Non-Archimedean Hypervolume in case of PC- and PL-MPL-MOPs

Minutella, Veltroni

1 Introduction

Evolutionary multi-objective optimization (EMO) has been an active research area in the last two decades. One important issue in this area is the performance evaluation of EMO algorithms. Since a set of non-dominated solutions is obtained by a single run of an EMO algorithm, performance evaluation in the EMO community usually means the comparison of different non-dominated solution sets. Various performance indicators have been proposed to evaluate the quality of a non-dominated solution set.

This project aims at investigating and implementing performance indicators for the following two classes of problems:

- **PC-MPL-MOPs**: Priority Chains Mixed Pareto Lexicographic Multi-objective Optimization Problems [1].
- **PL-MPL-MOPs**: Priority Levels Mixed Pareto Lexicographic Multi-objective Optimization Problems [2].

In order to evaluate performance, the most widely used indicator is *hypervolume* [3], primarily because none of the other indicators are Pareto compliant and, as often pointed out in the literature, they could lead to misleading results. However, hypervolume calculation needs a huge computation load for a large solution set of a many-objective problem.

This paper describes the implementation and revisiting of the hypervolume indicator for the classes of problems listed above. For PC-MOP-MPLs problems it is a very simple and straightforward implementation, while for PL-MOP-MPLs it is not trivial to find a solution.

After hypervolume, other performance indicators were considered for comparisons. The indicators implemented for BANs are *GD* and *IGD*, their modified versions, and *Averaged Hausdorff Distance* (Δ).

Remark. *Hereinafter, we will always refer to minimization problems.*

2 Multi-objective Optimization and Performance Indicators

This section recalls notions of multi-objective problems that will be recalled several times in this paper.

Definition 2.1. Let us consider two objective vectors $a = (a_1, a_2, ..., a_m)$ and $b = (b_1, b_2, ..., b_m)$, the Pareto dominance relation "$\succ$" and the weak Pareto dominance relation "$\succeq$" are defined for the minimization problem between them as follows:

Pareto Dominance: $a \succ b \Leftrightarrow \forall i, a_i \leq b_i$ and $\exists j, a_j < a_i$.

Weak Pareto Dominance: $a \succeq b \Leftrightarrow \forall i, a_i \leq b_i$.

The weak Pareto dominance relation $a \succeq b$ includes $a = b$ while it is excluded from the Pareto dominance relation $a \succ b$.

Definition 2.2. Let A be a set of objective vectors. When no objective vector in A is dominated by any other objective vector in A, A is called a non-dominated set. Let us denote two non-dominated sets of objective vectors as $A = \{a_1, a_2, ..., a_{|A|}\}$ and $B = \{b_1, b_2, ..., b_{|B|}\}$ where $|A|$ and $|B|$ are the cardinality of A and B, respectively.

Pareto Dominance for Sets: $A \succ B \Leftrightarrow \forall b_i \in B, \exists a_i \in A : a_i \succ b_j$

Weak Pareto Dominance for Sets: $A \succeq B \Leftrightarrow \forall b_i \in B, \exists a_i \in A : a_i \succeq b_j$

$A \succ B$ does not allow the existence of any shared objective vector in A and B. That is, $A \succ B$ requires $(A \cap B) = \emptyset$. Whereas $A \succ B$ does not allow any overlap between A and B, $A \succeq B$ allows $A = B$.

In order to handle partially overlapping sets, has been defined an intermediate relation called "**better**" denoted by "$\triangleright$" as follows:

Relation "better" for Sets: $A \triangleright B \Leftrightarrow A \succeq B$ and $A \neq B$

2.0.1 Hypervolume

The *hypervolume indicator* is a measure of the quality of a set $P = \{p_1, p_2, ..., p_n\}$ of n non-dominated objective vectors produced in a run of a multi-objective optimizer , such as a multi-objective evolutionary algorithm. Assuming a minimization problem involving d objectives, this indicator consists in the measure of the hypervolume of the region which is simultaneously dominated by P and bounded above by a reference point $r \in \mathbb{R}^d$ such that $r \geq (\max_p p_1, ..., \max_p p_d)$, where $p = (p_1, ..., p_d) \in P \subset \mathbb{R}^d$, and the relation $\geq$ applies componentwise.

The *hypervolume* is favored because it captures in a single scalar both the closeness of the solutions to the optimal set and, to some extent, the spread of the solutions across objective space.

As mentioned in the introduction, the *hypervolume indicator* is very often used as a performance indicator because it is the only Pareto compliant one. The concept of the Pareto compliance of an indicator $I(\cdot)$ can be defined using this relation as follows:

Definition 2.3 (**Pareto Compliant Indicator**)**.** Whenever $A \triangleright B$ holds between two non-dominated sets A and B, $I(A) < I(B)$ always holds: $A \triangleright B \Rightarrow I(A) < I(B)$.

2.0.2 GD/IGD

Two other possible and well known metrics are the *generational distance* (GD) and the *inverted generational distance* (IGD), described in [4].
Both indicators evaluate the quality of a non-dominated objective vector set $A = \{a_1, a_2, ..., a_{|A|}\}$ with respect to the given reference point set $Z = \{z_1, z_2, ..., z_{|Z|}\}$ where $|Z|$ is the cardinality of Z.
The former is defined as:

$$GD(A) = \frac{1}{|A|}(\sum_{i=1}^{|A|} d_i^p)^{\frac{1}{p}}$$

where d_i is the Euclidean distance from a_i to its nearest reference point in Z, and p is an integer parameter (in this paper will always be considered p = 1).
The IGD indicator is the inverted form of the previous one and its definition is:

$$IGD(A) = \frac{1}{|Z|}(\sum_{j=1}^{|Z|} d_j^p)^{\frac{1}{p}}$$

where d_j is the Euclidean distance from z_j to its nearest objective vector in A.

2.0.3 GD$^+$/IGD$^+$

As demonstrated in [4], modifying the distance calculation in GD and IGD creates new indicators GD$^+$ and IGD$^+$ that are better than the former.
Using this new method of calculating the distance IGD$^+$ became weakly Pareto-compliant.

