EVALUACIÓN BLOQUE III: ALGORITMOS EVOLUTIVOS

Parte II: Problemas Multiobjetivo con restricciones

RAMOS GONZÁLEZ, VÍCTOR *(vicramgon@alum.us.es)*

Fecha de elaboración: *2 de abril de 2021*
Profesor: *Dr. Francisco V. Fernández Fernández*
Asignatura: *Aplicaciones de Soft-Computing*.
Departamento: *Electrónica y Electromagnetismo.*
GII. Tecnologías Informáticas - Universidad de Sevilla.

Contenidos

1. ntroducción

Presentado previament el algoritmo MOEA/D con la evaluación de 3 operadores evolutivos, se propone ahora realizar una extensión del algoritmo para trabajar con Problemas de Optimización Multiobjetivo Restringidos (CMOP), de forma que se considera el CMOP de la forma:

$$\begin{aligned} &minimize \quad F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})) \\ &subject\,to \quad g_j(x) \geq 0,\ j = 1, \ldots, n \end{aligned}$$

de manera que el espacio de búsqueda $\boldsymbol{x} \in \Omega$ es acotado y corresponde a $\Omega = X_1 \times \cdots \times X_p$, con X_i el conjunto continuo de valores posibles para la componente x_i acotado superior e inferiormente por x_{Li} y x_{Ui} respectivamente. Pero además ahora no todos los puntos son soluciones posibles para el problema, sino que sólo aquellos que cumplen todas las restricciones son factibles para el problema. Esto hace que frecuentemente la región factible sea una pequeña parte dentro del espacio total de búsqueda en ocasiones, incluso disconexa.

El objetivo al igual que en el caso no restringido es encontrar los puntos o regiones que forma el frente de Pareto o frente Pareto-óptimo y que además son factibles, esto es, cumplen las restricciones.

En el estado del arte, existen multitud de aproximaciones a esta categoría de problemas (*CMOP*), pero nosotros presentaremos una basada en la extensión del algoritmo presentado en la primera parte *MOEA/D*. Utilizando también la formulación de Tchebycheff para la descomposición de los problemas y utilizando el operador EOP1 presentado en el primer apartado. De hecho las únicas variaciones sobre el método propuesto serán llevados a cabo en dos aspectos, la evaluación de los individuos, dado que ahora habrá que evaluarlos también frente a las restricciones y en el método de selección (actualización de la vecindad), en lo que se denominan métodos de manejo de restricciones.

Dentro de este campo existen multitud de aproximaciones, aunque las principales vías de aproximación se distribuyen entre las aproximaciones mediante métodos o *funciones penalti*, cuyo funcionamiento se basa en la penalización de los individuos por incumplimiento de restricciones, y las técnicas basadas en *separación de objetivos y restricciones*, que anteponen el cumplimiento de restricciones ante la mejora de objetivos. En este trabajo proponemos un método mixto basado en ambas aproximaciones y en otro artículos extraídos de la literatura, evaluando la aproximación propuesta frente al algoritmo NSGAII (con manejo de restricciones) para el problema CF-6 en sus versiones 4 y 16 dimensionales.

2. Metodología

2.1. Funcionamiento general del algoritmo

La *figura 1* muestra el diagrama de flujo general del algoritmo, ajustándose a un esquema típico de algoritmo evolutivo. De manera superficial, el algoritmo recibe las funciones objetivo, las funciones de restricción, el espacio de búsqueda y una serie de parámetros que intervienen en el algoritmo (detallados más adelante), lleva a cabo una primera etapa de inicialización de la población, los vectores de pesos y vecindad de los subproblemas, así como el punto de referencia y las evaluaciones de la población frente tanto a los objetivos como a las restricciones. A partir de ahí se lleva a cabo un proceso iterativo, en el que se va actualizando la población así como el punto de referencia. De forma que al fin del proceso la población constituye la aproximación del algoritmo al frente de Pareto para el problema tratado, distinguiendose para cada una de ellas las que son factibles (cumplen las restricciones) y las que no.

2.2. Datos de entrada

El algoritmo recibe como datos de entrada:

- **Funciones objetivo**: Un conjunto ordenado de funciones $f_1, \ldots, f_m$ consideradas como las funciones objetivo a optimizar por el algoritmo, tratando todas ellas como caso de minimización.
- **Restricciones**: Un conjunto ordenado de funciones $g_1, \ldots, g_n$ consideradas como las restricciones del problema en formato $g_j(\boldsymbol{x}) \geq 0$.
- **El espacio de búsqueda** Ω: Dado por el producto cartesiano de los espacios de búsqueda de cada

una de las variables, acotados superior e inferiormente por x_{Lj} y x_{Uj} respectivamente.

