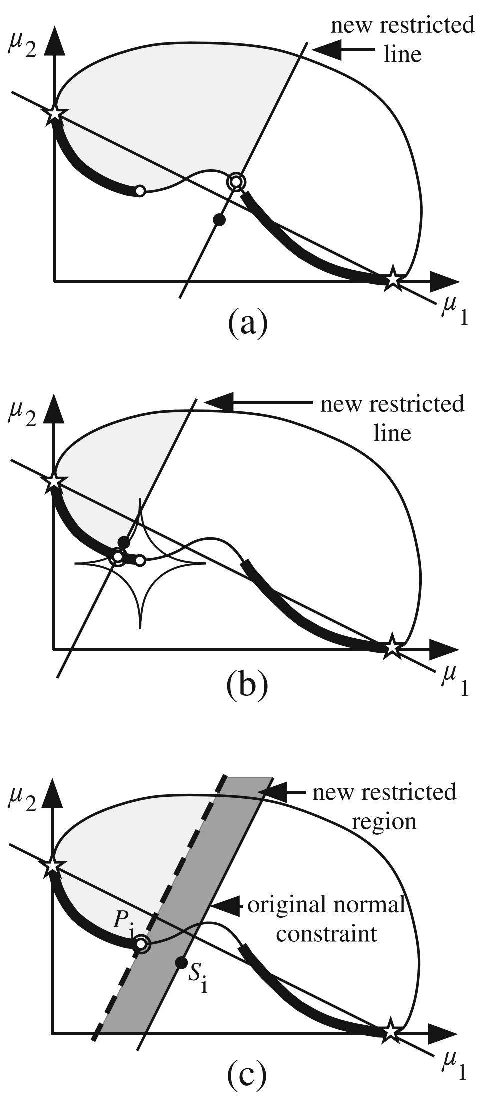


Fig. 7 Newly generated points that exhibit the following restriction traits have been circled: (**a**) dominated, (**b**) redundant, and (**c**) separated. For the sake of simplicity, only the approximation points that were selected for constructing the normal constraints are shown in these plots

frontier is discontinuous. By being separated from its own constraint lines, the point shows that certain tighter constraints could have been placed on it that would have yielded the same result. Using this information, a region of the design space can be restricted for the remainder of the optimization process. That region contains all points between the normal constraint that was used to find the point and the parallel normal constraint that could be constructed to pass through the new point. Points in the restricted region will satisfy (19) and (20):

$$N_k(\mu(\mathbf{x}) - S_i)^T \leq 0 \qquad (1 \leq k \leq n-1) \tag{19}$$

$$N_k(\mu(\mathbf{x}) - P_i)^T \geq 0 \qquad (1 \leq k \leq n-1) \tag{20}$$

Because the Pareto frontier approximation vertices generated in Step 1 span the entire feasible design space, it is possible for some design spaces that a particular SOO will be restricted such that no feasible solution is obtainable. Where this occurs, a restriction may be constructed wherein a point need only satisfy (19).

5 Numerical examples

In this section, we consider three well-known example problems from the literature to compare the effectiveness and efficiency of generating smart Pareto sets using the SNC method versus using the NC method with a smart Pareto filter. For each of these examples, the SNC method was applied using parameters that would result in a sufficiently low number ($n_\mathrm{p} \leq 20$) of smart Pareto points being generated for the designer to consider. Then, the shortest distance between any two smart Pareto points in a direction parallel to the utopia plane was calculated (see $\delta_{\min}$ in Fig. 5c). The NC was applied using this value for δ_j in the construction of the utopia plane (see (8)). This ensured that the two methods had the same maximum resolution of Pareto points on the frontier. Then, using the smart Pareto filter, a smart Pareto set was extracted with nearly identical solutions to the ones produced by the SNC method. With nearly identical final products, the methods are easily compared for efficiency.