For this purpose an inferiority vector $d^+ = (d_1^+, d_2^+, ..., d_m^+)$ is defined as follow:

$$d_i^+ = \max\{a_i - z_i, 0\},\ i = 1, 2, ..., m$$

Using this vector, the calculation of the modified distance in the GD and IGD indicators becomes:

$$d^+(z, a) = \sqrt{{d_1^+}^2 + ... + {d_m^+}^2} = \sqrt{(\max\{a_1 - z_1, 0\})^2 + ... + (\max\{a_m - z_m, 0\})^2}$$

Combining the indicators described above we obtain two new indicators

$$\Delta(A) = \max(GD(A), IGD(A))$$

$$\Delta(A)^+ = \max(GD(A)^+, IGD(A)^+)$$

3 PC-MPL-MOPs

PC-MPL-MOPs are problems in which objectives are inserted in priority chains (PC), so PC is a sequence of objectives lexicographically ordered by importance. The general mathematical structure is:

$$\min \begin{bmatrix} \operatorname{lex\,min} \left[f(\mathbf{x})_1^{(1)},\ f(\mathbf{x})_1^{(2)},\ \ldots,\ f(\mathbf{x})_1^{(p_1)} \right] \\ \operatorname{lex\,min} \left[f(\mathbf{x})_2^{(1)},\ f(\mathbf{x})_2^{(2)},\ \ldots,\ f(\mathbf{x})_2^{(p_2)} \right] \\ \vdots \\ \operatorname{lex\,min} \left[f(\mathbf{x})_m^{(1)},\ f(\mathbf{x})_m^{(2)},\ \ldots,\ f(\mathbf{x})_m^{(p_m)} \right] \end{bmatrix}$$

Where m is the number of priority chains and $f_i^{(j)}$ is the j_{th} objective in the i_{th} chain. The reformulation to address this type of problem by means of Alpha theory is:

$$\min \begin{bmatrix} f(\mathbf{x})_1^{(1)} + \alpha^{-1} f(\mathbf{x})_1^{(2)} + \ldots + \alpha^{1-p_1} f(\mathbf{x})_1^{(p_1)} \\ f(\mathbf{x})_2^{(1)} + \alpha^{-1} f(\mathbf{x})_2^{(2)} + \ldots + \alpha^{1-p_2} f(\mathbf{x})_2^{(p_2)} \\ \vdots \\ f(\mathbf{x})_m^{(1)} + \alpha^{-1} f(\mathbf{x})_m^{(2)} + \ldots + \alpha^{1-p_m} f(\mathbf{x})_m^{(p_m)} \end{bmatrix}$$

As described in [1] the definition of Pareto Dominance per set (given in section 2) can be extended for this class of problems as well, being careful that the numbers will no longer be in $\mathbb{R}$ but in $\mathbb{E}$.

3.1 Hypervolume Indicator

The problem to be addressed at this point is to find, and subsequently implement, a definition of hypervolume indicator for PC-MPL-MOPs by adapting the standard definition to the new class of problems while still having to comply with the defintion of *Pareto Compliant Indicator* defined in 2.0.1.

The problem can be solved thanks to the transfer principle, since the hypervolume indicator is a measure of the region between sets of points and a reference point, we can simply extend its standard definition in $\mathbb{E}$ and apply it to PC problems; thus, we can define it as:

Definition 3.1. Given a point set $S \subset \mathbb{E}^d$ and a reference point $r \in \mathbb{E}^d$, the hypervolume indicator of S is the measure of the region weakly dominated by S and bounded above by r

$$H(S) = \Lambda(q \in \mathbb{E}^d \mid \exists p \in S : p \leq q \ \wedge \ q \leq r)$$

where $\Lambda(\cdot)$ denotes the Lebesgue measure.

3.1.1 Implementation

As reported in [3], Fonseca, Paquete and López-Ibañez's hypervolume algorithm is one of the most recommended from a complexity perspective, so we chose to implement this algorithm of course making all the necessary changes to make it work in $\mathbb{E}$.

The Fonseca, Paquete and López-Ibañez's algorithm [5] or *FPL* in short, is an algorithm based on *HSO* (Hypervolume by Slicing Objectives), which is a dimension-sweep approach, but improving it in several aspects. First, the set of non-dominated points is stored in a dedicated data structure throughout the operation of the algorithm; then, this data structure is extended to store intermediate hypervolume values.

Below a simplified pseudo code to follow the reasoning behind the *FPL* algorithm.

Algorithm 1 PC_MPL_MOPs Hypervolume

Discard dominated points of the front
Sort for each dimension in ascending order points of the front
Initialize the tree T
for each point in front **do**
    for each dimension in point **do**
        Save in T the next point
        Save in T the prev point
if $d = 2$ **then**
    for each point in front **do**
        $hvol = hvol + (point_x * (point_y - point.prev_y))$
if $d = 3$ **then**
    for each point in front **do**
        $area$ = *area of the projection in (x-y)-plane* ▷ Return to case d=2
        $hvol = hvol + area * (point_z - point.prev_z)$
if $d > 3$ **then**
    for each point in front **do**
        Recursively decrement the point dimension by 1 until the case d=3.
        Multiply the obtained volume by the distance between adjacent points in the following dimensions

The following are the main data structures and constructors implemented in Julia that are needed to make the hypervolume algorithm work:

```
mutable struct Node
    cargo
    next::Array{Node}
    prev::Array{Node}
    ignore
    area
    volume
    Node() = new()
    Node(c, n, p, i, a, v) = new(c, n, p, i, a, v)
end
Node(numberLists) = Node(nothing, fill(Node(), numberLists), fill(Node(), numberLists),
    0, fill(0η, numberLists), fill(0η, numberLists))
Node(numberLists, cargo) = Node(cargo, fill(Node(), numberLists), fill(Node(),
    numberLists), 0, fill(0η, numberLists), fill(0η, numberLists))
mutable struct MultiList
    numberLists
    sentinel::Node
end
function MultiList(numberLists)
    m = MultiList(numberLists, Node(numberLists))
    m.sentinel.next = fill(m.sentinel, numberLists)
    m.sentinel.prev = fill(m.sentinel, numberLists)
    return m
end
mutable struct HyperVolume
    referencePoint
    list
```

```
end
HyperVolume(referencePoint) = HyperVolume(referencePoint, [])
```

The following function is the one to call to get the value of the hypervolume calculated based on the reference point and front passed as an argument.

```
function compute(hv::HyperVolume, front::AbstractArray)
    relevantPoints = []
    dimensions = length(hv.referencePoint)
    for point in front
        if weaklyDominates(point, hv.referencePoint)
            Base.append!(relevantPoints, [point])
        end
    end
    if any(a -> a != 0, hv.referencePoint)
        for j in 1:length(relevantPoints)
            relevantPoints[j] .= relevantPoints[j] - hv.referencePoint
        end
    end
    preProcess!(hv, relevantPoints)
    bounds = fill(Ban(-Inf), dimensions)
    hyperVolume = hvRecursive!(hv, dimensions, length(relevantPoints), bounds)
    return hyperVolume
end
```

3.1.2 Testing

In order to test the algorithm implemented for the hypervolume a script in python was built using the library Pygmo thanks to which it was possible to create complex fronts (in 2, 3, 4 and 5 dimensions) whose points were then saved on a file and then passed to the function written above to calculate the hypervolume.

```
from pygmo import *
from matplotlib import pyplot as plt
import json
prob = dtlz(prob_id = 7, fdim = 5, dim=7)
alg = algorithm(nsga2(gen=30))
pop = population(prob, 100)
pop = alg.evolve(pop)
points = pop.get_f()
with open("test.json", "w") as outfile:
    json.dump(points.tolist(), outfile)
hv = hypervolume(points)
ref_point = [50, 50, 50, 50, 50]
hv_value = hv.compute(ref_point)
```

```
include("./hv.jl")
import JSON
points = JSON.parsefile("test.json")
hv = HyperVolume([50, 50, 50, 50, 50])
res = compute(hv, points)
```

The correct functionality of the algorithm has been verified comparing the result we obtained with the resulting value of the hypervolume algorithm implemented in the Pygmo library. In order to then test the algorithm with BANs, small fronts were constructed by hand.

