Multi-objective Optimization using Pareto Optimality

Dhruv Shah, Vatsal Shah, Pranav Patel

November 19, 2023

Abstract

In real-world applications, optimizing multiple objective functions simultaneously, known as multi-objective or multi-criteria optimization problems, is a common challenge. This report explores the application of Pareto Optimality, a traditional technique for addressing such complex optimization problems. Pareto Optimality provides a structured approach to navigating trade-offs among conflicting objectives, identifying non-dominated solutions on the Pareto front. Through a case study involving a regional airline's marketing strategy, this report demonstrates the efficacy of Pareto Optimality in achieving a balance between conflicting goals. The findings contribute valuable insights for strategic decision-making in diverse real-world scenarios.

1 Introduction

Multi-objective optimization problems require optimization of multiple conflicting objectives, i.e. an improvement in one objective leads to a deterioration in the other. Such problems may not have a single unique optimal solution, but a set of Optimal solutions known as the **Pareto Front**. This report navigates through essential concepts, methodologies, and practical applications related to Pareto Optimality.

Beginning with a General Mathematical Model in Section 2, we have further defined the concept of Criterion Space and Design Space, later illustrating their construction through a detailed example in Section 2.2.1. Section 2.3 sheds light on the concept of Pareto Solution. Further sections define the methods of Pareto front generation and the algorithm for the same.

To ground the theoretical discussions in real-world relevance, Section 3.3 demonstrates the application of the Pareto Front to a tangible example– a case study of an airline's marketing strategy. Through this exploration, we aim to simplify the concept and showcase its relevance in solving complex real-world challenges.

2 Mathematical Model

2.1 Problem Formulation

We define a multi-objective optimization problem as follows:
Minimize

$$f(x) = (f_1(x), f_2(x), f_3(x), ..., f_k(x))$$

where the constraints set can have the form $\Omega = \{h(x) = 0, g(x) \leq 0\}$. where $h(x) : \mathbb{R}^n \to \mathbb{R}^m$ and $g(x) : \mathbb{R}^n \to \mathbb{R}^p$
In general, we may have three different types of multi-objective optimization problems:

- Minimize all the objective functions.
- Maximize all the objective functions
- Maximize some and minimize others

However, any of these can be converted into equivalent minimization problems.

2.2 Criterion Space and Design Space

Design Space is a representation of an optimization problem where the constraints and the objective functions are plotted as functions of the variables of the problem.
But, multi-objective optimization problems are usually represented in the Criterion Space where the different axes represent the objective functions of the problem. The objective function curve in the design space can be converted into a curve in the criterion space by evaluating the values of the objective function at different points on the constraint curve in the design space. Therefore, the feasible points in the design space map onto a set of points in the criterion space known as the feasible criterion space. Formally, a feasible **Criterion Space** is defined as $Z = \{f(x)|x \in \Omega\}$.

Having defined the basic terminologies in a multi-objective optimization problem, we will now move forward to define the solution concepts involved in such problems. For these concepts, we refer to Chapter 17 of the book by Jasbir S. Arora. [1]

2.2.1 An example showing the construction of Criterion Space and Design Space

Consider the following objective function:
Minimize

$$f_1 = (x_1 - 2)^2 + (x_2 - 5)^2$$

$$f_2 = (x_1 - 4.5)^2 + (x_2 - 8.5)^2$$

subject to

$$g_1 = -x_1 - x_2 + 10 \leq 0$$

$$g_2 = -2x_1 + 3x_2 - 10 \leq 0$$

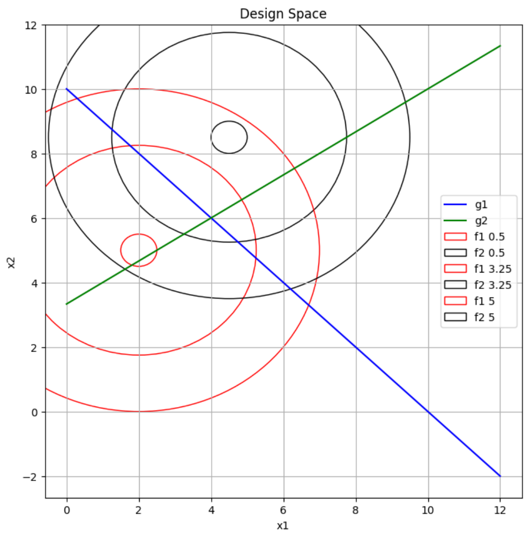

Figure 1: Design Space Construction of the above problem

We will now attempt to draw the Design Space and the Criterion Space of the above problem using numerical techniques.

