

24 DE OCTUBRE DE 2022



# COMPETICIÓN. RESULTADOS EXPERIMENTALES

EJERCICIO OBLIGATORIO APLICACIONES DE SOFT COMPUTING

IVÁN MORENO GRANADO

UNIVERSIDAD DE SEVILLA

IVAMORGRA@ALUM.US.ES

## CONTENIDO

<b>1.- Resultados experimentales con capacidad de cómputo de 10000 evaluaciones .....</b>	4
<b>1.1.- Población de 40 individuos y 250 generaciones.....</b>	5
.....	11
<b>1.2.- Población de 100 individuos y 100 generaciones.....</b>	12
<b>1.3.- Población de 200 individuos y 50 generaciones.....</b>	19
<b>2.- Resultados experimentales con capacidad de cómputo de 4000 evaluaciones .....</b>	25
<b>2.1.- Población de 40 individuos y 100 generaciones.....</b>	25
<b>2.2.- Población de 80 individuos y 50 generaciones.....</b>	30
<b>2.3.- Población de 100 individuos y 40 generaciones.....</b>	36
<b>2.- Conclusión.....</b>	42
<b>3.- Resultados experimentales de CF6 4D con capacidad de cómputo de 10000 evaluaciones .....</b>	43
<b>3.1.- Población de 40 individuos y 250 generaciones.....</b>	43
<b>3.2.- Población de 100 individuos y 100 generaciones.....</b>	44
<b>3. 3.- Población de 200 individuos y 50 generaciones.....</b>	44
.....	44
<b>4.- Resultados experimentales de CF6 4D con capacidad de cómputo de 4000 evaluaciones .....</b>	45
<b>4.1.- Población de 40 individuos y 100 generaciones.....</b>	45
<b>4.2.- Población de 80 individuos y 50 generaciones.....</b>	45
<b>4.3.- Población de 100 individuos y 40 generaciones.....</b>	46
<b>5.- Resultados experimentales de CF6 16D con capacidad de cómputo de 10000 evaluaciones .....</b>	46
<b>5.1.- Población de 40 individuos y 250 generaciones.....</b>	46
<b>5.2.- Población de 100 individuos y 100 generaciones.....</b>	47
<b>5.3.- Población de 200 individuos y 50 generaciones.....</b>	47
<b>6.- Resultados experimentales de CF6 16D con capacidad de cómputo de 4000 evaluaciones .....</b>	48
<b>6.1.- Población de 40 individuos y 100 generaciones.....</b>	48
<b>6.2.- Población de 80 individuos y 250 generaciones.....</b>	48
<b>6.3.- Población de 100 individuos y 40 generaciones.....</b>	49

## Tabla de contenidos

Versión	Descripción	Fecha
1.0	Realización de la introducción y resultados experimentales de ZDT3	23/10/22
1.1	Comentarios sobre los experimentos de ZDT3	24/10/22
1.2	Graficas de soluciones de ZDT3	27/10/22
1.3	Inserción de resultados de CF6	30/10/2022

## Introducción

En este documento se detalla una comparativa entre el algoritmo conocido como NSGAII y el algoritmo basado en agregación. Para ello, se ha implementado este último para posteriormente comparar mediante métricas y estadísticas el algoritmo dado (NSGAII) y el desarrollado (agregación).

Para ello, se han ejecutado métricas comparativas entre el algoritmo conocido (NSGAII) y el algoritmo implementado (Agregación). Para ello, en primer lugar se mostrará gráficas que representarán el hipervolumen, spacing y el coverage set (de tres ejecuciones con diferente semilla). Posteriormente, mediante scripts que ya venían dados y algunos modificados se han hecho estadísticas mediante diez ejecuciones. Teniendo en cuenta que contamos con una capacidad de cómputo de 10000 evaluaciones y 4000 evaluaciones las configuraciones son las siguientes:

1. 10000 evaluaciones:
  - 1.1. Población 40 individuos y 250 generaciones
  - 1.2. Población de 100 individuos y 100 generaciones
  - 1.3. Población de 200 individuos y 50 generaciones
2. 4000 evaluaciones:
  - 2.1. Población de 40 individuos y 100 generaciones
  - 2.2. Población de 80 individuos y 50 generaciones
  - 2.3. Población de 100 individuos y 40 generaciones

Los parámetros **t** y **cr** valen, respectivamente, un 20% y 0.5 para todos los experimentos. Tan sólo se ha variado en la configuración de los parámetros de población y número de individuos.

A continuación, se procede a explicar el significado de cada una de las tres gráficas que se generan al ejecutar las métricas:

- Gráfica hipervolumen: Nos da una aproximación al frente Pareto mediante hipercubos. En el eje X vienen dadas las generaciones y en el eje Y el valor de ese hipervolumen.
- Gráfica Coverage Set: Sólo tiene sentido esta gráfica si se tienen 2 frentes a comparar (2 conjuntos de soluciones), ya que establece dominancia entre ambos conjuntos.  $C(1,2)$  establece el porcentaje de soluciones de 2 que son dominadas por 1. Si  $C(1,2) = 1$  significa que el 100% de soluciones de 2 son dominadas por las soluciones de 1, y viceversa. El eje X representa las generaciones mientras que el eje Y representa el porcentaje de soluciones dominadas de un frente respecto a otro.
- Gráfica Spacing: Representa la desviación estándar de las distancias entre las soluciones vecinas. Cuanto menor sea esta métrica mejor, ya que esto significa que las soluciones están distribuidas de manera más uniforme. El eje X representa las generaciones mientras que el eje Y representa el valor de Spacing.

Cabe destacar que cada una de las estadísticas comparativas se hace en base a un conjunto de 10 ejecuciones diferentes (con 10 semillas). Dicha comparación se hace mediante los ficheros de salida de NSGAII proporcionados en EV y los ficheros de salida generados por el algoritmo implementado.

La representación gráfica de dichas funciones debemos tenerla en cuenta para la interpretación de los resultados. Son las siguientes:

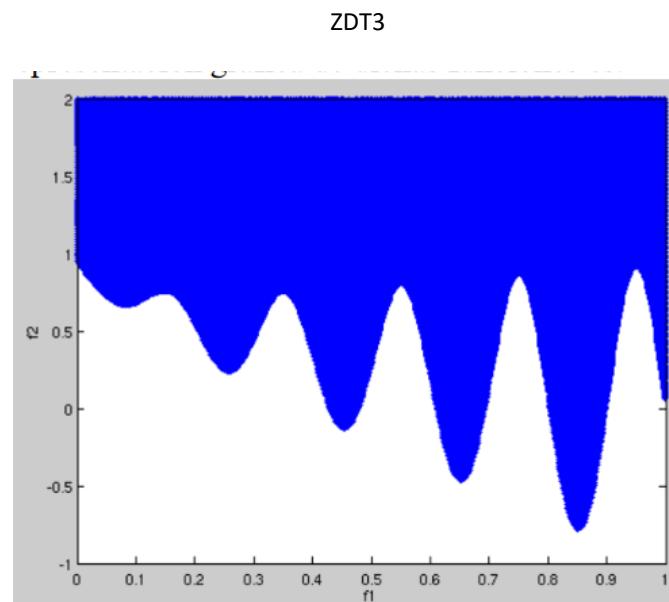


Fig 1: Representación gráfica de ZDT3

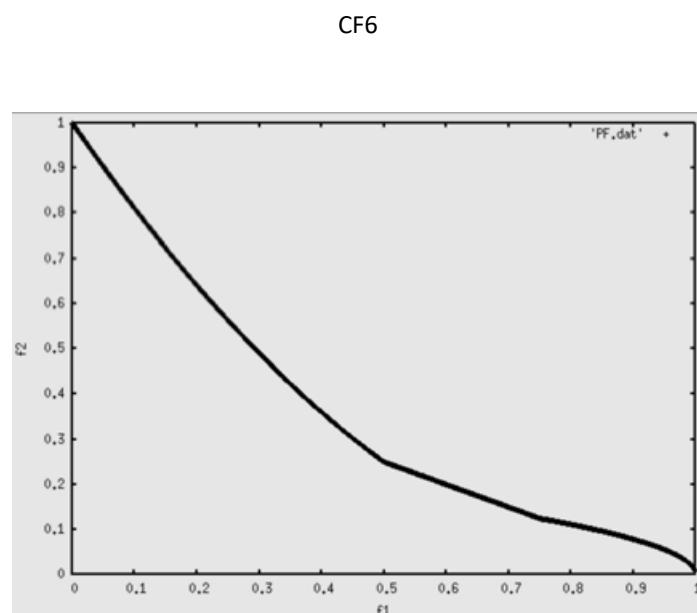


Fig 2: Representación gráfica de CF6

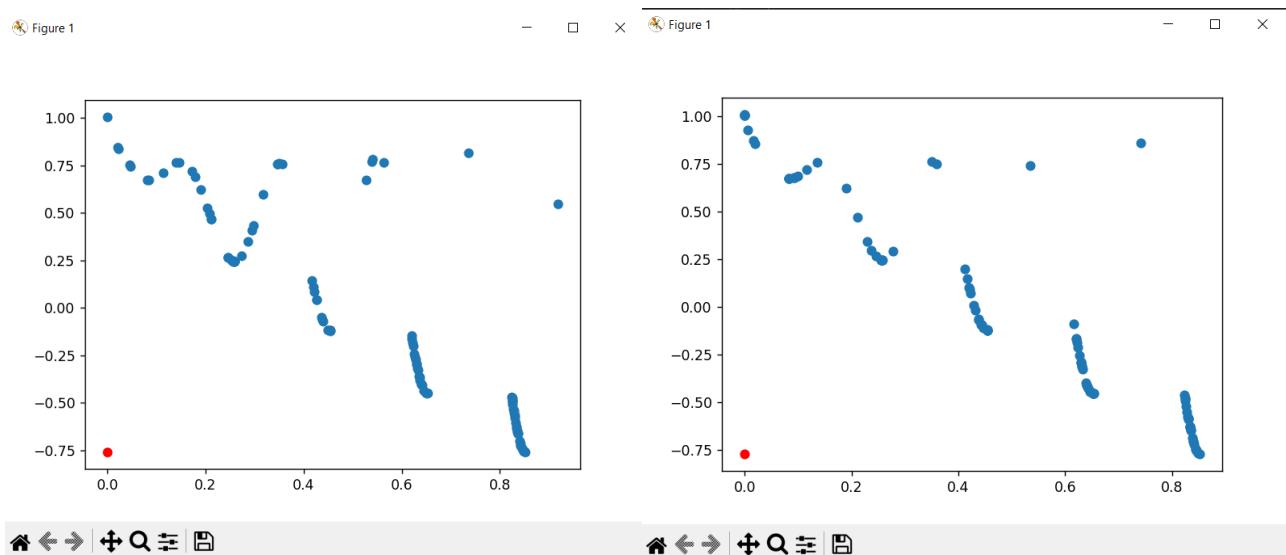
## Contenido

### 1.- Resultados experimentales con capacidad de cómputo de 10000 evaluaciones

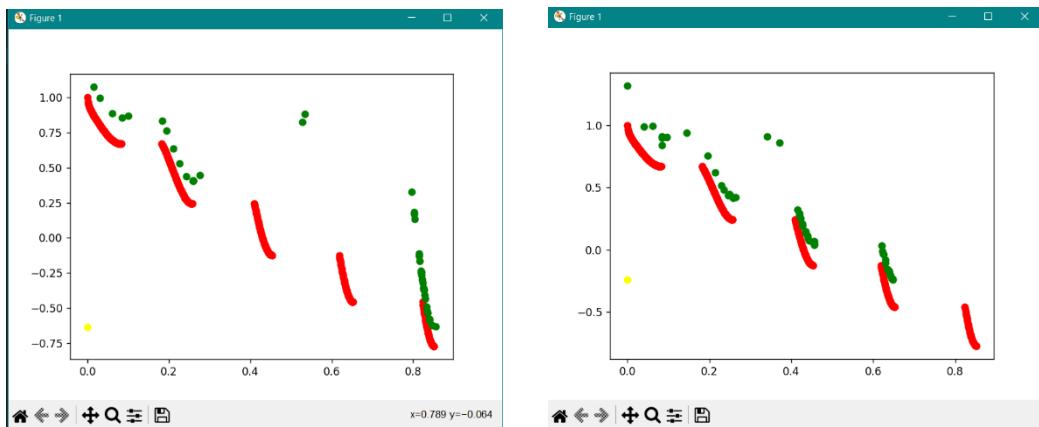
### 1.1.- Población de 40 individuos y 250 generaciones

A continuación, vamos a ver las gráficas correspondientes a los experimentos de las diferentes versiones del algoritmo basado en agregación. En el eje Y se encuentra el resultado de una de las dos funciones resultantes, en concreto,  $f_2$ , y en el eje X la función  $f_1$ . En estos casos, las gráficas representan la última generación y se trata de una sola ejecución por gráfica.

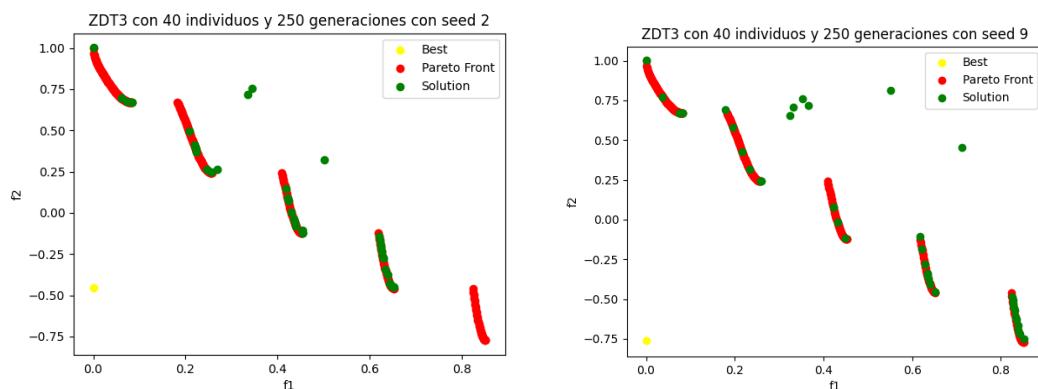
En primer lugar, se hicieron varios experimentos al desarrollar una primera versión de ZDT3. Con esta configuración se obtuvieron varias gráficas:



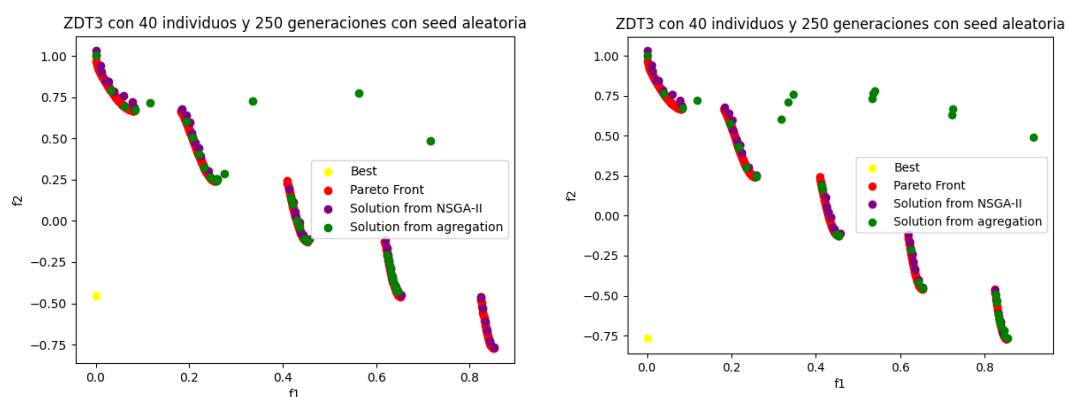
Para esta solución no se había implementado aún la mutación gaussiana y tampoco se había implementado la representación del frente de Pareto.



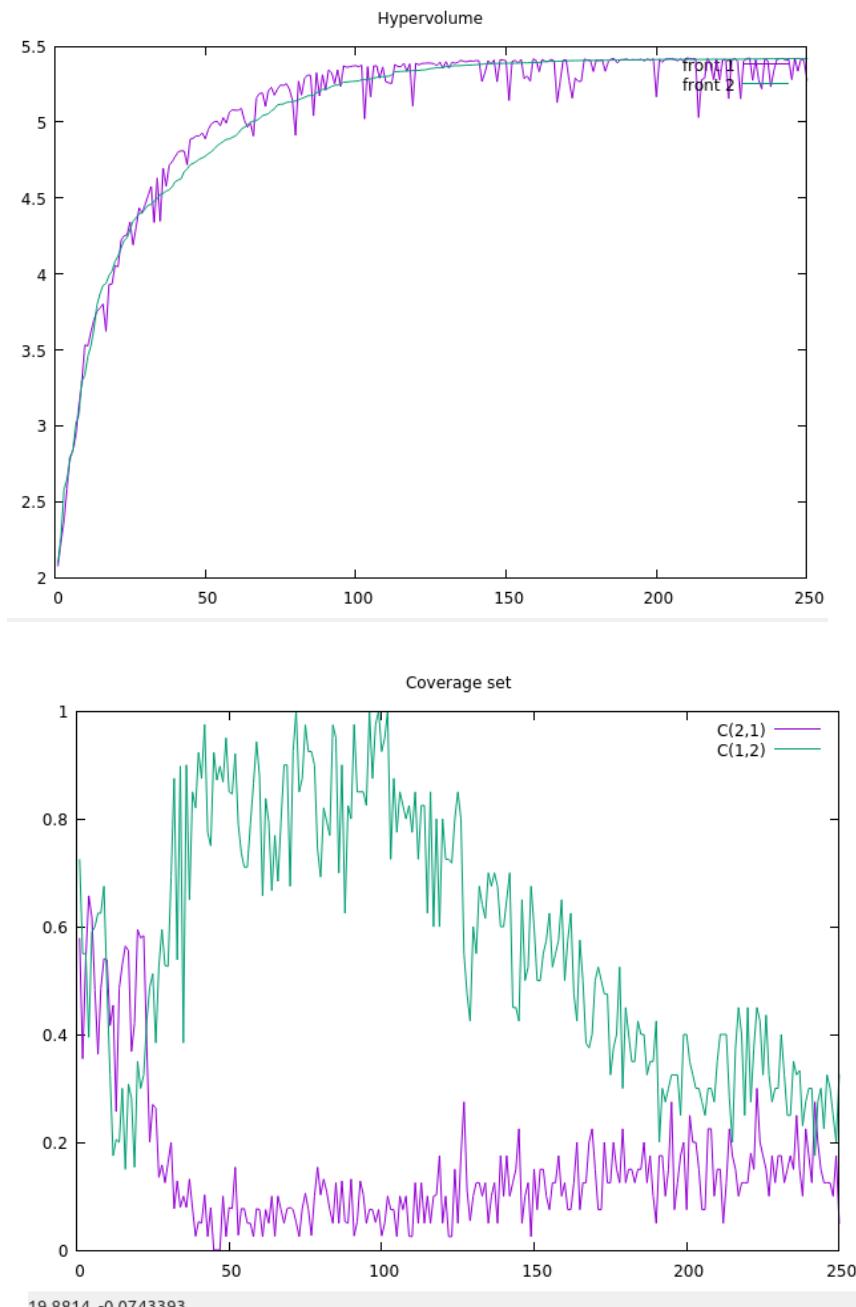
Esta representación gráfica se corresponde ya a la segunda versión del código, donde se actualizó la ejecución de las iteraciones (generaciones). En la antigua versión, se evaluaba únicamente a cada individuo una vez por generación. En esta versión, se evalúa la función una vez el individuo ha sido transformado mediante las operaciones de cruce y mutación.

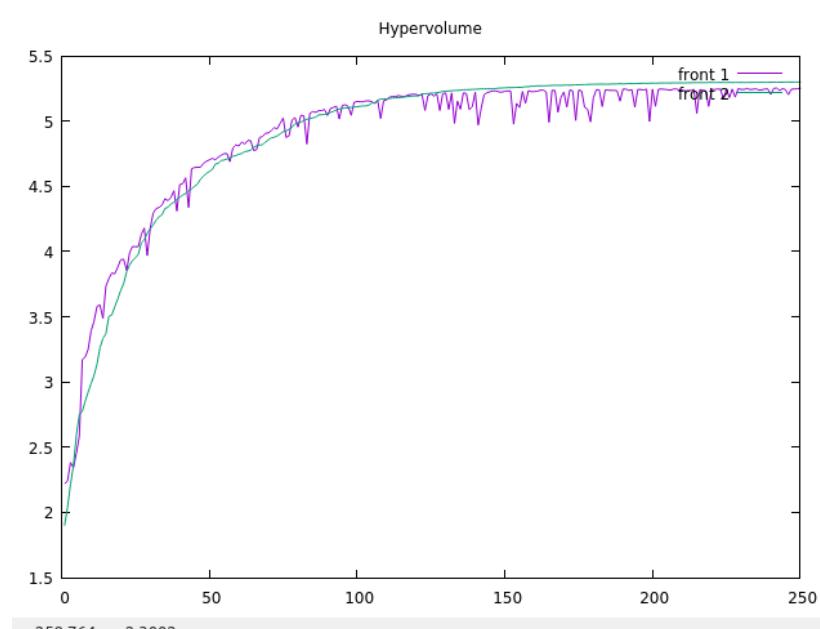
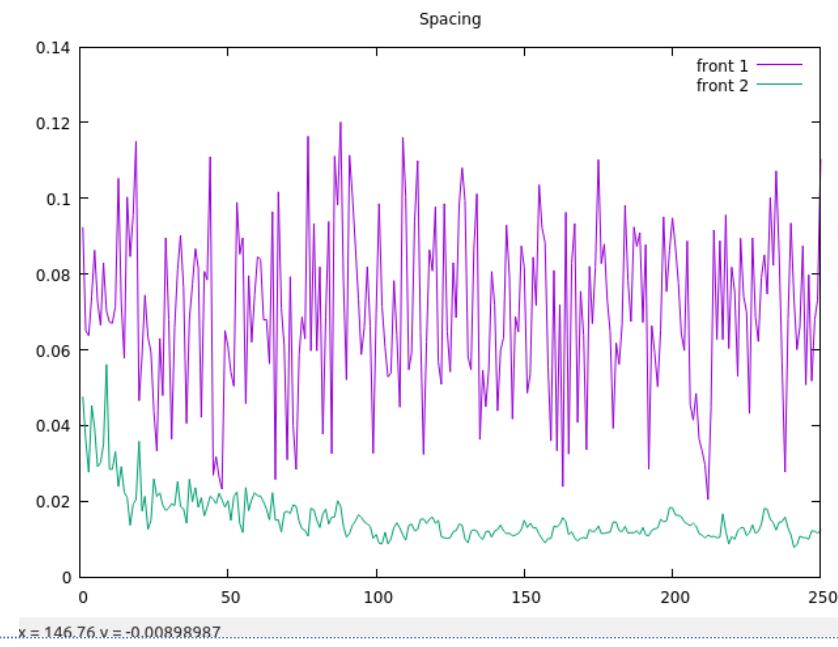