3.1.3 Complexity

The complexity of the algorithm in the version of the paper [5] is $\mathcal{O}(n^{d-2} log\, n)$, assuming we have a hardware component that can handle BAN operations as simple operations , the algorithm we implement maintains the same complexity.

3.2 GD/IGD, GD$^+$/IGD$^+$ and Delta metric

In PC problems in addition to calculating hypervolume, thanks to the transfer principle we can also implement the other performance indicators such as GD and IGD, their modified versions and the Delta metric and then redefine them on a vectorial space over $\mathbb{E}$.

3.2.1 Implementation

```
using LinearAlgebra
include("./src/ArithmeticNonStandarNumbersLibrary/src/BAN.jl")
using .BAN
function distance(ei, set, plus = false, p::Integer = 1, igd = false)
    tmp = []
    for v in set
        Base.append!(tmp, [ei - v])
    end
    if plus
        for i in 1:length(tmp)
            if igd
                tmp[i] = -tmp[i]
            end
            tmp[i][tmp[i] .< 0] .= 0
        end
    end
    distances = []
    for v in tmp
        Base.append!(distances, LinearAlgebra.norm(v))
    end
    return min(distances...)
end
function gd(solutions, reference_point_set, plus = false, p = 1, igd = false)
    sum = 0
    for s in solutions
        sum += distance(s, reference_point_set, plus, p, igd)
    end
    return sum / length(solutions)
end
function igd(solutions, reference_point_set, plus = false, p = 1)
    return gd(reference_point_set, solutions, plus, p, true)
end
function delta(solutions, reference_point_set, plus = false)
    return max(gd(solutions, reference_point_set, plus), igd(solutions,
     reference_point_set, plus))
end
```

3.2.2 Testing

The correct functionality of the implemented algorithms was tested using the fronts of the paper [4] and comparing the presented results with those we obtained. The same approach as described in the hypervolume algorithm was followed for fronts constructed with BANs.

3.3 Comparison of indicators

Comparisons of the various performance indicators are shown in the following examples. The tables below each graph show, for each indicator and for each set of points, whether the result obtained is consistent, i.e., the Pareto dominance is respected, or inconsistent.

3.3.1 Exemple 1

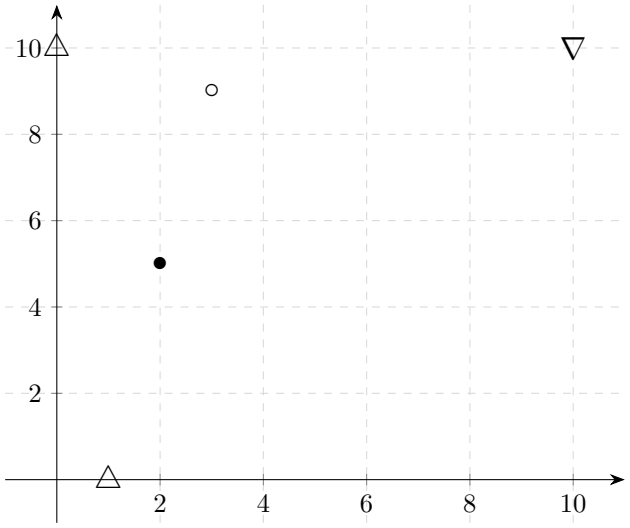

Figure 1

• Solution Set A = $\{(2 + 4\eta,\ 5 + 5\eta)\}$
◦ Solution Set B = $\{(3 + 4\eta,\ 9 + 5\eta)\}$
△ Reference Points for GD, IGD, GD+, IGD+, Delta, Delta+ = {(0, 10), (1, 0)}
▽ Reference Point for HV = (10, 10)

Indicator	A	B	I(A)<I(B) (HV = I(A)>I(B))
Hypervolume	$40 - 60\eta + 20\eta^2$	$7 - 39\eta + 20\eta^2$	consistent
GD	$5.1 + 5.7\eta + 0.85\eta^2$	$3.2 + 2.2\eta + 5.7\eta^2$	inconsistent
IGD	$5.2 + 1.3\eta + 1.0\eta^2$	$6.2 + 4\eta + 3.1\eta^2$	consistent
Delta	$5.2 + 1.3\eta + 1.0\eta^2$	$6.2 + 4\eta + 3.1\eta^2$	consistent
GD+	$2 + 4\eta$	$3 + 4\eta$	consistent
IGD+	$3.5 + 4.8\eta + 0.4\eta^2$	$6.1 + 4.9\eta + 0.2\eta^2$	consistent
Delta+	$3.5 + 4.8\eta + 0.4\eta^2$	$6.1 + 4.9\eta + 0.2\eta^2$	consistent

3.3.2 Exemple 2

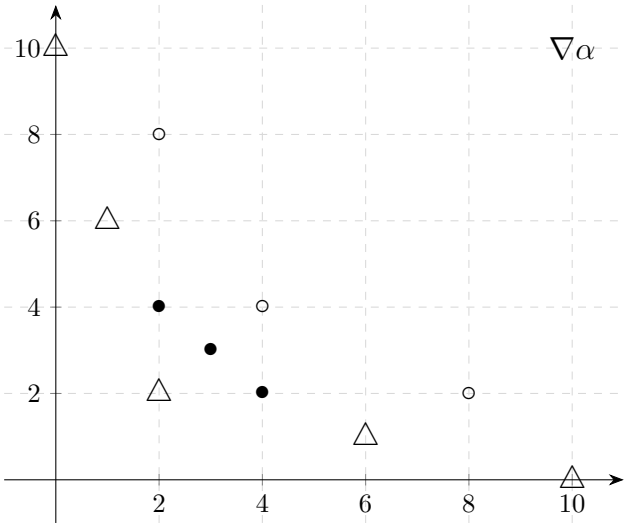

Figure 2

• Solution Set A = $\{(1 + 2\eta, 1 + 4\eta), (1 + 3\eta, 1 + 3\eta), (1 + 4\eta, 1 + 2\eta)\}$
◦ Solution Set B = $\{(1 + 2\eta, 1 + 8\eta), (1 + 4\eta, 1 + 4\eta), (1 + 8\eta, 1 + 2\eta)\}$
△ Reference Points for GD, IGD, GD+, IGD+, Delta, Delta+ = $\{(1, 1 + 10\eta), (1 + \eta, 1 + 6\eta), (1 + 2\eta, 1 + 2\eta), (1 + 6\eta, 1 + 1\eta), (1 + 10\eta, 1)\}$
∇ Reference Point for HV = (10, 10)

Indicator	A	B	I(A)<I(B) (HV = I(A)>I(B))
Hypervolume	$81 - 36\eta + 1\eta^2$	$81 - 36\eta - 16\eta^2$	consistent
GD	1.8η	2.4η	consistent
IGD	3.7η	2.6η	inconsistent
Delta	3.7η	2.6η	inconsistent
GD+	1.1η	2.3η	consistent
IGD+	1.5η	2.3η	consistent
Delta+	1.5η	2.3η	consistent

