EE609A: Multiobjective Optimization using Pareto Descent and MGDA

Nilotpal Pathak *150456*, Pratyush Garg *15807521*

***Abstract*—The problem of multiobjective optimization is taken up here, in which there are multiple objective functions which we try to optimize simultaneously. Unlike single objective problems, it is unlikely here that the same input will minimize all the objective functions simultaneously. This necessitates that we first study the notions of Pareto optimality to clarify the solutions to strive for. Two methods for finding such Pareto optimal solutions to multiobjective problems are discussed- Multiple Gradient Descent Algorithm (MGDA) and Pareto Descent Method (PDM). Both algorithms iteratively move the input in a direction that is opposite to a convex combination of the gradients of the objective functions at the present point. However, the methods employed to find the coefficients for the said convex combination are slightly different in the two algorithms. In this paper, we look at the implementation of the algorithms and analyze the similarities between the two algorithms and the impact of the differences. We also provide some clarifications and modifications to the suggested methods while providing a brief discussion on the convergence of PDM.**

I. Introduction

WE find applications of optimization problems everywhere in life. Single objective optimization problems have been well researched and have direct heuristics for a solution in most cases. However, not all problems are so simple- eg. in economics, finance, optimal control and many other areas, we find that we would like to simultaneously solve multiple objectives, even in the presence of trade-offs between them. Therefore, an important area of work, today, is finding solutions for multiobjective optimization problems.

Motivated by the need for such algorithms, we selected two algorithms, the Multiobjective Gradient Descent (MGDA) [1][2] and the Pareto Descent Method (PDM) [3] for analysis. The rest of the term paper is organized as follows: Section 2 presents the basic theoretical background required for analysis of multiobjective problems, Section 3 discusses the algorithm that we implement for solving the said problem, Section 4 describes the experiments conducted and presents the results of the simulation, Section 5 comments on the convergence of the algorithm, Section 6 points out modifications and critiques the aforementioned techniques and we finally conclude in Section 7.

II. Theoretical Background

The problem we will discuss here is the simultaneous minimization of a number of objective functions defined by

$$f(x) = (f_1(x), f_2(x)...., f_m(x)) \tag{1}$$

Department of Electrical Engineering

Each function f_i is dependent on a set of n input variables given by $x = (x_1, x_2, ..., x_n)$. Here, since all the objective functions are not minimized at the same point, two such points may exist, between which we cannot decide which is optimal. For example, for m=2, without further specifications, it is impossible to decide whether a point where $f = (1, 2)$ is better than a point where $f = (3, 1)$. To disambiguate in such cases, the notion of Pareto optimal points is introduced.

A. Pareto Optimal Points

To understand Pareto optimality, we first define the notion of superior points. A point x_1 is said to be superior to x_2 if

$$f_i(x_1) \leq f_i(x_2) \forall i \in \{1, 2, ..m\}$$

$$\text{and } \exists\ j \in \{1, 2, ..m\} \text{ s.t. } f_j(x_1) < f_j(x_2)$$

A point x_0 is Pareto optimal iff there are no points in the feasible region that are superior to x_0. Similarly, we define $locally\ Pareto\ optimal\ points$. These are points such that in a small neighbourhood around them, there is no point that is superior to them i.e. no point exists which can improve all the objective functions simultaneously over. In the paper, we study iterative approaches that converge on a locally Pareto optimal point. Using a number of initial points with these approaches, we can obtain the complete set of locally Pareto optimal points. After that, the globally Pareto optimal points can be found out by comparison.

B. Descent Directions

We define descent directions as the directions in which we can move at a particular point to improve all the objective functions. **Note that if at a particular point, descent directions do not exist, it implies that the point is locally Pareto optimal** since there exist no points in its neighbourhood that improve all the objective functions. Also, this is not necessarily a unique direction. Let d be a descent direction at a point x. Since moving towards d improves all the objective functions, we have

$$d.(-\nabla f_i(x) \geq 0\ \forall\ i \in \{1, 2, ..., m\}) \tag{2}$$

Note here that the linear combination (with positive coefficients) of descent directions is also a descent direction. Hence the set of descent directions forms a convex cone.