In Fig. 1, the red and black circles represent the function contours of the above-stated problem. The blue and the green lines represent the constraints of the problem.
As discussed above, we will now convert the above design space into a criterion space.

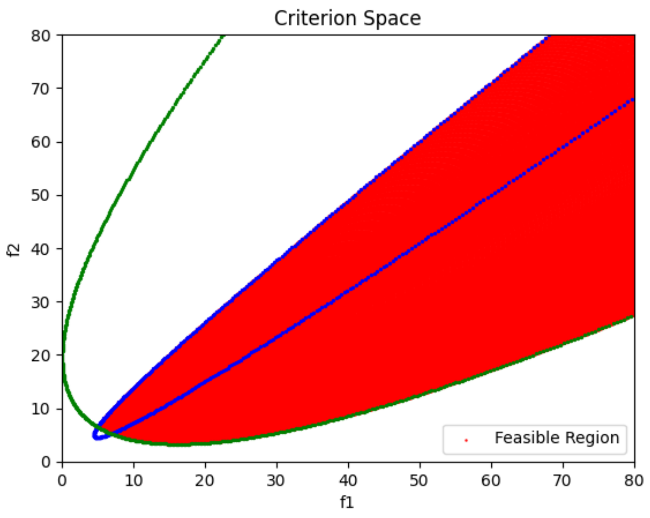

Figure 2: Criterion Space Construction using the obtained Design Space

Fig. 2 shows the criterion space; the blue and green lines are the objective function values obtained at different points on the constraint functions g_1 and g_2. The shaded red region represents the feasible criterion space, obtained by evaluating the objective function values at the points in the feasible set of the Design Space.

2.3 Pareto Solution

In a multi-objective optimization problem, a solution is optimal if there exists no other solution within the feasible set, that gives improved performance with regard to all the objectives. Such solutions are known as Pareto minimizers. We define the Pareto minimizer points mathematically as below.

A point $x^* \in \Omega$ is called a **Pareto Minimizer** if there does not exist another point x in the set Ω such that $f(x) \leq f(x^*)$ with at least one $f_i(x) < f_i(x^*)$.
In simple words, the point x^* is known as a Pareto minimizer, if there exists no other point x that would reduce at least one of the objective functions without increasing another one.

A vector of objective functions f in the feasible criterion space is **Non-Dominated** if there does not exist another vector f in the set, such that $f \leq f^*$, with at least one $f_i < f_i^*$. Else, f^* is dominated.

We will try to understand this concept using an example. Let there be three points in the Design Space, whose co-ordinates are given by $x_1 = [5,6], x_2 = [6,8]$ and $x_3 = [1,4]$ and their corresponding points in the criterion space are given by $f(x_1) = [30,45]$, $f(x_2) = [41,75]$ and $f(x_3) = [13,45]$. Clearly, since $f(x_1) < f(x_2)$, we can say that $f(x_2)$ is dominated by $f(x_1)$. But there also exists $f(x_3)$ where one function value is decreased without simultaneously increasing the other. Therefore only $f(x_3)$ is a non-dominated point in the above example.

We will now move forward to explain a basic algorithm to compute the set of Pareto optimizers and the set of Non-Dominated points.

3 Pareto Front generation Method

3.1 Computing the Pareto Front

To compute the Pareto Front, two solutions are compared, and the dominant solution is eliminated from the set of candidates of Pareto optimizers. Let,

$$x^{*r} = [x_1^{*r}, x_2^{*r}, \ldots, x_n^{*r}]^T$$

be the r^{th} candidate Pareto optimal solution, $r = 1, 2, \ldots, R$, where R is the number of current candidate Pareto solutions.
Let,

$$\mathbf{f}(x^{*r}) = [f_1^{x^{*r}}, f_2^{x^{*r}} \ldots, f_l^{x^{*r}}]^T$$

be the corresponding value of the objective function vector.
For a new solution x^j, objective function vector $f(x^j)$ is calculated. Further, the new candidate solution is being compared with existing Pareto solutions. There can be three cases:

- If x^j dominates at least one candidate solution, we delete the dominated solution and add the new solution to our set of candidates.

- If x^j does not dominate any existing candidate solutions, then add x^j to the set of candidate Pareto solutions.
- If x^j is dominated by a candidate solution, we do not change the set of candidate Pareto solutions.

3.2 Algorithm for generating Pareto points

3.2.1 A Pseudo-Code for generating non-dominated points

For the below Algorithm, we refer to Chapter 24 of the book [2] by Chong et. al.
Before we begin with the algorithm, we will introduce some notations that we will use in writing the following pseudo-code. Let J be the number of candidate solutions to be checked for Optimality, while R be the number of candidates Pareto Solutions. Also let l be the number of objective functions, and n be the dimension of the decision space.

