Proceedings of the Fields–MITACS Industrial Problems Workshop, 2006

Normalization and Other Topics in Multi-Objective Optimization

Problem Presenter: Helmut Mausser (Algorithmics Group)

Academic Participants: Yichuan Ding (University of Waterloo), Sandra Gregov (McMaster University), Oleg Grodzevich (University of Waterloo), Itamar Halevy, Zanin Kavazovic (Université de Laval), Oleksandr Romanko (McMaster University), Tamar Seeman (Bar Ilan University), Romy Shioda (University of Waterloo), Fabien Youbissi (Université de Laval)

Report prepared by: Oleg Grodzevich[1], Oleksandr Romanko[2]

1 Introduction

A *multi-objective optimization* typically arises in various engineering modelling problems, financial applications, and other problems where the decision maker chooses among several competing objectives to satisfy (see, e.g. [5]). In this report we consider a multi-objective optimization problem that comes from the financial sector. However, the analysis is applicable to any problem that retains similar characteristics. In particular, we focus on techniques for the normalization of objective functions. The normalization plays an important role in ensuring the consistency of optimal solutions with the preferences expressed by the decision maker. We also compare several approaches to solve the problem assuming that a linear or a mixed integer programming solver, such as CPLEX, is available. Namely, we consider weighted sum and hierarchical or ε-constraint methods, see also [1, 3, 6].

Although this report tends to provide a general discussion about multi-objective optimization, the proposed analysis and the resulting algorithm are specifically aimed at practical implementation. The reader should keep in mind that different approaches may be more effective should one not be constrained by various factors that are induced by the environment, and that are not discussed within the scope of this paper.

A multi-objective optimization problem can be written in the following form

$$\begin{array}{rl} \min & \{f_1(x), f_2(x), \ldots, f_k(x)\} \\ \text{s.t} & x \in \Omega, \end{array} \tag{1.1}$$

[1] ogrodzev@uwaterloo.ca
[2] romanko@mcmaster.ca

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are (possibly) conflicting objective functions and $\Omega \subseteq \mathbb{R}^n$ is the feasible region.

For consistency, we transform all the maximization problems of the type $\max f_i$ into equivalent minimization problems $\min(-f_i)$.

The goal of multi-objective optimization is to simultaneously minimize all of the objective functions. In this paper we restrict our attention mainly to the case of convex objectives and convex feasible region. The situation when integrality restrictions are present is also briefly addressed. We further consider only linear or convex quadratic objectives and constraints. In general we distinguish among the following three types of problems:

- linear – with linear objectives and constraints;
- quadratic – with linear and quadratic objectives, but only linear constraints;
- quadratic-quadratic – with both objectives and constraints being either linear or quadratic.

Define the set $Z \subseteq \mathbb{R}^k$ as the mapping of the feasible region into the objective space and denote it as *objective feasible region*:

$$Z = \{z \in \mathbb{R}^k \,:\, z = ((f_1(x), f_2(x), \ldots, f_m(x))^T \,\forall\, x \in \Omega)\}.$$

Since we assume that objective functions compete (or conflict) with each other, it is possible that there is no unique solution that optimizes all objectives simultaneously. Indeed, in most cases there are infinitely many optimal solutions. An optimal solution in the multi-objective optimization context is a solution where there exists no other feasible solution that improves the value of at least one objective function without deteriorating any other objective.

This is the notion of *Pareto optimality* [1, 2, 4, 5, 6]. Specifically, a decision vector $x^* \in \Omega$ is *Pareto optimal* if there exists no another $x \in \Omega$ such that $f_i(x) \leq f_i(x^*)\ \forall i = 1, \ldots, k$ and $f_j(x) < f_j(x^*)$ for at least one index j. The vector of objective function values is Pareto optimal if the corresponding decision vector x is Pareto optimal. The set of Pareto optimal solutions $\mathcal{P}$ forms a Pareto optimal set, which is also known as the *efficient frontier*, see [4].

The definition above refers to global Pareto optimality. In addition, local Pareto optimal solutions can be defined if points in the neighbourhood of an optimal solution are considered (rather than considering all points in the feasible region). Any global Pareto optimal solution is locally Pareto optimal. The converse is true for problems which feature convex Pareto set. More specifically, if the feasible region is convex and objective functions are quasi-convex with at least one strictly quasi-convex function, then locally Pareto optimal solutions are also globally Pareto optimal, see [6].

2 Decision making with multi-objective optimization

From the mathematical point of view, every Pareto optimal solution is equally acceptable as the solution to the multi-objective optimization problem. However, for practical reasons only one solution shall be chosen at the end. Picking a desirable point out of the set of Pareto optimal solutions involves a *decision maker* (DM). The DM is a person who has insights into the problem and who is able to express preference relations between different solutions. In the case of Algorithmics Inc., the DM is the customer running their software.

