A Fair, Sequential Multiple Objective Optimization Algorithm

Can KIZILKALE

July 1,2012

Abstract

In multi-objective optimization the objective is to reach a point which is Pareto efficient. However we usually encounter many such points and choosing a point amongst them possesses another problem. In many applications we are required to choose a point having a good spread over all objective functions which is a direct consequence of the notion of fairness. In this paper we propose two things. The first one is the concept of "Feasible Fairness" and the second one is a novel, simple to implement and fast algorithm which is guaranteed to converge to a "feasibly fair" point if the objective functions are continuous and strictly quasi-convex on a compact and convex domain.

1 Introduction

The standard definition of a multiple-objective optimization problem is as follows:

$$\begin{aligned} &\text{Minimize }_x(f_1(x), \cdots, f_M(x)) \\ &\text{Subject to } g_j(x) \leq 0, j \in \{1, \cdots, J\} \end{aligned} \tag{1}$$

Where $g_1(x), \cdots, g_1(x)$ are the constraint functions and $f_1(x), \cdots, f_M(x)$ are the objective (also called cost) functions (we can always add equality constraints as pairs of inequality constraints so they were not given in the definition). In this problem definition we have x as the decision variable, which is usually chosen from R^n or some decision space X. We will denote the set of feasible variables x by Z that is $Z = \{x \in X \| g_j(x) \leq 0, j \in \{1, \cdots, J\}\}$.
An important question about this problem is how minimization is defined for a vector of cost functions. The necessary and the most fundamental condition for a point $x^* \in Z$ to be minimum is that the vector of cost functions has to be Pareto efficient ([1] for definition of Pareto efficiency) at that point, meaning that we can not have another point $x' \in Z$ such that $f_i(x^*) \geq f_i(x^{'})$ for every $i \in \{1, \cdots, M\}$

moreover $\exists i'$ such that $f_{i'}(x^*) > f_{i'}(x^{'})$. This condition by itself leads to a set of points, which can be infinitely many, and if there exists more than one point, choosing a point from this set possesses another problem. Many different methods for choosing such a point are proposed in the literature, surveys such as [2] and [3] give necessary overview on the subject.
In many cases we want to choose a point which makes the costs as close to each other as possible. This need comes from the desire to be fair among the objects of optimization($f_1(x), \cdots, f_M(x)$ in our case). However without a proper definition of "fairness" it is hard to come up with a meaningful method. So we are going to give a new and simple definition for fairness in the following section. In section 3 we give our algorithm, in section 4 we give an analysis of the algorithm show that it converges after at most $2M^2$ repetitions.

2 Fairness

The true definition of Fairness is a deep philosophical question which is out of the scope of this paper. What we will focus on as fairness is the elimination of envy. Cake-cutting problem is probably the most famous example for this kind of fairness argument where we are trying to share a cake amongst people who have different valuations on the size of the portion they receive. The problem is to divide the cake so that no person envies another's portion. A nice treatment of this kind of problem can be found in [4]. In the definition of multi-objective optimization problem (1) we can think of each $i \in \{1, \cdots, M\}$ as an agent and $w_i = -f_i(x)$ as the utility of that agent. For this utility definition, a point x^* which eliminates envy has to be the one which assigns equal value to each cost function, that is $\forall i, j \ f_i(x^*) = f_j(x^*)$. However in constrained optimization it is highly unlikely that we have such a point which is also Pareto efficient, hence a solution to the multi-objective optimization problem (1). To overcome this problem we are going to relax the notion of "envy" and define "Feasible-envy" as follows.

2.1 Feasible Envy

Let A and B be two human beings. We say that A envies B if A desires the utility of B more than her own. This definition however is too strict and does not reflect the effect of feasibility on envy. Let's think about the cake sharing example where we are deciding on how to share the cake between two people. Here our decision variable is the size of the cake we allocate to each person. In this case if both desire the cake equally, dividing the cake in half will remove envy between these two persons. Now assume that we have a cherry on top of the cake and our knife is not sharp enough to divide the cherry, no matter how we divide the cake cherry must stay on only one piece. Now let both people desire cherry equally much and more than the whole cake. It is clear that no matter what we decide the person who does not get

the cherry will always envy the other one. Here there is envy that is not possible to eliminate because of a feasibility constraint, which is that the cherry can not be cut. We are going to say that in such a case there is no feasible-envy or no envy that can be justified. Now we can give the following definition.