1. Randomly get the initial candidate Pareto Solutions from the obtained Criterion Space. Set initial indices $R := 1$ and $j := 1$
2. Increment j. If $j \leq J$, then move to step 3. Else, stop, because all the candidate solutions have been considered.
3. Set $r := 1$ and $q := 0$ (q is the number of eliminated solutions from the existing set of Pareto solutions).
4. If for all $i = 1, 2, 3, ..., l$,
$$f_i(x^j) < f_i(x^{*r})$$
, then set $q := q+1$, $f^{*R} := f(x^j)$. This stores the solution that should be eliminated. Go to step 6.
5. If for all $i = 1, 2, ..., l$,
$$f_i(x^j) \geq f_i(x^{*r})$$
, then go to step-2
6. Set $r := r + 1$. If $r \leq R$, go to step 4.
7. If $q \neq 0$, remove the solutions that are eliminated in Step-4 and then move to Step-2.
8. Set $R := R + 1$, $x^{*r} := x^j$, $f^{*R} := f(x^j)$. Go to Step 2.

3.2.2 Example

Consider a randomly generated candidate Pareto solutions set given as

$x^{(i)T}$	$\boldsymbol{f}(x^{(i)T})$
(48, 62)	(-130, 100)
(15, 37)	(-88, 42)
(84, 29)	(5, 103)
(53, 91)	(-212, 134)
(76, 18)	(30, 84)
(40, 59)	(-129, 89)
(23, 77)	(-200, 90)
(68, 12)	(40, 70)
(56, 83)	(-185, 129)
(95, 20)	(43, 105)

The algorithm works as described in section 3.1: Here $x^{(i)T}$ is our set of candidate Pareto solutions, which we have generated randomly from the feasible space, which we denoted as J in the algorithm.
Now, one candidate Pareto solution is taken for J and compared with all the elements in J.

- Assume (48, 62) $\in R$ i.e., to be a Pareto solution, then compare it with other points J.
- (15, 37) dominates (48, 62); hence, we eliminate (48, 62) from R and include (15, 37)
- (84, 29) is dominated by (15, 37); thus, we do not change R.
- Now (53, 91) does not dominate (15, 37). Hence we add (53, 91) to R.

This way, the algorithm continues, and we will find the *non-dominated* points for this problem as,

$x^{(i)T}$	$\boldsymbol{f}(x^{(i)T})$
(15, 37)	(-88, 42)
(53, 91)	(-212, 134)
(40, 59)	(-129, 89)
(23, 77)	(-200, 90)

3.2.3 Algorithm for computing Pareto Optimal points given objective functions and constraints

We can use the algorithm discussed in section 3.2.1 on the feasible space of a given problem to find all the Pareto Optimal points. But notice that it would take so long to compare all points in feasible space with each other.
Due to limitations in computational power, we could choose only some randomly selected points from the entire feasible space and find non-dominating points. Doing this process iteratively will eventually help to find Pareto Optimal solutions for a given problem.

1. Set POP_SIZE and NUM_ITER variable for the number of random points we want to generate and the number of iterations we want to perform.
   Initialize xP and fP as an array of our Pareto solutions and objective function values at Pareto points.

2. For all $i = 1, 2, \ldots, num\_iter$,
   Set $POPULATION :=$ randomly generate POP_SIZE points from feasible space.

3. If $i \neq 1$, append xP values in $POPULATION$, otherwise go to step 4.

4. Run the algorithm in section 3.2.1 on $POPULATION$.

5. If $i <= num\_iter$, move to step 2.

Using the above pseudo-code, we have developed a Python Code to obtain the Pareto Front/Non-Dominated set of points for any of the above solutions.
Consider the same example mentioned in Section 2.2.1; we have obtained Fig. 3 by using the Python Code of the above-mentioned algorithm.

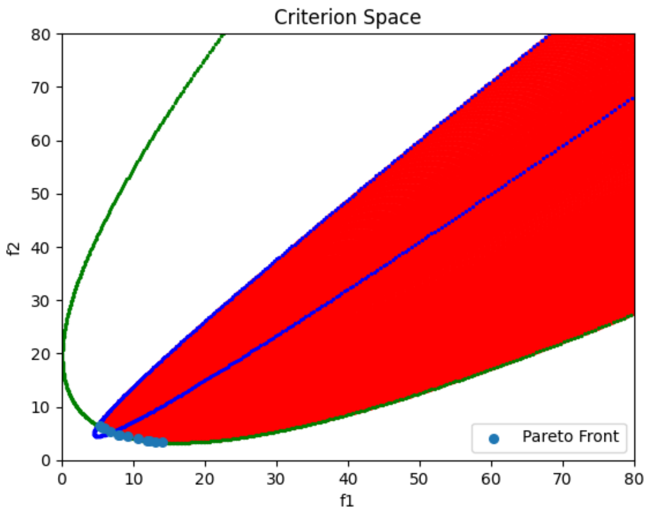

Figure 3: Pareto Front Obtained using the above-mentioned Algorithm. The Algorithm was simulated for 500 points in the feasible Criterion Space for 200 iterations