A process of solving a multi-objective optimization problem typically involves the co-operation between a decision maker and an analyst. The analyst in our situation is represented by a piece of software that is responsible for performing mathematical computations required during the solution process. This analytical software generates information for

the decision maker to consider and assists in the selection of a solution by incorporating preferences expressed by the DM. For example, the DM can assign importance levels, such as "high", "medium", or "low", to each objective or rank objectives in some specific order.

In the context of this report we seek to find a solution that is both Pareto optimal and also satisfies the decision maker. Such a solution, providing one exists, is considered a desired solution to the multi-objective optimization problem and is denoted as a *final solution*.

3 Numerical example

A small portfolio optimization problem is used to test and to illustrate the multi-objective optimization methodology. In portfolio optimization, investors need to determine what fraction of their wealth to invest in which stock in order to maximize the total return and minimize the total risk. In our experiments, the data includes expected returns, return covariances and betas for 8 securities, as well as their weights in the initial portfolio x_0. If we define our decision variables to be the weights of the securities x then expected return $r^T x$ and beta $\beta^T x$ are linear functions and variance of return $\frac{1}{2}x^T Q x$ is a quadratic function.

We put box constraints on the weights x ($0 \leq x \leq 0.3$) and use three objectives:

1) minimize the variance of return;
2) maximize the expected return;
3) set a target beta of 0.5 and penalize any deviation from this target.

Moreover, we also need to add a constraint that makes the weights sum to 1. The data for the problem is presented in Tables 1 and 2.

Security	x_0	$E(Return)$	Beta
1	0	0.07813636	0.1
2	0.44	0.09290909	0
3	0.18	0.11977273	0.7
4	0	0.12363636	0.5
5	0	0.12131818	0.3
6	0.18	0.09177273	0.25
7	0.13	0.14122727	0.4
8	0.07	0.12895455	-0.1

Table 1 Portfolio data

Security	1	2	3	4	5	6	7	8
1	0.000885	-8.09E-05	9.99E-05	5.80E-05	-0.000306	0.000261	-0.001255	0.000803
2	-8.09E-05	0.022099	0.010816	0.010107	0.011279	0.010949	0.010534	-0.013429
3	9.99E-05	0.010816	0.026997	0.028313	0.031407	0.007148	0.020931	-0.017697
4	5.80E-05	0.010107	0.028313	0.030462	0.035397	0.006782	0.022050	-0.015856
5	-0.000306	0.011279	0.031407	0.035397	0.047733	0.007278	0.023372	-0.015692
6	0.000261	0.010949	0.007148	0.006782	0.007278	0.006194	0.004195	-0.010970
7	-0.001255	0.010534	0.020931	0.022050	0.023372	0.004195	0.052903	-0.013395
8	0.000803	-0.013429	-0.017697	-0.015856	-0.015692	-0.010970	-0.013395	0.121308

Table 2 Return Covariances Matrix

Thus, the multi-objective portfolio optimization problem looks like:

$$\begin{array}{ll} \min & f_1(x) = -r^T x, f_2(x) = |\beta^T x - 0.5|, f_3(x) = \frac{1}{2} x^T Q x \\ \text{s.t} & \sum_i x_i = 1, \\ & 0 \leq x_i \leq 0.3 \ \forall i. \end{array}$$

Let us rewrite the beta constraint as $\beta^T x + t_1 - t_2 = 0.5$, in this case $f_2(x) = t_1 + t_2,\ t_1 \geq 0, t_2 \geq 0$. We get the following problem:

$$\begin{array}{ll} \min\limits_{x,t} & f_1(x,t) = -r^T x, f_2(x,t) = t_1 + t_2, f_3(x,t) = \frac{1}{2} x^T Q x \\ \text{s.t} & \sum_i x_i = 1, \\ & \beta^T x + t_1 - t_2 = 0.5, \\ & 0 \leq x \leq 0.3,\ t \geq 0. \end{array} \tag{3.1}$$

Suppose that

$$\Big(f_1(x^*), f_2(x^*), f_3(x^*)\Big) = (-12, 0.1, 26)$$

and

$$\Big(f_1(x'), f_2(x'), f_3(x')\Big) = (-5, 0.1, 15)$$

are both Pareto optimal solutions. If the DM prefers the first objective over the third, then the DM may prefer solution x', whereas he may prefer x^* if the opposite scenario holds. The challenge in this multi-objective portfolio optimization problem is to find the Pareto optimal point that meets the DM's given preferences. We propose to focus on two approaches: the weighted sum approach outlined in Section 4 and the hierarchical approach discussed in Section 5.