For these examples, the NC method was used with the improvements of Messac and Mattson (2004) and Haddock et al. (2008) mentioned at the end of Section 2.3, which are built into the SNC method. Thus, advantages that the SNC method exhibits can be attributed to the

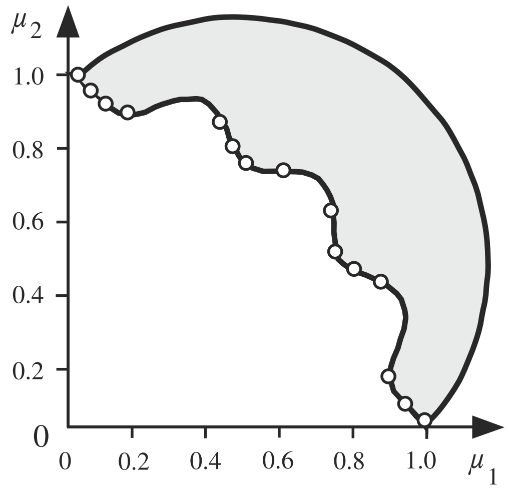


Fig. 8 The smart Pareto set generated by the SNC method for the problem TNK (Tanaka et al. 1995)

Table 1 A comparison of the efficiency of the SNC method vs. the NC* method for creating a smart Pareto set

Problem	Obj	Var	Con	# Smart Pareto points	Method	# SOOs	Func. calls per Pareto point	Total func. calls
TNK	2	2	2	15	NC*	42	73	3066
					SNC	20	66	1320
Gear box	3	7	11	10	NC*	176	100	17600
					SNC	85	73	6205
WATER	5	3	7	20	NC*	2500	164	410000
					SNC	26	130	3380

*Denotes that the NC method has been implemented with the improvements of Messac and Mattson (2004) and Haddock et al. (2008) from Section 2.3 and used in conjunction with a smart Pareto filter

aspects of it that are unique from existing NC method variations.

The three chosen problems are TNK (Tanaka et al. 1995), a gear box design (Huang et al. 2006), and WATER (Ray et al. 2001). TNK is notable for having concave regions in both the horizontal and vertical directions, which causes difficulties for many Pareto set generation algorithms. Figure 8 shows the smart Pareto set generated by the SNC method sumperimposed on an outline of the feasible design objective space. The gear box design problem has been used a number of times to demonstrate the robustness of Pareto set generation algorithms in handling dependent objective functions and multiple nonlinear constraints (Motta et al. 2012; Sanchis et al. 2008). The problem WATER was chosen to demonstrate the functionality of the SNC method in problems with a large number of objectives. Table 1 presents the results of all three example problems.

The extent to which the SNC method and NC method differ in efficiency naturally varies by problem. Nevertheless, the trends shown in Table 1 are typical. First, the SNC method in nearly all cases will require fewer function calls per Pareto point generated. This is because the iteratively updated approximation of the Pareto frontier provides the algorithm with a generally more accurate initial value for each SOO. Second, the number of SOOs required to produce the same smart Pareto set is nearly always less for the SNC method than for the NC method, as the NC method must equally space all of its utopia plane points based upon the closest two smart Pareto points that it is designed to be capable of finding. The SNC method, on the other hand, can adjust to search more closely in regions of high curvature where smart Pareto points may be found close together. As the number of objectives or maximum curvature of the Pareto frontier increase, the advantages of using the SNC method over the NC method increase as well.

The SNC method has been introduced in this paper for implementation in sequence. The SNC method can also be implemented in parallel by simultaneously performing SOOs for multiple selected approximation points. However, this may decrease the ability of the algorithm to identify the most likely regions where new smart Pareto points will be discovered, as the Pareto frontier approximation will be updated less frequently.

6 Conclusion

This paper presented a novel method for directly generating a smart Pareto set of solutions for an MOP. This method avoids the inefficiencies of existing approaches for generating minimal Pareto sets, which generate a significant number of solutions that will not be part of the set being presented to the designer. The development of a scalar value for smart distance which reflects the amount of significant tradeoff between points enables this method. We showed how iteratively updating an approximation of the Pareto frontier allows for searches for smart Pareto solutions to be made in those regions of the design space that are calculated to be most likely to yield them. Because of its ability to more accurately select initial values for SOOs and dynamically select the location for normal constraints in each SOO, the SNC method results in significantly fewer function calls than the predominant existing method for generating smart Pareto sets in nearly all cases. The proposed method was tested on three challenging numerical problems from the literature and demonstrated its expected effectiveness and efficiency.

Acknowledgments We would like to recognize the National Science Foundation (Grant CMMI-0954580) for funding this research.