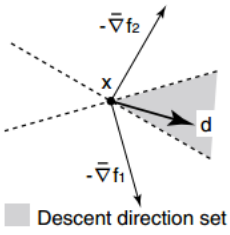

Figure 1: A descent direction: ***at a solution x of a 2-variable-2-objective problem***

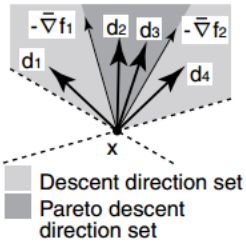

Figure 2: Pareto descent directions: ***at a solution x of a 2-variable-2-objective problem***

Fig. 1. Pareto Optimality (courtesy: Harada et al)

C. Pareto Descent Directions

Taking the concept of Pareto optimality to descent directions, a directiond is a Pareto descent direction if there exists no other direction which improves all the objective functions at a faster rate than d. Any convex combination of the gradients of the objective functions at a particular point is a Pareto descent direction and vice versa. This has been illustrated below graphically for two dimensions and can be extended to higher dimensions as well. Hence, if d is a Pareto descent direction

$$\exists\ \alpha_i \geq 0 \text{ for } i = 1, 2, ..m \text{ s.t. } d = \sum_{i=1}^{m} \alpha_i(-\nabla f_i(x))$$

Similar to descent directions, the set of all Pareto descent directions also forms a convex cone. Fig 1 illustrates what the set of descent directions and Pareto descent directions looks like, given the gradient directions at a particular point.

III. Description of Algorithm

A. Unconstrained Case

Both the algorithms we are discussing are iterative algorithms that move the input in Pareto descent directions, thus improving all the objective functions maximally. This happens until there are no Pareto descent directions left to move in, which implies that we are at a locally Pareto Optimal point, as discussed above. The detailed method is given below. We first specify the method for unconstrained case and then discuss how constraints can be tackled.

i) Choose an initial point x_0. Calculate the gradients of all objective functions at this point - $\{\nabla f_1(x_0), \nabla f_2(x_0), .., \nabla f_m(x_0)\}$. Say the normalized gradients are $\bar{\nabla} f_1(x_0), \bar{\nabla} f_2(x_0), .., \bar{\nabla} f_m(x_0)$.

ii) Solve the following linear programming problem to find the coefficients α_i for the convex combination of gradients. Define β_{ij} = $< \bar{\nabla} f_i, \bar{\nabla} f_j >$

$$\max\ \alpha_k$$

$$\text{s.t. } \sum_{i=1}^{m} \alpha_i \beta_{ij} \geq 0\ (j = 1, 2, ..m),$$
$$\sum_{i=1}^{m} \alpha_i \leq 1;\ \alpha_i \geq 0\ i = (1, 2, ..m)$$

The first constraint ensures that the resultant direction is a descent direction and improves all the functions. The last two constraints ensure that the resultant direction is a convex combination of the gradients of the objective functions. The objective function is not unique and alternatives like MGDA (discussed later) and others exist. Choosing this particular objective however ensures that the trivial solution $\alpha = 0$ is avoided if other feasible solutions exist. It is also linear and hence efficiently solvable.

iii) This gives us M possible Pareto descent directions we could move in- one for each value of k in the previous problem. The possibilities that arise from having multiple optimal directions have been discussed later. Assume we move in any of these directions to reach a new point x_1, which improves all objectives functions from x_0

iv) We continue moving our point in this fashion until α comes out to be zero, which would imply that we are already at a Pareto Optimal point. Note that α = 0 is a trivial solution to the linear programming problem but is attained only when no other solutions exist. Also, if feasible solutions exist, sum of α_i is one because it is a maximization problem and other constraints are scalable.

Note: The major difference between PDM and MGDA lies in step ii). The method given above discussed above is PDM. In MGDA, they solve the following problem instead:

$$\alpha = arg\ min(j(u)) \text{ where } j(u) = u^T u$$

$$\text{s.t. } \sum_{i=1}^{m} \alpha_i = 1,\ \alpha_i \geq 0 :\ i = (1, 2, ..m)$$

This however is a quadratic problem, rather than the LP discussed earlier. So, we expect that the time taken per iteration in MGDA is larger.

B. Constrained Case

Equality constraints : As discussed in class assignments, if equality constraints are given, we can just transform the problem to a lower dimensional problem with no equality constraints. Say t linear equality constraints are given. We define a linear transformation that transforms input x (n-dimensional) to a y(n-t dimensional) for which the equality constraints are always satisfied.