4 The weighted sum method

The weighted sum method allows the multi-objective optimization problem to be cast as a single-objective mathematical optimization problem. This single objective function is constructed as a sum of objective functions f_i multiplied by weighting coefficients w_i, hence the name. These coefficients can be normalized to 1, while this is not necessary in general.

4.1 Basics of the weighted sum method. In the weighted sum method the problem (1.1) is reformulated as:

$$\begin{array}{ll} \min & \sum\limits_{i=1}^{k} w_i f_i(x) \\ \text{s.t} & x \in \Omega, \end{array} \tag{4.1}$$

where $w_i \geq 0$, $\forall i = 1, \ldots, k$ and $\sum_{i=1}^{k} w_i = 1$.

Under the convexity assumptions, the solution to (4.1) is Pareto optimal if $w_i > 0$, $\forall i = 1, \ldots, k$. The solution is also unique if the problem is strictly convex.

In principle, every Pareto optimal solution can be found as a solution to (4.1), if convexity holds. However, as we will see in Section 4.4, depending on the problem geometry and the solution method some of the Pareto optimal solutions can never be obtained.

4.2 Normalization in the weighted sum method. Ideally, weights of each objective function are assigned by the DM based on the intrinsic knowledge of the problem. However, as different objective functions can have different magnitude, the normalization of objectives is required to get a Pareto optimal solution consistent with the weights assigned by the DM. Hence, the weights are computed as $w_i = u_i\theta_i$, where u_i are the weights assigned by the DM and θ_i are the normalization factors.

Some possible normalization schemas are:

- normalize by the magnitude of the objective function at the initial point x_0, here $\theta_i = \frac{1}{f_i(x_0)}$;
- normalize by the minimum of the objective functions, $\theta_i = \frac{1}{f_i(x^{[i]})}$, where $x^{[i]}$ solves $\min_x\{f_i(x) : x \in \Omega\}$;
- normalize by the differences of optimal function values in the Nadir and Utopia points that give us the length of the intervals where the optimal objective functions vary within the Pareto optimal set (details are provided below).

The first two schemas have proved to be ineffective and are not practical. The initial point may provide very poor representation of the function behaviour at optimality. Moreover, $f_i(x_0)$ is often equal to zero and can not be used at all. Use of the optimal solutions to individual problems can also lead to very distorted scaling since optimal values by themselves are in no way related to the geometry of the Pareto set.

Let us consider the last normalization schema in more details. It is not difficult to see that ranges of the objective Pareto optimal set provide valuable information for the solution process. The components $z_i^* = f_i(x^{[i]}) \in R$ of the ideal objective vector $z^* \in R^k$ are obtained by minimizing each of the objective functions individually subject to the original constraints, i.e.,

$$z_i^* = f_i(x^{[i]}) \qquad \text{where} \quad z_i^* = \operatorname{argmin}_x\{f_i(x) : x \in \Omega\}.$$

The ideal objective vector $z^U = z^*$, called the *Utopia point*, is not normally feasible because of the conflicting nature of the individual objectives. The Utopia point provides the lower bounds of the Pareto optimal set.

The upper bounds of the Pareto optimal set are obtained from the components of a *Nadir point* z^N. These are defined as

$$z_i^N = \max_{1\le j\le k}(f_i(x^{[j]})),\ \forall i = 1,\dots,k.$$

The normalization schema that uses the differences of optimal function values in the Nadir and Utopia points gives the following values of θ_i,

$$\theta_i = \frac{1}{z_i^N - z_i^U}.$$

This normalization schema provides the best normalization results as we normalize the objective functions by the true intervals of their variation over the Pareto optimal set. Intuitively, it is not difficult to see that all objective functions after normalization will be bounded by

$$0 \le \frac{f_i(x) - z_i^U}{z_i^N - z_i^U} \le 1,$$

that gives the same magnitude to each objective function.

4.3 Computing normalization weights. In order to compute true normalization ranges $z_i^N - z_i^U$ it is necessary to solve k optimization problems of the form $\min_x\{f_i(x) : x \in \Omega\}$ to obtain $x^{[i]}$ values.

Knowing $x^{[i]}$ is crucial, as they are required to calculate $z_i^U = f_i(x^{[i]})$ and $z_i^N = \max_j(f_i(x^{[j]}))$.

In practise, it may be computationally expensive to solve k optimization problems if they are mixed integer programming problems or quadratically constrained problems. Even solving k linear or convex quadratic problems can take a significant amount of time if the problem dimensions are large (more than 10,000-100,000 variables/constraints).

On the other hand, it is evident that exact solutions to the individual optimization problems are not required. One can come up with acceptable normalization factors if the estimates of the Utopia and Nadir points are known. For that reason, we propose the following relaxations or modifications to cope with the expensive computational costs.