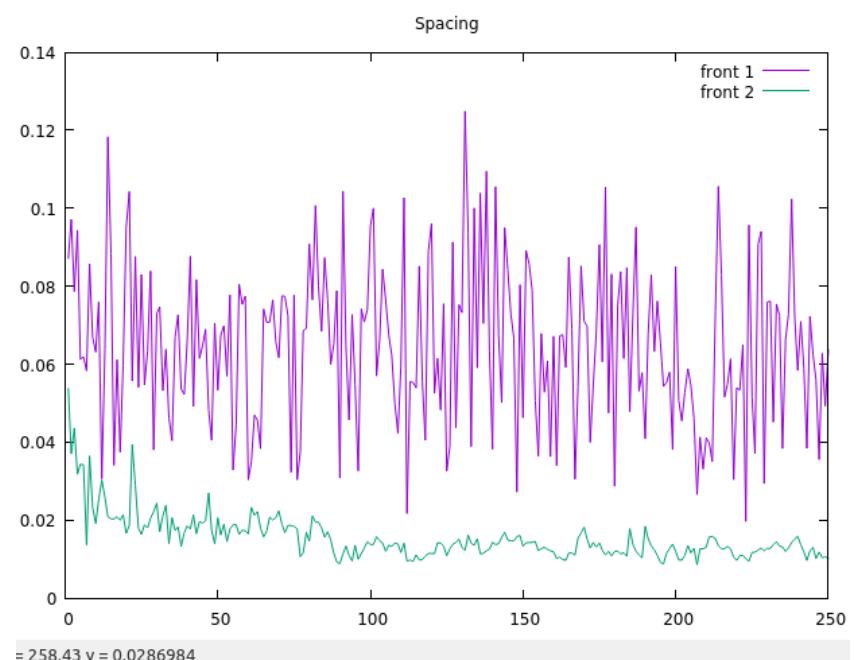
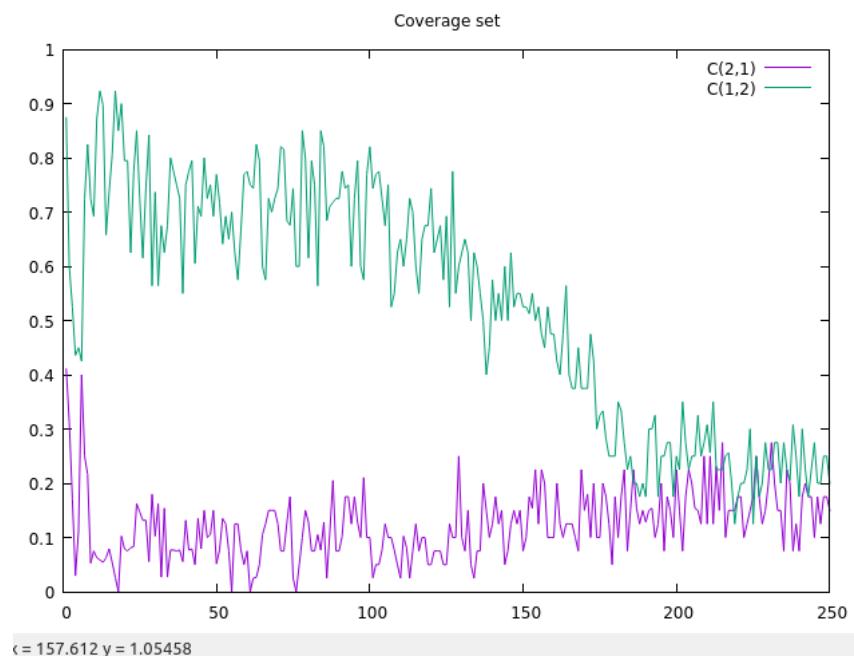
Por último, estas dos gráficas representan la solución obtenida con diferentes semillas una vez aplicada la mutación de Gauss, que logra suavizar la solución respecto a las anteriores. Se puede observar un cambio notorio en el acercamiento de la solución al Pareto óptimo. También se puede ver el impacto de la aleatoriedad en la solución. Al ser un algoritmo estocástico, la solución depende mucho de la semilla que parte la población inicial y varía notablemente la solución. En esta última versión ya se ha hecho una comparativa con la solución generada por NSGA-II:



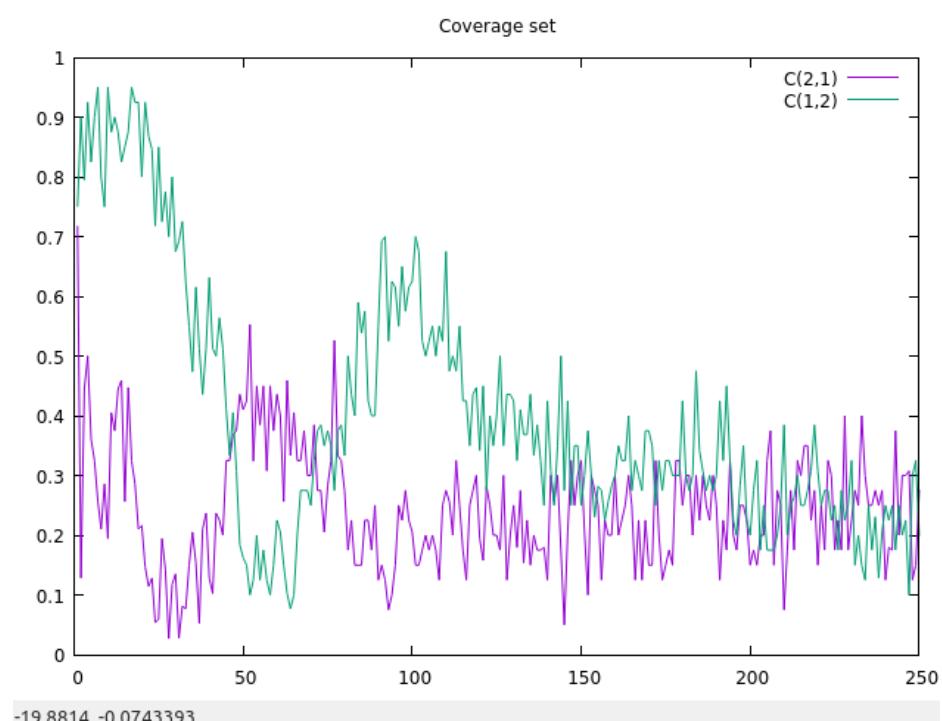
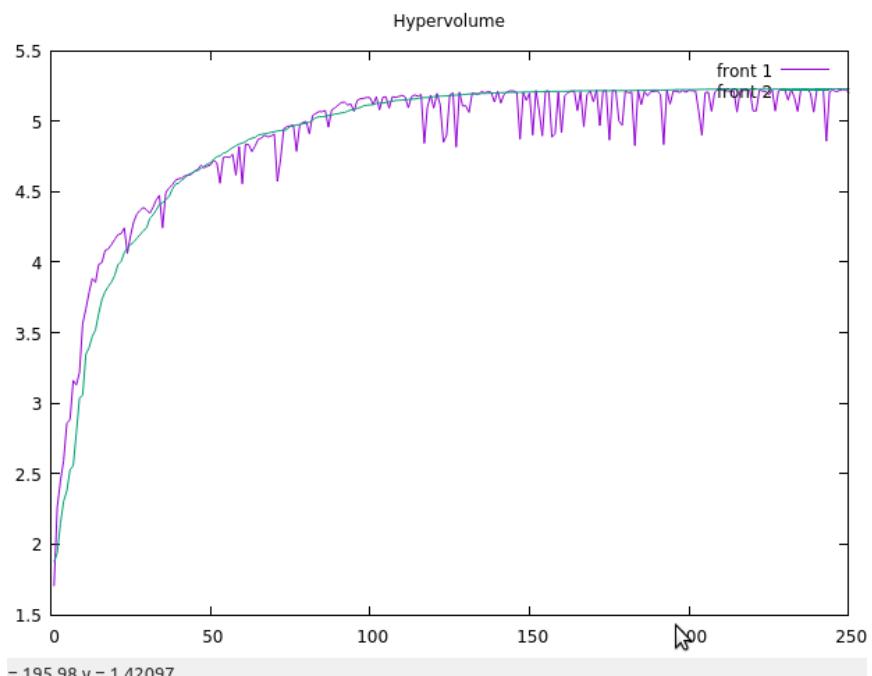
### 1.1.1.- SEMILLA 7

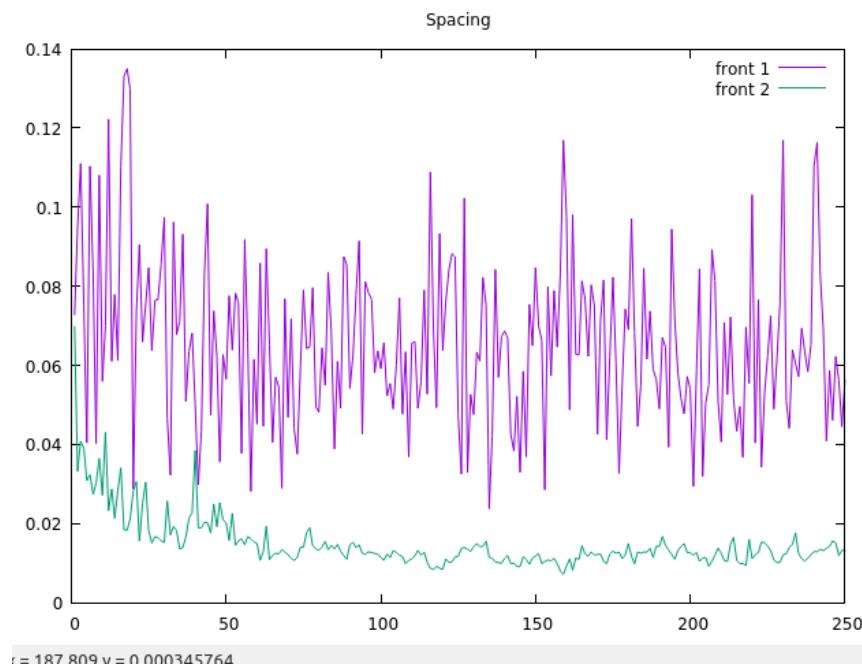






### 1.1.3.- SEMILLA 2





#### 1.1.4.- ESTADÍSTICAS SOBRE MÉTRICAS (10 EJECUCIONES)

```
usuario@LinuxMintVM: ~/Escritorio/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
max 30 characters)      hypervol.out      hypervol2.out      Makefile
Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)
Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ef[1]=0.9581990000      spacing2.out      zdt3_all_
ef[2]=1.0628800000      spacing2.out      popmp40g250.out      zdt3_all_
front 1 hypervolume: 1.0150547693      zdt3_all_
front 1 spacing: 0.0146897451      popmp100g100.out
configuration UNKNOWN p40g250: Hypervolume mean: 0.987792      zdt3_all_
configuration UNKNOWN p40g250: Hypervolume standard deviation: 0.044541      popmp100g100.out
configuration NSGAII p40g250: Hypervolume mean: 1.00786      zdt3_all_
configuration NSGAII p40g250: Hypervolume standard deviation: 0.0124088      popmp100g100.out
configuration UNKNOWN p40g250: Spacing mean: 0.0314527      zdt3_all_
configuration UNKNOWN p40g250: Spacing standard deviation: 0.00760969      popmp100g100.out
configuration NSGAII p40g250: Spacing mean: 0.0145729      zdt3_all_
configuration NSGAII p40g250: Spacing standard deviation: 0.00105278      popmp100g100.out
usuario@LinuxMintVM:~/Escritorio/METRICS$
```

Como podemos observar, se obtiene un hipervolumen similar en el algoritmo NSGAII (algo superior) y el algoritmo de agregación (llamado UNKNOWN). Además, el spacing es menor en NSGAII respecto al implementado. Por tanto, el algoritmo no supera a NSGAII con 40 individuos y 250 generaciones, aunque se acerca bastante.

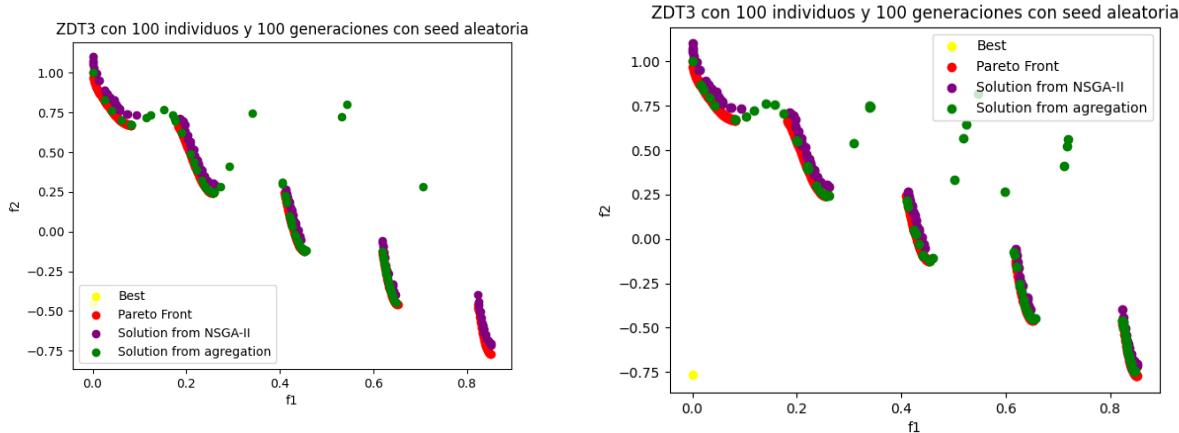
En cuanto a las gráficas sobre métricas vistas anteriormente, podemos destacar varios patrones generados en las tres ejecuciones:

Aunque vemos que finalmente se alcanza un hipervolumen mayor en NSGAII, con el algoritmo basado en agregación obtenemos con menor generaciones un hipervolumen más alto (véase a las 50 generaciones). Por tanto, se alcanza mayor hipervolumen con menor generaciones (menos cómputo). Sin embargo, cuando aplicamos más generaciones NSGAII es óptimo.

El Coverage Set depende de la ejecución y suele estar muy disputado entre ambos algoritmos. En las primeras ejecuciones con semilla 7 y 5 se puede ver como durante casi toda la ejecución (en el caso de la ejecución con semilla 5 es durante toda la ejecución) hay mayor porcentaje de soluciones de NSGAII (2) superadas por las soluciones del algoritmo basado en agregación (1). También es verdad que durante las últimas generaciones aproximadamente acaban casi con el mismo porcentaje superadas tanto un algoritmo como el otro. En el rango de 50-100 generaciones, las soluciones proporcionadas por el algoritmo NSGAII se ven muy superadas por las soluciones proporcionadas por el algoritmo basado en agregación.

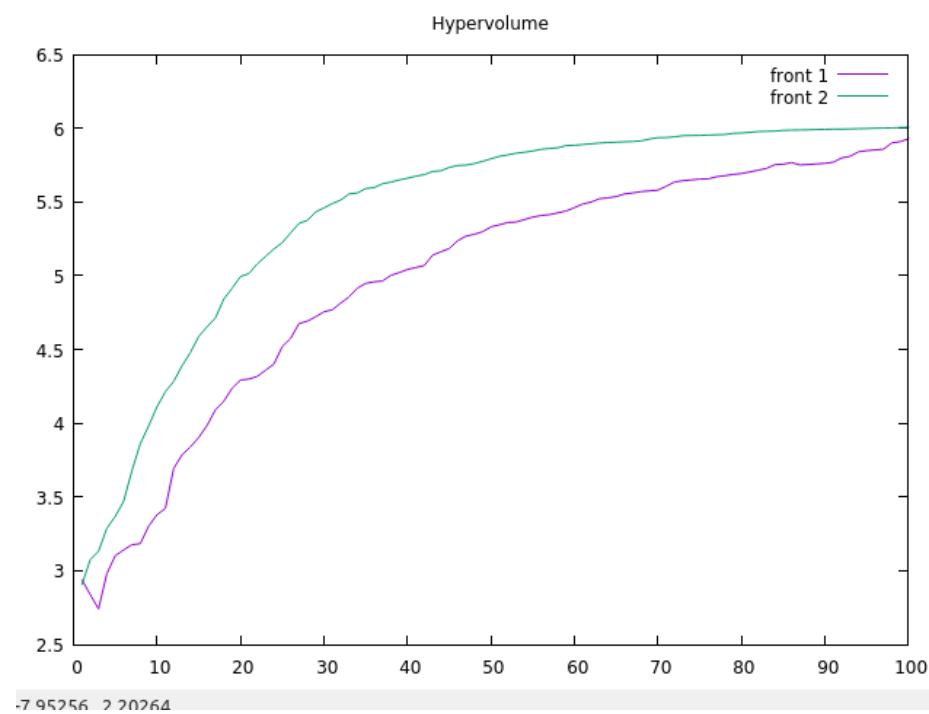
En cuanto al Spacing, sin duda el algoritmo NSGAII proporciona un conjunto de soluciones mejor distribuidas respecto al algoritmo implementado, ya que los valores de spacing son más bajos. Por otro lado, el spacing generado por el nuevo algoritmo depende mucho de la generación, hay muchos cambios en los valores. Por último, el spacing mejora muy poco si nos fijamos en la primera generación y en la última.

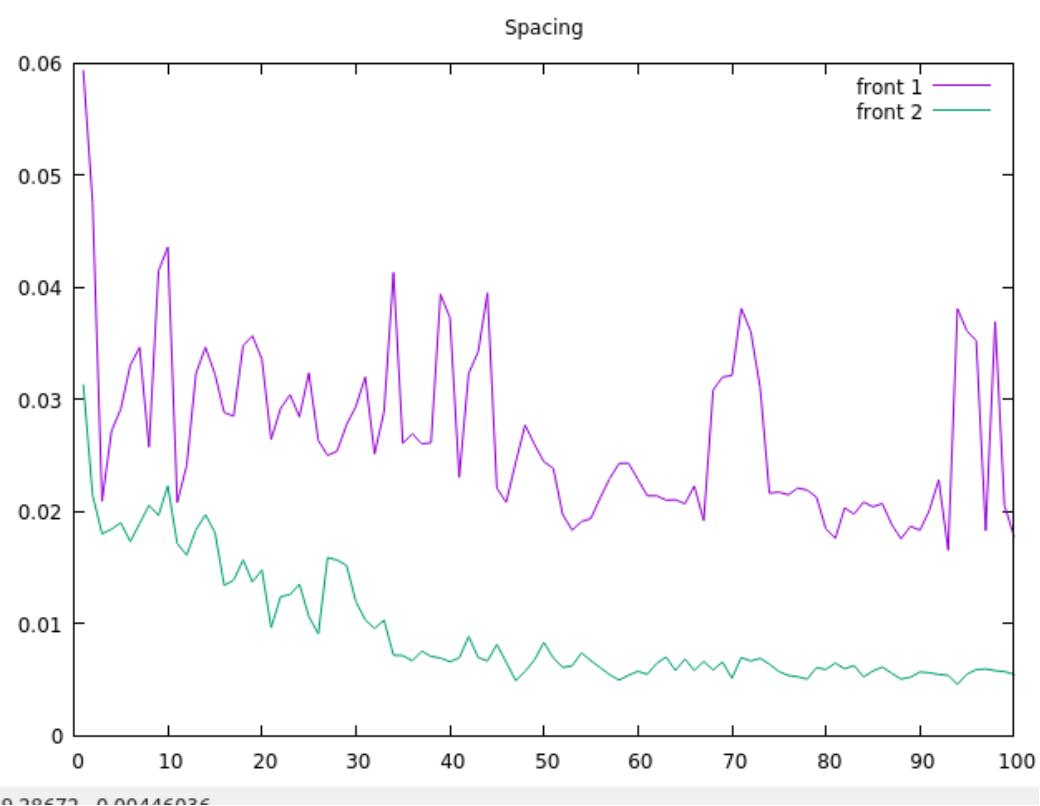
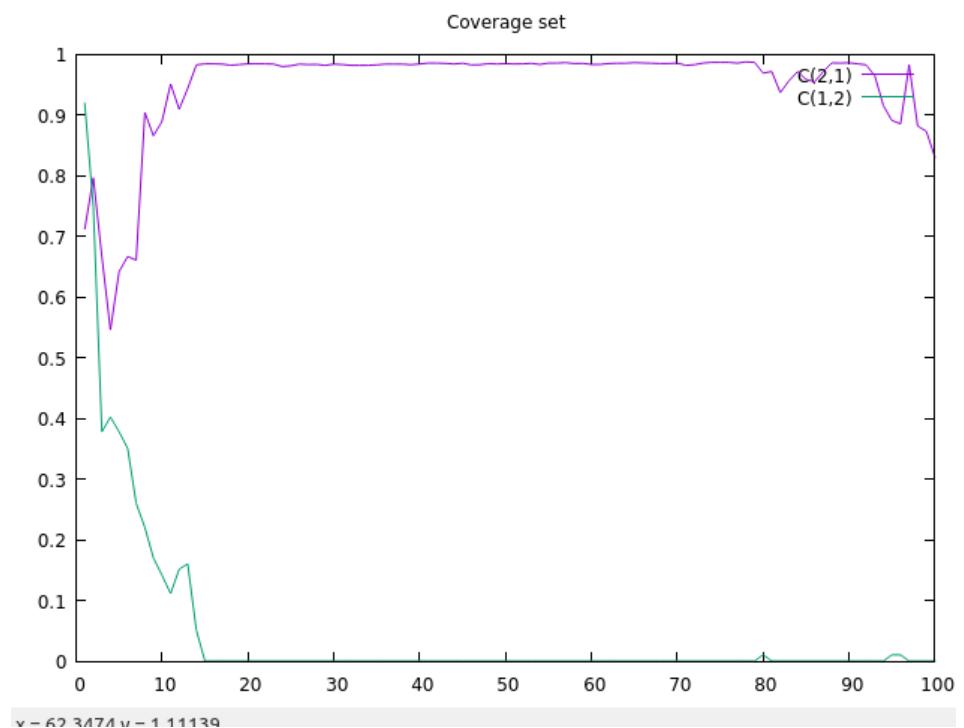
### 1.2.- Población de 100 individuos y 100 generaciones



En ambas gráficas podemos ver una diferencia de cobertura entre ambos algoritmos. En la primera gráfica podemos ver como la zona más baja del frente (respecto al eje Y) no recoge ninguna solución el algoritmo basado en agregación. Sin embargo, en la segunda gráfica se ajusta mejor en dicha zona las soluciones generadas por este algoritmo. Por tanto, podemos observar como la cobertura de las diferentes zonas del frente depende de las semillas que recoja el algoritmo. Por otro lado, NSGA II no presenta ningún desvío respecto del Pareto en sus soluciones, mientras que nuestra solución no llega a minimizar del todo algunas de las soluciones obtenidas.

1.2.1.- SEMILLA 2

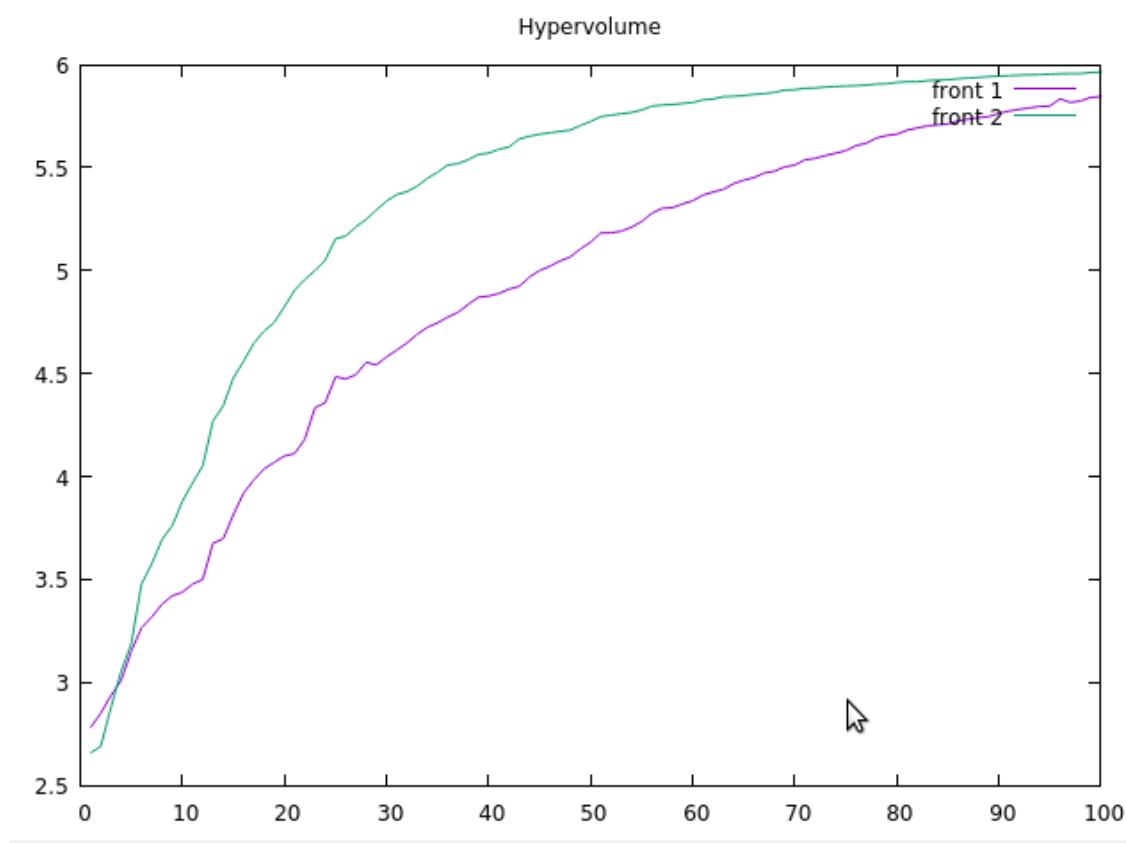


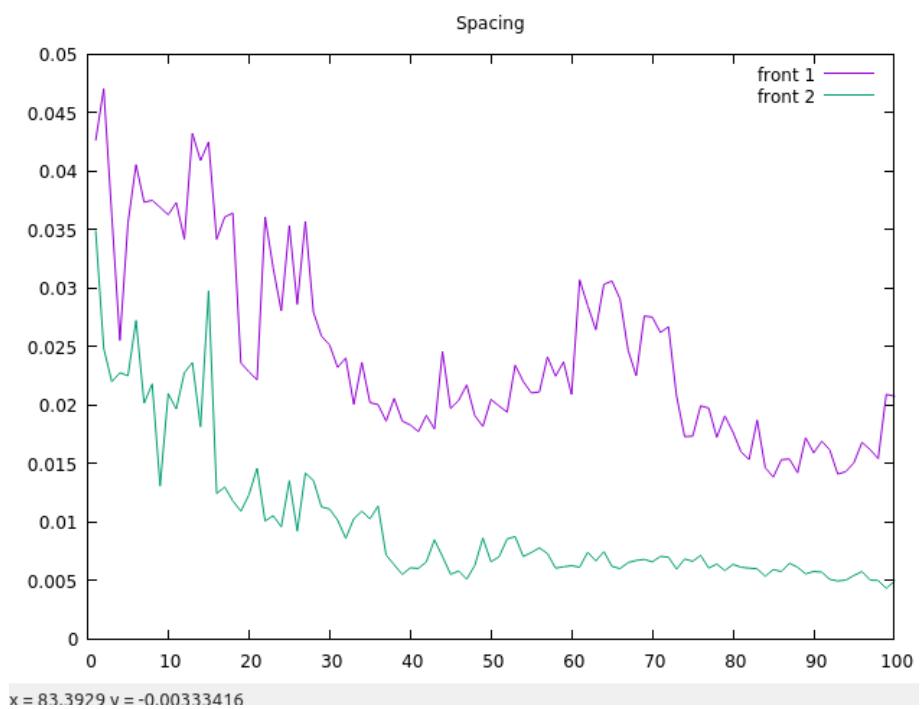
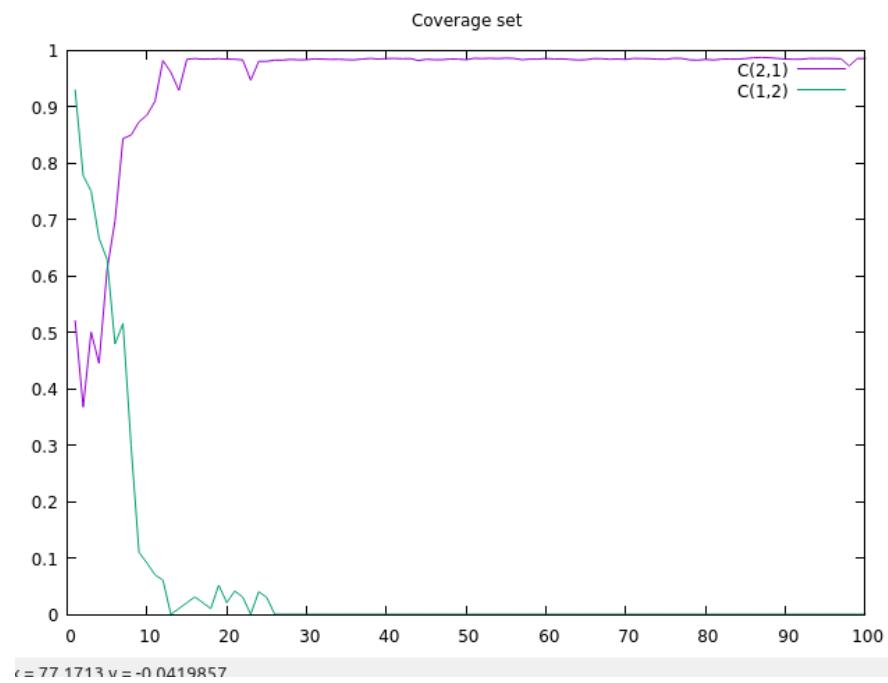


Aunque no podemos establecer conclusiones con una sola ejecución, parece ser que con 100 individuos y con 100 generaciones el spacing mejora más, el hipervolumen empeora y no llega a ser mayor en ninguna fase de la ejecución y claramente las soluciones del algoritmo basado en agregación son superadas por las soluciones de NSGAII.