Inequality constraints : For the purpose of this paper we consider linear inequality constraints only. As pointed out by Harada et al in their paper, non-linear inequality constraints too can be approximated as linear by Taylor aproximation. Say we have p linear equality constraints of the form $g_j^T x \geq 0$. Then we use the following process to ensure convergence to a locally Pareto optimal point:

i) Before each update step ($x_k \rightarrow x_{k+1}$), we check if the new point lies in the feasible region. If that is the case, the update is made.

ii) However, if it lies in the infeasible region, we go back and try to find a different Pareto optimal descent direction by changing the value of k in the LP problem.

iii) If all of the m possible Pareto descent directions lead us out of the feasible region, we assume that there exist no feasible Pareto descent directions.

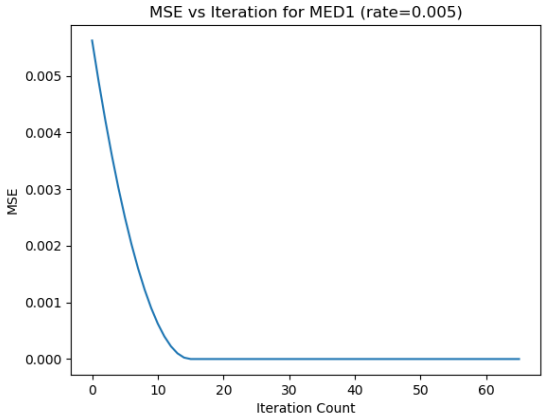

Fig. 2. MED 1: Convergence in 14 Iterations

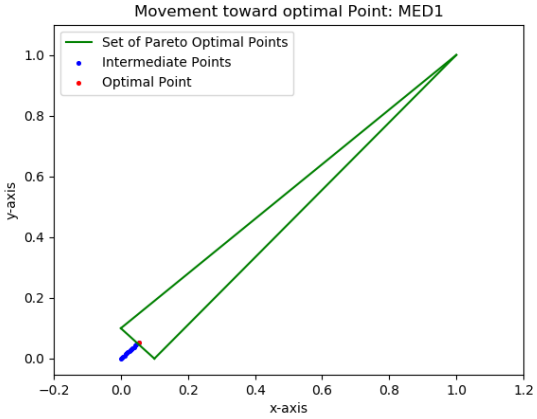

Fig. 3. MED 1: the triangle is the boundary of all possible solutions

iv) We then try to find a descent direction d instead, by solving the following linear program as suggested in [3].

$$\max\ w.d$$

$$\text{s.t.}\ \ \text{d.}(-\nabla f_i(x)) \geq 0: \ \ (j = 1, 2, ..m),$$
$$\text{-1} \leq d_i \leq 1: \ \ (i = 1, 2, ..n)$$

v) Harada et al [3] suggest that w here be a random vector. They suggest we try to obtain such descent directions for a large number of w vectors until we either obtain a feasible direction or hit a decided threshold on the number (say D) of attempts.

However, it makes more sense to keep track of the inequality constraints g_j for which violations occur, and then maximize dot product with these g_j instead, to move away from the infeasible region by the maximum possible amount while reducing all objective functions. We will talk about this modification in Section 6.

vi) If even after D attempts of finding a descent direction, we fail to find one that is feasible, we conclude that there are no feasible descent directions and hence the current point is locally Pareto optimal.

IV. Simulation Analysis

We used the algorithm on the problems mentioned in [3] and also created a toy example to validate the results of the simulation. We describe these so called 'Benchmark Problems' below.

A. MED 1: The 3 Objective Problem

We used the 3-variable 3-objective benchmark problem of Multiple Euclidean Distances (MED) with the function:

$$f_{11}(x) = ||x - c_{11}||^2; f_{12}(x) = ||x - c_{12}||^2; f_{13}(x) = ||x - c_{13}||^2;$$

where, $c_{11} = (1, 1, 0);\ c_{12} = (0.1, 0, 0);\ c_{13} = (0, 0.1, 0)$. We used this problem on an unconstrained implementation of the algorithm. The theoretical Pareto optimal solutions of this problem form a triangle with the vertices at $c_{11},\ c_{12},\ c_{13}$. Hence, every point inside it (Fig 3) is a pareto optimal solution.