- Tweak CPLEX parameters to reduce the solution time, i.e. increase one or more of the stopping criteria tolerances, e.g. increase the duality gap in the barrier solver from 10^{-6} to 10^{-3} or 10^{-4}.
- Relax some of or all "difficult" constraints, i.e. relax all quadratic and integer constraints.
- Estimate z_i^U and z_i^N without solving any optimization problem, by relying instead on a random sampling of points x subject to a subset of problem constraints Ω (such as, for example, box-constraints $l_i \leq x_i \leq u_i$). Calculate z_i^U and z_i^N as the minimum and the maximum over the set of sampled points.
- If known, use the initial solution x_0 for normalization when all other methods are still computationally expensive.
- In certain cases, there is a closed form solution for z_i^U. For example, in Problem (3.1), $z_i^U = -0.3(r_{(1)} + r_{(2)} + r_{(3)}) - 0.1r_{(4)}$ where $r_{(i)}$ is the i^{th} order statistic of r_i[2]. The corresponding solution is $x_{(1)} = 0.3, x_{(2)} = 0.3, x_{(3)} = 0.3, x_{(4)} = 0$, and all other $x_i = 0$, which can be used to find z_i^N.

  This result is due to the fact that the constraints in the problem define the standard simplex.

  Similarly, $z_2^U = 0.5 - \beta_{(1)}$, if $\beta_{(1)} < 0.5$ (and the corresponding optimal solution will be $x_{(1)} = 1$ and $x_i = 0, \forall i \neq (1)$), $z_2^U = \beta_{(8)} - 0.5$, if $\beta_{(8)} > 0.5$ (and the corresponding optimal solution will be $x_{(8)} = 1$ and $x_i = 0, \forall i \neq (8)$), and $z_2^U = 0$ otherwise (in this case, if $\beta_k \leq 0.5 \leq \beta_j$, then $x_j = \frac{0.5-\beta_k}{\beta_j-\beta_k}$, $x_k = \frac{\beta_j-0.5}{\beta_j-\beta_k}$ and $x_i = 0, \forall i \neq j, k$.) Clearly, there can be multiple optimal solutions.

4.4 Weaknesses of the weighted sum method. The weighted sum method has a major weakness when objectives are linear and a simplex-type method is used to solve the weighted sum problem. As we noted earlier, the specific Pareto optimal solution can be obtained by the proper choice of weights. However, if the objective Pareto optimal set features a linear face, the simplex method will always pick up a vertex as the possible solution. There does not exist a set of weights that yield points in the interior of the linear face as long as the simplex method is considered. Different solution techniques, such as interior-point methods, may overcome this disadvantage.

[2]The i^{th} order statistic of r_i is the i^{th} largest value in $\{r_1, \ldots, r_n\}$, i.e., $r_{(1)} \geq r_{(2)} \geq \cdots \geq r_{(n)}$.

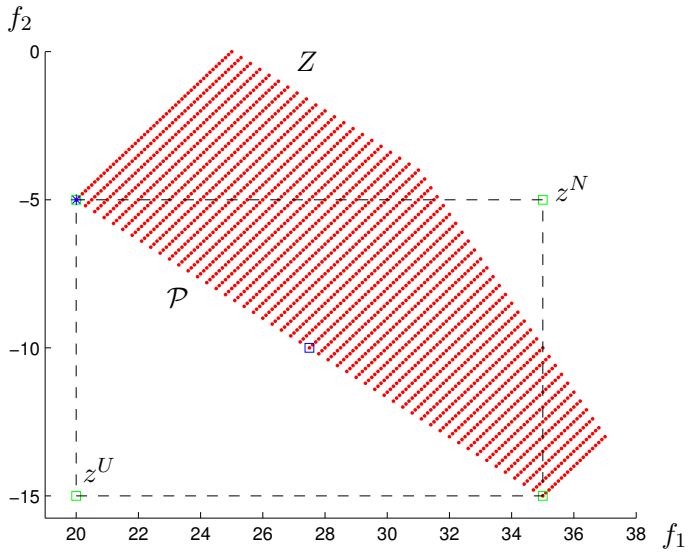

Figure 1 Objective Feasible Region: Linear Case

In order to better understand this situation, let us consider a small example with two linear objective functions subject to a polyhedral feasible region,

$$\begin{array}{rl} \min & \{w_1(3x_1 + x_2), w_2(-2x_1 + x_2)\} \\ \text{s.t} & x_1 + x_2 \leq 17, \\ & 5 \leq x_1 \leq 10, \\ & 5 \leq x_2 \leq 10. \end{array}$$

Figure 1 displays the objective feasible region Z (shaded polyhedron) for this problem. It also shows Utopia z^U and Nadir z^N points, as well as the Pareto efficient frontier $\mathcal{P}$.