3.3.3 Exemple 3

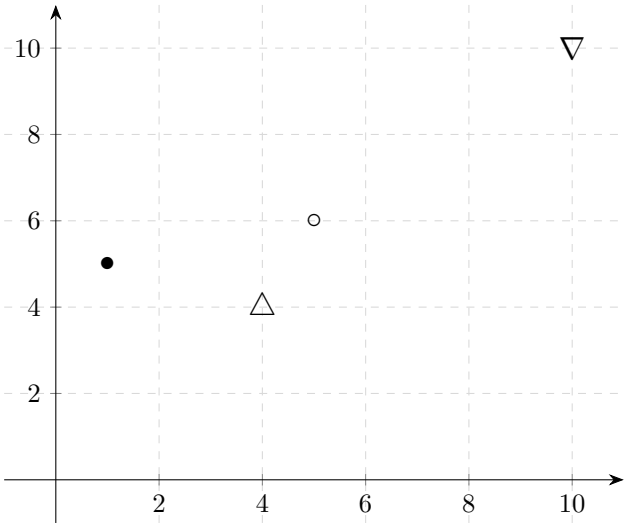

Figure 3

• Solution Set A = $\{(1 + 5\eta,\ 5 + \eta)\}$
◦ Solution Set B = $\{(5 + 10\eta,\ 6 + 9\eta)\}$
△ Reference Points for GD, IGD, GD+, IGD+, Delta, Delta+ = $\{(4,\ 4)\}$
▽ Reference Point for HV = (10, 10)

Indicator	A	B	I(A)<I(B) (HV = I(A)>I(B))
Hypervolume	$45 - 34\eta + 5\eta^2$	$20 - 85\eta + 90\eta^2$	consistent
GD	$3.2 + 4.4\eta + 1\eta^2$	$2.2 + 12.5\eta + 5.4\eta^2$	inconsistent
IGD	$3.2 + 4.4\eta + 1\eta^2$	$2.2 + 12.5\eta + 5.4\eta^2$	inconsistent
Delta	$3.2 + 4.4\eta + 1\eta^2$	$2.2 + 12.5\eta + 5.4\eta^2$	inconsistent
GD+	$1 + \eta$	$2.2 + 12.5\eta + 5.4\eta^2$	consistent
IGD+	$1 + \eta$	$2.2 + 12.5\eta + 5.4\eta^2$	consistent
Delta+	$1 + \eta$	$2.2 + 12.5\eta + 5.4\eta^2$	consistent

3.3.4 Example 4

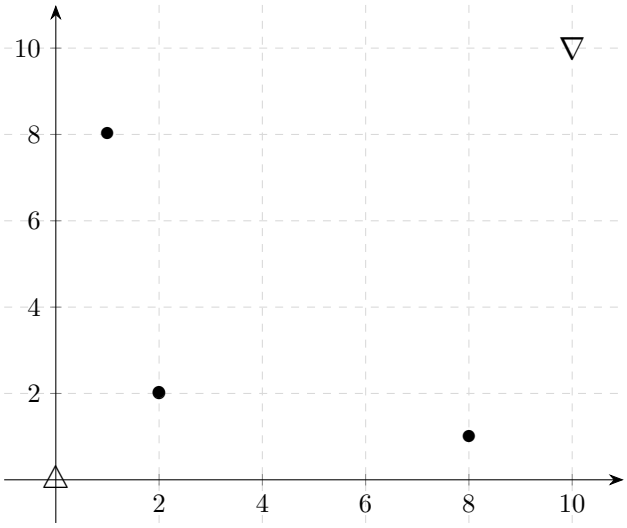

Figure 4

• Solution Set A = $\{(1 + 10\eta, 8 + 3\eta), (8 + 4\eta, 1 - 3\eta), (2 - 1\eta, 2 - 1\eta)\}$
◦ Solution Set B = $\{(2 - 1\eta, 2 - 1\eta)\}$
△ Reference Points for GD, IGD, GD+, IGD+, Delta, Delta+ = {(0, 0}
▽ Reference Point for HV = (10, 10)

Indicator	A	B	I(A)<I(B) (HV = I(A)>I(B))
Hypervolume	$68 - 9\eta + 26\eta^2$	$64 + 16\eta + 1\eta^2$	consistent
GD	$6.3 + 2.1\eta + 2.1\eta^2$	$2.8 - 1.4\eta$	inconsistent
IGD	$2.8 - 1.4\eta$	$2.8 - 1.4\eta$	I(A)=I(B) weak consistent
Delta	$6.3 + 2.1\eta + 2.1\eta^2$	$2.8 - 1.4\eta$	inconsistent
GD+	$6.3 + 2.1\eta + 2.1\eta^2$	$2.8 - 1.4\eta$	inconsistent
IGD+	$2.8 - 1.4\eta$	$2.8 - 1.4\eta$	I(A)=I(B) weak consistent
Delta+	$6.3 + 2.1\eta + 2.1\eta^2$	$2.8 - 1.4\eta$	inconsistent

3.4 Monte Carlo Hypervolume

Monte Carlo simulation is a commonly used strategy for approximating the hypervolume. In order to calculate this approximation, a number of samples are uniformly generated in a hyperrectangle containing Ω (Ω is the region of which we want to calculate the hypervolume, the celestial region of the figure below), and then the proportion of points dominated by those in the front is used to estimate the hypervolume.
Suppose that N points $x_1, ..., x_N$ are uniformly sampled in Ψ,the generated hyperrettangle, the basic Monte Carlo method provides an approximation of the volume V by

$$V(N) = \frac{U}{N} \sum_{i=1}^{N} I_\Omega(x_i)$$

where U is the Volume beetween Reference point and Ideal Point (the ideal point is the minimum value present in the front for each dimension, z^l in the figure below) and $I_\Omega(x_i)$ is the Indicator function of Ω set.