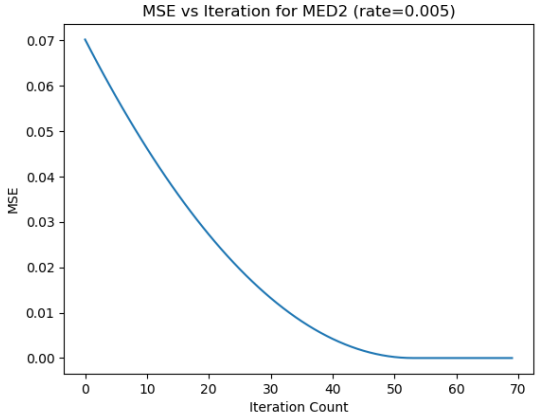

Fig. 4. MED 2: Convergence in 54 Iterations

Since there are many possible end-points for the algorithm, the particular solution we stop at, is a function of the initialization x_0 and the step-size ρ. In our simulation, we chose $x_0 = (0, 0)$ and $\rho = 0.005$. The algorithm converged in 14 iterations (Fig 2) and the convergence point as illustrated in Fig 3 was just greater than $(0.05, 0.05)$ where our movement encounters the first optimal point.

B. MED 2: Solutions on the Constraint Boundary

To ensure that the algorithm was accurately handling cases on the boundary of the feasible region in the constrained cases, the second benchmark problem is a 2-variable 2-objective problem with the function:

$$f_{21}(x) = ||x - c_{21}||^2; f_{22}(x) = ||x - c_{22}||^2;$$

where, $c_{21} = (0, -1);\ c_{22} = (1, -1)$ and the feasible region is such that $(x_1, x_2) \in [-1, 2]X[0, 1]$. Here, it can be theoretically calculated that the Pareto optimal solutions of this problem lie on the line joining the points $(0, 0)$ and $(1, 0)$.

Again, since there are many possible end-points for the algorithm, the particular solution we stop at, depends on