Definition 1. *Feasible Envy. In the multi-objective optimization problem defined in (1), $\forall i$ and some $x \in Z$ let $-f_i(x) = w_i$ be the utility value of agent i. Let $i \neq j$ in M , such that i prefers w_j to his/her utility w_i. If there exists a feasible x' such that $\forall k \notin \{i, j\}$, $-f_k(x')$ is at least as good as w_k($-f_k(x') \geq w_k$) for agent k, $-f_i(x')$ is at least as good as $w_j(-f_i(x') \geq w_j)$, for agent i and agent j prefers $-f_j(x')$ to $w_j(-f_j(x') > w_j)$, then we will say agent j feasibly envies agent i.*

To put this in words, if an agent j can be made better by not making agent i worse than what j is getting right now and keeping the remaining agents at least as good, then agent j envies agent i and this envy is feasible hence can be justified. Since fairness is defined as envy-freeness, we make the following relaxation on it to define a feasibly-fair decision point x.

Definition 2. *Feasible Fairness. For a feasible point $x \in Z$ and utility values of agents $(w_1, \cdots, w_M) = (-f_1(x), \cdots, -f_M(x)$ if there is no feasible-envy for every tuple $i, j \in \{1, \cdots, M\}$ then we call such x a feasibly-fair point of the problem 1 and w is a feasibly-fair utility vector.*

Lemma 1. *There exists a feasibly fair allocation if the space is convex and compact, utility functions are continuous and strictly quasi-convex.*

The proof of this lemma follows from the convergence proof in the analysis section. From lemma 3 we know that there exists a point that the algorithm converges and from lemma 5 we know that the point leads to a feasibly fair allocation which concludes the proof. □

Now we have a nice extension of the fairness for the multi-objective optimization problem, the next thing we are going to do is to give the algorithm which is going to return such a fair and optimum point if it is feasible.

3 The Algorithm

The algorithm we are proposing is given in the table 1. Our aim is not only to find a Pareto efficient point but also a point that is feasibly fair. In our definition of feasible-fairness we said that it is the elimination of envy that leads to such a point. Our algorithm does just that in a sequential manner. At each iteration we choose one of the cost functions ($f_i(x)$)to work with(the simplest way to chose is to choose them one by one in order). We search for a point which will make $f_i(x)$ smaller, while for every $j \in \{1, \cdots, M\}$ if the value of $f_j(x)$ was smaller or equal to $f_i(x)$

it will stay that way and if $f_j(x) > f^i(x)$ then $f_j(x)$ will not get any higher then its current value. We can also see this as we make i better by taking from the ones which are already better than i while not making the ones who are getting less than i any worse. This idea is very much like "taking from the rich and giving to the poor".

Algorithm 1 Fair, Sequential Multiple Objective Optimization Algorithm

---

Input : Problem of the form 1 , a feasible $x_0 \in Z$, maximum number of iterations $n \in N$.
Output : A feasibly fair and pareto efficient $x^* \in Z$ if there exists one else outputs a pareto efficient point.
Initiliaze : $(u_1^0, \cdots, u_M^0) = (f_1(x_0), \cdots, f_M(x_0))$

1. For $k = 1$ to n

   2. Choose an $i \in \{1, \cdots, M\}$ [i= (k mod M) + 1]

   3. Set x_k to an optimizer of

$$\begin{array}{rc} \text{Minimize} & f_i(x) \\ \text{subject to} & x \in Z \\ & f_j(x) \leq max(f_i(x), u_j^{k-1}) \text{ for every } j \end{array}$$

   4. Update optimum value and decision variable if there is an improvement.
   [if $f_i(x_k) < u_i^{k-1}$ then $u_j^k = f_j(x^k)$ for every $j \in \{1, \cdots, M\}$
   else $u_j^k = u_j^{k-1}$ and $x_k = x_{k-1}$].

---

4 Analysis

Analyzing the algorithm for arbitrary settings is surely complicated as the inequalities $f_j(x) \leq max(f_i(x), u_j^{k-1})$ may lead to some hard problems, however under some reasonable assumptions we will see that convergence is guaranteed, the algorithm converges in a relatively small number of repetitions moreover the converged point is feasibly-fair.
For the analysis in this section the following are assumed to hold. The domain of optimization, Z, is compact and convex and the objective functions $f_i(x)$ are bounded, continuous and strictly quasi-convex.

Lemma 2. *Let $u^k_{max} = max_i\{u^k_i \| i = 1 \dots M\}$. u^k_{max} is monotonically non-increasing on k.*

Proof. The proof of this lemma can be seen as follows: For every iteration $\exists x \in Z$ such that for every $j \in \{1 \dots M\}$ we have $f_j(x) = u^{k-1}_j \Rightarrow f_j(x) \leq max(f_i(x), u^{k-1}_j)$, then since x_k is a minimizer (step 2 in the algorithm) we have $f_i(x_k) \leq f_i(x) = u^{k-1}_i \leq u^{k-1}_{max}$. So we have $u^k_j \leq max(f_i(x_k), u^{k-1}_j) \leq u^{k-1}_{max} \Rightarrow u^k_{max} \leq u^{k-1}_{max}$. $\square$
This lemma shows that the behavior of the algorithm is monotonic. We are going to use this result in the following lemma to show that the algorithm converges.