Solving the above problem with the simplex-type algorithm yields an optimal solution which is either one of two points (z_1^N, z_2^U) or (z_1^U, z_2^N). The choice of weights defines which one will be obtained, and the jump occurs for some values w_1 and w_2 which depend on the tangency ratio $\frac{w_1}{w_2}$. However, it is not possible to escape these corner solutions, as no other point in the Pareto set can be obtained as a solution of the weighted sum problem.

This deficiency of the weighted sum method can seriously puzzle the decision maker as the results obtained by the analytical software may be not in line with his or her expectations. In fact, the following anomalies may be observed:

- sensitivity issues – an optimal solution experiences huge jumps when weights are changed slightly, but at the time is unaffected by broad changes in weights;
- optimality issues – optimal solutions are corner solutions on the Pareto optimal set, which usually are not practically desirable solutions.

These shortcomings are especially evident if we have only linear objective functions that yield an objective feasible region like the one in Figure 1. If objective functions are all quadratic these problems may not be encountered, since the efficient frontier is likely to be non-linear as in Figure 2.

One way to overcome the linearity of a Pareto face is to square some of the linear objective functions and to keep the quadratic ones. To understand this idea we look at

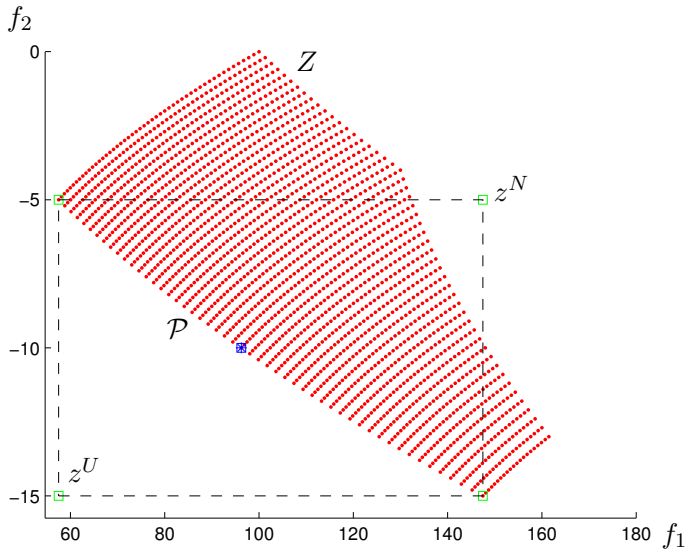

Figure 2 Objective Feasible Region: Non-Linear Case

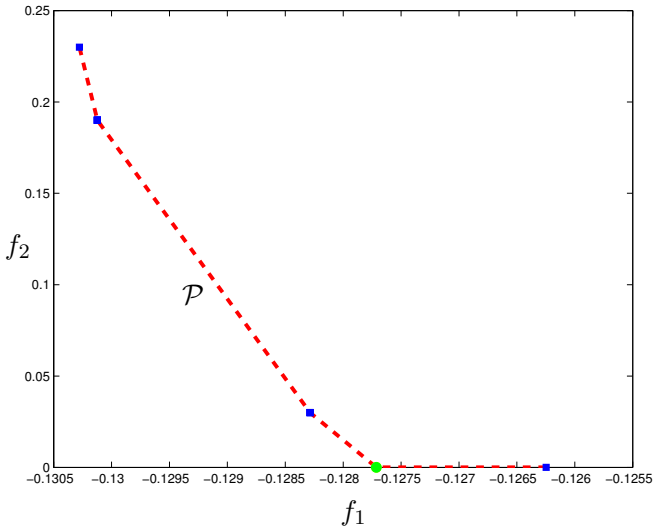

Figure 3 Pareto Optimal Set: Illustrative Portfolio Problem

the illustrative portfolio problem (3.1). In that problem the objective functions f_1 and f_2 are linear, while f_3 is quadratic. Figure 3 shows the Pareto optimal set in the (f_1, f_2) plane. Squares denote the corners of the set, i.e. solutions we can possibly obtain with the weighted sum approach.

We can square the objective function f_2 since

$$\min f_2(x) = \min |\beta^T x - 0.5| = \min f_2^2(x) = \min(\beta^T x - 0.5)^2.$$

Using $f_2^2(x)$ instead of $f_2(x)$ and keeping f_1, f_3 as before, we come up with a more populated Pareto set, as depicted in Figure 4.

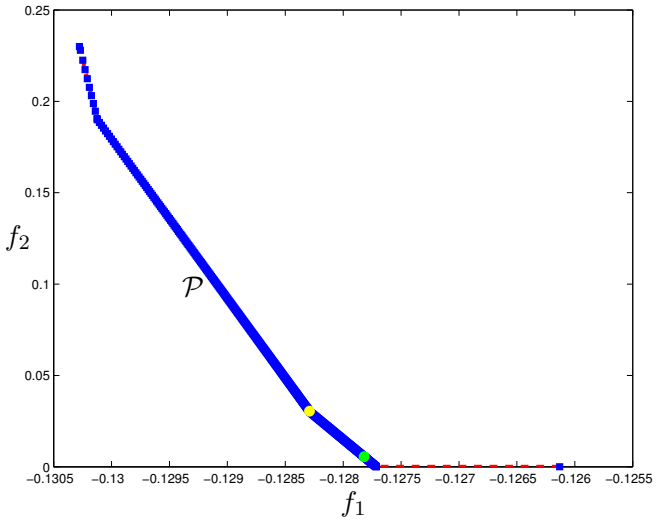

Figure 4 Pareto Optimal Set: Squared Linear Function f_2

Clearly, we cannot square all linear objective functions as squaring some of them may make no sense for the original multi-objective problem formulation. The above discussion illustrates that reformulating the problem may sometimes help in overcoming the weaknesses of the weighted sum method, while keeping the complexity roughly the same.

Another approach that does not suffer from problems experienced by the weighted sum method is the hierarchical (ε-constraint) method, described in the next section.

5 The hierarchical method

This method allows the decision maker to rank the objective functions in a descending order of importance, from 1 to k. Each objective function is then minimized individually subject to a set of additional constraints that do not allow the values of each of the higher ranked functions to exceed a prescribed fraction of their optimal values obtained on the corresponding steps.

In other words, once the optimal value for the specific objective has been obtained, the possible values this objective function can take on the following steps are restricted to a box (or a ball) around this optimal solution. Depending on the size of the restriction this strategy effectively puts more weight on the higher ranked objective functions.

5.1 Basics of the hierarchical method. For illustration purposes, we consider the problem with two objective functions. Suppose that f_2 has higher rank than f_1. We then solve,

$$\begin{aligned} \min \quad & f_2 \\ \text{s.t} \quad & x \in \Omega \end{aligned}$$

to find the optimal objective value f_2^*.

Next we solve the problem,

$$\begin{aligned} \min \quad & f_1 \\ \text{s.t} \quad & x \in \Omega, \\ & f_2(x) \leq f_2^* + \varepsilon. \end{aligned}$$

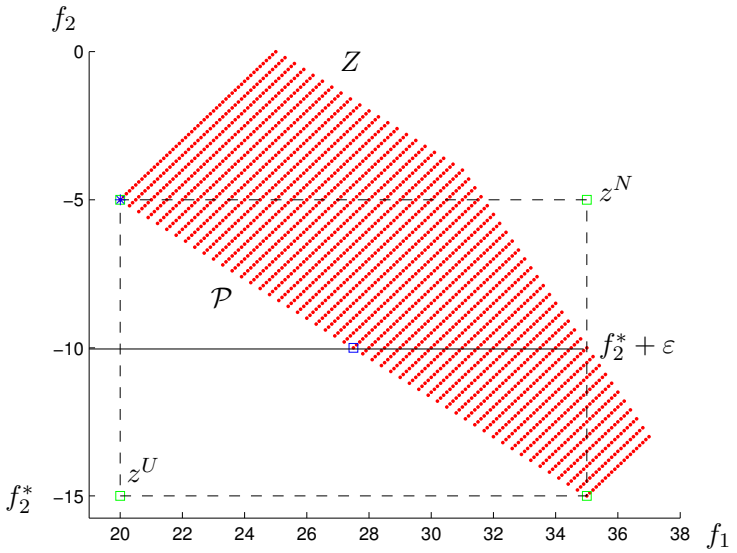

Figure 5 Objective Feasible Region: The Hierarchical Method Cut

Intuitively, the hierarchical method can be thought as saying "f_2 is more important than f_1 and we do not want to sacrifice more than 20% (or 30% or 50%) of the optimal value of f_2 to improve f_1."

In the general case of k objective functions and the strict preferential ranking f_1, f_2, …, f_k, we start by solving the problem

$$\begin{aligned} \min \quad & f_1 \\ \text{s.t} \quad & x \in \Omega, \end{aligned}$$

to obtain the optimal solution $x^{[1]}$. Next, for $j = 2, \ldots, k$, we find the optimal solution $x^{[j]}$ for the j-th objective function by solving the problem with additional constraints of the following form

$$f_l(x) \leq (1 + \varepsilon_l) f_l(x^{[l]}),\ \forall l = 1, \ldots, j-1.$$

If both Utopia and Nadir points are known, we can interpret ε_l as the distance from optimality in percents,

$$f_l(x) \leq \Big(1 + \varepsilon_l (z_l^N - z_l^U)\Big) f_l(x^{[l]}).$$

5.2 Application of the hierarchical method. The hierarchical method provides the decision maker with another way to specify the relative importance of the objective functions: instead of weighting each of them, the DM ranks them and specifies how much of the more important objectives can be traded in order to improve the less important ones.