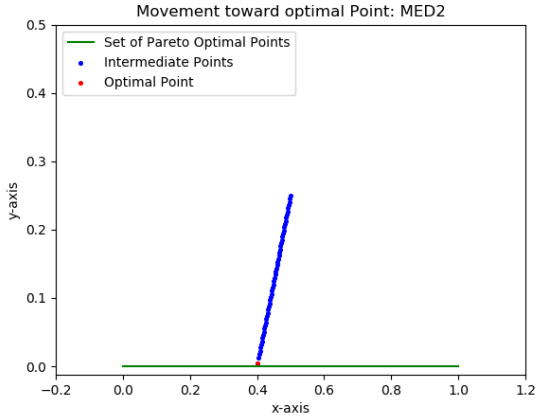

Fig. 5. MED 2: Towards Pareto Optimality

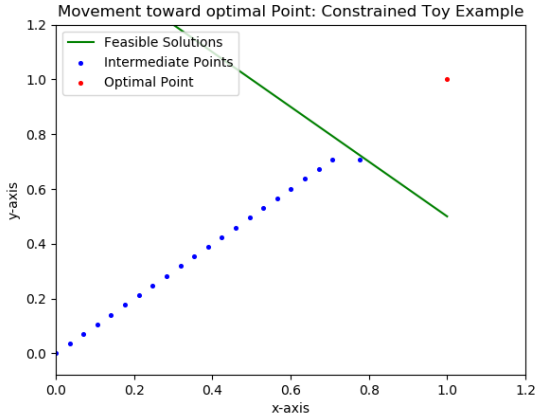

Fig. 7. Toy Constrained: Feasible Solutions all lie on $x_1 + x_2 = 1.5$

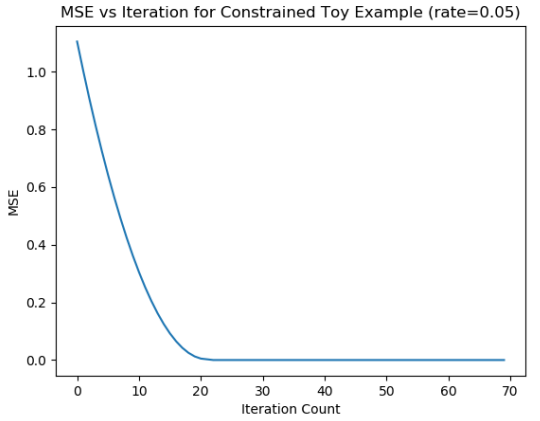

Fig. 6. Toy Constrained: Convergence in 23 Iterations

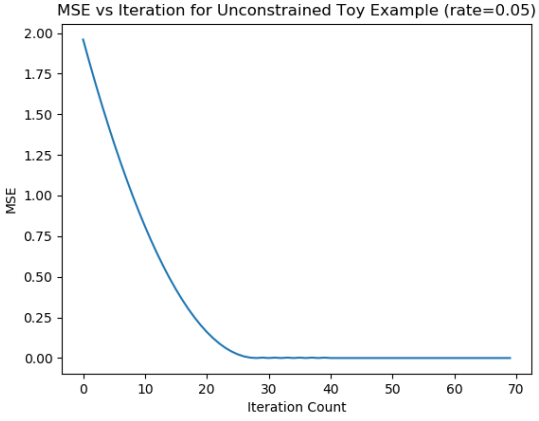

Fig. 8. Toy Unconstrained: Oscillation in 34 iterations

the initialization x_0 and the step-size ρ. In our simulation, we chose $x_0 = (0.5, 0.25)$ and $\rho = 0.005$. The algorithm converged in 54 iterations (Fig 4) and the convergence point as illustrated in Fig 5 is just around $(0.4, 0)$ where our movement encounters the first optimal point.

C. Toy Example: Single Optimal Solution

Finally, we also tested the code on a toy example where there was a single solution because both the objective functions could be minimized simultaneously at that point. The function is a 2-variable 2-objective problem as:

$$f_{31}(x) = (x_0 - 1)^2(x_1 - 1)^2; f_{32}(x) = (x_0 - 1)^2 + (x_1 - 1)^2;$$

We tested this for both the constrained case (with constraint as $x_0 + x_1 = 1.5$) and the unconstrained case with the initial point $x_0 = (0, 0)$ and rate $\rho = 0.05$. While the constrained case converged in 23 iterations (Fig 6,7), we uncovered an interesting trade-off when dealing with the unconstrained problem.

Here, the rate massively effected the convergence. When it was taken too small (0.005), the progress was slow and the number of iterations required exceeded the limit. When it was taken large (0.05), the algorithm quickly reached optimal point (1,1) but owing to the larger rate, was unable to converge and began oscillating across the optima. This resulted in errors in the converged value but they were small and therefore cannot be aptly seen on Fig 8.

This concludes the section on simulation analysis of the algorithm. The variety of cases tried ensures that the code (entirely self-written) is robust and very adaptable using the hyperparameters of x_0, ρ and the limit on max iterations.

V. Comments on Convergence

Note: The papers that we have read and analyzed do not contain any of the points that we discuss below with respect to MGDA and PDM.

In this section we will try to analyze what kinds of functions this method is able to handle. We then try to compare the convergence of this algorithm to that of gradient descent algorithm (one dimension) and discuss why similar comments can not be made about rate of convergence even if all objective functions are μ-strongly convex and Lipschitz continuous. First we consider some cases which can become problematic for the algorithm. Before beginning, it is important to point

out that the method only works if all the objective functions are differentiable at all the points inside the feasible region.

A. One or more of the objective functions are concave:

Even if one function is concave, we can not guarantee that the algorithm will converge because it is possible that the problem become unbounded. A simple example of such a case would be the minimization problem $f(x) = ((x_1 - 1)^2, -x_2^2)$ where m = 2 and n = 2. Here, we can easily observe that irrespective of the starting point we choose, the algorithm never converges. For a fixed value of f_1, we can keep changing x_2 to progressively reduce f_2 and hence the problem becomes unbounded.

B. One or more of the objective functions is neither concave nor convex:

Here, convergence cannot be guaranteed for all initial points though some points might lead to convergence to locally Pareto-optimal points. If other functions are optimized in a region where the non-convex functions display concavity, problems similar to the previous part might arise and we may obtain an unbounded solution.

C. All the objective functions are convex:

Even in such problems, it is not necessary that the algorithm converge because it is possible that despite the convexity of all functions, no Pareto optimal points exist. Consider the example of the problem $f = ((x_1 - 1)^2, e^{x_2})$ where Pareto optimal points do not exist and our algorithm will iteratively keep reducing x_2 but never converge.