Appendix: Flowchart comparison of NC* and SNC methods

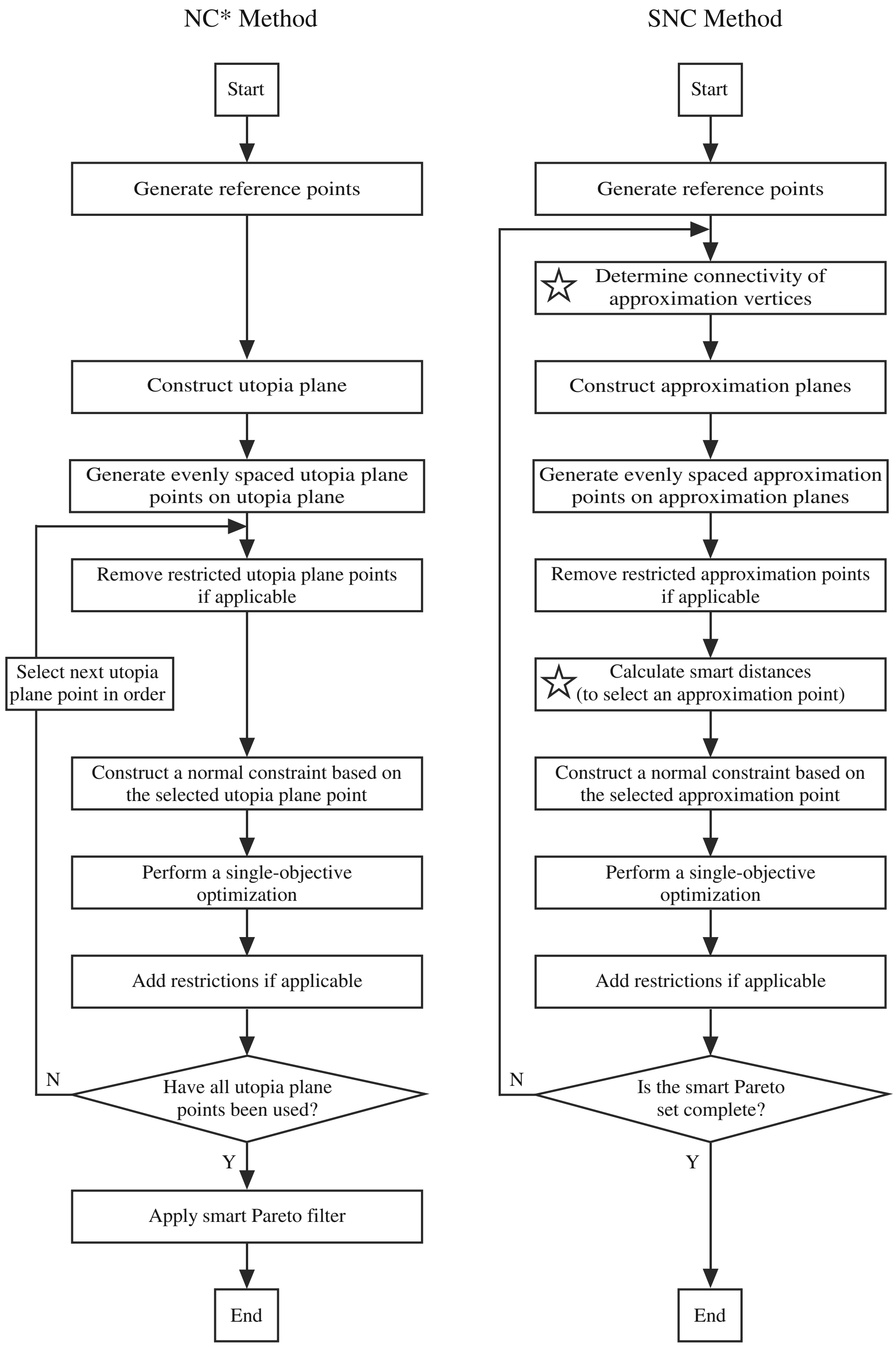

NC* Method
Start
Generate reference points
Construct utopia plane
Generate evenly spaced utopia plane points on utopia plane
Remove restricted utopia plane points if applicable
Select next utopia plane point in order
Construct a normal constraint based on the selected utopia plane point
Perform a single-objective optimization
Add restrictions if applicable
Have all utopia plane points been used?
N
Y
Apply smart Pareto filter
End
SNC Method
Start
Generate reference points
Determine connectivity of approximation vertices
Construct approximation planes
Generate evenly spaced approximation points on approximation planes
Remove restricted approximation points if applicable
Calculate smart distances (to select an approximation point)
Construct a normal constraint based on the selected approximation point
Perform a single-objective optimization
Add restrictions if applicable
Is the smart Pareto set complete?
N
Y
End

Fig. 9 This chart gives the flow of the NC* and SNC methods, aligning corresponding steps horizontally. As shown here, the primary differences are (1) the greater number of steps included in the SNC method within each iteration (as a result of the Pareto frontier approximation being updated), (2) the introduction of two new steps in the SNC method (identified by stars in this figure), and (3) the application of the smart Pareto filter at the end of the NC* method, as opposed to the direct generation of a smart Pareto set by the SNC method. As in Table 1, the asterisk denotes that the NC* method has been implemented with the improvements of Messac and Mattson (2004) and Haddock et al. (2008) from Section 2.3 and used in conjunction with a smart Pareto filter, so as to draw attention to the novel aspects of the SNC method

References

Aittokoski T, Ayramo S, Miettinen K (2009) Clustering aided approach for decision making in computationally expensive multiobjective optimization. Optim Method Softw 24:157–174

Barber CB, Dobkin DP, Huhdanpaa HT (1996) The quickhull algorithm for convex hulls. ACM Trans Math Softw 22:469–483

Bechikh S, Said LB, Ghedira K (2010) Searching for knee regions in multi-objective optimization using mobile reference points. In: Proceedings of the 2010 ACM symposium on applied computing

Boyce NO, Mattson CA (2008) Reducing computational time of the normal constraint method by eliminating redundant optimization runs. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference

Deb K, Tiwari S (2006) Reference point based multi-objective optimization using evolutionary algorithms. Int J Comput Intell Res 2:273–286

Haddock ND, Mattson CA, Knight DC (2008) Exploring direct generation of smart Pareto sets. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference

Handl J, Knowles J (2007) An evolutionary approach to multiobjective clustering. IEEE Trans Evol Comput 11:56–76

Huang HZ, Gu YK, Du X (2006) An interactive fuzzy multi-objective optimization method for engineering design. Eng Appl Artif Intell 19:451–460

Ismail-Yahaya A, Messac A (2002) Effective generation of the Pareto frontier using the normal constraint method. In: 40th Aerospace Sciences Meeting and Exhibit

Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26:369–395

Martinez M, Sanchis J, Blasco X (2007) Global and well-distributed Pareto frontier by modified normalized normal constraint methods for bicriterion problems. Struct Multidiscip Optim 34: 197–209

Martinez M, Garcia-Nieto S, Sanchis J, Blasco X (2009) Genetic algorithms optimization for normalized normal constraint method under Pareto construction. Adv Eng Softw 40:260–267

Mattson CA, Mullur AA, Messac A (2004) Smart Pareto filter: Obtaining a minimal representation of multiobjective design space. Eng Optim 36:721–740

Messac A, Mattson CA (2004) Normal constraint method with guarantee of even representation of complete Pareto frontier. AIAA J 42:2101–2111

Messac A, Ismail-Yahaya A, Mattson CA (2003) The normalized normal constraint method for generating the Pareto frontier. Struct Multidiscip Optim 25:86–98

Motta RS, Afonso SMB, Lyra PRM (2012) A modified NBI and NC method for the solution of n-multiobjective optimization problems. Struct Multidiscip Optim 46:239–259

Pareto V (1964) Cour deconomie politique. Librarie Droz-Geneve (the first edition in 1896)

Rachmawati L, Srinivasan D (2009) Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front. IEEE Trans Evol Comput 13:810–824

Ray T, Tai K, Seow C (2001) An evolutionary algorithm for multiobjective optimization. Eng Optim 33:399–424

Ruzika S, Wiecek MM (2005) Approximation methods in multiobjective programming. J Optim Theory Appl 126(3):473–501

Rynne B (2007) Linear functional analysis. Springer, New York

Sanchis J, Martinez M, Blasco X, Salcedo JV (2008) A new perspective on multiobjective optimization by enhanced normalized normal constraint method. Struct Multidiscip Optim 36:537–546

Schutze O, Laumanns M (2008) Approximating the knee of an MOP with stochastic search algorithms. Springer-Verlag, New York

Tanaka M, Watanabe H, Furukawa Y, Tanino T (1995) GA-based decision support system for multicriteria optimization. In: Proceedings IEEE international conference systems

Zitzler E, Thiele L (1998) Multiobjective optimization using evolutionary algorithms–a comparitive case study. In: Parallel Problem Solving From Nature