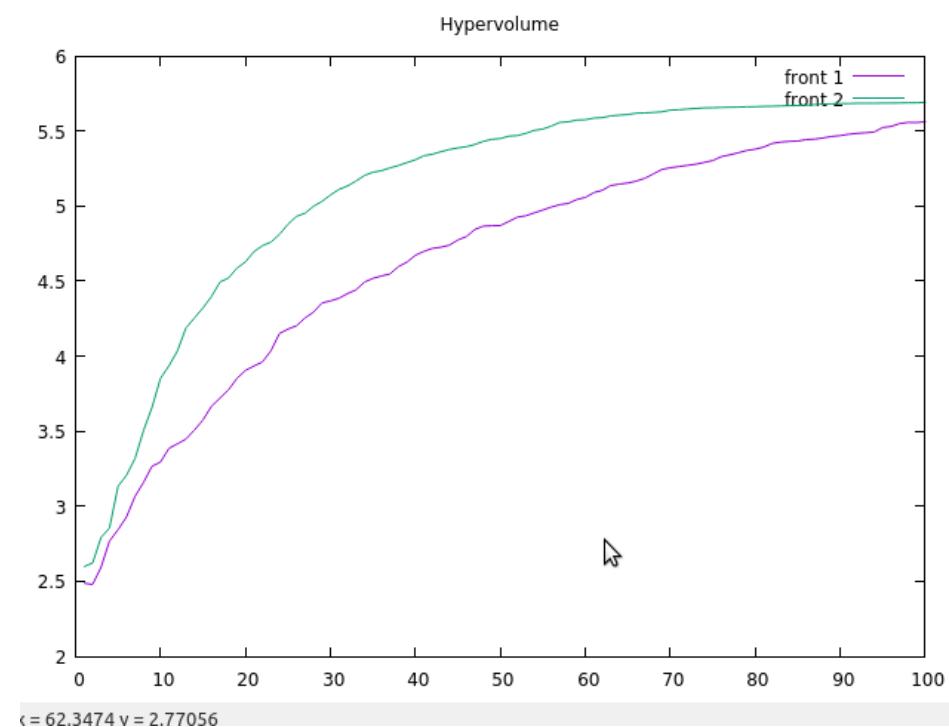
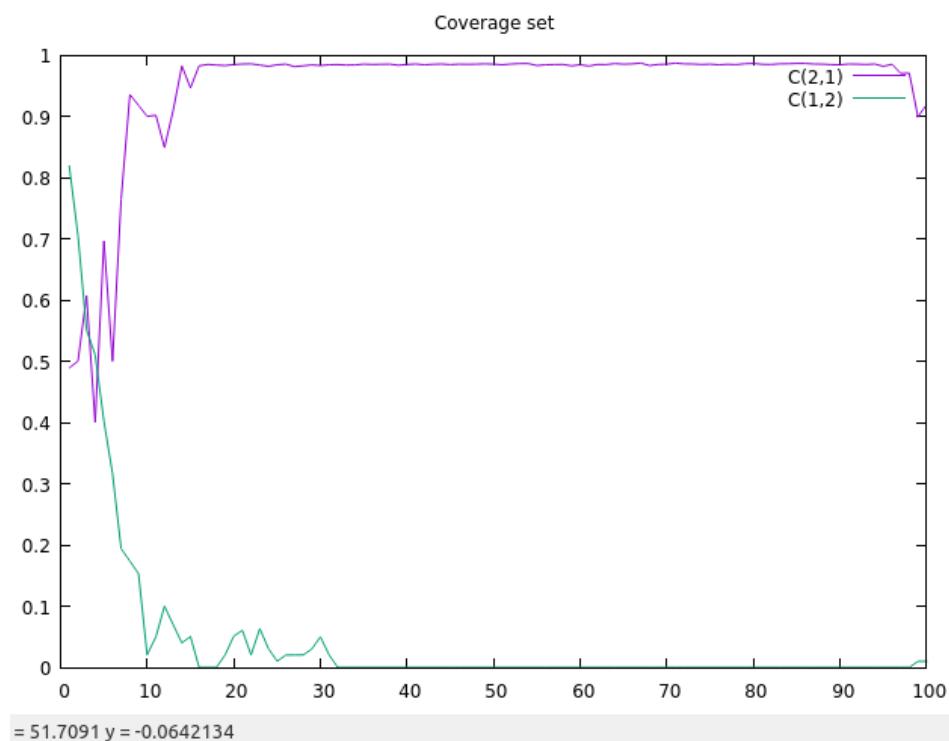
---

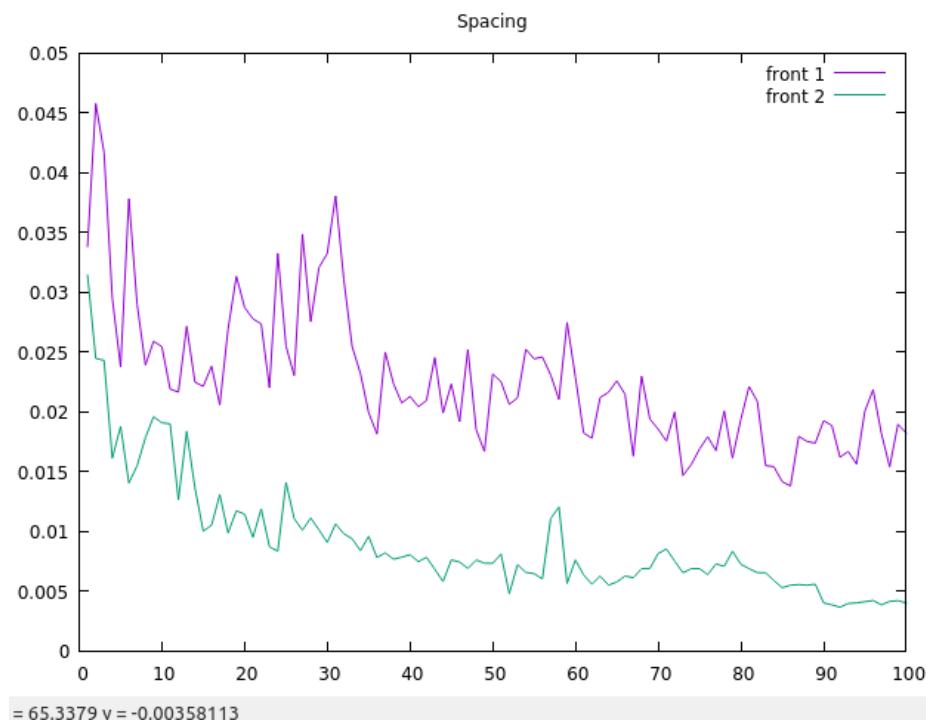
#### 1.2.2.- SEMILLA 8





### 1.2.3.- SEMILLA 5





#### 1.2.4.- ESTADÍSTICAS SOBRE MÉTRICAS (10 EJECUCIONES)

```
usuario@LinuxMintVM: ~/Escritorio/E1/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

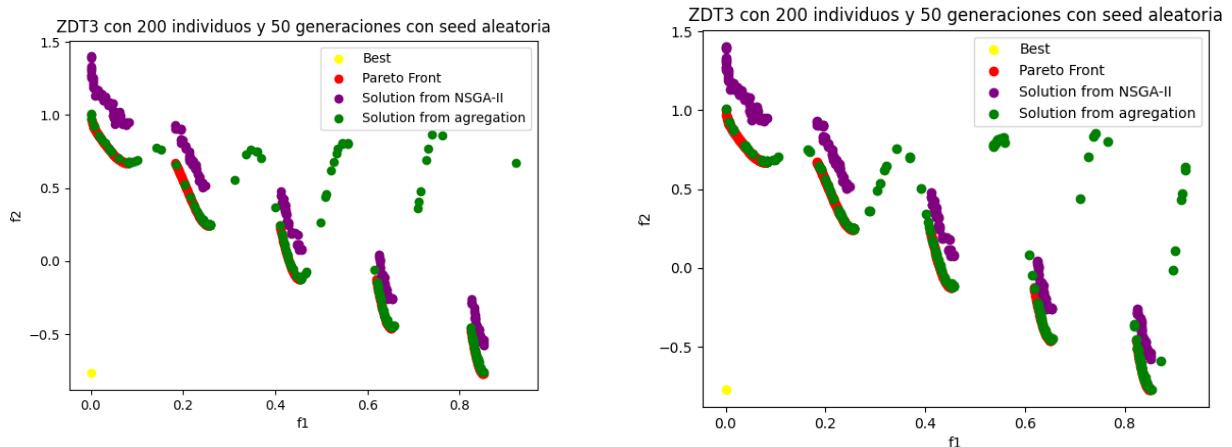
Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=0.9230377000
ref[2]=1.1615780000

Front 1 hypervolume: 0.9977827527
Front 1 spacing: 0.0047575538
Configuration UNKNOWN p100g100: Hypervolume mean: 1.01954
Configuration UNKNOWN p100g100: Hypervolume standard deviation: 0.0367299
Configuration NSGAIID p100g100: Hypervolume mean: 1.00601
Configuration NSGAIID p100g100: Hypervolume standard deviation: 0.00787497
Configuration UNKNOWN p100g100: Spacing mean: 0.0379932
Configuration UNKNOWN p100g100: Spacing standard deviation: 0.0103846
Configuration NSGAIID p100g100: Spacing mean: 0.00488778
Configuration NSGAIID p100g100: Spacing standard deviation: 0.000611439
usuario@LinuxMintVM:~/Escritorio/E1/METRICS$
```

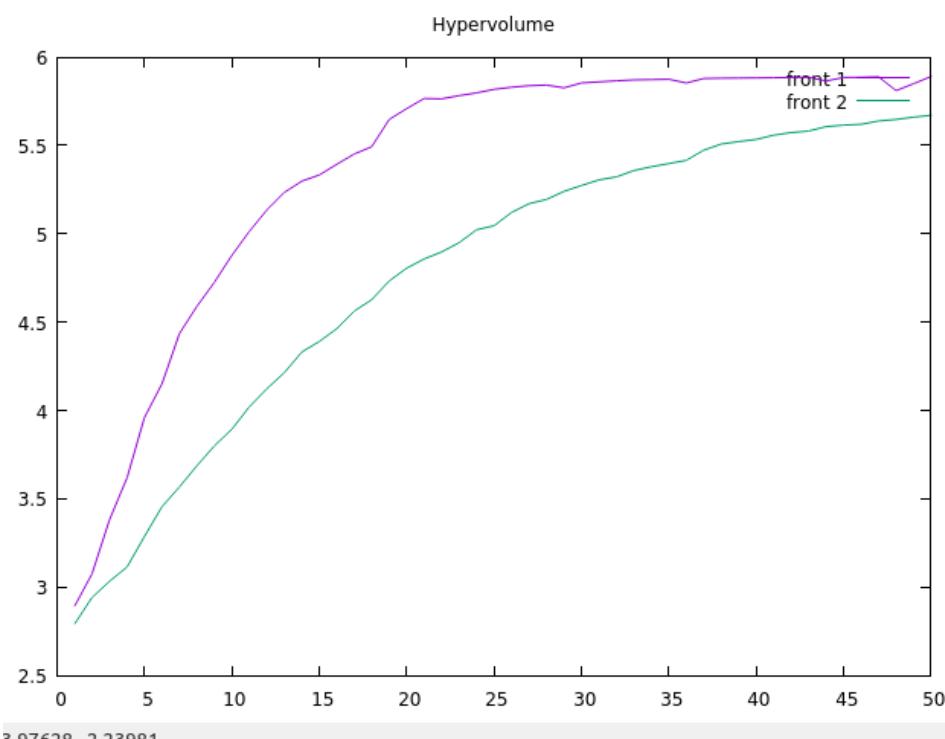
Como podemos ver, para este caso, el hipervolumen medio del algoritmo propuesto es ligeramente superior al NSGA II (diferencia muy pequeña) con una desviación muy grande y el valor medio del spacing es bastante mayor.

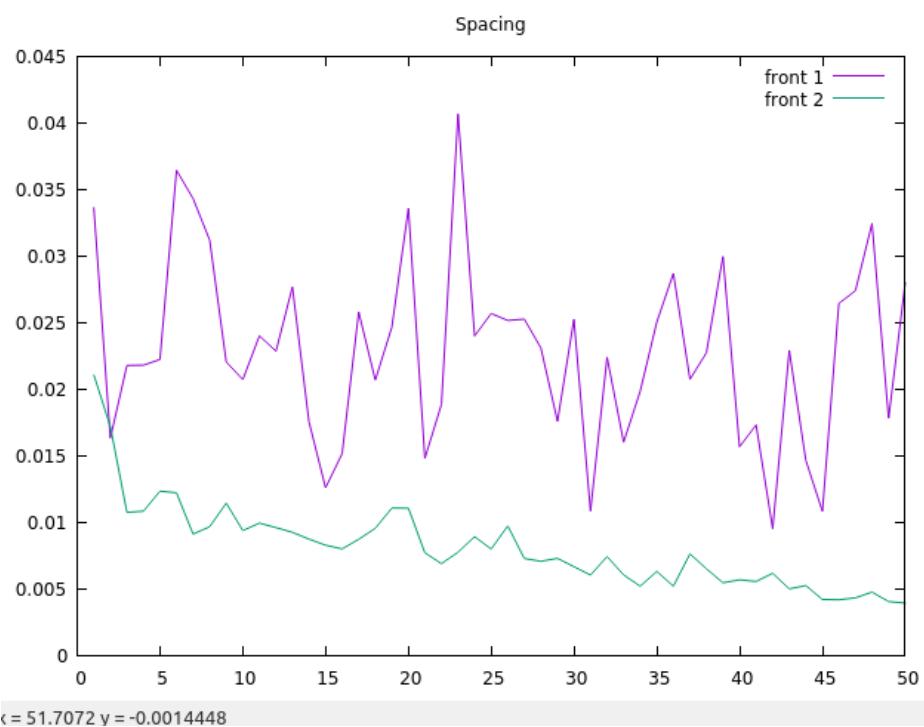
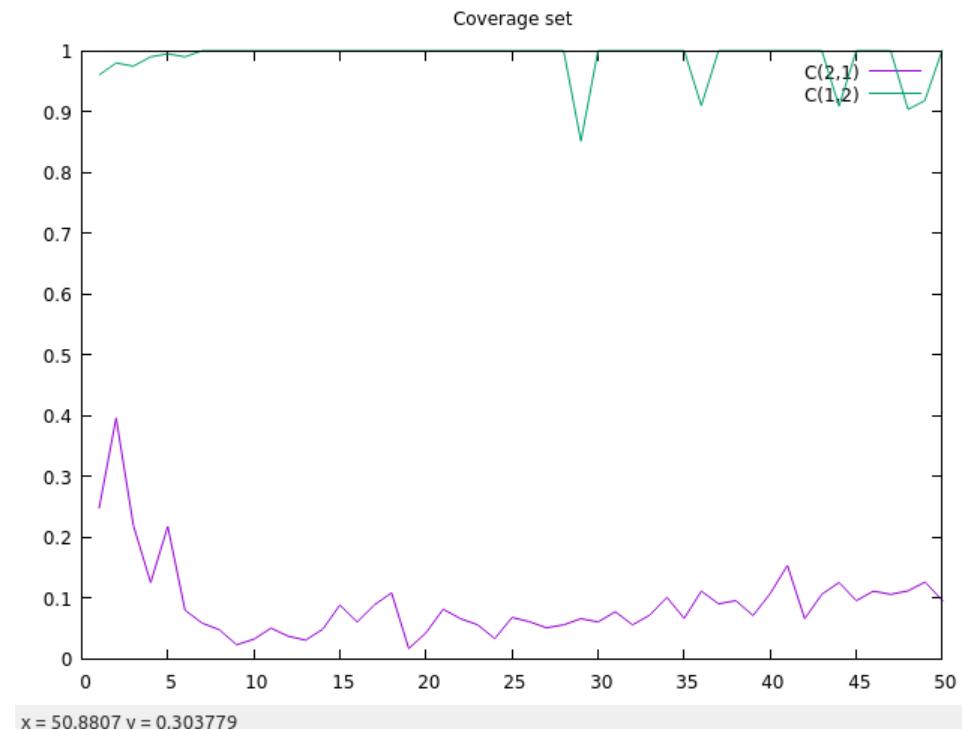
### 1.3.- Población de 200 individuos y 50 generaciones



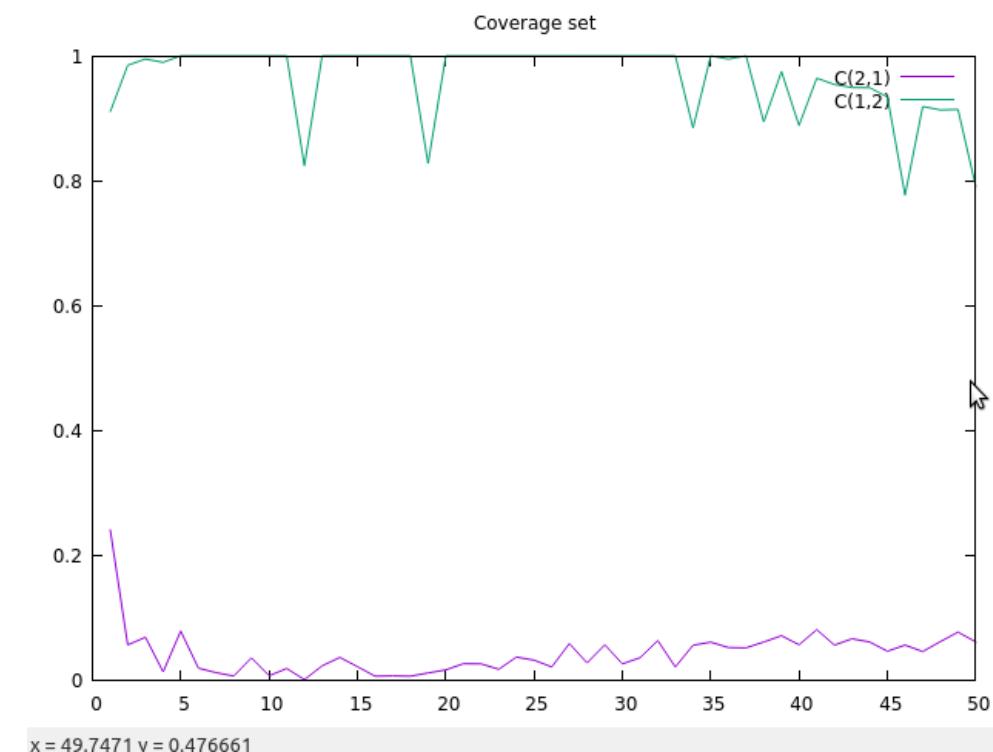
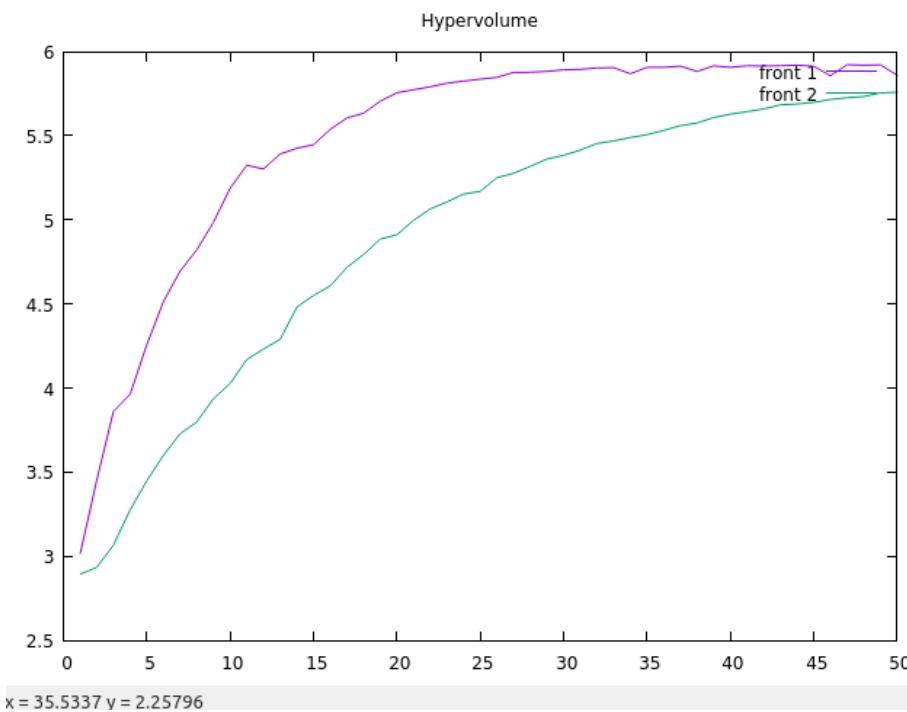
Podemos ver cómo claramente parece que el algoritmo propuesto supera al algoritmo NSGAII en cuanto a aproximación al Pareto se refiere. Sin embargo, la distribución de NSGAII sigue siendo bastante mejor que la del algoritmo de agregación. Para asegurarnos de que esta configuración realmente favorece al algoritmo propuesto a la comparativa de ambos, vamos a analizar las métricas generadas.