Lemma 3. *The algorithm converges to a point after at most $2M^2$ number of iterations.*

Proof. Let's first observe the sequence u^k_{max}. Since all the functions are bounded and continuous then $u^k_{max} = max_i(f_i(x_k))$ is a sequence in a compact set in R and since it is monotonically non-increasing then it will converge to some value $\mu = max_i(f_i(x_\mu))$ at some point $x_\mu \in Z$. Let i be the function being minimized at some iteration k, if $f_i(x_{k-1}) > \mu$ then $f_i(x_k) \leq \mu$ since x_μ is a feasible point for the optimization problem in step 1. After M iterations the algorithm must have visited every $f_i(x)$ at least once hence for $k \geq M$ we have $f_i(x_k) \leq \mu$ for every i. After this iteration, every j having $u^k_j < \mu$ will stay that way in the following iterations since for such function to increase we have to have an i such that $f_i(x^k) > f_i(x^{k+1}) \geq f_j(x^{k+1}) = u^{k+1}_j$ and for $k \geq M$ we have $u^{k+1}_j \leq f_i(x^{k+1}) < f_i(x^k) \leq \mu$.

Let $k \geq M$ and $i' =$ (k mod M) +1 (hence at iteration k objective function $f_{i'}(x)$ is being optimized), If $f_{i'}(x_{k+1}) = \mu$ then there is no $x \in Z$ such that $f_{i'}(x) < \mu$ and $f_j(x) \leq \mu$ for every j, otherwise since $f_i(x)$ is assumed to be continuous and strictly quasi-convex, we can find a small enough $\lambda > 0$ such that $f_{i'}(x_k) > f_j(x_k)$ $\Rightarrow \mu > f_{i'}((1-\lambda)x_k + \lambda x) \geq f_j((1-\lambda)x_k + \lambda x)$. So if $f_{i'}(x_{k+1}) = \mu$ then it will stay that way after kth iteration.

If $f_{i'}(x_{2M+1}) < \mu$ then for $k' \geq 2M$ we have $\mu > c \geq f_{i'}(x_{k'+1})$ for some $c \in R$. The proof is as follows. For $k' \geq 2M$ every $i \in \{1, \cdots, M\}$ is chosen at least twice by the step 2 of our algorithm. From the previous case we know that $u^{k'}$ will not be updated for the i at which $f_i(x_{2M+1}) = \mu$ (the second visit is the previous case). Now for $f_{i'}(x_{k'+1}) > f_{i'}(x_{k'})$ we have to have $f_{i'}(x_{k'}) < f_i(x_{k'})$ (because of the constraint$f_{i'}(x) \leq max(f_i(x), u^{k'}_i)$) and $f_i(x_{k'}) < \mu$ (otherwise there will be no update) where i is chosen by step 2 at iteration k'. But then $max_j\{u^{k'}_j \| u^{k'}_j < \mu\} \geq f_{i'}(x_{k'+1})$ and $max_j\{u^{k'}_j \| u^{k'}_j < \mu\} \geq max_j\{u^{k'+1}_j \| u^{k'+1}_j < \mu\}$ (the maximum valued function can not get larger by the same argument) hence $\mu > max_j\{u^{2M+1}_j \| u^{2M+1}_j < \mu\} = c \geq max_j\{u^{k'+1}_j \| u^{k'+1}_j < \mu\} \geq f_{i'}(x_{k'+1})$ for

every $k' \geq 2M$.

The two cases above clearly show that for $k \geq 2M$, $u^k_{max} = \mu$ moreover if $f_i(x^k) < \mu$ then $\liminf f_i(x^k) \leq c < \mu$. We can remove the functions $f_{i'}(x_{2M+1}) = \mu$ and add them as equalities to the the constraint set, that is $Z' = Z \cap \{x \| f_{i'}(x) = \mu, \forall i'$ where $f_{i'}(x_{2M+1}) = \mu\}$ and wlog. assume that $\{1 \dots M^1\} = \{i \| f_i(2M+1) < \mu\}$ (we can always rename the functions to satisfy this), clearly $M^1 \leq M-1$. Since the algorithm does not update at such i', if we initialize another instance of the algorithm with Z' , $x'_0 = x_{2M+1}$, and consider the cost vector $(f_1(x'_k), \cdots, f_{M^1}(x'_k)) = (u')^k$ we will have $x_k = x'_{k-2M-1}$. By the same argument on this reduced instance we can see that after another 2M iterations we will have at least one $i \in \{1 \dots M^1\}$ such that $f_i(x'_{2M+1}) = \mu^1 = \liminf_k (u')^k_{max}$ and it will stay that way for the remaining iterations of the algorithm. Now we can continue reducing the algorithm at every $2M$ iterations by removing the fixed functions, $f_i(x^l_{2M+1}) = \mu^l$, and embedding them to the constraint set. Since at least one more function gets fixed at each reduction we have $M^l \leq M^{l-1} - 1$ and after at most M reductions every function gets fixed hence the algorithm converges to a fixed point. Since one reduction takes place at every $2M$ iterations we conclude that the algorithm will converge to its limit at no more than $2M^2$ iterations. □