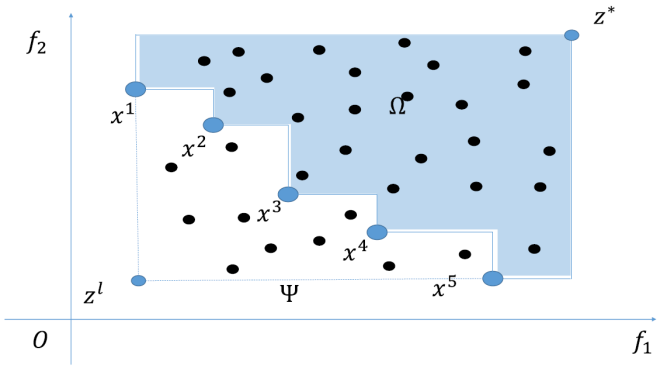

Figure 5

3.4.1 Basic algorithm

The following code is the implementation of the first version of the Monte Carlo algorithm.

```
include("./src/ArithmeticNonStandarNumbersLibrary/src/BAN.jl")
using .BAN
using Distributions
function weaklyDominates(point::AbstractArray, other::AbstractArray)
    for i in 1:length(point)
        if point[i] > other[i]
            return false
        end
    end
    return true
end
function indicator(point, front)
    for p in front
        if weaklyDominates(p, point)
            return true
        end
    end
    return false
end
```

```
function cleanFront(relevantPoints)
    tmp = []
    for i in 1:length(relevantPoints) - 1
        for j in i + 1:length(relevantPoints)
            if weaklyDominates(relevantPoints[i], relevantPoints[j])
                if j ∉ tmp
                    append!(tmp, j)
                end
            end
        end
    end
    res = []
    for i in 1:length(relevantPoints)
        if i ∉ tmp
            append!(res, [relevantPoints[i]])
        end
    end
    return res
end
function shiftPointToZero(relevantPoints, referencePoint)
    min = []
    max = []
    for i in 1:length(referencePoint)
        Base.append!(min, [Inf])
    end
    for i in 1:length(referencePoint)
        Base.append!(max, [-Inf])
    end
    for j in 1:length(relevantPoints)
        if any(a -> a != 0, referencePoint)
            relevantPoints[j] .= relevantPoints[j] - referencePoint
        end
        for i in 1:length(referencePoint)
            if relevantPoints[j][i] < min[i]
                min[i] = relevantPoints[j][i]
            end
            if relevantPoints[j][i] > max[i]
                max[i] = relevantPoints[j][i]
            end
        end
    end
    return relevantPoints, min, max
end
function compute_mc(referencePoint, front::AbstractArray, iteration = 100000)
    relevantPoints = []
    for point in front
        if weaklyDominates(point, referencePoint)
            Base.append!(relevantPoints, [point])
        end
    end
    relevantPoints, min, max = shiftPointToZero(relevantPoints, referencePoint)
    relevantPoints = cleanFront(relevantPoints)
    hv = 0
    hit = 0
    for i in 1:iteration
        tmp = [rand()*i for i in min]
        if indicator(tmp, relevantPoints)
            hit += 1
        end
    end
    volume = 1
    for i in min
        volume *= (i * -1)
    end
    hv += volume * hit / iteration
    return hv
end

referencePoint = [20, 20]
front = [[2,5], [2, 2]]
hyperVolume = compute_mc(referencePoint, front)
```

Studying deeper this version of the algorithm we realized that it presented a big issue: the calculation of the hypervolume is correct only for the leading term.

3.4.2 First issue and first solution

In order to understand the motivations let's take an example (this example is a borderline case as we are trying to calculate the hypervolume between front and reference points with very different orders of magnitude, while usually the reference point is chosen to have the same order of magnitude as the front points):

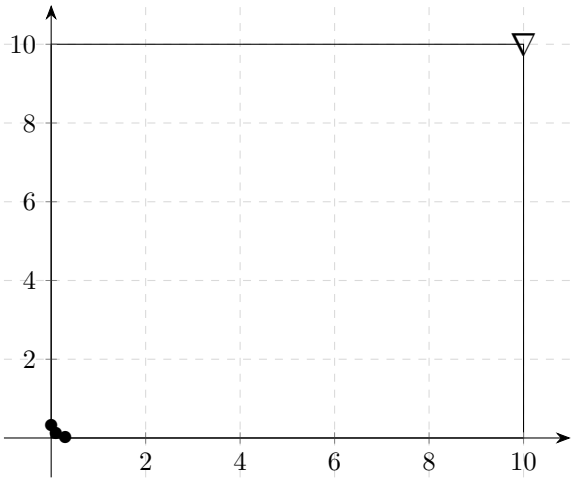

Figure 6:
• Front = $\{(0, \eta), (0.5\eta, 0.5\eta), (\eta, 0)\}$
∇ Reference Point = (10,10)
Extact Hypervolume = $100 - 0.75\eta^2$
Monte Carlo Hypervolume = 100

If we apply the Monte Carlo hypervolume to this case, it is easy to see that all the sampled points will be in the area of the rectangle delimited by the origin and the reference point because the probability that some points are sampled in the infinitesimal area not dominated by the front is very low and therefore the implemented algorithm will return as hypervolume value all the area between the reference point and the origin.

The example proposed in Fig. 6, as already mentioned, is a borderline case used only for the purpose of emphasizing the problem, but by taking even a more realistic case the problem continues to exist.

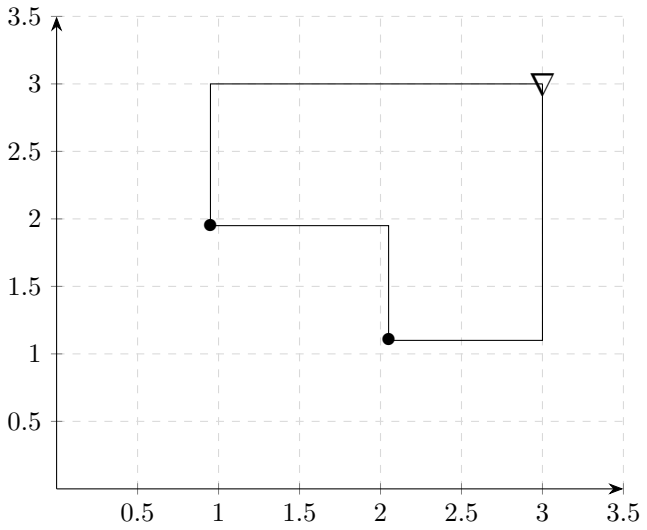

Figure 7:
- Front = $\{(1-\eta,\, 2-\eta),\, (2+\eta,\, 1+2\eta)\}$

∇ Reference Point = (3,3)
Extact Hypervolume = $3-\eta+4\eta^2$
Monte Carlo Hypervolume = $3-1.5\eta-1.5\eta^2$

So we need to think of a strategy to be able to sample in infinitely smaller spaces than the leading term. The solution we came up with, involves sampling in the neighborhood of each point while keeping one dimension fixed at a time to find out all the information needed to calculate the hypervolume after which we go on to sampled in the infinitely smaller neighborhood.

One of the main advantages between the classical hypervolume algorithm, described in Section 4.1, from the Monte Carlo hypervolume is that the latter does not need to reorder the points according to one of the dimensions to compute the value. Also in our approach for PC problems this idea has been maintained making more complicated the solution we want to adopt because, not doing the reordering, we do not know the distance between a point and its next/previous.

In order to find out this length our idea is to sample along the red dotted line shown in Fig. 8 that starts from the considered point of the front and arrives at the reference point. Sampling along this line is used to figure out the length to calculate the hypervolume of the infinitesimal term of the point $(1-\eta,\, 2-\eta)$ for the second dimension, as outlined by the blue line in Fig. 8.