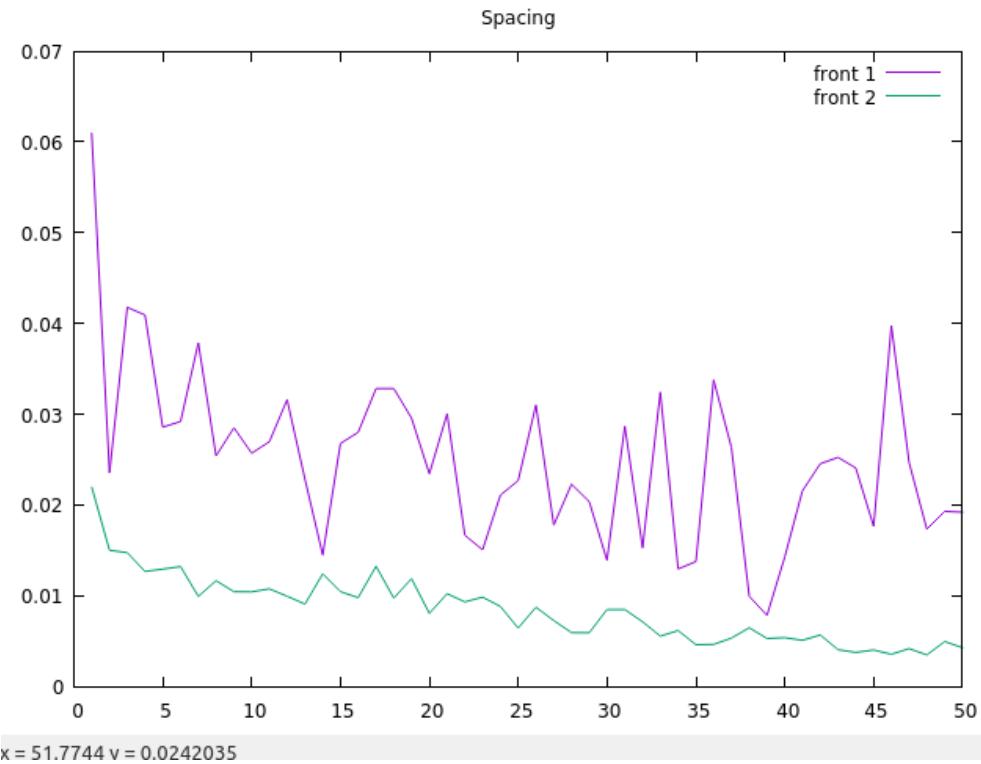
#### 1.3.1.- SEMILLA 1



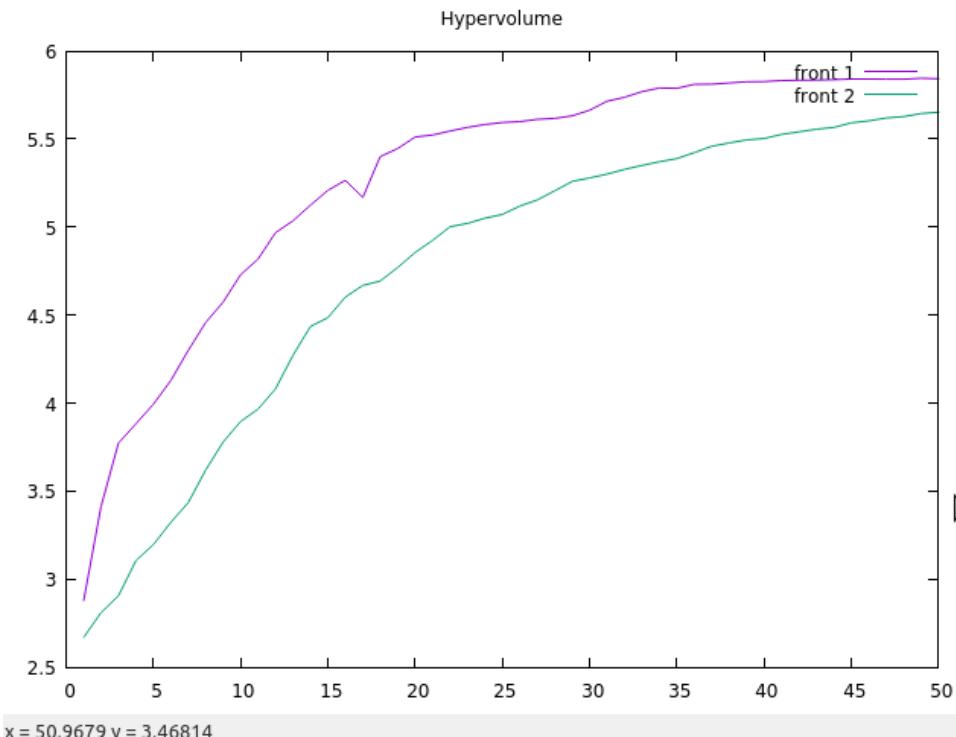


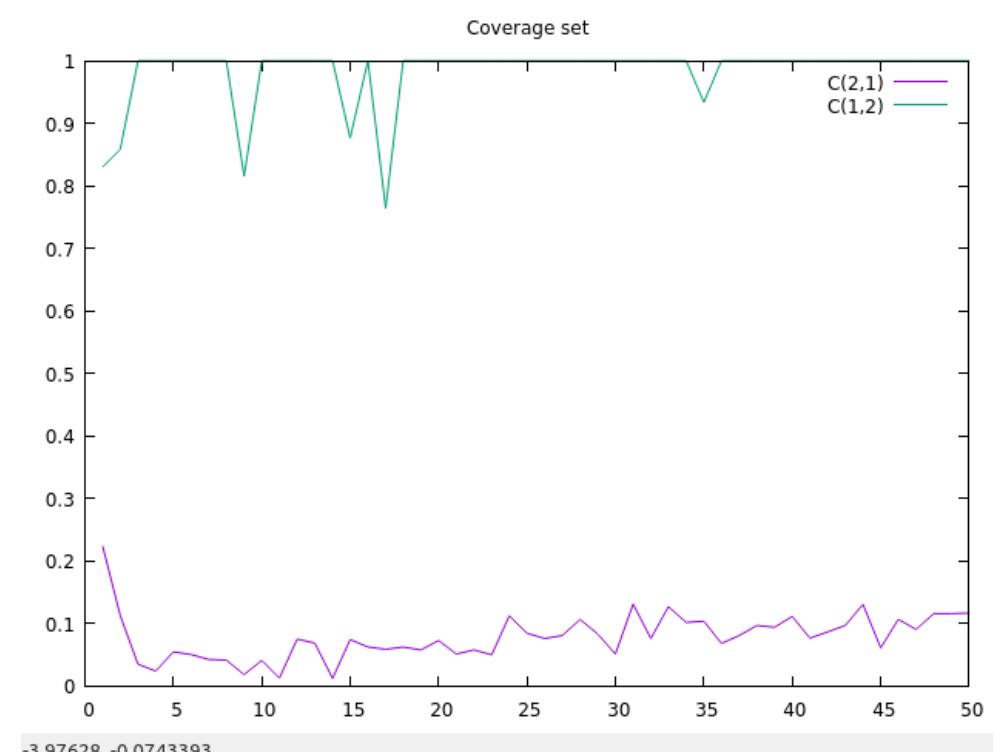
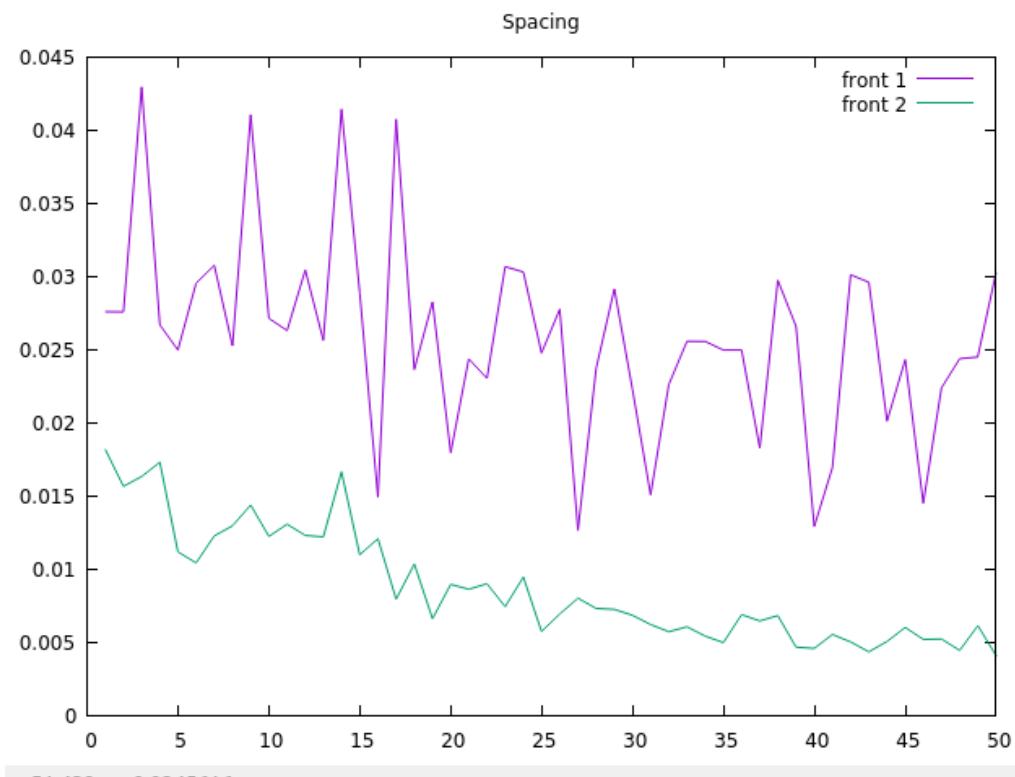
### 1.3.2.- SEMILLA 4





### 1.3.3.- SEMILLA 10





#### 1.3.4.- ESTADÍSTICAS SOBRE MÉTRICAS (10 EJECUCIONES)

```
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)      popmp100g100.out      popmp100g100...      popmp200g50.out
                           unknown.out

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=0.8558197000
ref[2]=2.0433990000
Front 1 hypervolume: 1.4985652992
Front 1 spacing: 0.0040560592
Configuration UNKNOWN p200g50: Hypervolume mean: 1.66798
Configuration UNKNOWN p200g50: Hypervolume standard deviation: 0.00593851
Configuration NSGAIID p200g50: Hypervolume mean: 1.49949
Configuration NSGAIID p200g50: Hypervolume standard deviation: 0.0187247
Configuration UNKNOWN p200g50: Spacing mean: 0.0252237
Configuration UNKNOWN p200g50: Spacing standard deviation: 0.00637858
Configuration NSGAIID p200g50: Spacing mean: 0.00410533
Configuration NSGAIID p200g50: Spacing standard deviation: 0.000534258
usuario@LinuxMintVM:~/Escritorio/METRIC$
```

Con esta configuración podemos concluir que el algoritmo basado en agregación supera como solución al algoritmo NSGA II, ya que presenta un hipervolumen medio bastante superior (con una desviación mucho menor).

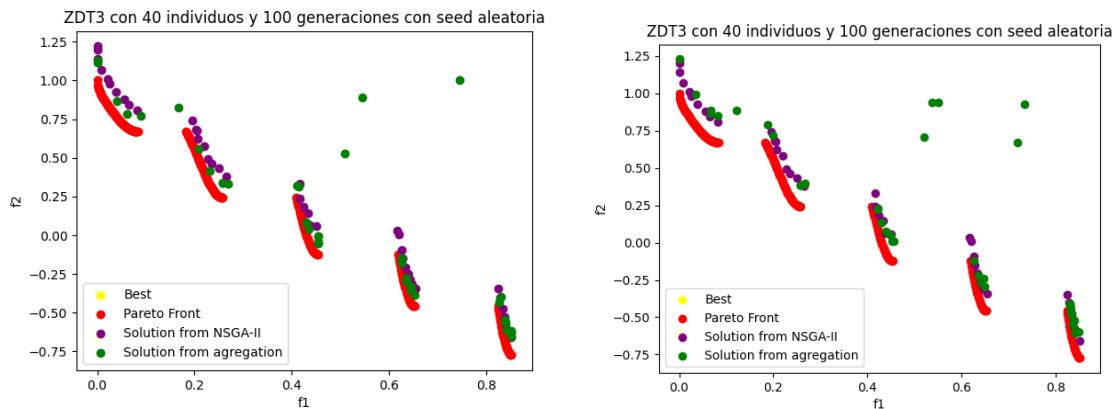
Por otro lado, el spacing medio sigue siendo mayor, parece que no supera en ninguna configuración con 10000 evaluaciones a NSGA II, pero es más bajo con respecto a otras configuraciones.

El Coverage Set según las gráficas es claramente superior, lo que hace que tenga sentido un valor de hipervolumen más alto, ya que las soluciones del algoritmo de agregación se acercan más al frente, lo cual quiere decir que son mínimas y ya contienen a la mayoría de soluciones de NSGA II.

## 2.- Resultados experimentales con capacidad de cómputo de 4000 evaluaciones

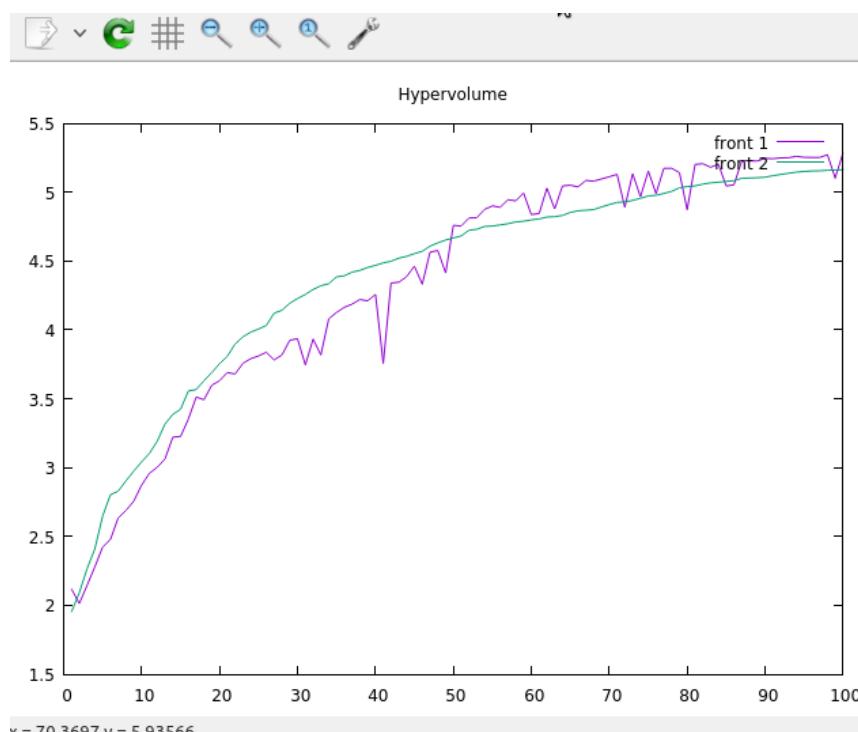
### 2.1.- Población de 40 individuos y 100 generaciones

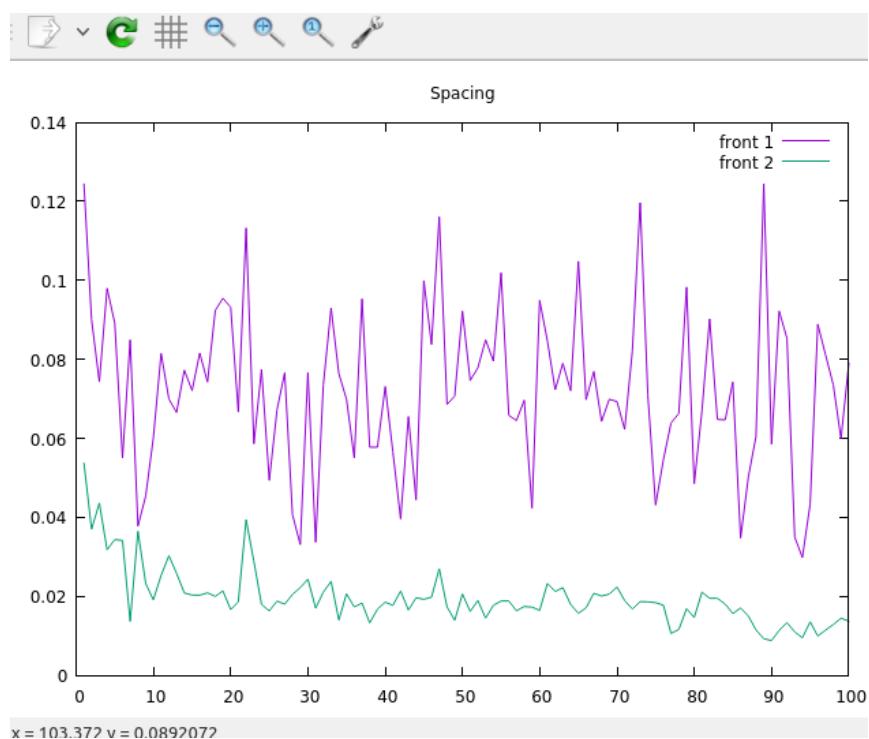
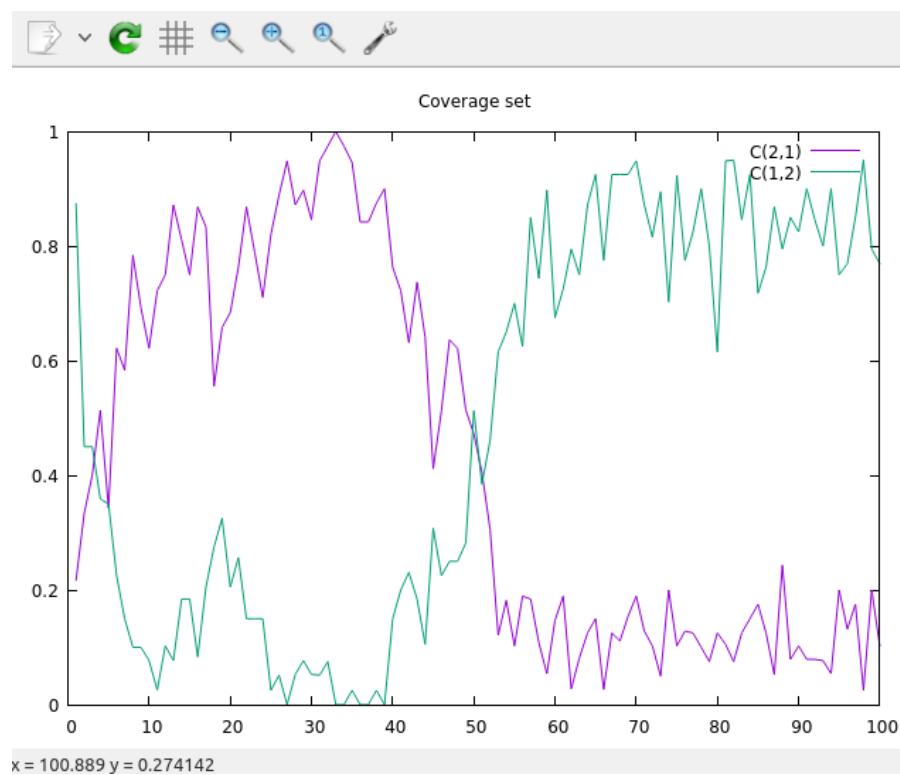
Los resultados gráficos registrados en un primer momento son los siguientes:



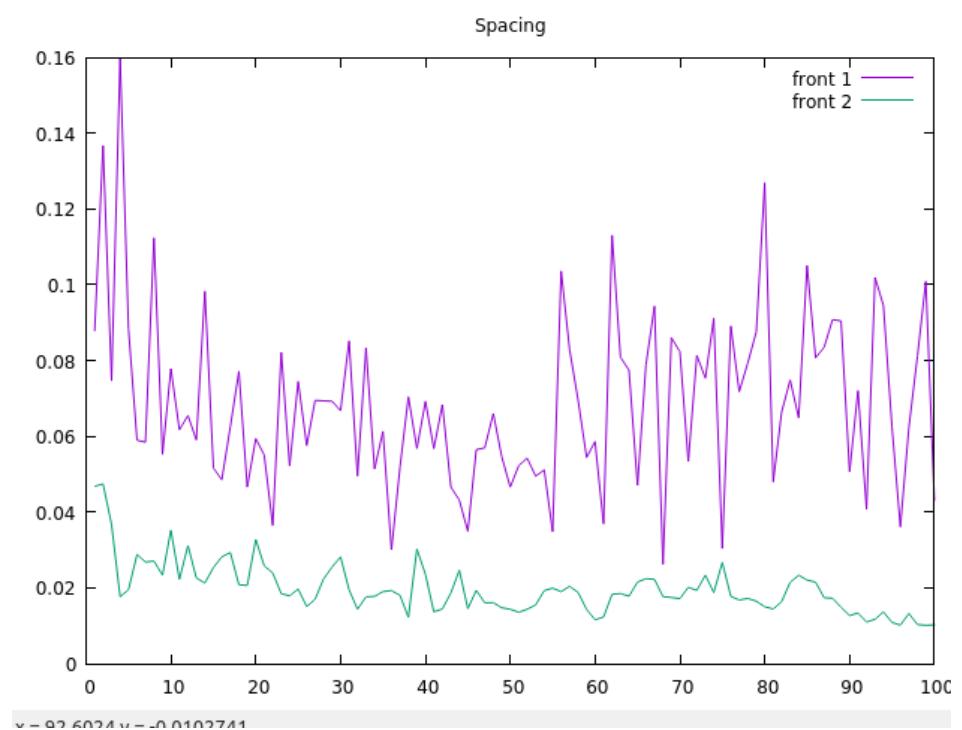
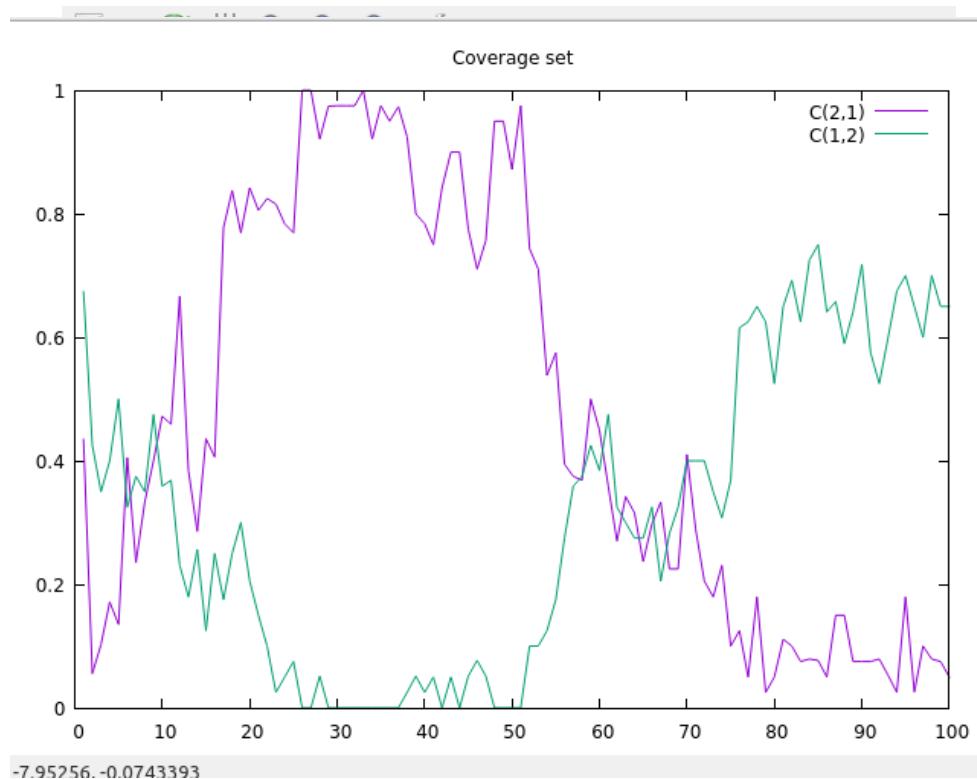
Podemos presenciar la principal diferencia que existe entre el algoritmo de NSGAII y el implementado en este trabajo. El NSGAII mantiene todas sus soluciones aproximadas al Pareto óptimo, mientras que el algoritmo basado en agregación proporciona algunas soluciones que no llegan a minimizarse, pero se encuentran en la frontera entre las soluciones posibles (ver Fig 1).