Lemma 4. *If we denote the point that the algorithm converges by x^* then $(u^*_1, \cdots, u^*_M) = (f_1(x^*), \cdots, f_M(x^*))$ has to be Pareto efficient.*

Proof. We will prove this lemma by contradiction. Let $(u^*_1, \cdots, u^*_M)$ not be Pareto efficient then there exists a feasible x' such that for every $i \in \{1, \cdots, M\}$, $u'_i = f_i(x')$, we have $u'_i \leq u^*_i$ and there exists an $l \in \{1, \cdots, M\}$ such that $u'_l < u^*_l$. From lemma (3) we know that the algorithm will reach x^* after at most $2M^2$ iterations. Let j be the smallest index such that $f_j(x^*) > f_j(x')$ and $k > 2M^2$ be the smallest index k where at kth iteration the algorithm minimizes $f_j(x)$. By lemma 3 we have $u^{k-1}_i = u^*_i$ but then $u^k_j < u^*_j = u^{k-1}_j$ hence the algorithm can not converge to x^* which is a contradiction. □

Now what remains to show is that the vector $(u^*_1, \cdots, u^*_M) = (f_1(x^*), \cdots, f_M(x^*))$ which is converged by our algorithm is an feasibly fair value vector.

Lemma 5. *If we denote the point that the algorithm converges by x^* then $(u^*_1, \cdots, u^*_M) = (f_1(x^*), \cdots, f_M(x^*))$ has to be Feasibly Fair.*

Proof. We will again prove by contradiction. Assume that there exists agents $l_1, l_2 \in \{1, \cdots, M\}$ where $u^*_{l_1} > u^*_{l_2}$ and a point x' such that $f_{l_1}(x') < u^*_{l_1}$, $f_{l_2}(x') \leq u^*_{l_1}$, $\forall k \neq l_1, l_2$ $f_k(x') \leq u^*_k$. We will focus on the line segment connecting x^* and x' and denote it by $p = \{\lambda x^* + (1-\lambda)x' \| \lambda \in [0,1]\}$. Since all functions are strictly quasi-convex then for every $x \in p$ we have $\forall k \neq l_1, l_2$ $f_k(x) < u_k$.

If $f_{l_1}(x') < f_{l_2}(x')$, since $f_{l_1}(x^*) > f_{l_2}(x^*)$ and $f_i(x)$ are continuous, then there exists a $z \in p$ such that z is in the relative interior of p and $f_{l_1}(z) = f_{l_2}(z)$. Since

functions are strictly quasi-convex we have $u^*_{l_1} > f_{l_1}(z) = f_{l_2}(z)$. Let $k > 2M^2$ be an iteration where $f_{l_1}(x)$ is minimized then we are going to get $f^{k+1}_{l_1} \leq f_{l_1}(z) < u^*_{l_1}$. Clearly the algorithm can not converge to x^* hence a contradiction.

If $f_{l_1}(x') \geq f_{l_2}(x')$, since we have $u^*_{l_1} > f_{l_1}(x')$, at the iteration where $f_{l_1}(x)$ is minimized, we are going to get $f^k_{l_1} \leq f_{l_1}(x') < u^*_{l_1}$ which shows that the algorithm can not converge to x^* hence a contradiction. $\square$

5 Conclusion

We have defined an extended notion for fairness and gave an easy to implement algorithm. Although we have not given empirical results we have given a comprehensive analysis of the algorithm and showed that it is guaranteed to converge in $2M^2$ iterations. The optimization step of our algorithm (step 3) can be efficiently solved via bisection since the functions are quasi-convex and the remaining steps are fairly straightforward. The algorithm can be seen to work fast for small problems and as future work algorithm will be implemented for some popular problems and empirical results will be given.

References

[1] Fudenberg, Triole *Game Theory.* The MIT Press, 1991.

[2] Altannar Chinchuluun and Panos M. Pardalos *A survey of recent developments in multiobjective optimization* Annals of Operations Research Volume 154, Number 1 (2007).

[3] R.T. Marler and J.S. Arora *Survey of multi-objective optimization methods for engineering* Struct Multidisc Optim 26, 369395 (2004).

[4] Robert J. Aumann and Michael Maschler *Game Theoretic Analysis of a Bankruptcy Problem from the Talmud** JOURNAL OF ECONOMIC THEORY 36, 195-213 (1985).