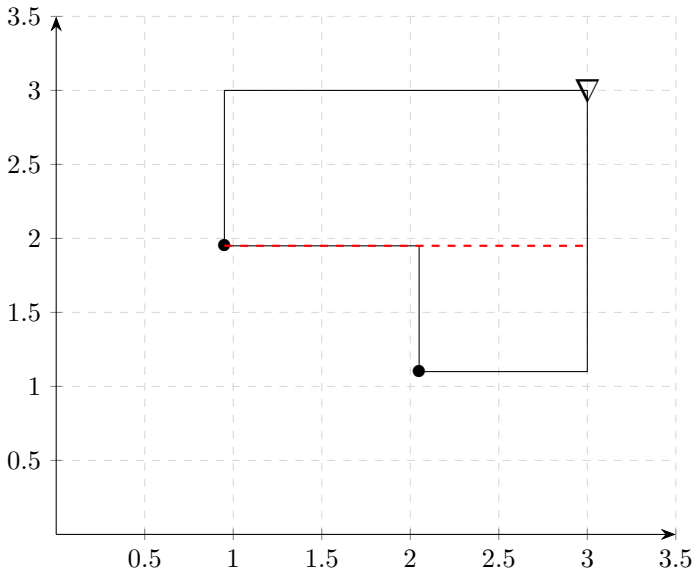

Figure 8:
• Front = $\{(1-\eta, 2-\eta), (2+\eta, 1+2\eta)\}$
∇ Reference Point = (3,3)

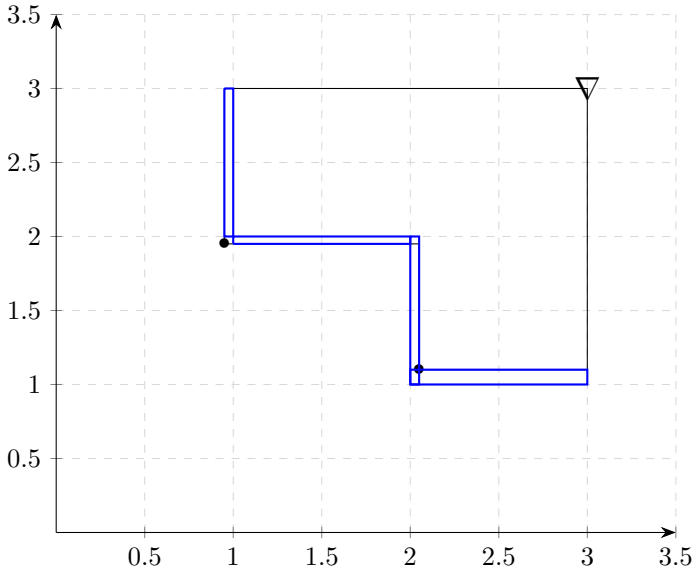

Figure 9:
• Front = $\{(1-\eta, 2-\eta), (2+\eta, 1+2\eta)\}$
∇ Reference Point = (3,3)

An important consideration to keep in mind during this approach is whether the hypervolume value of the infinitesimal terms just computed should be added to or subtracted from the hypervolume of the leading terms.

With this approach, described below in Algorithm 2, we actually have an improvement in the approximation of the hypervolume. As a reference we take the result obtained using the exact hypervolume described in Section 3.1.1, $hv = 3-\eta+4\eta^2$. The result obtained from the first version of the Monte Carlo hypervolume, reported in Section 3.4.1, actually correctly approximates the leading term but not the infitesimal terms. In fact the result obtained is $hv = 3 - 1.5\eta - 1.5\eta^2$ (It is easy to derive it considering that the volume is calculated from the ideal point $(1-\eta, 1+2\eta)$ to the reference point). Modifying the algorithm to improve the approximation of the infinitesimal terms we obtain this result $hv = 3 - \eta$ where the infinitesimal term is calculated by sampling the blue areas drawn in Fig. 9.

The pseudocode below describes the algorithm to be implemented for the calculation of the approximate hypervolume with Monte Carlo, algorithm that must be repeated iteratively on each monosemium of the BANs used.

Algorithm 2 PC_MPL_MOPs Hypervolume Monte Carlo

Input: front, reference point
Require: front and reference point should have the same order of magnitude
Result: approximate hypervolume value

```
leading_term_front =[]
for each point in front_points do
    leading_term_front.add(extract-leading-term(point)
leading_term_ref = extract-leading-term(reference_point)
hv = compute_monte_carlo(leading_term_ref, leading_term_front)          ▷ see section 3.4.1
visited = []
for each p in front do
    if p in visited then
        continue
    tmp_front = find_same_leading_term(p, front)
    visited.add(tmp_front)
    for each dim in length(p) do
        hv += compute_monte_carlo_along_an_axis(reference_point, front, tmp_front, dim)
    ▷ see section 3.4.2
    if length(tmp_front) > 1 then
        remove_leading_term(tmp_front)
        nadir = get_nadir(tmp_front)
        hv += compute_monte_carlo_for_each_monosemium(nadir, front, tmp_front)
return hv
```

3.4.3 Second issue

As you can see from the solution obtained in the previous section and as shown in the following example also this approach has problems. Even if we are going to sample in an infinitesimal area using a sampling strategy the area bounded by the red line in Fig. 10 will never be considered in the hypervolume calculation.

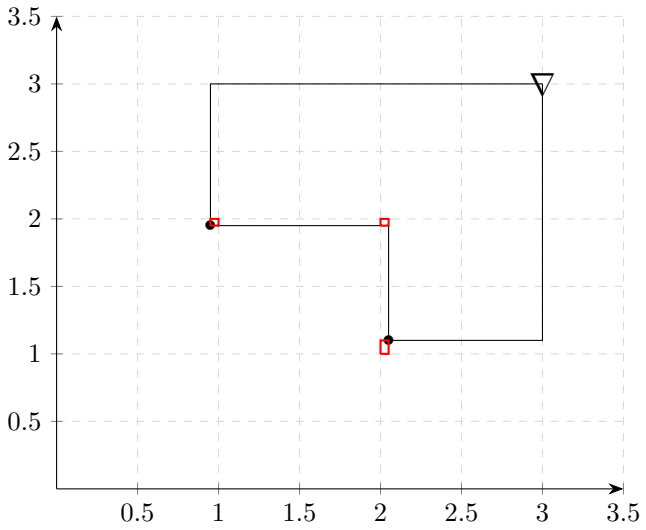

Figure 10:
• Front = $\{(1-\eta, 2-\eta), (2+\eta, 1+2\eta)\}$
∇ Reference Point = (3,3)

These red areas are the components needed to calculate the order-of-magnitude term η^2. In fact by manually calculating these areas we see that they have exactly the value of $4\eta^2$ as reported the exact hypervolume result, given in Section 3.4.2.

3.4.4 Error estimation

To give an upper bound to the error in calculating the hypervolume we focused on the 2D problem. To find the error we have to calculate the area given by the maximum infitesimal monosemium taken in absolute value for each dimension and multiply it by twice the number of points minus one. This upperbound is always constructed considering the worst case which is the largest area that can be generated by the infinitesimal monosemium.

The error is:

$$|error| \leq \max_{p \in P}(|p_x|) \max_{p \in P}(|p_y|)(2n-1)$$

n = number of points in the front
p_x = infinitesimal part of point p in the x-coordinate
p_y = infinitesimal part of point p in the y-coordinate
P = the population approximating the front

Note that if we are calculating the error we have from monosemium of order of magnitude η we will have an error in η^2.