#### 2.1.2.- SEMILLA 2

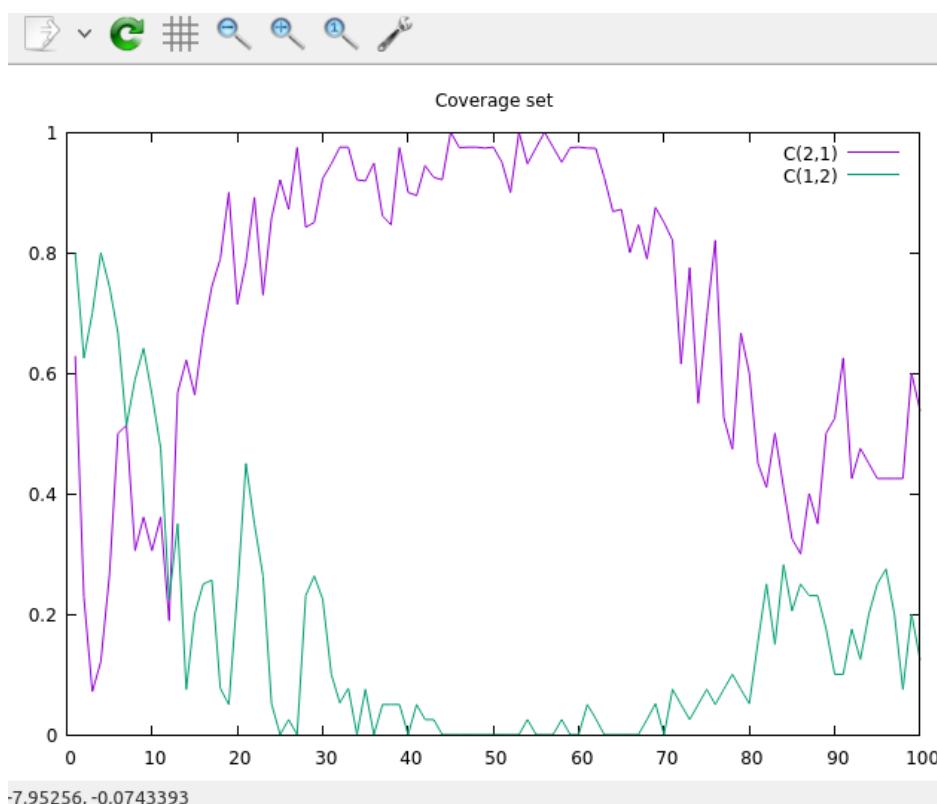
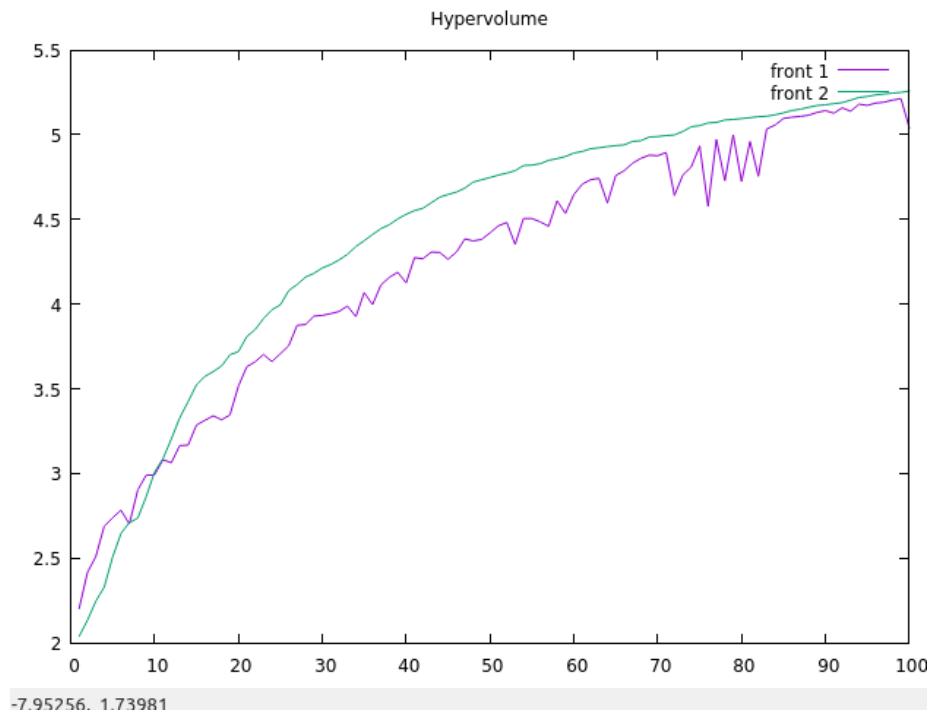


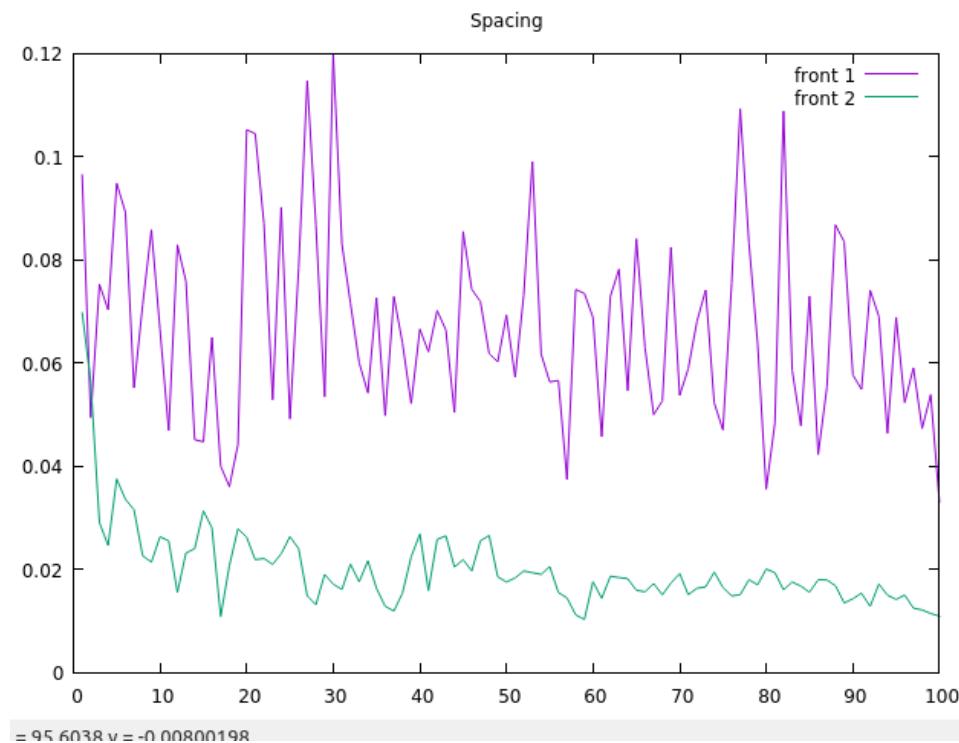


### 2.1.2.- SEMILLA 3



### 2.1.3.- SEMILLA 8





#### 2.1.4.- ESTADÍSTICAS SOBRE MÉTRICAS

```
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=0.9935885000    4dp40g100stallref.in    4dp40g100stsingl
ref[2]=5.5609560000    4dp40g100stsinglba
sic.in

Front 1 hypervolume: 5.4103289721
Front 1 spacing: 0.0105973558
Configuration UNKNOWN p40g100: Hypervolume mean: 5.34715
Configuration UNKNOWN p40g100: Hypervolume standard deviation: 0.122147
Configuration NSGAIID p40g100: Hypervolume mean: 5.38011
Configuration NSGAIID p40g100: Hypervolume standard deviation: 0.047646
Configuration UNKNOWN p40g100: Spacing mean: 0.0585758
Configuration UNKNOWN p40g100: Spacing standard deviation: 0.0201767
Configuration NSGAIID p40g100: Spacing mean: 0.0146074
Configuration NSGAIID p40g100: Spacing standard deviation: 0.00807253
```

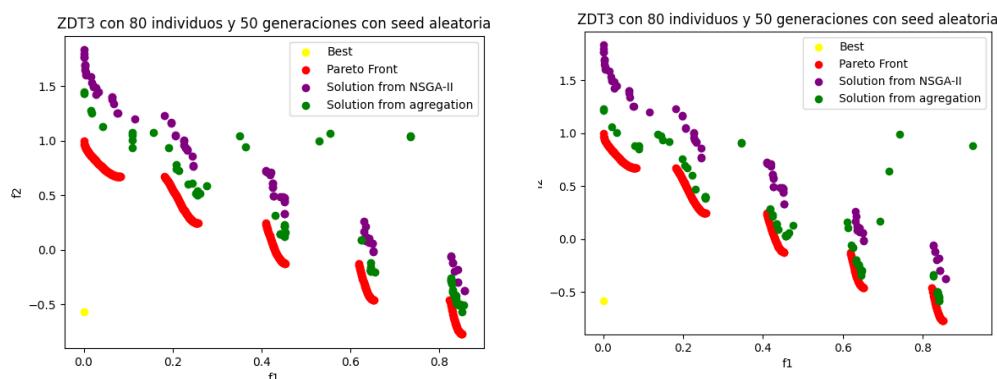
El hipervolumen medio es muy similar en ambos algoritmos, aunque podríamos decir que NSGAII supera ligeramente al algoritmo de agregación (es algo mayor y presenta menor desviación).

El Spacing sigue siendo mejor el algoritmo NSGAII (presenta menores valores). Cabe destacar la diferencia de desviación entre ambos. La desviación estándar del algoritmo a implementar es mucho mayor (un patrón que se lleva repitiendo en estas pruebas). Esto quiere decir que varía mucho el valor de spacing en cada una de las ejecuciones.

El Coverage Set según las tres ejecuciones presentadas anteriormente depende bastante de la ejecución. Las dos primeras ejecuciones acaban con mayor porcentaje de soluciones de NSGAII superadas por el algoritmo de agregación. Sin embargo, en el rango de 15-55 generaciones las soluciones proporcionadas por NSGAII en porcentaje superan a las del algoritmo implementado. Con menor cómputo se consiguen mejores soluciones en NSGAII. Esto tiene relación con el hipervolumen. En las tres ejecuciones en ese rango de generaciones el hipervolumen es mayor en NSGAII (las soluciones están más cerca del Pareto y son mejores).

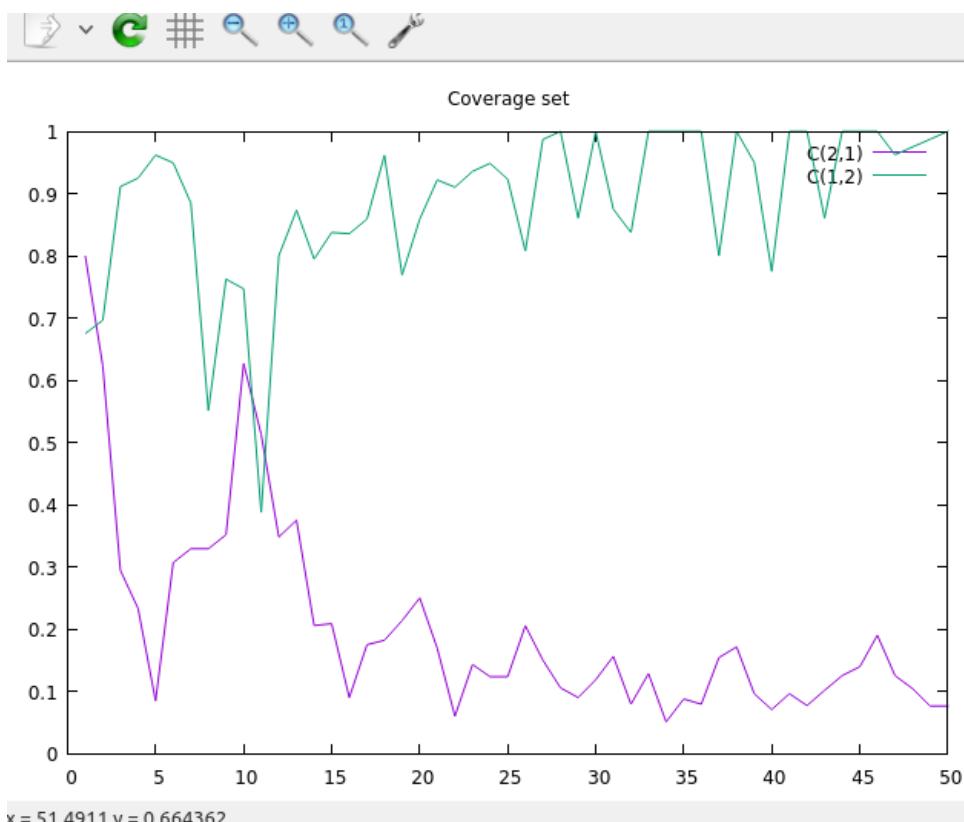
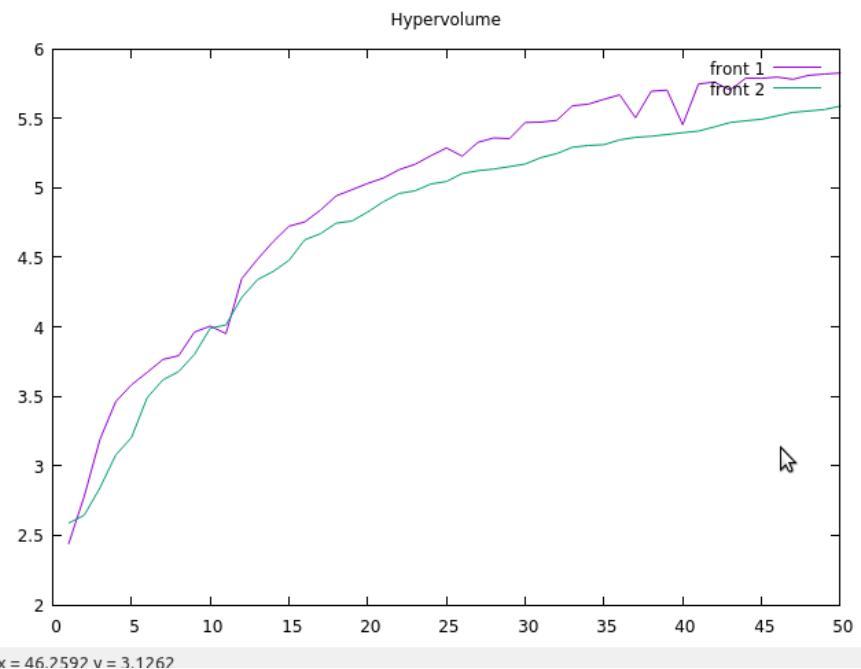
Con esta configuración el algoritmo propuesto lucha y se acerca a NSGAII pero no llega a superarle.

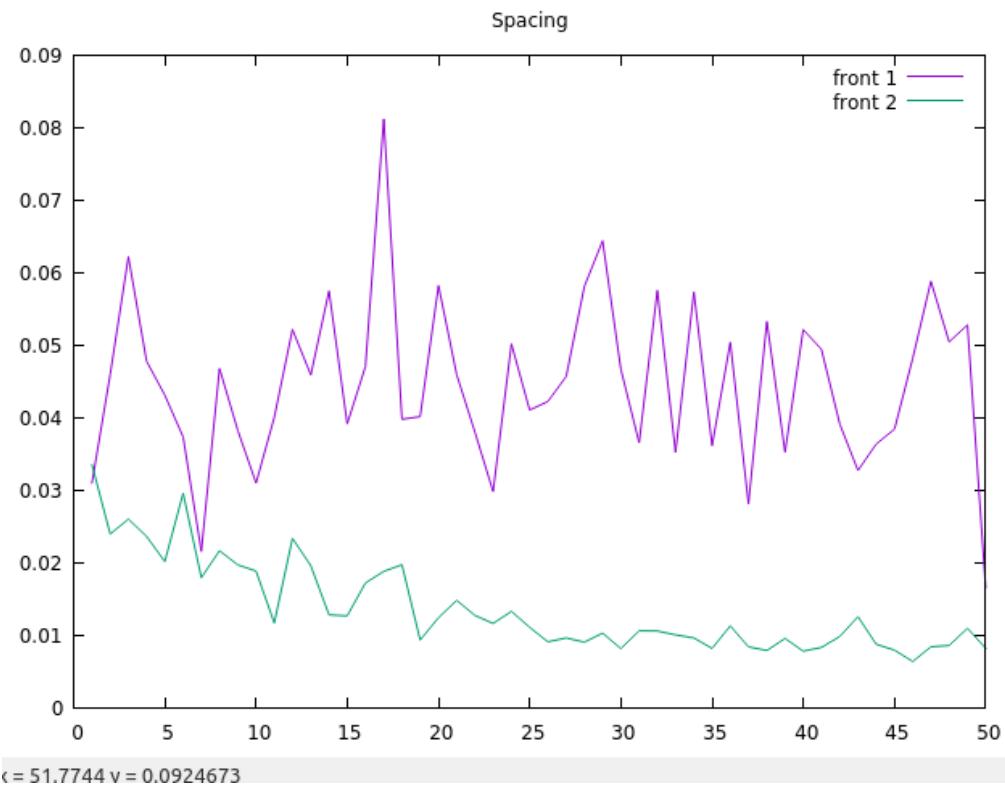
## 2.2.- Población de 80 individuos y 50 generaciones



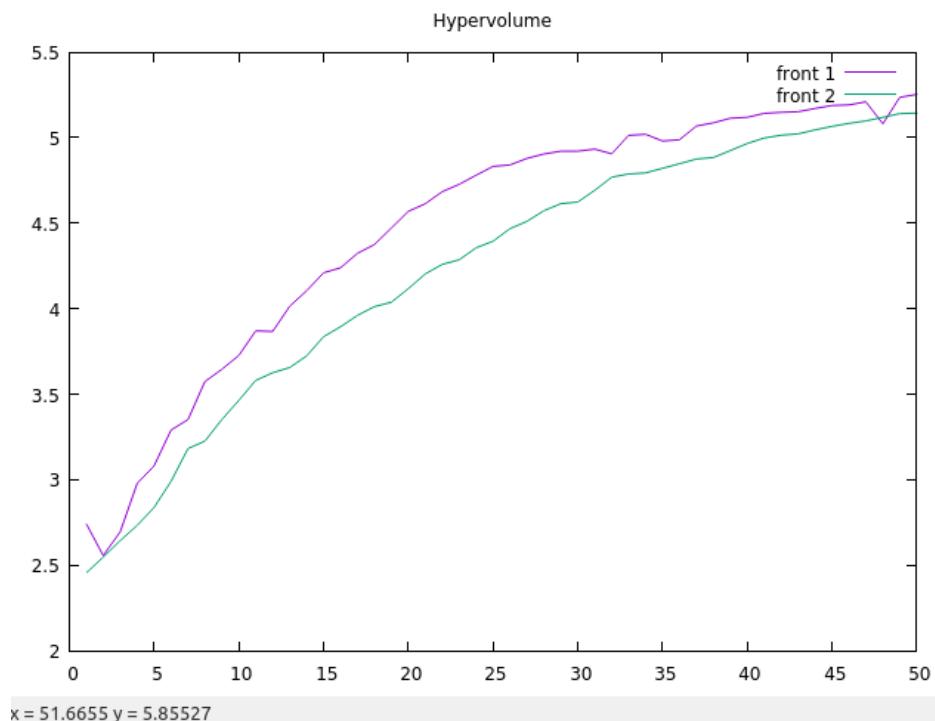
Aunque todavía no pueden extraerse conclusiones parece aparentemente que el algoritmo de agregación proporciona con esta configuración una solución mejor (en cuanto a minimización) respecto a NSGAII, ya que las soluciones aparecen más cercanas al frente. Vamos a comprobarlo con el uso de métricas.

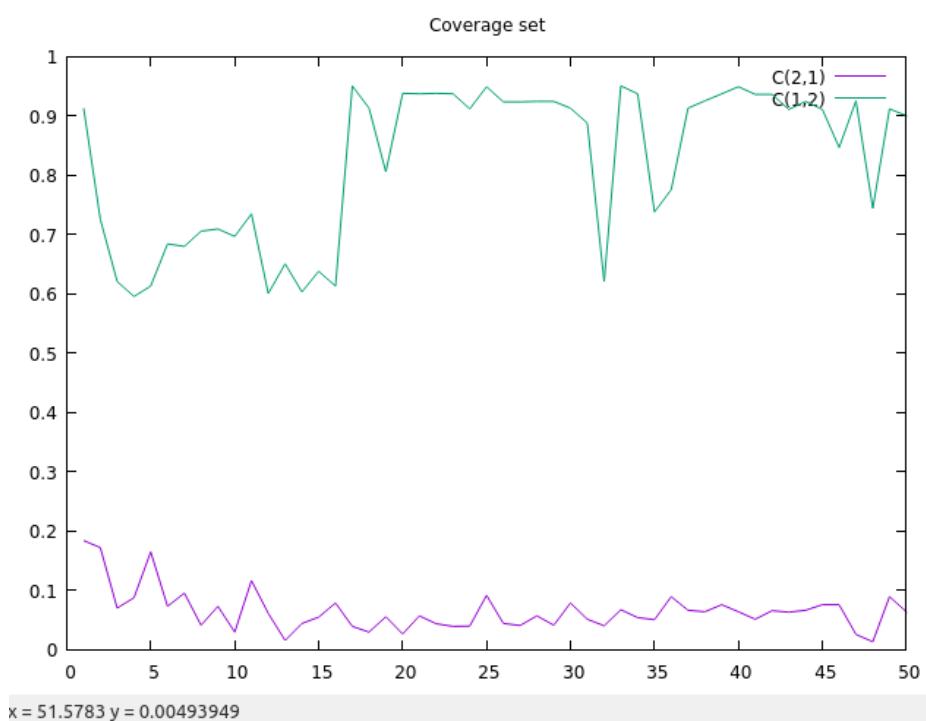
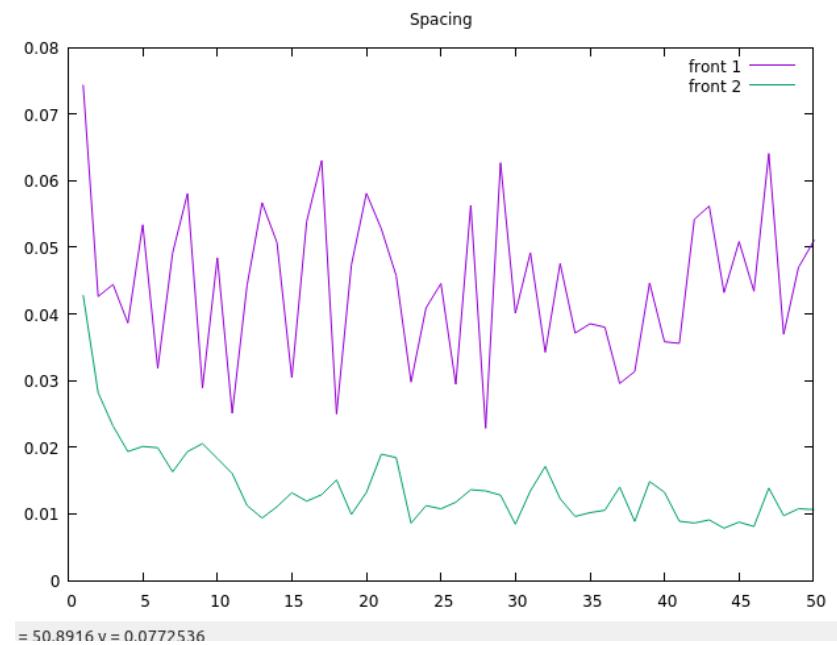
### 2.2.1.- SEMILLA 1



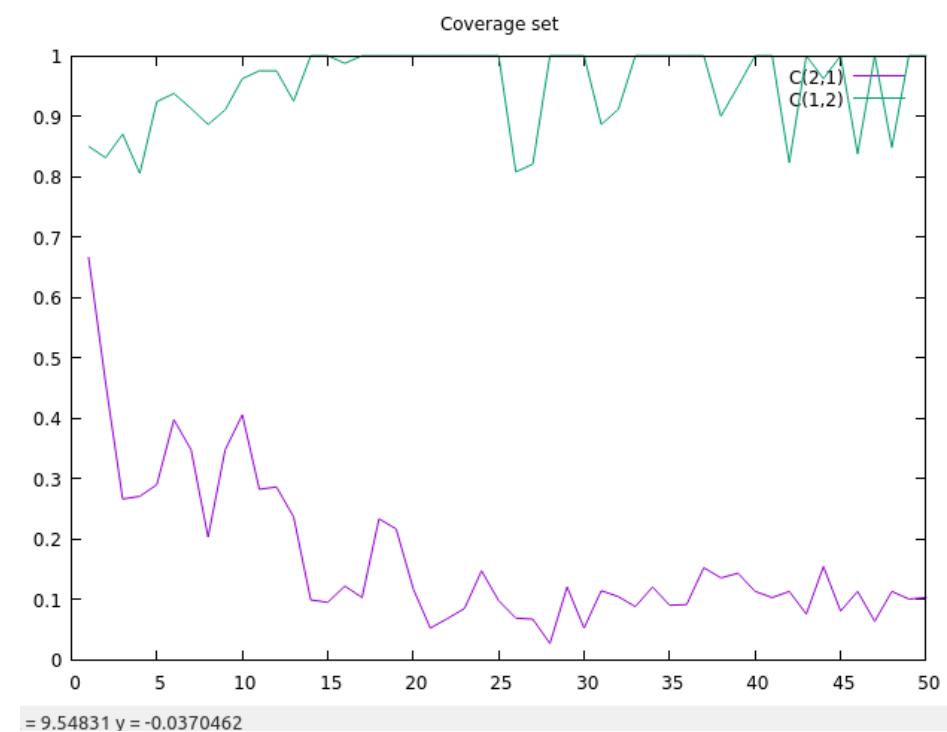
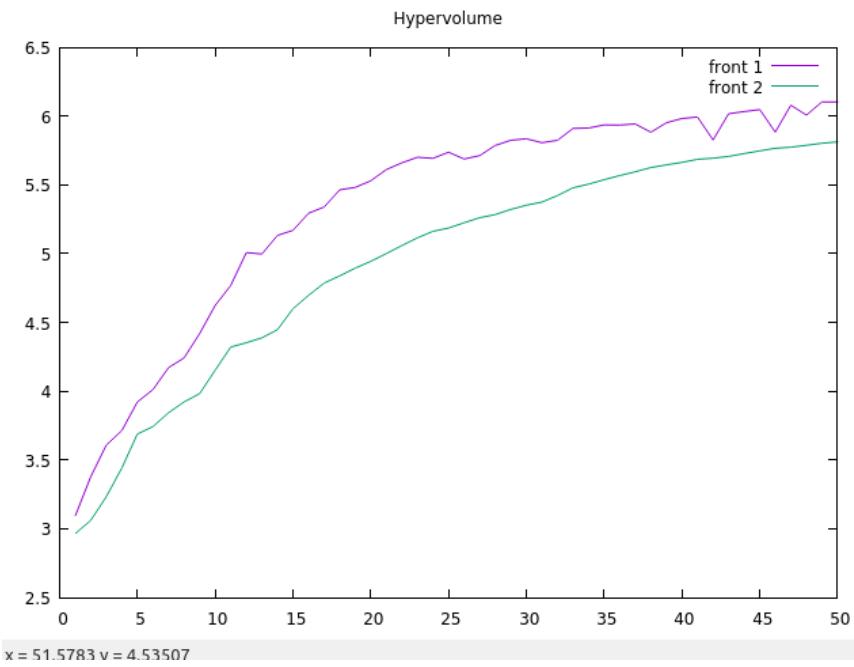


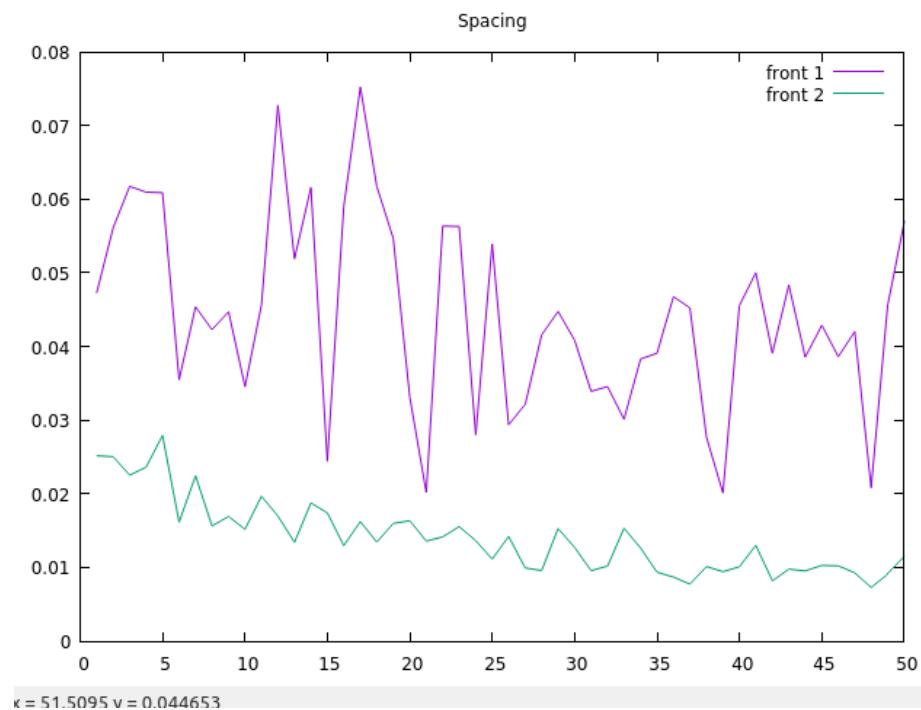
#### 2.2.2.- SEMILLA 5





### 2.2.3.- SEMILLA 7





#### 2.2.4.- ESTADÍSTICAS SOBRE MÉTRICAS

```

Documentos                               usuario@LinuxMintVM: ~/Escritorio/METRICS
Descargas
Música
Imágenes
Vídeos
Papelera
d
Navega la red

Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=0.9981977000
ref[2]=6.0896360000

Front 1 hypervolume: 5.6823084014
Front 1 spacing: 0.0099076895
Configuration UNKNOWN p80g50: Hypervolume mean: 5.92767
Configuration UNKNOWN p80g50: Hypervolume standard deviation: 0.0978135
Configuration NSGAII p80g50: Hypervolume mean: 5.7031
Configuration NSGAII p80g50: Hypervolume standard deviation: 0.0581127
Configuration UNKNOWN p80g50: Spacing mean: 0.0407019
Configuration UNKNOWN p80g50: Spacing standard deviation: 0.0112049
Configuration NSGAII p80g50: Spacing mean: 0.00988131
Configuration NSGAII p80g50: Spacing standard deviation: 0.00234144
usuario@LinuxMintVM:~/Escritorio/METRICS$ 

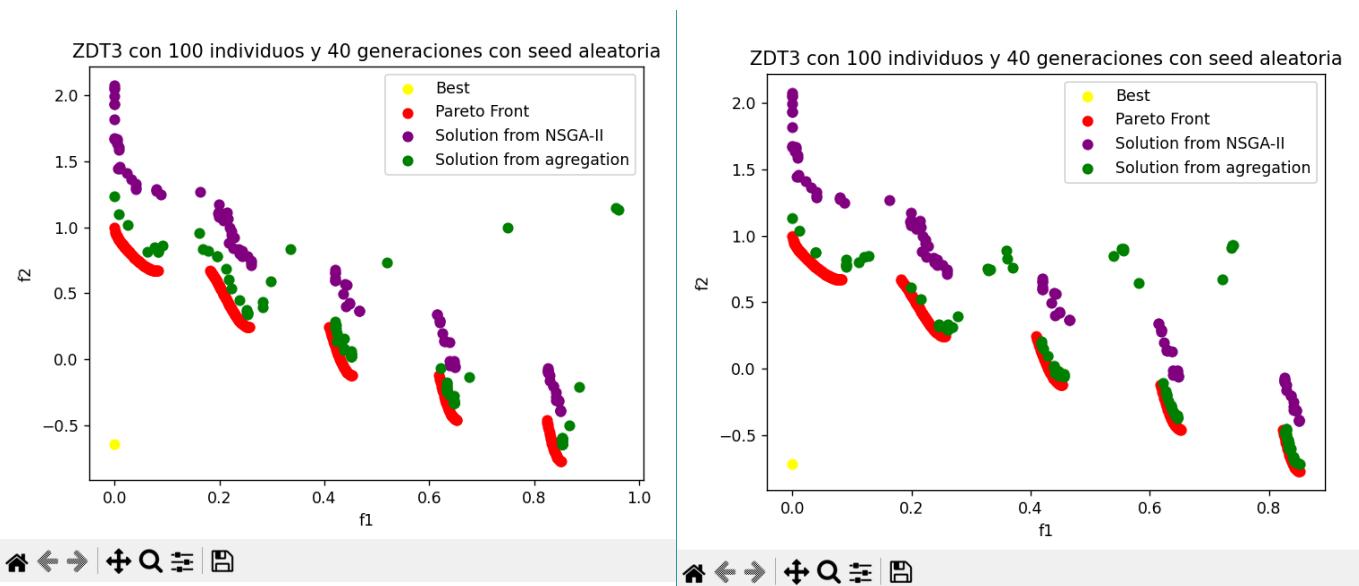
```

Con 80 individuos y 50 generaciones se presenta un hipervolumen medio mayor en el algoritmo propuesto (aunque no hay que perder de vista la desviación que es mayor respecto a la desviación de NSGAII).

En cuanto al Spacing, NSGAII sigue superándolo claramente al igual que con el resto de configuraciones.

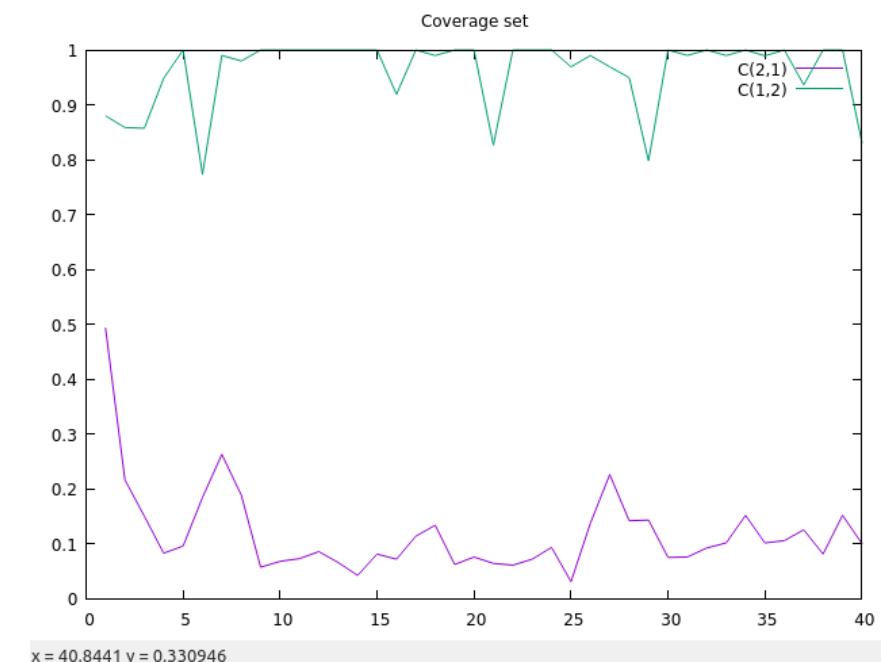
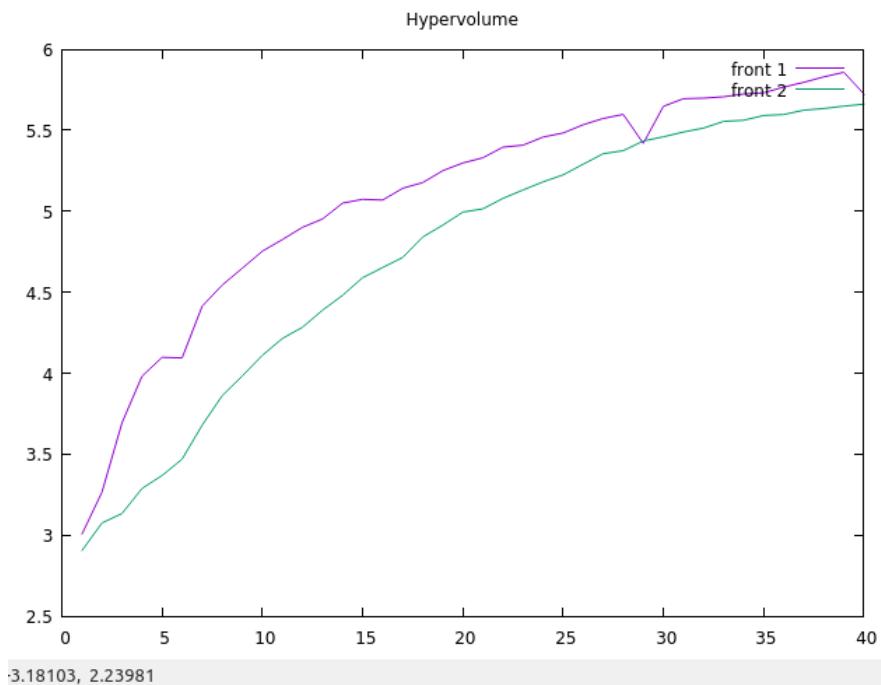
El Coverage Set visto en las tres ejecuciones diferentes muestra claramente cómo las soluciones del algoritmo propuesto superan a las soluciones de NSGAII. Esto quiere decir que son óptimas y minimizan más las soluciones del problema. Podemos determinar que con esta configuración el algoritmo propuesto supera a NSGAII.

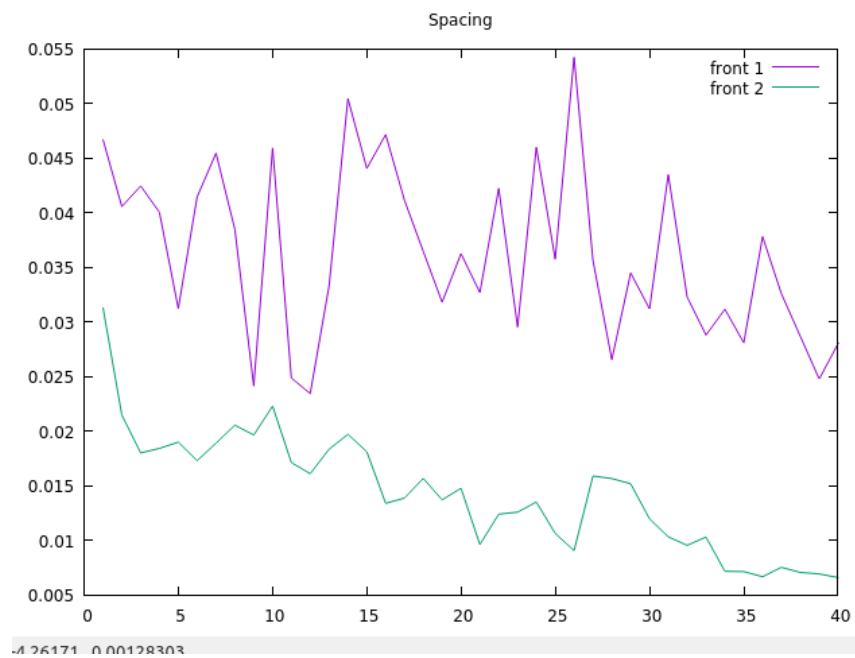
### 2.3.- Población de 100 individuos y 40 generaciones



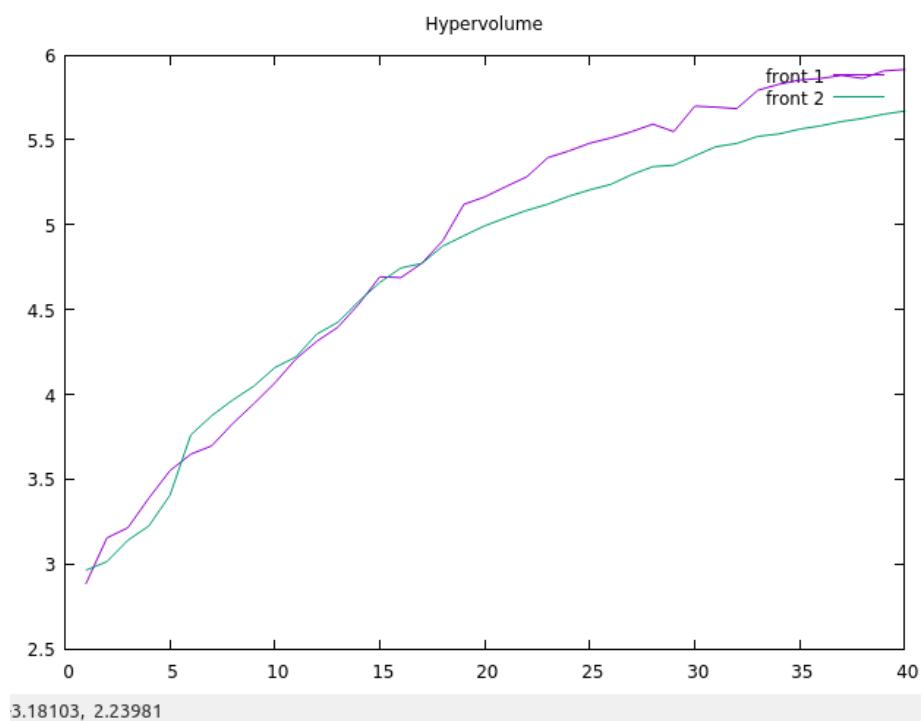
Podemos observar como la solución proporcionada por el algoritmo de agregación es bastante superior en un principio a la solución del algoritmo NSGA II, ya que en ambas ejecuciones puede verse como los puntos proporcionados ( $f_1, f_2$ ) del algoritmo implementado se acercan mucho más al frente Pareto que los puntos proporcionados por NSGA II. Sin embargo, procedemos a analizar las métricas resultantes para validar este hecho.

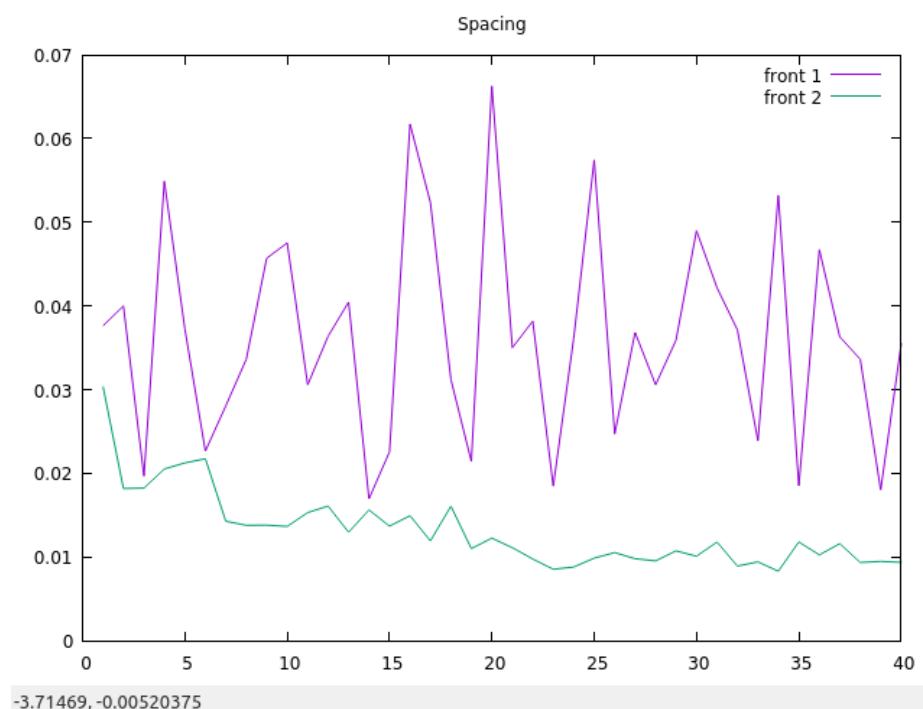
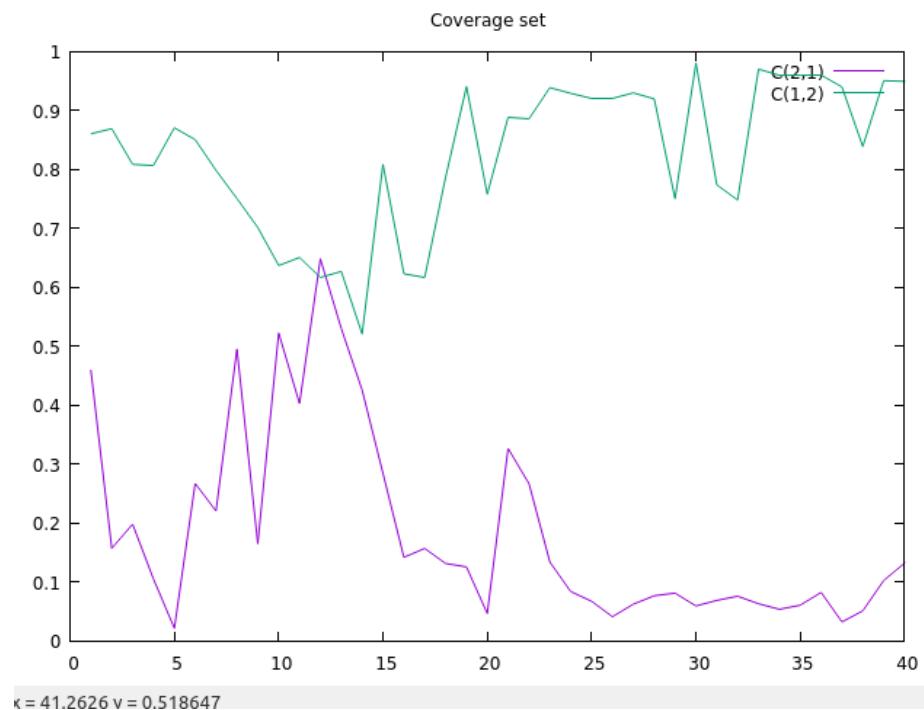
### 2.3.1.- SEMILLA 2



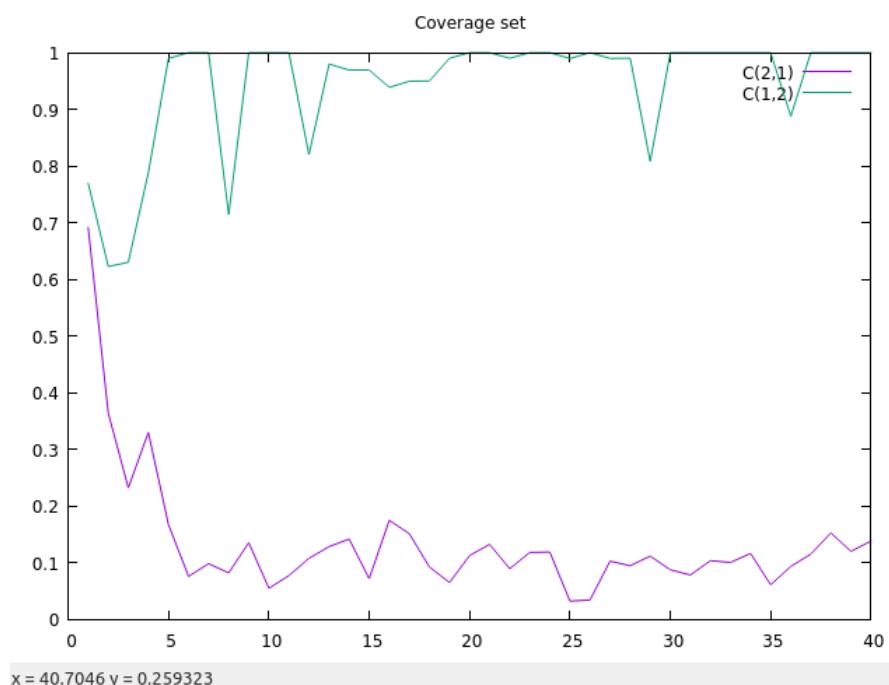
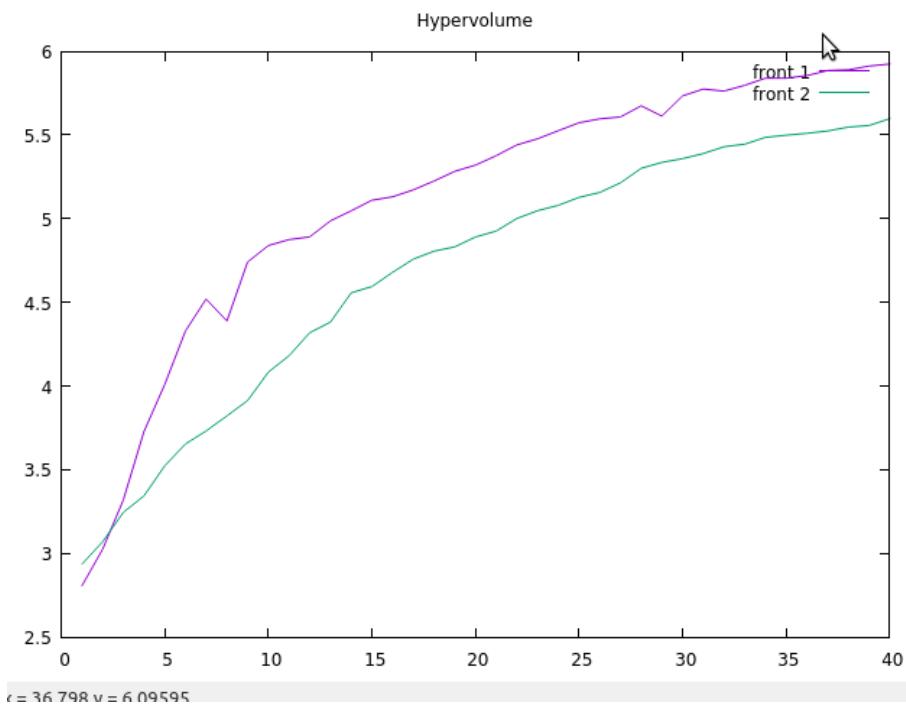


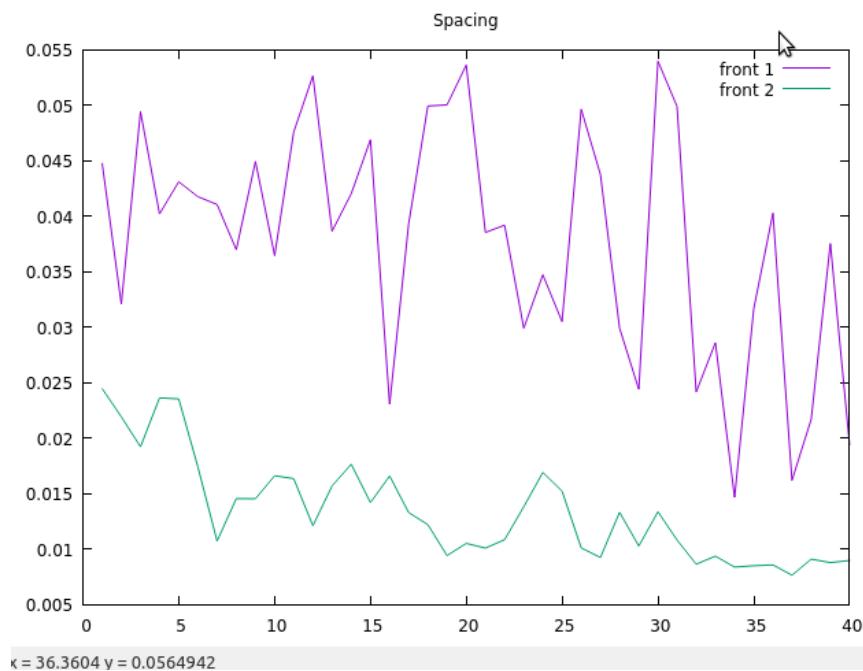
### 2.3.2.- SEMILLA 6





### 2.3.3.- SEMILLA 7





#### 2.3.4.- ESTADÍSTICAS SOBRE MÉTRICAS

```
usuario@LinuxMintVM: ~/Escritorio/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation      4dp40g100stallbasic.      4dp40g100stallref.in
ref[1]=0.9981977000
ref[2]=6.0896360000

Front 1 hypervolume: 5.6789927122
Front 1 spacing: 0.0077322247
Configuration UNKNOWN p100g40: Hypervolume mean: 5.84942
Configuration UNKNOWN p100g40: Hypervolume standard deviation: 0.0885661
Configuration NSGAII p100g40: Hypervolume mean: 5.65505
Configuration NSGAII p100g40: Hypervolume standard deviation: 0.0453189
Configuration UNKNOWN p100g40: Spacing mean: 0.0315827
Configuration UNKNOWN p100g40: Spacing standard deviation: 0.0123798
Configuration NSGAII p100g40: Spacing mean: 0.00817355
Configuration NSGAII p100g40: Spacing standard deviation: 0.00165721
usuario@LinuxMintVM:~/Escritorio/METRICS$
```

Como podemos observar, el hipervolumen medio es algo mayor en el algoritmo basado en agregación (con una desviación algo mayor en el algoritmo implementado), concordando con las gráficas de las tres ejecuciones comparativas.

Por otro lado, el spacing medio es mayor también en el algoritmo basado en agregación. Esto significa que las soluciones están peor distribuidas sobre el Pareto óptimo.

En cuanto al Coverage Set se puede ver claramente cómo las soluciones obtenidas por NSGAII (2) son superadas durante toda la fase de ejecución por las soluciones proporcionadas por el algoritmo basado en agregación (1).

Podemos determinar que esta configuración supera a NSGAII en líneas generales.

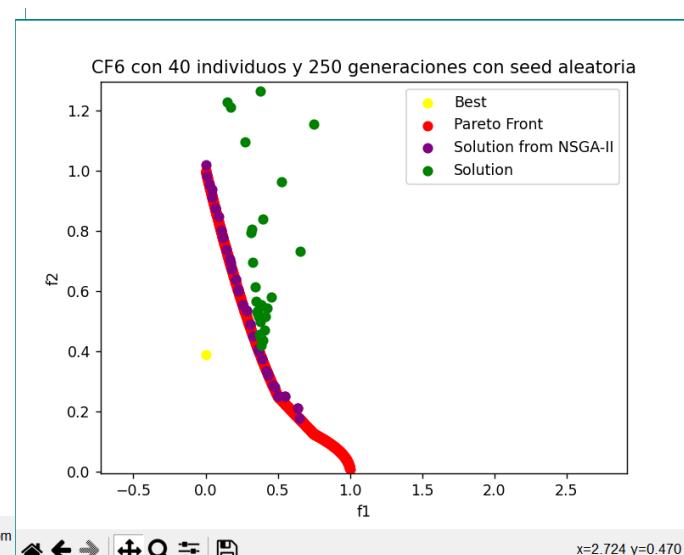
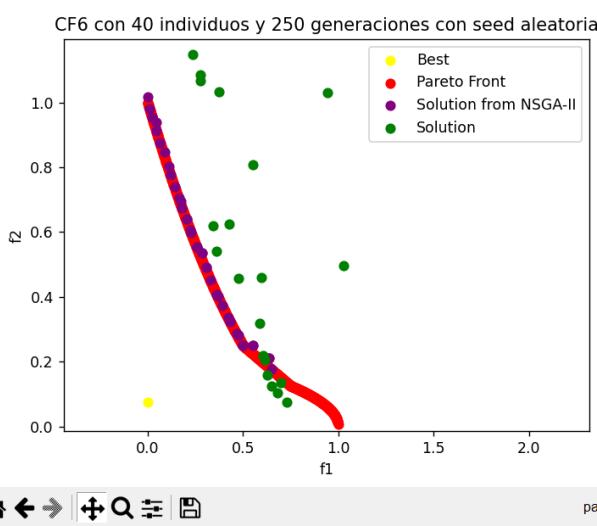
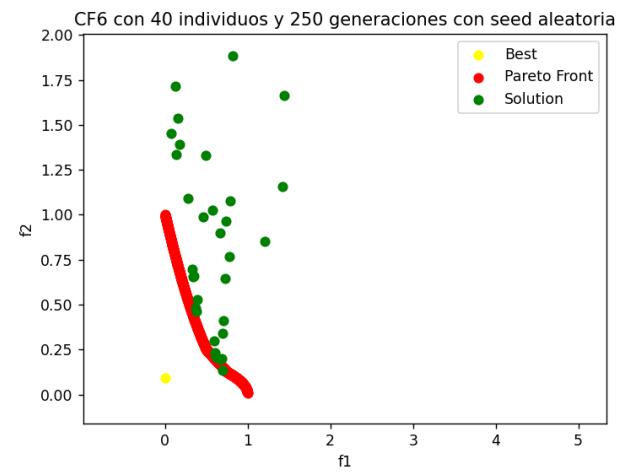
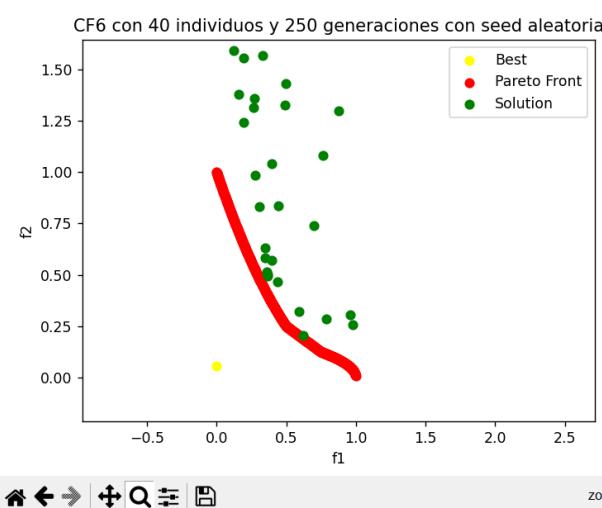
### **3.- Conclusión**

Tras ver los resultados experimentales sobre el problema ZDT3 podemos establecer algunas conclusiones generales:

- Para el caso de capacidad de cómputo 4000, las soluciones proporcionadas por el algoritmo basado en agregación son mejores que las soluciones de NSGA II, o por lo menos, compiten y en la mayoría de las configuraciones gana.
- Para el caso de capacidad de cómputo 10000 NSGA II gana, pero agregación lucha y llega a obtener soluciones bastante próximas.
- La implementación del algoritmo basado en agregación no logra obtener una mejor distribución de las soluciones, ya que el spacing medio siempre sale por encima de NSGA II.
- Aunque el coverage Set lo hemos visto en pocas ejecuciones en cada configuración, se puede ver cómo en las soluciones generadas por el algoritmo propuesto con capacidad de cómputo 4000 superan claramente a las soluciones generadas por el algoritmo NSGA II. Esto quiere decir que las soluciones de agregación ya engloban las soluciones de NSGA II. Sin embargo, con capacidad de cómputo 10000 es totalmente inverso. Esta conclusión no la podemos determinar del todo puesto que los coverage set siempre los hemos valorado mediante gráficas de 3 ejecuciones, aunque si nos puede dar una pista general de la comparación de ambos frentes en cada una de ellas.

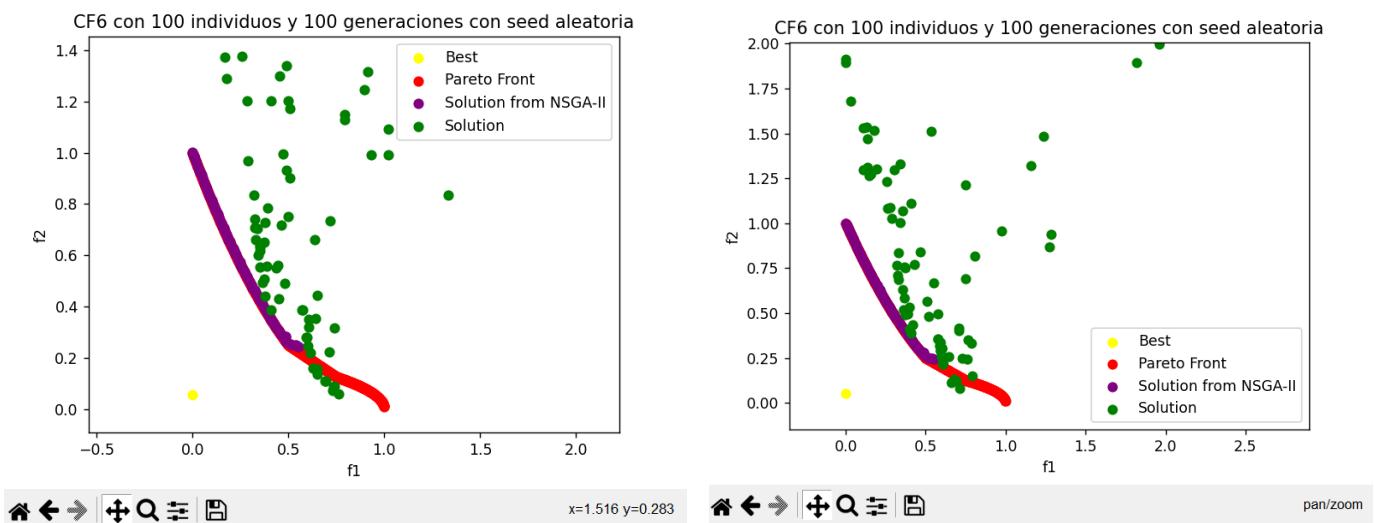
**4.- Resultados experimentales de CF6 4D con capacidad de cómputo de 10000 evaluaciones**

**4.1.- Población de 40 individuos y 250 generaciones**



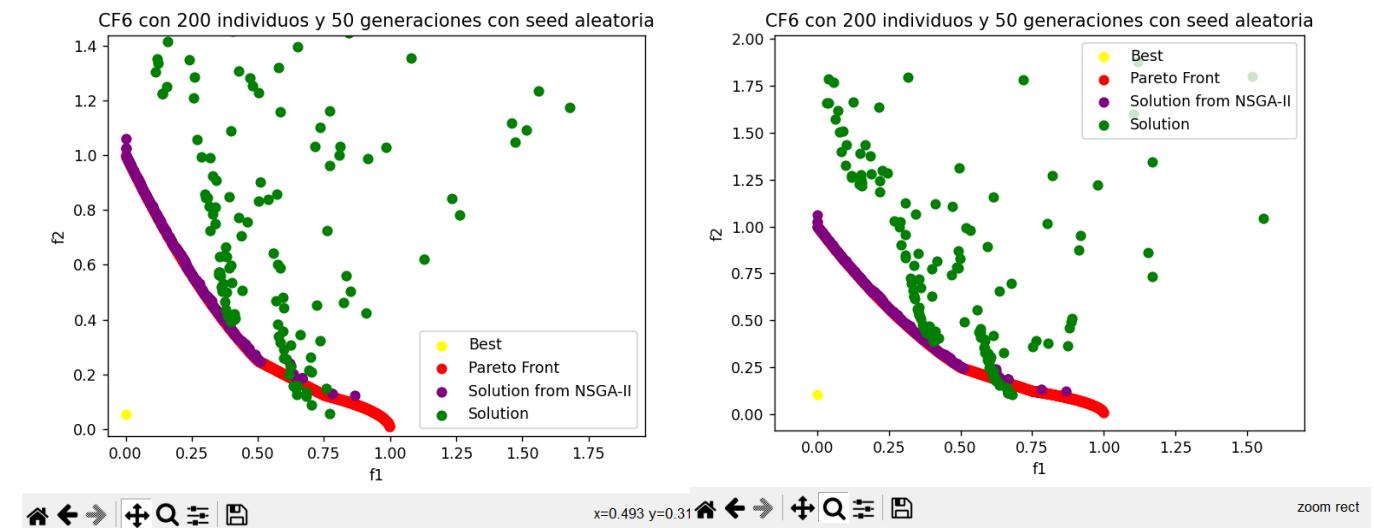
Como puede verse, el algoritmo implementado no logra tener una buena cobertura de la solución óptima (Pareto óptimo) e incluso en algunos casos como puede verse en la tercera figura proporciona soluciones que son imposible de darse, ya que resultan estar por debajo del límite que muestra el Pareto.

#### 4.2.- Población de 100 individuos y 100 generaciones



En este caso, con esta configuración pasa igual. Se dan soluciones por debajo de los límites, las cuales no pueden darse y esto nos lleva a concluir que el algoritmo no proporciona una buena solución. Analizar métricas sobre estas soluciones carece de sentido, aunque más adelante se proporcionará capturas de ellas.

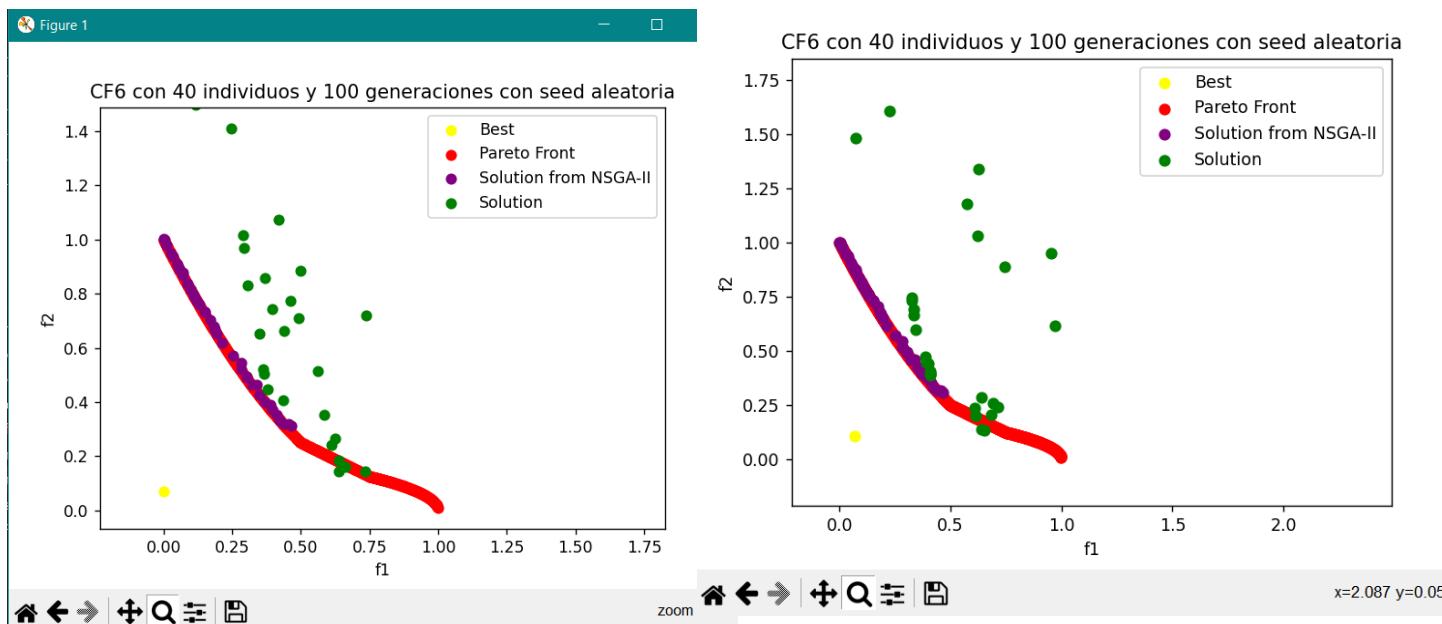
#### 4.3.- Población de 200 individuos y 50 generaciones



Exactamente igual pasa con estas ejecuciones.

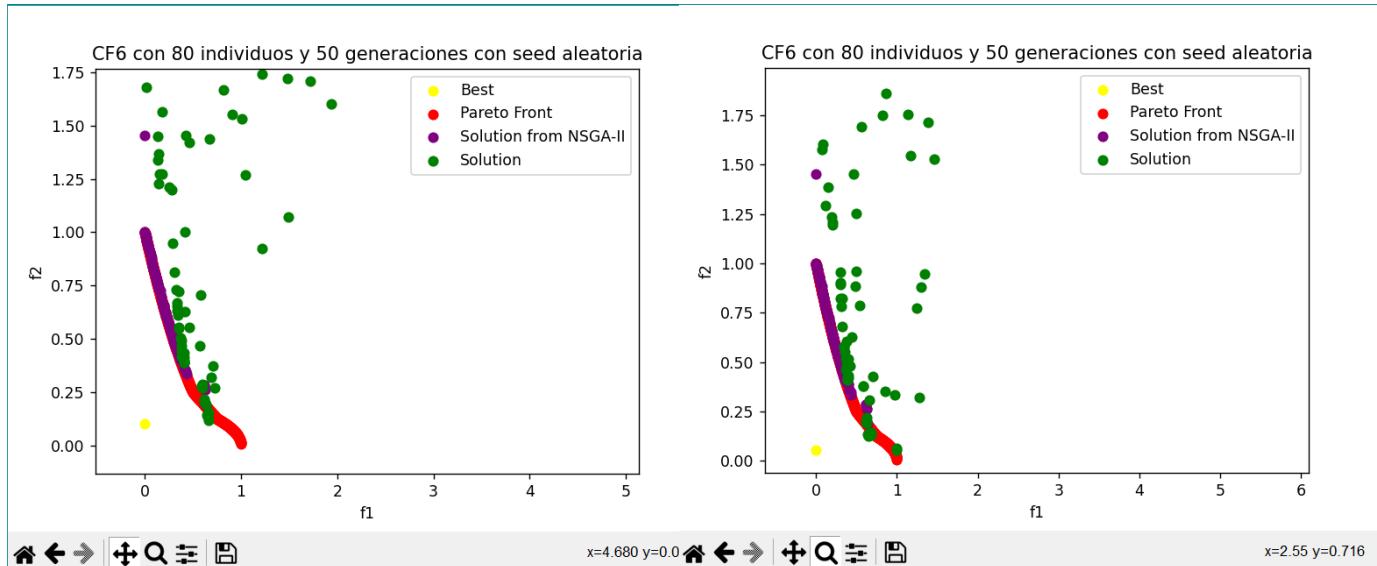
## 5.- Resultados experimentales de CF6 4D con capacidad de cómputo de 4000 evaluaciones

### 5.1.- Población de 40 individuos y 100 generaciones



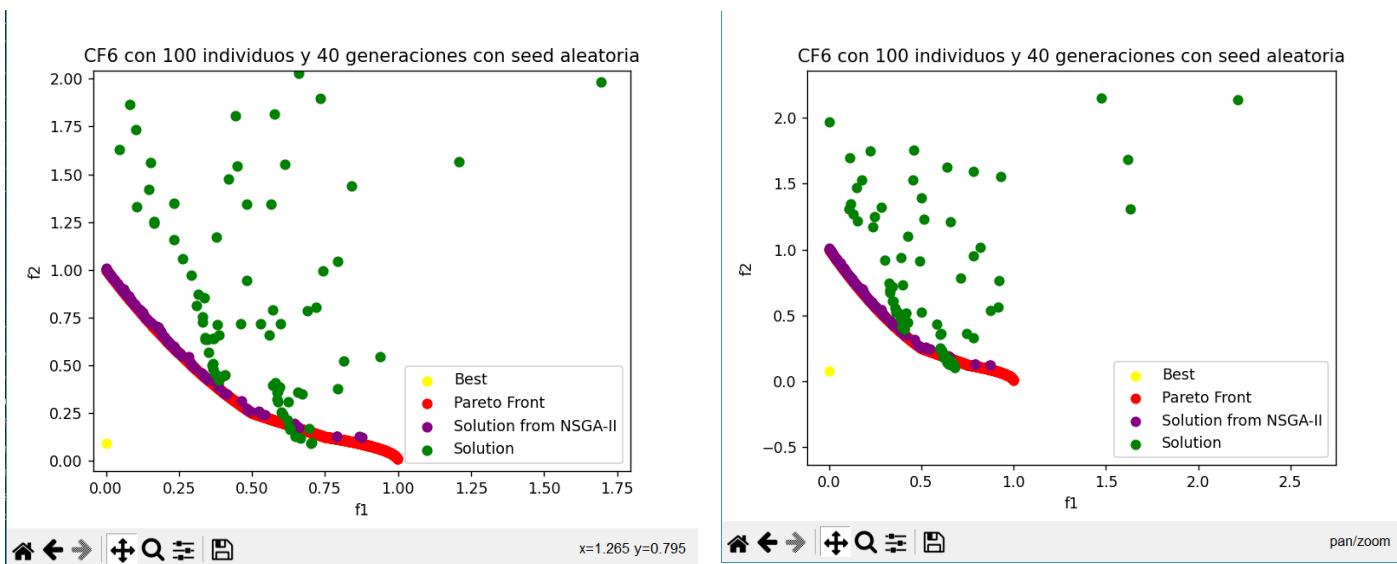
Se siguen proporcionando soluciones no válidas, aunque parece que la solución a pesar de no acercarse demasiado tampoco se dispersa tanto.

### 5.2.- Población de 80 individuos y 50 generaciones



Con esta configuración las soluciones se acercan más al frente, pero se sigue proporcionando, en menor cantidad, soluciones no válidas. Por tanto, no tiene sentido analizar métricas con esta configuración.

### 5.3.- Población de 100 individuos y 40 generaciones

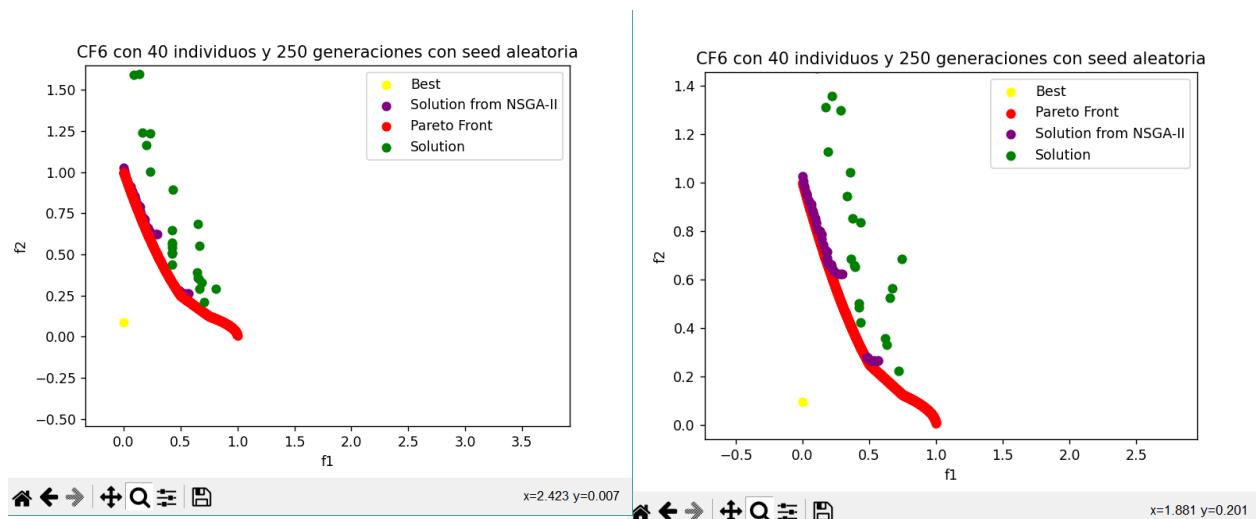


Por último, para esta configuración vuelve a pasar lo mismo. Parece ser que la mejor configuración para 4000 evaluaciones es con 80 individuos y 50 generaciones, pero no llega a ser válida ya que al igual que las otras dos se dan soluciones que no son posibles.

### 6.- Resultados experimentales de CF6 16D con capacidad de cómputo de 10000 evaluaciones

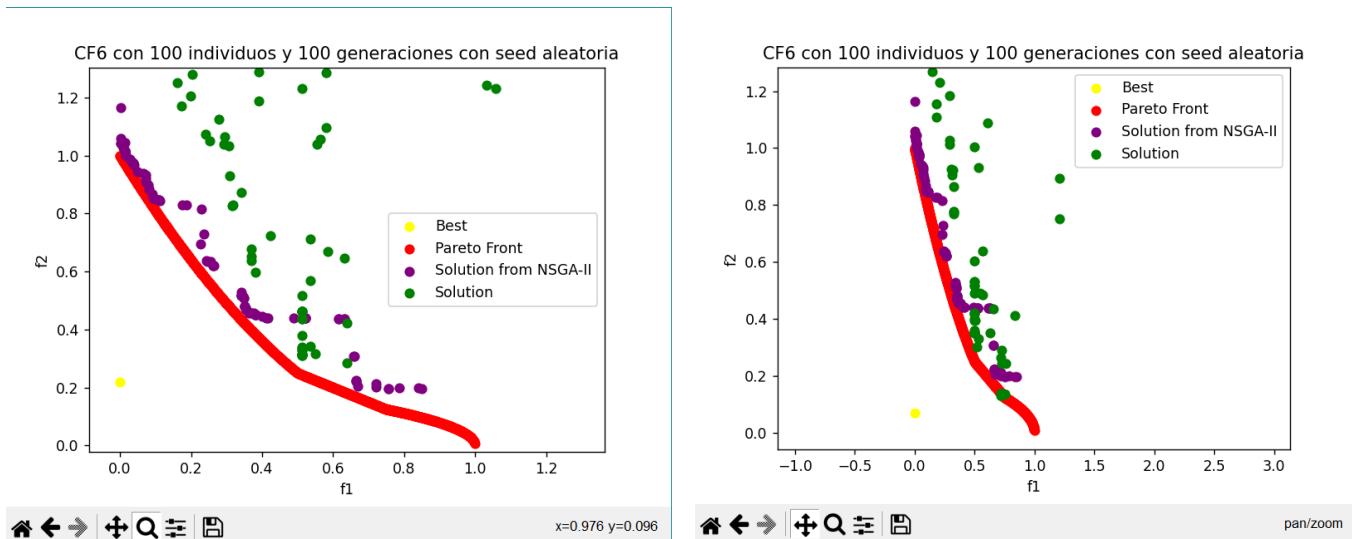
Hemos establecido los mismos pesos (100) que para CF6 4D y los mismos parámetros de mutación y cruce. También se ha dejado los mismos parámetros para la mutación gaussiana.

#### 6.1.- Población de 40 individuos y 250 generaciones



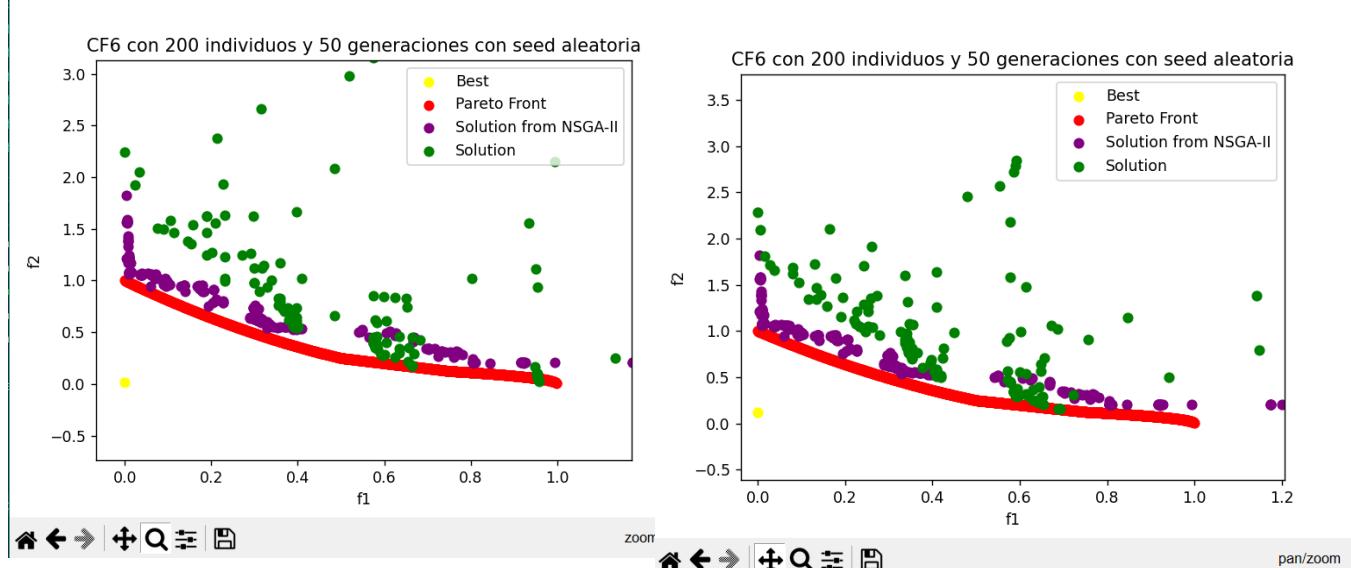
Parece ser que con 16 dimensiones con esta configuración sí se obtienen resultados que son posibles. Todos cumplen restricciones, pero parece ser que el algoritmo implementado sigue siendo mucho peor a NSGA II.

### 6.2.- Población de 100 individuos y 100 generaciones



Con esta configuración el algoritmo implementado por lo menos lucha con NSGA II aunque a simple vista las soluciones de NSGA II se acercan mucho más al Pareto. Parece ser que se siguen dando soluciones posibles y por ello en el apartado ANEXO se analizarán las métricas.

### 6.3.- Población de 200 individuos y 50 generaciones

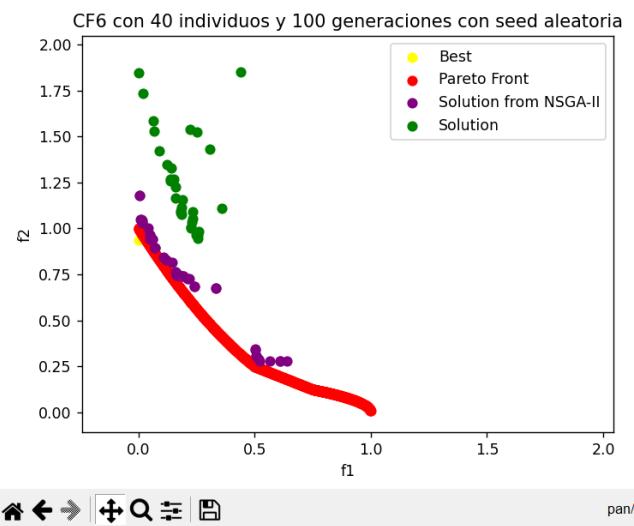
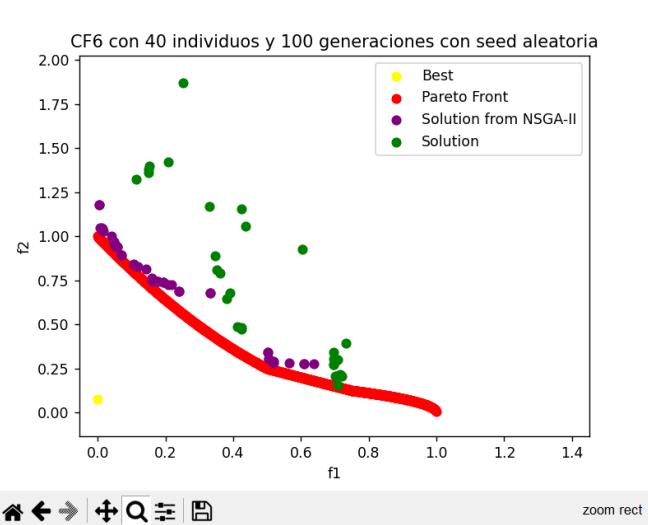


Aunque parece ser que el algoritmo NSGA II proporciona un mayor número de soluciones cercana al Pareto, el algoritmo de agregación consigue minimizar algunas soluciones y competir en algunas zonas del Pareto. Se analizarán las métricas en el apartado ANEXO.

## 7.- Resultados experimentales de CF6 16D con capacidad de cómputo de 4000 evaluaciones

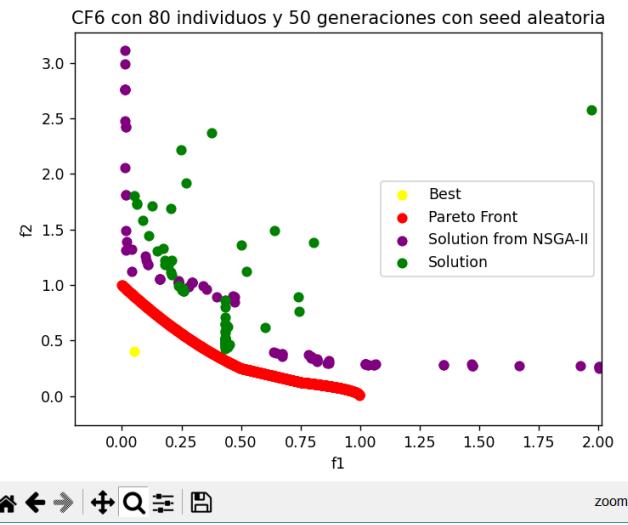
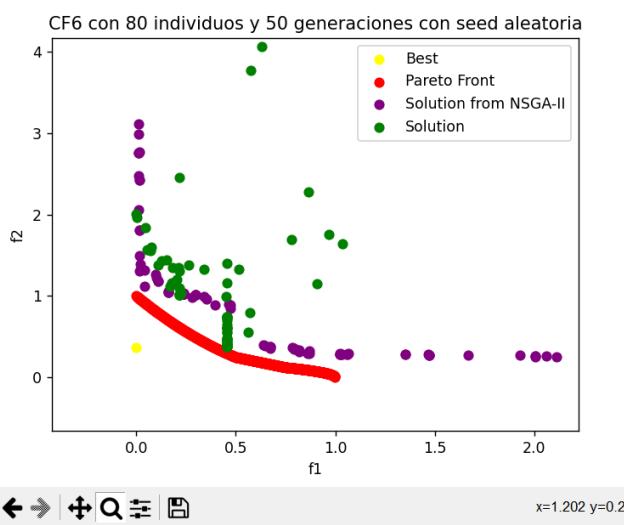
Hemos establecido los mismos pesos (100) que para CF6 4D y los mismos parámetros de mutación y cruce. También se ha dejado los mismos parámetros para la mutación gaussiana.

### 7.1.- Población de 40 individuos y 100 generaciones



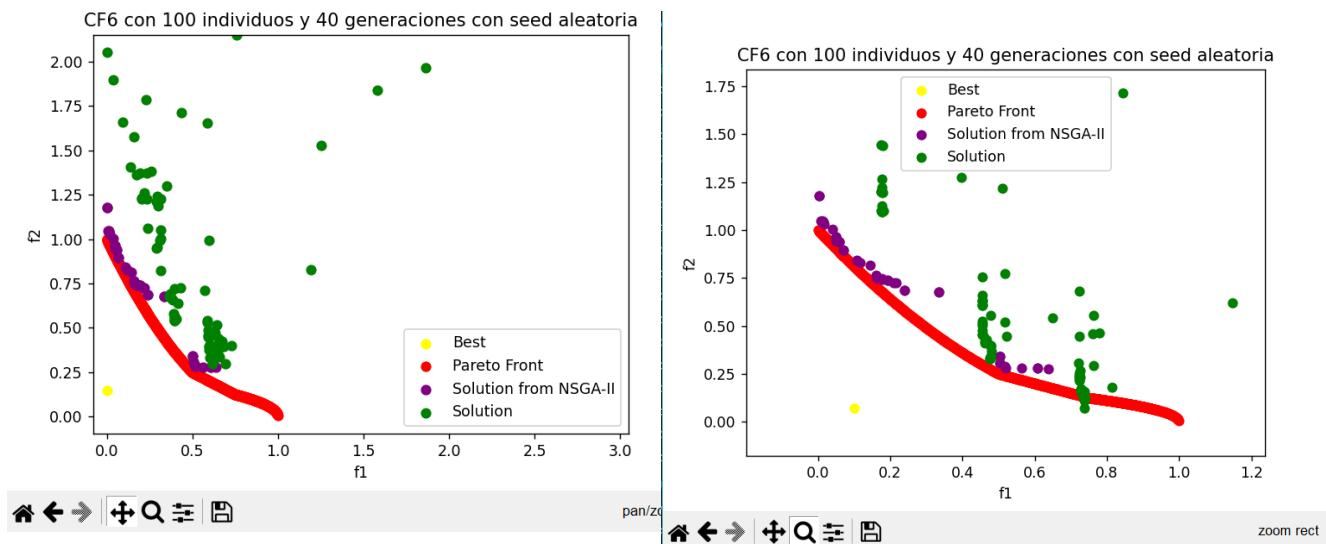
Aunque se obtienen soluciones válidas, a simple vista puede verse que el algoritmo implementado no gana a NSGA II.

### 7.2.- Población de 80 individuos y 50 generaciones



Con esta configuración parece ser que ni NSGA II ni el algoritmo implementado pueden obtener soluciones bastante buenas. En cuanto a la distribución de cada una de ellas parece que NSGA II gana.

### 7.3.- Población de 100 individuos y 40 generaciones



Con esta configuración se puede ver como se proporciona soluciones no posibles, ya que se encuentran por debajo del frente Pareto. Carece de sentido analizar sus métricas.

### 8.- Anexo

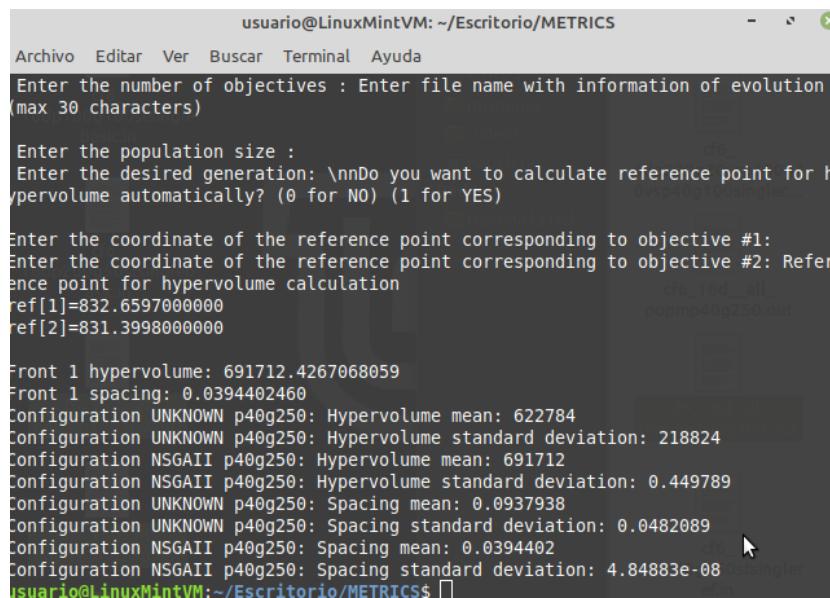
```
usuario@LinuxMintVM: ~/Escritorio/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=832.168300000
ref[2]=829.647900000

Front 1 hypervolume: 689846.7995203817
Front 1 spacing: 0.0284568644
Configuration UNKNOWN p100g100: Hypervolume mean: 621113
Configuration UNKNOWN p100g100: Hypervolume standard deviation: 218237
Configuration NSGAIID p100g100: Hypervolume mean: 689847
Configuration NSGAIID p100g100: Hypervolume standard deviation: 0.211324
Configuration UNKNOWN p100g100: Spacing mean: 0.0467914
Configuration UNKNOWN p100g100: Spacing standard deviation: 0.0269919
Configuration NSGAIID p100g100: Spacing mean: 0.0284569
Configuration NSGAIID p100g100: Spacing standard deviation: 3.75257e-08
usuario@LinuxMintVM:~/Escritorio/METRICS$
```

Podemos ver claramente que se trata de las estadísticas sobre métricas de ambos algoritmos. Están basadas en 10 ejecuciones (como en el caso del problema de ZDT3) y en ellas podemos determinar el hipervolumen medio y el spacing medio. Podemos ver que con esta configuración (100 individuos y 100 generaciones) el hipervolumen medio es mucho mayor en el caso del algoritmo NSGA II. Esto quiere decir que las soluciones en líneas generales son más próximas al Pareto. Por otro lado, cabe destacar que la desviación estándar es muy grande para el algoritmo basado en agregación. Puede deberse a la técnica de manejo de restricciones, que se usa penalti o simplemente a un fallo en la implementación del mismo algoritmo. Por último, el spacing medio es ampliamente superado por NSGA II, que presenta unos valores muy pequeños, lo que quiere decir que sus soluciones están muy bien distribuidas.



```

usuario@LinuxMintVM: ~/Escritorio/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

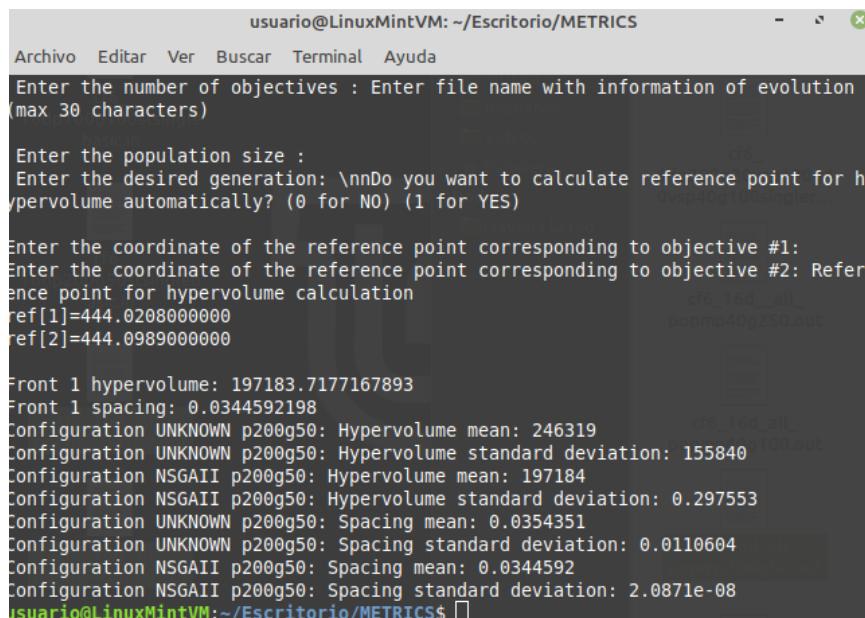
Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=832.6597000000
ref[2]=831.3998000000

Front 1 hypervolume: 691712.4267068059
Front 1 spacing: 0.0394402460
Configuration UNKNOWN p40g250: Hypervolume mean: 622784
Configuration UNKNOWN p40g250: Hypervolume standard deviation: 218824
Configuration NSGAI1 p40g250: Hypervolume mean: 691712
Configuration NSGAI1 p40g250: Hypervolume standard deviation: 0.449789
Configuration UNKNOWN p40g250: Spacing mean: 0.0937938
Configuration UNKNOWN p40g250: Spacing standard deviation: 0.0482089
Configuration NSGAI1 p40g250: Spacing mean: 0.0394402
Configuration NSGAI1 p40g250: Spacing standard deviation: 4.84883e-08
usuario@LinuxMintVM:~/Escritorio/METRICS$ 

```

En este caso podemos apreciar lo mismo que para el caso anterior. Con 40 individuos y 250 generaciones se obtiene un hipervolumen medio bastante menor y una desviación muy grande. Además, el spacing medio sigue siendo muy alto con respecto al spacing que obtiene NSGA II. Claramente esta solución no supera a NSGA II



```

usuario@LinuxMintVM: ~/Escritorio/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=444.0208000000
ref[2]=444.0989000000

Front 1 hypervolume: 197183.7177167893
Front 1 spacing: 0.0344592198
Configuration UNKNOWN p200g50: Hypervolume mean: 246319
Configuration UNKNOWN p200g50: Hypervolume standard deviation: 155840
Configuration NSGAI1 p200g50: Hypervolume mean: 197184
Configuration NSGAI1 p200g50: Hypervolume standard deviation: 0.297553
Configuration UNKNOWN p200g50: Spacing mean: 0.0354351
Configuration UNKNOWN p200g50: Spacing standard deviation: 0.0110604
Configuration NSGAI1 p200g50: Spacing mean: 0.0344592
Configuration NSGAI1 p200g50: Spacing standard deviation: 2.0871e-08
usuario@LinuxMintVM:~/Escritorio/METRICS$ 

```

Con esta configuración se obtiene un hipervolumen bastante mayor por parte del algoritmo propuesto pero la desviación sigue siendo mucho más alta en comparación con NSGA II. Por otro lado en spacing no se acerca al spacing medio que obtiene el algoritmo a comparar (NSGA II). Con 200 individuos y 50 generaciones obtenemos soluciones que compiten algo más en comparación con el resto de las configuraciones pero que siguen siendo deficientes. Si observamos las gráficas de su apartado correspondiente vemos como es cierto que son las que más igualadas quedan respecto del resto.

```

usuario@LinuxMintVM: ~/Escritorio/METRICS
Archivo Editar Ver Buscar Terminal Ayuda
Enter the number of objectives : Enter file name with information of evolution
(max 30 characters)

Enter the population size :
Enter the desired generation: \nDo you want to calculate reference point for h
ypervolume automatically? (0 for NO) (1 for YES)

Enter the coordinate of the reference point corresponding to objective #1:
Enter the coordinate of the reference point corresponding to objective #2: Refer
ence point for hypervolume calculation
ref[1]=0.0000000000
ref[2]=0.0000000000

Front 1 hypervolume: -0.4377712602
Front 1 spacing: 0.0384047181
configuration UNKNOWN p80g50: Hypervolume mean: 69171.1
configuration UNKNOWN p80g50: Hypervolume standard deviation: 218739
configuration NSGAIID p80g50: Hypervolume mean: -0.437771
configuration NSGAIID p80g50: Hypervolume standard deviation: 2.74275e-07
configuration UNKNOWN p80g50: Spacing mean: 0.0613535
Configuration UNKNOWN p80g50: Spacing standard deviation: 0.0596831
configuration NSGAIID p80g50: Spacing mean: 0.0384047
configuration NSGAIID p80g50: Spacing standard deviation: 1.90791e-08
usuario@LinuxMintVM:~/Escritorio/METRICS$ 

```

## 9.- Conclusión

Podemos establecer las siguientes conclusiones:

- Para CF6 en líneas generales (4 y 16 dimensiones) el algoritmo que he implementado (basado en agregación) no compite en la mayoría de las ocasiones con NSGA II. No sólo es eso, sino que ni siquiera proporciona soluciones válidas en la mayoría de las configuraciones que se han evaluado.
- Cuando proporciona soluciones válidas está lejos de proporcionar soluciones cercanas y bien distribuidas sobre el frente Pareto óptimo.
- Con 16 dimensiones parece ser que podemos obtener mejores soluciones respecto a 4 dimensiones
- Con 16 dimensiones y capacidad de cómputo de 10000 evaluaciones podemos al menos comparar ambos algoritmos. Para el resto de configuraciones es una pérdida de tiempo.
- En ningún caso el algoritmo de agregación implementado gana al NSGA II.
- Las desviaciones que se producen en hipervolumen medio y spacing medio son muy altas en comparación con las desviaciones de NSGA II.

## 10.- Problemas encontrados

- Principalmente el uso de scripts. Se han modificado y creado todos los scripts posibles pero no se sabe si del todo funcionan bien.
- Por falta de tiempo no se ha podido mejorar la implementación de cf6.
- No se ha podido analizar el coverage set en CF6 y ZDT3 ya que no ha habido tiempo para cambiar más scripts relativos a dicha métrica.