In fact, we show below that even the properties of Lipschitz continuity and μ-strong convexity, under which convergence of 1-d gradient descent algorithm is guarranteed to converge with number of iterations of $O(1/\epsilon)$, does not really guarantee convergence in the same fashion here. Assume x* is the solution that the algorithm eventually converges to. Also, all objective functions are strongly convex with respect to parameter μ and Lipschitz continuous with parameter L. ρ is the step size. We proceed in similar fashion to convergence analysis of gradient-descent and show why the same may not work here.

D. Comparison with Convergence of Gradient Descent

For our algorithm: $x_{k+1} = x_k - \rho \sum_{i=1}^{m} \alpha_i \bar{\nabla} f_i(x_k)$. Now, let $\delta_i = \alpha_i / ||\nabla f_i(x)||$

$$\implies x_{k+1} = x_k - \rho \sum_{i=1}^{m} \delta_i \nabla f_i(x_k)$$

$$\implies ||x_{k+1} - x^*||^2 = ||x_k - x^* - \rho \sum_{i=1}^{m} \delta_i \nabla f_i(x_k)||^2$$

$$= ||x_k - x^*||^2 + \rho^2 ||\sum_{i=1}^{m} \delta_i \nabla f_i(x_k)||^2 - 2\rho A, \text{ where:}$$

$$A = < x_k - x^*, \sum_{i=1}^{m} \delta_i \nabla f_i(x_k) >$$

.

Now, to display convergence rate analogous to gradient descent, we need to show that the second and third terms are bounded above by a term of the form $c||x_k - x^*||^2$ In the case of gradient descent, we were able to derive:

$$< \nabla f(x), x - x^* > \geq \mu ||x - x^*||^2 \text{ and,}$$

$$||\nabla f(x_k)||^2 \leq L ||x_k - x^*||^2$$

using strong convexity and Lipschitz continuity respectively. This gave:

$$||x_{k+1} - x^*||^2 \leq \gamma^k ||x_k - x^*||^2$$

where, $\gamma^2 = 1 + 2\rho^2 L^2 - 2\rho\mu$.

But in the pareto optimal case, as is apparent from the MED problems, it is not necesssary that x^*, the point at which we converge satisfy $f(x^*) = 0$. So, here, using the same properties we get:

$$< \nabla f_i(x^*) - \nabla f_i(x), x^* - x > \geq \mu ||x^* - x||^2 \text{ and,}$$

$$||\nabla f_i(x_k) - \nabla f_i(x^*)||^2 \leq L^2 ||x_k - x^*||^2$$

But, since we have no limits (in this case) on $\nabla f_i(x^*)$ for pareto optimal x^*, similar conclusions cannot be made for this algorithm despite strong continuity and Lipschitz continuity of all objective functions.

E. Intuition for Convergence in Well Behaved Cases

However, the condition of $d.(-\nabla f_j(x_k)) \geq 0$ for search directions d implies that the algorithm would, in most cases, improve one or more functions by $\rho d.(-\nabla f_j(x_k))$ without increase in the other objective functions.

The only case where the algorithm does not improve any of the objective functions despite update, is when

$$\sum_i \alpha_i \beta_{ij} = 0 \text{ for } j = 1, 2, 3...$$

Say B is the mxn matrix whose ij^{th} entry is β_{ij}. Since, β_{ij} is the dot product between the i^{th} and j^{th} gradients, we have $B_{ii} = 1$. No improvement happens in all of the functions iff $B\alpha = 0$. For this, α must lie in the null space of the matrix B. Since α is calculated from an independent route of linear optimization, this is unlikely and for relatively well behaved functions, we can rely on the algorithm for convergence.

VI. Some Notes: Critical Analysis and Modifications

We present a few notes critiquing the techniques developed and also suggest modifications and scope for future work.

A. Selection of α_k

While Harada et al [3] mention that the choice of α_k for maximization in the linear program results in different descent directions and hence different solutions, they never describe any strategies for selection. We propose 3 separate strategies and employ one of them in the code. One, is to always choose the same α_k at each iteration to ensure that a pareto optimal point is reached in the quickest possible way. Second, is to cycle between different k's at each iteration so that a point that is close to the minima of every individual objective is attained. Finally, we can also solve multiple linear programs at each iteration and choose a convex combination of the consequent gradients which is also guaranteed to be a pareto descent direction. In such a way, with the same hyperparameters, we can get multiple pareto optimal points. Further intelligent schemes can be derived to serve other specific aims , for eg. when we have a significance/importance ordering on the objective functions.