These ideas had for the 2D case can also be expanded for more dimensions with some more consideration.

With more thought we could also have a more precise upperbound by studying the signs of infinitesimal monosemium, which could allow us to estimate the number of hypervolumes to add and subtract to the total hypervolume.

3.4.5 Complexity

The complexity to calculate the Hypervolume of the leanding term is $\mathcal{O}(mn)$ where m is the number of points that are sampled and n are the numbers of the front with which the sampled numbers must be compared to know if they are or not within the region bounded by the points of the front and the reference point. To calculate the contribution of the infinitesimal terms to the hypervolume it is necessary to sample along each dimension for each point then $\mathcal{O}(nmd)$ (m refers to the number of fronts to compare).

To find out the contributions of infinitesimals around each point seen from the leading term (this is the case where there are points with the same leading term, but different infinitesimal components), we need to sample for each infinitesimal around the point, so $\mathcal{O}(nmk)$ where k is the number of monosemium. The overall complexity is therefore $\mathcal{O}(nm+nmd+nmk)$ which is equal to $\mathcal{O}(ndm)$ if $d \geq k$ and to $\mathcal{O}(nmk)$ otherwise.

4 PL-MPL-MOPs

PL-MPL-MOPs are problems in which objectives are grouped in levels and lexicographic priority exists among them.
The general mathematical structure is:

$$\text{lex}\min \left[\min \begin{pmatrix} f(\mathbf{x})_1^{(1)} \\ f(\mathbf{x})_2^{(1)} \\ \cdot \\ \cdot \\ \cdot \\ f(\mathbf{x})_{m1}^{(1)} \end{pmatrix}, \min \begin{pmatrix} f(\mathbf{x})_1^{(2)} \\ f(\mathbf{x})_2^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ f(\mathbf{x})_{m2}^{(2)} \end{pmatrix}, \ \ldots \ , \min \begin{pmatrix} f(\mathbf{x})_1^{(p)} \\ f(\mathbf{x})_2^{(p)} \\ \cdot \\ \cdot \\ \cdot \\ f(\mathbf{x})_{mp}^{(p)} \end{pmatrix} \right]$$

where p is the number of priority levels and $f_i^{(j)}$ is the i_{th} objective in the j_{th} level of importance.
The reformulation to address this type of problem by means of Alpha theory is:

$$\min \left[\begin{pmatrix} f(\mathbf{x})_1^{(1)} \\ f(\mathbf{x})_2^{(1)} \\ \cdot \\ \cdot \\ \cdot \\ f(\mathbf{x})_{m1}^{(1)} \end{pmatrix} + \alpha^{-1} \begin{pmatrix} f(\mathbf{x})_1^{(2)} \\ f(\mathbf{x})_2^{(2)} \\ \cdot \\ \cdot \\ \cdot \\ f(\mathbf{x})_{m2}^{(2)} \end{pmatrix} + \ \ldots \ + \alpha^{1-p} \begin{pmatrix} f(\mathbf{x})_1^{(p)} \\ f(\mathbf{x})_2^{(p)} \\ \cdot \\ \cdot \\ \cdot \\ f(\mathbf{x})_{mp}^{(p)} \end{pmatrix} \right]$$

With this type of problem, as opposed to PC-MPL-MOPs, the definition of Pareto Dominance must be redefined since the standard definition is not suitable because it ignores the priority information.

PL-Dominance (correct) Given two solutions A and B, belonging to the subfronts F_i and F_j, respectively, A "PL-dominates" B if and only if i is strictly smaller than$j$: $A \prec^* B \Leftrightarrow i < j$.

With this definition the non-dominance relation is not only a function of two arguments (solutions), but has a global dependence on all other individuals, it is not possible to say

whether two elements are dominated before assigning a fitness degree to the whole population).
Here is the problem to define a correct method to calculate the Hypervolume indicator:
As indicator we want to have a order, so, given a solution set we want to assign a number that evaluate the solution, given another solution we want to assign another number that evaluate this solution and than compare two or more solutions using the indicators value, but without the need to know all the solution in advance.
With this observation is clearly that we can't leverage the correct PL-Dominance definition.

4.1 Some ideas and why they don't work

In all the following example and reasoning when we refer to A, B, C solutions we are indicating the solutions used to show the problem with Pareto Dominance.

4.1.1 Sum together various priority levels

An idea can be sum together the priority levels and we obtain:

$$A = \begin{pmatrix} 2 + 2\alpha^{-1} \\ 5 + 2\alpha^{-1} \end{pmatrix}, \; B = \begin{pmatrix} 4 + 7\alpha^{-1} \\ 3 + 6\alpha^{-1} \end{pmatrix}, \; C = \begin{pmatrix} 1 + 9\alpha^{-1} \\ 4 + 8\alpha^{-1} \end{pmatrix}$$

Here we try to reconduct the problem in a situation similarly to PC-MPL-MOPs, this solution can not be used easily because priority levels can have different numbers of objective functions and the the sum between vectors of different dimension cannot be used. Anyway this solution is not Pareto Compliant either having vectors of equal dimensions, or the more complicated case where the vectors do not have equal dimensions (an example is shown in 5.2).

4.1.2 Calculate iteratively Hypervolume in each level

Here we calculate the Hypervolume in each level separately and than sum together the result obtained using different power of α, for example using solution A:

$$level_1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix} \; Hypervolume = h_1 \;\; level_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \; Hypervolume = h_2$$

$$Hypervolume = h_1 + \alpha^{-1} h_2$$

To understand why this solution can not work we have to think a set of solutions found by an algorithm: this solution can be dominated or non dominated solutions, it is quite simple to find solution dominated using the PL-Dominance but with higher Hypervolume value (example in 4.2), so using this solution the Hypervolume is not Pareto Compliant and can be found tons of ideas that suffer of this situation, but we want list another one to start a reasoning over it:

4.1.3 Using objective functions space

Let S be a PL-MPL-MOPs like this one:

$$S = \min \left[\begin{pmatrix} f(\mathbf{x})_1^{(1)} \\ f(\mathbf{x})_2^{(1)} \end{pmatrix} + \alpha^{-1} \begin{pmatrix} f(\mathbf{x})_1^{(2)} \\ f(\mathbf{x})_2^{(2)} \end{pmatrix} \alpha^{-2} + \begin{pmatrix} f(\mathbf{x})_1^{(3)} \\ f(\mathbf{x})_2^{(3)} \end{pmatrix} \right]$$

We can write:

$$S = \min \begin{bmatrix} f(\mathbf{x})_1^{(1)} \\ f(\mathbf{x})_2^{(1)} \\ f(\mathbf{x})_1^{(2)} \\ f(\mathbf{x})_2^{(2)} \\ f(\mathbf{x})_1^{(3)} \\ f(\mathbf{x})_2^{(3)} \end{bmatrix}$$

Here we can make example where also this idea doesn't work and as we said at the beginning of the PL-MPL-MOPs section the original definition of Pareto Dominance is an "issue" for this class of problems (example in 4.2). But the Hypervolume indicator can be used in a set of non-dominated solution after checking the PL-Dominance (correct) in some population, this ensure that all solutions are non dominated and it's simple, using the 4.1.2 method (we will show 4.1.2 is better than 4.1.3), create a valid indicator to evaluate between non dominated solutions.
So now the problem is how to find the population of non dominated solutions.