The hierarchical method avoids the pitfalls of the weighted sum method as it starts with the corner solution and attempts to move away towards the centre of the Pareto set.

Figure 5 shows that the constraint $f_2(x) \leq (1+\varepsilon_2) f_2(x^{[2]})$ introduces the cut that forces the optimal solution to move out of the corner.

One of the drawbacks of the hierarchical method is the introduction of a quadratic constraint on subsequent steps when the previous problem has a quadratic objective function. Quadratic constraints are generally very hard to deal with and, thus, should be avoided.

However, as we already discussed in Section 4.4, the problem with purely quadratic objectives usually features non-linear efficient frontier and can be tackled by the weighted sum approach. This leads us to the following proposition:

1) if linear objective functions are present, rank and apply the hierarchical method to eliminate them from the problem, i.e. replace these functions with the corresponding cuts;
2) use the weighted sum method to solve the remaining problem where only quadratic objectives are present.

These ideas are key to the algorithm we present in the next section.

6 The algorithm

We propose the following algorithm to solve the general convex multi-objective optimization problem with linear and quadratic objectives. The algorithm combines both weighted sum and hierarchical approaches and requires only linear or quadratic programs to be solved at each step.

1. Separate linear and quadratic objectives into two index sets $\mathcal{L}$ and $\mathcal{Q}$, denote linear objectives as f^L and quadratic objectives as f^Q.
2. Rank the linear objective functions in $\mathcal{L}$ from the most important to the least important.
3. Solve (or approximate if possible) each of
$$\min_x \{f_i(x) \,:\, x \in \Omega\}.$$
Let x_i^* denote an optimal solution and f_i^* be the optimal objective value. Compute the Nadir vector z^N.
4. Sequentially solve problems
$$\begin{array}{ll} \min & f_i^L \\ \text{s.t} & x \in \Omega, \\ & f_l^L(x) \le f_l^{L^*} + \delta_l, l = 1, \ldots, i-1, \\ & \delta_l = \varepsilon_l(z_l^N - f_l^{L^*}). \end{array}$$
One may also consider updating elements of z^N by considering optimal solutions to these problems, as they become available.
5. If no quadratic objectives are present, return the last optimal solution as the final solution; otherwise proceed to the next step.
6. Compute scaling coefficients θ_i for objectives in $\mathcal{Q}$ and set $w_i = u_i\theta_i$, where u_i are the decision maker preferences.
7. Solve for the final solution
$$x^* = \text{argmin}_x \Big\{ \sum_{q \in \mathcal{Q}} w_q f_q^Q \,:\, x \in \Omega, f_l^L \le f_l^{L^*} + \delta_l, \forall l \in \mathcal{L} \Big\}.$$

6.1 Choosing weights based on the geometrical information. In the fourth step of the algorithm, we aim to solve the sequential program $P_i = \{\min f_i^L \,:\, x \in \Omega, f_l^L(x) \le f_l^{L^*} + \varepsilon_l(z_l^N - f_l^{L^*}), l = 1, \ldots, i-1\}$ to get the optimal value $f_i^{L^*}$. Each time we obtain an optimal objective value $f_i^{L^*}$, the constraint
$$f_i^L(x) \le f_i^{L^*} + \varepsilon_i(z_i^N - f_i^{L^*})$$

is added in succeeding programs to cut off a portion of the feasible region where the i-th objective can't be well achieved. A challenging problem is to decide the proportion we want to cut off. If we choose a small ε_i, we may overly favour the i-th objective ($\frac{f_i(x^*) - f_i^*}{z_i^U - f_i^*} \leq \varepsilon_i$), while sacrificing subsequent objectives; in contrast, choosing a large ε_i may result in a poor i-th objective, but leaves more space to optimize other objectives. In the multi-objective optimization it is important to develop a fair and systematic methodology to decide on ε_i according to the given information.

The choice of ε_i should depend on the importance of objective f_i^L and the potential conflict between objective f_i^L and other objectives. The importance of f_i^L could be represented by the priority factor u_i, which is scaled to satisfy $\sum_i u_i = 1$. However, there is no explicit indicator measuring the conflicts among different objectives.