4 Case study - Marketing Strategies for an Airline Company

A regional airline company is planning its marketing strategy to attract passengers and improve customer satisfaction. The company has two advertising options: traditional media campaigns and personal appearances. Each traditional media campaign costs 2000 rupees and results in 2 new passengers and 1 positive rating per month. Each Personal appearance costs 500 rupees, generating 2 new passengers and 5 positive ratings. Further, each personal appearance takes 2 hours, and each Advertisement campaign takes 1 hour.

The company aims to achieve a minimum of 16 new passengers and 28 positive ratings per month to stay competitive in the market.

Minimize

$$cost = 2000x_1 + 500x_2$$

$$time : x_1 + 2x_2$$

subject to the constraints,

$$2x_1 + 2x_2 \geq 16$$

$$x_1 + 5x_2 \geq 28$$

We will now use the above-understood concepts and generate the Pareto Front using the algorithm learned above.
First, we construct the Design Space of the problem using the objective functions.

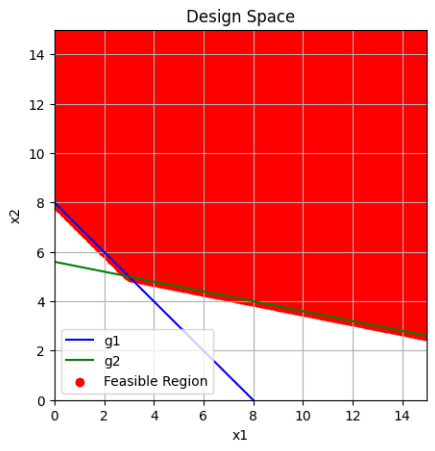

Figure 4: Desgin Space of the above-mentioned problem. g_1 and g_2 represent the objective functions

We now construct the Criterion Space using the Design Space and then run the Algorithm in Section 3.2 to Generate the Pareto Front. We obtained the below plot for the problem.

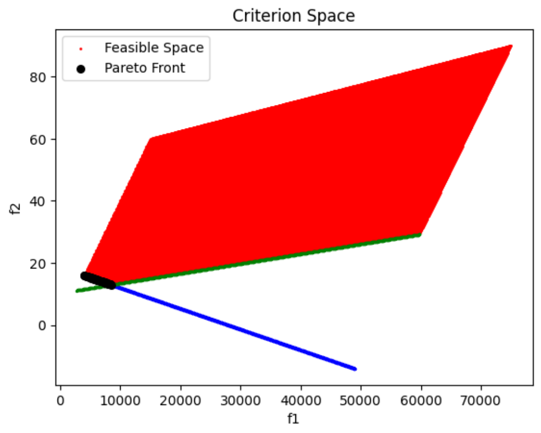

Figure 5: The above Pareto Front is obtained from generating around 1000 points while iterating the algorithm for 500 iterations

From the above Fig. 5, we observe that for the points on the black dotted line, which is known as the Pareto Front is considered as Pareto Optimal because as we move along this line, we could either minimize cost at the expense of time or minimize time at the expense of cost. Say, for example, we have points $x = (8500, 13)$ and $y = (4000, 16)$ on the black dotted line; therefore, on moving from x to y, we could minimize the cost but only at the expense of time. Hence, we can say that all these values on the black line can be an optimal choice for our multi-objective optimization problem.

5 Conclusion and Future Scope

In this report we present our study on the concept of Pareto Optimal Points for a Multi-Objective Optimization problem. By analysing the Pareto Minimizers or the Pareto Front we understand the trade-off and the relationships between conflicting objectives. Through the real world problem of a small firm with limited resources, we highlight the significance of Pareto Optimality in guiding the decision making of the company, offering a nuanced understanding of optimal solutions that balance diverse, often competing, objectives.
This technique finds enormous applications in wide range of fields, ranging from algorithm development to finance, logistics and beyond. This helps in developing efficient solutions to many real world multi-objective problems.

References

[1] Jasbir S. Arora. *Introduction to optimum design.* Academic Press, 3 edition, 2011.

[2] Stanislaw H. Zak Edwin K. P. Chong. *An Introduction to Optimization.* Wiley Series in Discrete Mathematics and Optimization. Wiley, 4 edition, 2013.