B. PDM vs MGDA

Both the algorithms are based on the same principles of gradient descent along a convex combination of the steepest descent directions. However, PDM [3] manages to formulate the problem as a linear program at every iteration instead of a quadratic program [1][2]. Additionally, the discussion on feasibility and boundary conditions is substantially more thorough in [3]. Hence, PDM seems to be superior to MGDA in time complexity and in handling of boundary cases.

C. Finding Descent Directions

As mentioned before during our description of the algorithm, Harada et al claim that any random weight vector w would result in a suitable descent direction after a few tries. This is a wasteful and unimaginative solution to the problem. We propose to use existing information on constraints by choosing the normal of the hyper-plane that violates feasibility as w. Maximizing $w.d$ in such a situation, would effectively be maximizing the distance from that culprit boundary, which makes intuitive sense. Further, moving away from the boundary may be a better solution that moving tangential to it as suggested in [1].

Again, [3] does not discuss when to stop trying random w's. This question is left unanswered and would be crucial in getting accurate solutions to special cases where feasibility regions are highly non-trivial.

D. Does absence of Pareto Descent Directions imply absence of Descent Directions?

This is a slightly non-trivial, non-intuitive and yet important question. The algorithm relies on it when it directly declares a point as pareto optimal in the absence of *pareto descent directions* (i.e. when $\alpha = 0$) without worrying about descent directions as a possibility. We find that while this is true in almost all cases, there exists an exception.

Consider Fig 1: the darker region represents the pareto dd's and the lighter are just descent directions. In this case, if we rotate the first steepest descent direction anticlockwise till both of them are at 180 degrees, we see that by definition, a descent direction exists at 90 degrees to both of them. However, this direction *cannot* be produced by a convex combination of the steepest descent directions and hence it is not a pareto dd.

Therefore, this exceptional case is one where even in the absence of a pareto dd, we can have a descent direction at that point. This becomes crucial because while the algorithm would declare the point to be pareto optimal, there is a chance that movement in the descent direction reveals an overall better point.

VII. Conclusion

This concludes our discussion on multiobjective gradient descent and pareto optimality. We implemented all the code from scratch (Section 4), commented on the complex nature of convergence analysis (Section 5) and pointed out some delicate points and modifications (Section 6).

We thank Prof. Ketan Rajawat for this opportunity and the wonderful learning adventure that he led us on.

VIII. Bibliography

[1]: Jean-Antoine Dsidri. "Multiple-Gradient Descent Algorithm (MGDA)." [ResearchReport] RR-6953,INRIA. 2009

[2]: Desideri, Jean-Antoine. "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization." Comptes Rendus Mathematique 350.5-6 (2012): 313-318.

[3]: Harada, Ken, Jun Sakuma, and Shigenobu Kobayashi. "Local search for multiobjective function optimization: pareto descent method." Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, 2006

[4]: Brown, Martin and Hutauruk, Nicky. "On the Convergence of Multi-Objective Descent Algorithms." Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making. MCDM, 2007

[5]: Singer, Yaron. "Convergence Analysis of Gradient Descent." Lecture 9, AM221: Advanced Optimization. Harvard University, Spring 2016.

[6]: Miettinen, Kaisa. "Nonlinear Multiobjective Optimization." University of Jyvaskyla. 1998

[7]: Shukla, Pradyumn Kumar, Dutta, Joydeep and Deb, Kalyanmoy. "On Properly Pareto Optimal Solutions."

[8]: Rajawat, Ketan. "EE609: Convex Optimization in Signal Processing and Communication." Indian Institute of Technology Kanpur. Fall Semester. 2018-19.

[9]: Svaiter, Benar F. "The Multiobjective Steepest Descent Direction is not Lipschitz Continuous, but is Hlder Continuous." Elsevier. 2018.

[10]: Boyd, Stephen and Vandenberghe, Lieven, "Convex Optimization." Cambridge University Press. 2004.

[11]: Branke, Jrgen, Deb, Kalyanmoy,Miettinen, Kaisa and Sowinski, Roman. "Multiobjective Optimization". Springer. 2008.