We are motivated to measure the conflict based on the geometry of the objective functions. In the second step of the algorithm, we have computed $x_i^* \in \mathbb{R}^n$ for each objective i. Next, a reasonable choice of the centre is the bycentre $x^C = \frac{1}{n}\sum_i x_i^*$. Clearly, there are many choices of the centre, e.g. the analytical centre of the polyhedral feasible set. But the bycentre of $x_i^*, i = 1, 2, \ldots, k$ is the easiest to compute.

We propose a metric for the conflict between objectives i and j. Let us call it the *conflict indicator*, denoted by c_{ij}. We first measure the cosine value of the angle θ_{ij} between the vectors from $x_i^* - x^C$ and $x_j^* - x^C$:

$$\cos\theta_{ij} = \frac{< x_i^* - x^C, x_j^* - x^C >}{\|x_i^* - x^C\|\|x_j^* - x^C\|}.$$

As discussed above, x^C denotes the point we set to approximate the centre of the feasible set. We define c_{ij} as

$$c_{ij} = \frac{1}{2}(1 - \cos\theta_{ij}).$$

Thus, the larger the angle between the two objectives, the bigger the conflict indicator c_{ij} is. The conflict indicator equals zero when the optimizer of the two objectives coincides; and it is close to one when the two objectives tend to conflict the most. Note that $c_{ii} = 0$.

We propose to set $\varepsilon_i = \alpha\sum_{j=1}^k u_j c_{ij}$, where α is some scalar factor which could be decided by numerical experiment. The weighted sum has two significance: if important objectives conflict with the i-th objective, i.e. they have high priority, then we do not want to impose strict constraints on the achievement of objective f_i^L because it tends to restrict the important objectives, and thus we select big ε. On the other hand, if objectives conflicting with i-th objective are all of low priority, even if there are many conflicts, we can still impose a small ε to the objectives. If i-th objective has a high priority u_i, then the weighted sum of conflict indicators tends to be low.

Writing it mathematically, we denote by u the vector of priority factors:

$$u = \{u_1, u_2, \ldots, u_k\}^T,$$

and let $X \in \mathbb{R}^{n,k}$ be

$$X := \{x_1^*, x_2^*, \ldots, x_k^*\}.$$

Then, we get a vector $\varepsilon = (E - X^T X)u$, which carries ε_j for each objective j. We could choose to compute ε once in the beginning or update x_j^* after each iteration of the sequential program and use this information to compute ε.

This technique could be generalized to the quadratic constraints, quadratic objective case (QCQP). For this case, if all quadratic constraints are convex, we could linearize the functions by replacing non-negative constraints with positive semi-definite constraints. This will result in a semi-definite programming problem to compute x_j^*, define the bycentre, and then compute conflict indicators c_{ij}. Thus, we can solve the sequential QCQP program and compute ε as one problem.

7 Conclusions and future work

We have reviewed multi-objective optimization problems and normalization methods – weighted sum and hierarchical approaches. Both methods are analyzed and their advantages, as well as disadvantages, are discussed. Based on the analysis, we propose an algorithm to solve convex multi-objective problems with linear and quadratic objectives. A version of the algorithm was implemented in MATLAB and tested on the financial problem discussed in Section 3.

This paper provides a lot of insight on how to attack and what to expect from these kinds of problems. We accept the fact that analysis is rather sketchy, but we feel that the algorithmic approach and, especially, ideas expressed in Section 6.1 may be innovative and of further interest. Since we did not have time to perform an extensive literature review, we should stress that these or related ideas might be already known or better developed elsewhere. We encourage an interested reader to consult numerous publications on multi-objective optimization.

8 Acknowledgements

We thank the organizers of Fields-MITACS Industrial Problem Solving Workshop and the Algorithmics Inc. for giving us the opportunity to work on this problem.

References

[1] R. Kahraman and M. Sunar (2001), A comparative study of multiobjective optimization methods in structural design. *Turkish Journal of Engineering and Environmental Sciences* 25: 69-78,

[2] I. Y. Kim and O. De Weck (2005), Adaptive weighted sum method for bi-objective optimization. *Structural and Multidisciplinary Optimization* 29(2):149-158.

[3] K. Klamroth and J. Tind (2006), Constrained Optimization using Multiple Objective Programming. Technical report. Accepted for publication in Journal of Global Optimization.

[4] P. Korhonen (1998), Multiple objective programming support. Technical Report IR-98-010, International Institute for Applied Systems Analysis, Laxenburg, Austria.

[5] R. Timothy Marler. *A Study of Multi-Objective Optimization Methods for Engineering Applications.* PhD thesis, The University of Iowa.

[6] Kaisa Miettinen (2001), Some methods for nonlinear multi-objective optimization. In Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), *Lecture Notes in Computer Science*, 1993:1–20.