4.2 If optimal front or a optimal population is available

If the optimal pareto front is available we can delete all the dominated solutions and than the hypervolume indicator can be used to evaluate this solution set, here we want to show how this approach can work: $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ different solution set (for sake of brevity with only one point), Reference Population = {E, F}.
hv_1 indicates hypervolume presented in 4.1.1.
hv_2 indicates hypervolume presented in 4.1.2.
hv_3 indicates hypervolume presented in 4.1.3.

Solution	hv_1	hv_2	hv_3
$\mathcal{A} = \left\{ \begin{pmatrix} 2 \\ 5 \end{pmatrix} + \alpha^{-1} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\}$	$270 - 66\alpha^{-1} + 4\alpha^{-2}$	$270 + 324\alpha^{-1}$	87480
$\mathcal{B} = \left\{ \begin{pmatrix} 4 \\ 3 \end{pmatrix} + \alpha^{-1} \begin{pmatrix} 7 \\ 6 \end{pmatrix} \right\}$	$272 - 215\alpha^{-1} + 42\alpha^{-2}$	$272 + 182\alpha^{-1}$	49504
$\mathcal{C} = \left\{ \begin{pmatrix} 1 \\ 4 \end{pmatrix} + \alpha^{-1} \begin{pmatrix} 9 \\ 8 \end{pmatrix} \right\}$	$304 - 296\alpha^{-1} + 72\alpha^{-2}$	$304 + 132\alpha^{-1}$	40128
$\mathcal{D} = \left\{ \begin{pmatrix} 2 \\ 3.5 \end{pmatrix} + \alpha^{-1} \begin{pmatrix} 15 \\ 4 \end{pmatrix} \right\}$	$297 - 319.5\alpha^{-1} + 60\alpha^{-2}$	$297 + 80\alpha^{-1}$	23760

Reference Population

$$E = \begin{pmatrix} 1.5 \\ 5 \end{pmatrix} + \alpha^{-1} \begin{pmatrix} 4 \\ 7 \end{pmatrix}$$

$$F = \begin{pmatrix} 5 \\ 0 \end{pmatrix} + \alpha^{-1} \begin{pmatrix} 3 \\ 8 \end{pmatrix}$$

hv_1, hv_2 and hv_3 was calculated with reference point (20, 20), (20, 20) and (20, 20, 20, 20) respectively.
$\mathcal{A}$ is dominated by E so it is not a optimal solution (or we can assign $\mathcal{A}$ to another front), it is worth noting that the value of hv_3 is the highest one.
$\mathcal{B}$ is in the same front of E and F.
$\mathcal{C}$ is dominated by F in the second layer so it is not a optimal solution (or we can assign $\mathcal{A}$ to another subfront), it is worth noting that the value of hv_1 and hv_2 is better than $\mathcal{B}$, but $\mathcal{C}$ is not in the optimal front.
$\mathcal{D}$ is in the same front of E and F.
So $\mathcal{B}$ and $\mathcal{D}$ are in the best subfront (we refer to [2] for the definition of the subfronts), in this case $\mathcal{B}$ is considered better than $\mathcal{D}$ using hv_3 and $\mathcal{D}$ is considered better than $\mathcal{B}$ using hv_1 and hv_2.
hv_2 is able to keep separate the calculation of hypervolume in each level, this allows you to compare the various levels in a more direct way, with hv_2 a level has infinitely more importance than those to follow. hv_2 is the method of hypervolume calculation most consistent with the definition of PL-MPL-MOPs problems.

4.3 Find non-dominated solutions

As we show in the above section we want to try to address the problem of the PL Dominance before input the solutions set in an algorithm to calculate the Hypervolume, some ideas can be:

- **Using gradients**: Try a local search in the proximity of each solution present in the solutions set and check if non dominated solution that dominates others are found. This solution implies to have differentiable objective functions and to check if the movement in some direction is in the feasible region.
- **Exploit Input Space**: Make a random search in the input space, here in the neighbourhood of the solution set we can try to exploit the input space, the problem is often input space is bigger than output space.
- **Exploit Output Space**: Make a random search in the output space (similar to Monte Carlo Approach) and check if the solution sampled are feasible. Here we can move for each level and make a search for each level, we need to pay attention to maintain the constraints received by optimal solution in previous levels (this problem is also present using gradients)
- **Use an algorithm to find other solutions**: For example, we can think of running an algorithm such as PL-NSGA-II to find the reference population on the problem or we can try to exploit the solution set in which we have to calculate the hypervolume as starting point for the PL-NSGA-II.

The solutions above are computationally very expensive.

4.4 Hypervolume Complexity

Using the algorithm hv_2 presented in 4.1.2 The complexity is $\mathcal{O}(ln^{d-2}log\, n)$ using the algorithm presented in [5] for each layer in our PL-MOP-MPL (l is the number of layers in the problem).

4.5 Monte Carlo Hypervolume

A Monte Carlo approach, once found the nadir population to filter out dominated solutions, is very simple to implement, considering the hypervolume calculation presented in 4.1.2, this is the code:

```
referencePoints = [[20, 20], [20, 20]]
front = [[[2,5], [2, 2]], [[2,5], [2, 1]]]
hyperVolume = 0
for i in 1:length(referencePoints)
    referencePoint = referencePoints[i]
    tmp_front = [f[i] for f in front]
    global hyperVolume += compute_mc(referencePoint, tmp_front) *η^(i-1)
end
println(hyperVolume)
```

We need a reference point for each level, and iterate and calculate the hypervolume for each level (this part is identical also in a Non-Monte Carlo Approach).
The hypervolume calculation is presented in the first figure of 3.4 subsection.

4.5.1 Complexity

The complexity is $\mathcal{O}(lnm)$ where l is the number of layers, n is the number of the points in the front that are needed to understand which portion of the m points sampled are in the Hypervolume per each layer.

References

[1] Leonardo Lai, Lorenzo Fiaschi, Marco Cococcioni: Solving Mixed Pareto-Lexicographic Multi-Objective Optimization Problems: The Case of Priority Chains.

[2] Leonardo Lai, Lorenzo Fiaschi, Marco Cococcioni, Kalyanmoy Deb: Handling Priority Levels in Mixed Pareto-Lexicographic Many-Objective Optimization Problems.

[3] Andreia P. Gurreiro, Carlos M. Fonseca, Luìs Paquete: The Hypervolume Indicator: Problems and Algorithms.

[4] Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki and Yusuke Nojim: Modified Distance Calculation in Generational Distance and Inverted Generational Distance.

[5] Carlos M. Fonseca, Luìs Paquete, and Manuel López-Ibañez,: An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator