Reverse Normal-Boundary Intersection for Multi-Objective Optimization for Power Plant Operation

Joel H. Van Sickel*, Kwang Y. Lee**

*Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802, USA
(e-mail: jhv107@psu.edu).
**Department of Electrical and Computer Engineering, Baylor University, Waco, TX 76798, USA
(e-mail: Kwang_Y_Lee@baylor.edu)

Abstract: This paper examines the Pareto front arising in the multi-objective optimization problem for power plant operation. However, having the Pareto front alone only provides more information about the operation of the power plant. To utilize this information, first, a point from the Pareto front must be chosen, which for this application turns out to be a very simple choice. Then a method is required to quantitatively distinguish this point from the other points on the Pareto front so that a computer algorithm can consistently choose this point when optimizing power plant operation. This is accomplished by a single optimization problem inspired by the normal-boundary intersection approach for generating Pareto fronts. This in turn, creates a way to choose the correct point over and over again, and in addition, removes the need to use aggregation methods to quantify the different objectives. This creates a much more intuitive approach to multi-objective optimization which will be referred to as reverse normal-boundary intersection.

Keywords: Multi-objective optimization, Pareto optimization, power plant operation, reference governor, reverse normal boundary intersection, differential evolution.

1. INTRODUCTION

Multi objective optimization has long been an area of interest in many fields such as engineering and business. There are constantly trade offs that have to be made between different objectives because so often there is no ideal solution. In many cases, the decision between objectives only needs to be made once or perhaps a few times, but other times, the decision must be made repeatedly, such as in power plant operation where the power demand can constantly be changing, making it a challenging area to address multi-objective optimization. How multi-objective optimization is actually done also differs widely from area to area, with numerous approaches available. Generally, these approaches can be split into two broad categories. The first category aggregates the objectives quantitatively into a single objective, the second approach uses the concept of Pareto optimality to find a trade-off curve. Aggregation of multiple objectives can be achieved through different ranking schemes such as weighted summation (Rao, 1996) or goal programming (Jones and Tamiz, 2002) to create a single function to be optimized. This assumes a quantitative knowledge of the relative importance of the different objectives. It is useful, however, because it returns a single solution. Finding the trade off curve, in which numerous points are returned, provides a comprehensive view of the optimization problem, but the large number of points are hard to visualize when more than three objectives are used. Also, most applications need a specific solution, not a set of points, to function properly. This creates a need to know how to choose a point on the trade-off curve.

Currently, multi-objective optimization work done for developing reference governors (Heo, et al., 2006; Garduno-Ramirez and Lee, 2001) has used aggregation to create a single objective function from multiple objectives for the control of power plants. Unfortunately, there is no explicit way to weight the different objectives, and without looking at the Pareto front, it is unknown what trade offs are being made. The Pareto front of a power plant was analyzed in (Van Sickel, et al., 2008). While the approaches showed accurate ways to generate the Pareto front of a power plant, it was not enough of a step to provide something that could be implemented in actual operation. A Pareto front by itself mainly serves as a method to allow humans to better visualize a multi-objective optimization problem. How a specific point is chosen is often subjective if not even arbitrary at times depending on the situation. For this application, choosing the point turned out to be quite simple based on the chosen objectives. However, once this point has been chosen, what is the appropriate method for finding it again later? A new optimization could be run that finds the point closest to the original point, but for this application that is not sufficient as the Pareto front moves a lot as the unit load demand (uld) changes, completely changing where the ideal optimization point would be located. To address this issue in a manner that allows a point to be chosen only once and not for every possible uld value, a modified version of normal-boundary intersection is used.
2. MULTI-OBJECTIVE OPTIMIZATION AND PARETO OPTIMALITY

Multi-objective optimization has always been important as it is not very often that there is an ideal solution that provides the best result for every aspect of a problem. As such, great effort has gone into trying to develop methods to provide better ways to make decisions where trade-offs are inevitable. When the problem can be quantified, equations can be used to give numerical values to different objectives representing how well they have been achieved. Initially, standard single objective optimization techniques were used, but they required that different objectives be aggregated. This was only a simplification. To truly deal with multi-objective solutions in their entirety, the concept of Pareto optimality was put forth.

Pareto optimality deals with multi-objective optimization where it is desired to minimize the $N$ objectives $f_i(x)$, where $i = 1, 2, \ldots, N$, $x$ is a vector of length $k$, and $k$ is the number of variables. This optimization can be posed as:

$$\text{minimize } F(x) = [f_1(x), f_2(x), \ldots, f_N(x)]$$

subject to constraints:

$$g_i(x) \leq 0$$

$$h_i(x) = 0$$

Terminology: Vector $F(a)$ is said to dominate vector $F(b)$, $F(a) \prec F(b)$, if and only if $f_i(a) \leq f_i(b)$ for all $i$ and $f_i(a) < f_i(b)$ for at least one $i$. Vector $F(a)$ is considered Pareto optimal or non-dominated if there exists no other vector $F(b)$ such that $F(b) \prec F(a)$.

There can be numerous points in a set that are all Pareto optimal. The collection of all of the Pareto optimal points is called the Pareto front. An in-depth discussion of Pareto optimality is provided by Kim and de Weck (2006).

3. POWER PLANT

The power unit used for in this paper is a 160 MW oil-fired boiler-turbine unit as modeled by Bell & Astrom (1987). It is a three-input three-output system with three differential equations (1). The three inputs $u_1$, $u_2$, and $u_3$ are all control valves that effect mass flow rates within the system. Input $u_1$ controls the mass flow rate, $u_2$ the steam to the turbine, and $u_3$ the feedwater to the drum. All three controls are normalized where for $u_1$, 0 - 1 corresponds to 0 - 14 kg/s, for $u_2$, 0 - 1 corresponds between closed and open, and for $u_3$, 0 - 1 corresponds to 0 - 140 kg/s. The three states $E$, $P$, and $\rho_f$ are electric power in MW, drum steam pressure in kg/cm$^2$, and fluid density in kg/m$^3$, respectively.

3.1 Mathematical Model

$$\frac{dP}{dt} = 0.9u_1 - 0.0018u_2 P^{0.8} - 0.15u_3$$ (1a)

$$\frac{dE}{dt} = ((0.73u_2 - 0.016)P^{0.8} - E)/10$$ (1b)

For the purposes of this paper, the steady-state equations are all that are important for deriving the Pareto front:

$$E_{ss} = \frac{(0.73u_1 - 0.16)(0.9u_2 - 0.15u_3)}{0.0018u_2}$$ (2a)

$$P_{ss1} = \frac{141u_1}{1.1u_2 - 0.19}$$ (2b)

$$P_{ss2} = \left(\frac{0.9u_2 - 0.15u_3}{0.0018u_2}\right)^{8/9}$$ (2c)

Solving the steady-state equations for $u_1$, $u_2$, and $u_3$ based on pressure limits yields operating windows for a given unit load demand ($uld$). This has previously been done in (Heo, et al., 2006) but the windows are provided in Fig. 1.

4. REFERENCE GOVERNOR

A reference governor provides feedforward control actions and set points to a feedback control system as shown in Fig. 2. Typically, a heuristic optimization is used in conjunction with a steady-state neural network model to search for ideal feed forward controls. Currently, multiple objectives are aggregated by optimizing a cost function that uses either weighted sums system (Heo, et al., 2006) or goal programming (Garduno-Ramirez and Lee, 2001). Unfortunately, creating this cost function is challenging and does not provide an idea of what trade-offs are being made between different set points. Generating a Pareto front of the
power plant’s steady state operation will allow a more informed decision to be made, but the reference governor needs an algorithm that only results in a single point, so more is required after generating the Pareto front. For this application, the operation of the power plant can change as it is a physical system subject to fatigue, friction, and numerous other conditions that slowly change its operating points. Additionally, the operating range is based on the given uld, so there is an infinite number of pareto fronts that can be generated, as uld is a continuous signal. This means that any usable method must not require a specific point to be chosen for each uld.

\[
\begin{align*}
x & = [uld, u_1, u_2, u_3] \\
f_1(x) &= uld - P \\
f_2(x) &= u_1 \\
f_3(x) &= -u_2 \\
\text{minimize } F(x) &= [f_1(x), f_2(x), f_3(x)]
\end{align*}
\]

This optimization minimizes error (between uld and power output) and fuel used while maximizing the steam valve opening which minimizes losses. The Results of this optimization, which were obtained in (Van Sickel, et al., 2008), are shown in Figs. 3 and 4 for different values of uld.

Having generated the Pareto front of a power plant, the next step is to choose which point is desired. For this power plant described in (1), the Pareto front is shown in Figs. 3 and 4. For a given error and uld each point from Fig. 3 matches up with the same point in Fig. 4. As can be seen in Fig. 3, the desired location of \( u_1 \) changes dramatically with uld. For this particular power plant choosing points is easy. Except for extremely low power demands at which this power plant probably would not be running, we can always have \( u_2 \) set to 1 (fully open). As for a tradeoff, the power plant will not purposely provide less power than needed to conserve fuel, so error will chosen to be 0 at all times, and then whatever \( u_1 \) corresponds to this specific Pareto point will be chosen. This decision would not be so simple if maximizing \( u_2 \) was possible under almost all operating conditions. However, even with this knowledge, we still must solve for this point for every uld as \( u_1 \) is constantly different. Using reverse NBI, it will be shown how this is not necessary.

Normal-boundary intersection (NBI) was developed by Das and Dennis (1996) for solving multi-criteria optimization problems. It is geometrically inspired and has been very successful in practice. This approach requires the knowledge of the solution to the optimization of the individual \( f_i(x) \), which is usually available. In this paper, NBI has been combined with differential evolution (DE) to solve the optimization problem.

Fundamentally, NBI is based on the concept of the Convex Hull of Individual Minima which is defined after a few preliminaries. For an \( n \)-objective minimization problem, where each objective is represented by \( f_i(x) \), \( i = 1, \ldots, n \), let \( F^* = [f_1^*, f_2^* \ldots f_n^*] \), where \( f_i^* \) is the solution to the
minimization of $f_i(x)$, and $x_i^*$ is the corresponding vector that minimizes $f_i(x)$. $\Phi$ is the matrix whose $i$-th column is $F(x_i^*) - F^*$. The Convex Hull of Individual Minima (CHIM) is then defined as the set of points that are convex combinations of $f_i$. It may help to see it expressed in terms of (6).

$$\Phi \omega : \omega \in \mathbb{R}^n, \sum_{i=1}^n \omega_i = 1, \omega_i \geq 0$$

NBI then proceeds by solving the following maximization problem (7) by projecting lines from the CHIM toward $F^*$ while varying $\omega$ to search across the CHIM.

$$\max_{s.t.} t$$

$$\Phi \omega + \hat{n} = F(x) - F^*$$

$$h(x) = 0$$

$$g(x) \leq 0$$

Where $\hat{n}$ is the unit normal vector to the CHIM and $t$ represents how far $\hat{n}$ can project from $\Phi \omega$ before crossing the Pareto front. Typically, all functions are shifted so that $F^*$ lies at the origin. A graphical example of NBI is provided in Fig. 5. There are a few issues unique to NBI that are important to be aware of. The above optimization only searches over the CHIM simplex and could possibly leave out some extreme optimal points. In addition, a point solved for $\omega$ is not restricted to have its elements sum up to 1. Regardless, NBI has proven to be a useful method of generating Pareto points, and these points of possible concern are not an issue in regard to reverse NBI as will be explained later.

First, a point $P_o$ from the Pareto front must be chosen to represent the desired point. This also means that the Pareto front needs to be generated. This can be seen as the first step in reverse NBI as shown in Fig. 6. Everything following seeks to describe this point in terms of its location in regard to the CHIM and is summarized in Fig. 7. Basically, $\omega_o$ needs to be solved for as it is independent of the specific values taken on by the Pareto front. To do this, it is required to find where $\hat{n}$ would intersect with the CHIM when projected from $P_o$. This is done through a Gram-Schmidt orthonormalization (Bay, 1999) and then projecting $P_o$ onto the CHIM simplex. To start with, the $n$ points $f_i^*$ must be found as in NBI and define the $n$-1 dimensional affine subspace on which the CHIM is located. This subspace is shifted to a linear subspace by subtracting $f_i^*$ from the other $n$-1 points, putting $f_i^*$ at the origin and leaving $n$-1 vectors $y_i, i = 1, \ldots, n-1$, with $n$ elements. These vectors can be turned into an orthonormal basis using Gram-Schmidt orthonormalization given in (8-9).

$$v_i = y_i - \sum_{j \neq i} \{v_i, y_j\} v_j$$

$$\tilde{v}_i = \frac{v_i}{|v_i|}$$

Once there is an orthonormal basis, it is simple to find the intersection point $P_e$. First $P_e$ should also be shifted by subtracting $f_i^*$ which yields a new point $P_{e,0}$ which can be used with the orthonormal basis. Using (10) solves for point $P_{e,1}$, which is easily transformed by to $P_o$ by adding $f_i^*$.

$$P_{e,1} = P_{e,0} - \sum_{i=1}^{n-1} \{P_{e,0}, \tilde{v}_i\}$$

Once $P_e$ is known, it can be used to solve for $\omega_o$, which is then used to represent the desired Pareto optimal point even if the Pareto front is shifted. This can be used in all subsequent NBI subproblems to directly find the equivalent point $P_o$, even if the objective space has been shifted, as in this application when the unit load demand changes.

**Fig. 5. Graphical example of NBI.**

**7. REVERSE NORMAL-BOUNDARY INTERSECTION**

While NBI started with a point on the CHIM and then projected $\hat{n}$ toward $F^*$, reverse NBI does the exact opposite and starts with a point that is known to be Pareto optimal, and projects $\hat{n}$ toward the CHIM and uses the intersection to solve for a vector $\omega_o$ that can be used to represent the Pareto optimal point in terms of an NBI subproblem. For this application, this represents the operating points in a manner that is independent to the value of $uld$ and allows for only a single point to be chosen. Then, to recalculate this point under different operating conditions, $\omega_o$ is used to solve a single NBI subproblem.
Now that $\omega_o$ is known, the NBI maximization problem in (6) can be solved substituting $\omega_o$ for $\omega$. Choosing $\omega_o$ in this manner eliminates the previous issues that were mentioned as possible areas of concern when using NBI to generate the Pareto front. As can be seen in (7), the NBI maximization problem has nothing to do with whether the optimal point is dominated or not. It is only solving for the point that lies in $F$ which can be projected by $\hat{n}$ the furthest from the CHIM in the direction of the origin. For a non-convex surface, it is possible to have solutions to this maximization problem that are definitely not Pareto-optimal. Reverse NBI does not have this constraint because the point has already been verified to be Pareto-optimal when it was chosen originally. Additionally, NBI cannot necessarily find all points on the Pareto front because it limits itself to searching the CHIM simplex. However, reverse NBI does not require the elements of $\omega_o$ to sum to 1 and can be used to represent points not constrained to the CHIM simplex.

It is important to remember that this method is only useful for representing Pareto-optimal points when the shape of the Pareto front does not vary much relative to the CHIM. It is robust against scaling and shifting, but if the Pareto front’s shape drastically changes, there is no solution other than to completely re-analyze the front. As can be seen in Figs. 3 and 4, the shape of the Pareto front is basically unchanged, and is only shifted based on changing unit load demand, which is why this technique will prove useful for this application.

8. IMPLEMENTING A REFERENCE GOVERNOR FOR A POWER PLANT WITH REVERSE NORMAL-BOUNDARY INTERSECTION

Implementing a reference governor with reverse NBI is a fairly straightforward process. With the multi-objective problem stated, the Pareto front found, we must first choose $P_o$. As stated before, we want to choose a point with an error of 0 and take whatever minimum value of $u_1$ can be obtained, and we know $u_2$ will be maximized to 1 regardless of what point we choose. We need to pick a specific value for $uld$ and pick a specific $P_o$ so that $\omega_o$ can be solved for. The unit load demand does not really matter as the results are going to be almost identical regardless of its value. All results are shown having chosen $P_o$ when $uld$ was set to 120 MW.

The specific values for $f_i^*$ should be available or very easily obtained for any application if a Pareto front has been generated, as they only represent single value optimization problems, and can easily be determined for this application by looking at Figs. 3 and 4. Error was constrained to have a magnitude no larger than 10 MW when calculating these values.

From this point, reverse NBI can be followed exactly to yield $\omega_o$ and this is then used to solve the NBI subproblem continuously as $uld$ varies. This subproblem is solved exactly the same as in (Van Sickel, et. al., 2008) using differential evolution and a static neural network modeling the power plant’s steady state operation.

Following all of the previous steps will provide the values for $uld$, $u_1$, $u_2$, and $u_3$. However, to operate correctly, the reference governor must also provide the set points for $E$ and $P$. $E$ is just $uld$ and $P$ is trivial to find since all other variables are known.

9. RESULTS

Shown below in Table 1 is the point $P_o$ chosen for $uld = 120$ MW and the resulting vector $\omega_o$. Fig. 8 shows the corresponding output of the reference governor for feedforward controls and Fig. 9 shows the setpoint values along with the performance of the power plant in simulation. This specific reference governor can calculate feed forward controls and set points for a given $uld$ in 0.44 seconds when implemented in Matlab on a Dell precision with a Xeon duel core 3.6 GHz processor with 3.25 GB ram. Note that when $uld$ is constant, the previous results are used and not recalculated every time.

| TABLE 1. Optimal Point and Simplex Vector used in Reverse NBI with Power Plant |
|---------------------------------|-----------------|
| $P_o$                           | 0.0, 0.546, -1.0 |
| $\omega_o$                      | 0.0142, 0.0296, 0.9563 |

Fig. 6. First Step in Reverse NBI, finding the Pareto front and choosing $P_o$.

Fig. 7. Second Step in Reverse NBI, solving for $\omega_o$. 

Implementing a reference governor with reverse NBI is a fairly straightforward process. With the multi-objective problem stated, the Pareto front found, we must first choose $P_o$. As stated before, we want to choose a point with an error of 0 and take whatever minimum value of $u_1$ can be obtained, and we know $u_2$ will be maximized to 1 regardless of what point we choose. We need to pick a specific value for $uld$ and pick a specific $P_o$ so that $\omega_o$ can be solved for. The unit load demand does not really matter as the results are going to be almost identical regardless of its value. All results are shown having chosen $P_o$ when $uld$ was set to 120 MW.

The specific values for $f_i^*$ should be available or very easily obtained for any application if a Pareto front has been generated, as they only represent single value optimization problems, and can easily be determined for this application by looking at Figs. 3 and 4. Error was constrained to have a magnitude no larger than 10 MW when calculating these values.

From this point, reverse NBI can be followed exactly to yield $\omega_o$ and this is then used to solve the NBI subproblem continuously as $uld$ varies. This subproblem is solved exactly the same as in (Van Sickel, et. al., 2008) using differential evolution and a static neural network modeling the power plant’s steady state operation.

Following all of the previous steps will provide the values for $uld$, $u_1$, $u_2$, and $u_3$. However, to operate correctly, the reference governor must also provide the set points for $E$ and $P$. $E$ is just $uld$ and $P$ is trivial to find since all other variables are known.

9. RESULTS

Shown below in Table 1 is the point $P_o$ chosen for $uld = 120$ MW and the resulting vector $\omega_o$. Fig. 8 shows the corresponding output of the reference governor for feedforward controls and Fig. 9 shows the setpoint values along with the performance of the power plant in simulation. This specific reference governor can calculate feed forward controls and set points for a given $uld$ in 0.44 seconds when implemented in Matlab on a Dell precision with a Xeon duel core 3.6 GHz processor with 3.25 GB ram. Note that when $uld$ is constant, the previous results are used and not recalculated every time.

| TABLE 1. Optimal Point and Simplex Vector used in Reverse NBI with Power Plant |
|---------------------------------|-----------------|
| $P_o$                           | 0.0, 0.546, -1.0 |
| $\omega_o$                      | 0.0142, 0.0296, 0.9563 |
10. CONCLUSIONS

As has been shown, reverse NBI created a way of solving the multi-objective optimization problem in a way that allows a single point to be found without resorting to aggregation practices typically used to convert multi-objective problems into single-objective problems. While this process directly eliminates the creation of an aggregation function, it requires more work in every other area of solving the problem.

First, this method requires generation of the Pareto front, or at least knowledge of the specific point desired, while an aggregation function can technically be realized without any knowledge of the Pareto front. However, it is obviously bad practice to arbitrarily make cost functions for an application and it is hoped that any serious solution to a multi-objective optimization problem will have at some point looked at the Pareto front, so requiring this knowledge is far from unreasonable.

Second, solving the NBI maximization subproblem is typically much harder than solving a typical aggregation function. It is foreseeable that for a given application, it might require so much computing power to solve the NBI subproblem that speed requirements would force the use of aggregation instead of reverse NBI.

The last big issue with using reverse NBI is the fact that knowledge of each $f_i^*$ is required. While this information is easily obtained in comparison to getting the Pareto front, it is information that needs to be verified every time the NBI subproblem is solved. The main point of this process is that it adapts to the Pareto front being shifted or scaled, which means that the values of $f_i^*$ will be changing. For a small problem like this, with only three objectives, it is not a computationally significant problem to continually find the new values. However, for problems with a large number of objectives, calculating the new values of each $f_i^*$ will effect the speed of this algorithm.

It will be stated again that this approach only works when the Pareto front maintains its general shape. If the Pareto front will be changing dramatically, there is no simple method that will consistently provide desired results.

If these constraints are not a problem, then reverse NBI becomes a very elegant approach that allows precision and knowledge to accurately solve a multi-objective problem in an unambiguous manner, without the hours of tweaking cost functions until one is found that happens to work.

REFERENCES