- N: Corresponde al número de subproblemas considerados para la división del objetivo mútiple en subobjetivos individuales (por agregación).

- G: Corresponde al número de generaciones máximo que realizará el algoritmo, estableciendo un criterio de parada para el mismo.

- T: Corresponde al número de vecinos de cada subproblema esto es, el número de subproblemas, incluyéndose a sí mismo, que serán considerados parte de la vecindad de cada uno de los subproblemas y que influirán en el proceso evolutivo de la población.

- **Operador evolutivo**: Durante el desarrollo del algoritmo son utilizados operadores de cruce y mutación que han de ser dados por el usuario y que obtendrán un solo individuo como descendiente de cada uno de los individuos de la población. En este trabajo utilizaremos el operador EOP1 presentado en la primera parte (optimización no restringida).

- **Otros parámetros**, que detallamos y justificamos en el apartado correspondiente.

2.3. Inicialización del algoritmo

Para poder llevar a cabo la ejecución del algoritmo (proceso iterativo) hemos de disponer de algunos elementos que debemos inicializar al comienzo del algoritmo como son:

- **El conjunto de vectores de pesos $\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_N}$**, vectores m-dimensionales que identifican a cada subproblema, de forma que para todos ellos debe cumplirse $||\boldsymbol{\lambda_i}||_1 = 1$ y además deben estar distribuidos uniformemente (equiespaciados, tomando como distancia la forma euclídea). Para el caso bi-objetivo, dado que los vectores han de estar normalizados, los N vectores pueden inicializarse siguiendo la expresión $\boldsymbol{\lambda_i} = \left(\frac{i-1}{N-1}, \frac{N-i}{N-1}\right)$ con $i \in \{1, \ldots, N\}$.

- **Vecindad $B(i)$ de cada vector peso $\boldsymbol{\lambda_i}$**, que corresponde al conjunto de los los T vectores más cercanos a $\boldsymbol{\lambda_i}$ de entre todos los vectores de pesos (incluyendo al propio $\boldsymbol{\lambda_i}$). Nótese que $B(i)$ será considerado el entorno del subproblema i, de manera que debe considerarse un entorno cercano (T pequeño) para la evolución del individuo asociado al subproblema considerado.

- **Inicialización de la población** P, con N individuos cada uno de ellos asociados a uno de los subproblemas. Para la inicialización se generará una población aleatoria (respetando los espacios de búsqueda para cada variable).

- **Inicialización del punto de referencia $\boldsymbol{z}$**. Recuérdese que según la formulación de Tchebycheff $\boldsymbol{z}^*$ debe contener los óptimos globales de cada una de las funciones objetivo en el espacio de búsqueda, pero esto implica resolver m problemas de optimización mono-objetivo. En vez de eso relajaremos $\boldsymbol{z}^*$ en $\boldsymbol{z}$ de forma que contendrá en cada momento el mejor (menor) valor para cada f_j ($j = 1, \ldots, m$) alcanzado por cualquiera de los individuos tanto factibles como infactibles, evaluados

hasta ese instante. De forma que dicho punto de referencia $\boldsymbol{z}$ se irá actualizando de forma síncrona al avance del algoritmo.

- **Inicialización del conjunto de soluciones no dominadas**. Opcionalmente, el algoritmo puede mantener un conjunto con las soluciones no dominadas por ninguna otra encontrada hasta el momento de forma que el conjunto será actualizado a lo largo del proceso evolutivo, realizándose dicha actualización de acuerdo al concepto extendido de Pareto-dominancia.

2.4. Desarrollo iterativo

Tras la inicialización, comienza un proceso iterativo (evolutivo) en el que la población, se actualiza por medio del uso de operadores evolutivos y métodos de manejo de restricciones (métodos de señección) que presentaremos a continuación. Como criterio de parada para este proceso iterativo se tomará el límite de generaciones fijado por el usuario (al igual que en otros algoritmos de carácter evolutivo o inteligencia colectiva se pueden fijar otros criterios de parada anticipada). De manera que para cada iteración (generación) se llevan a cabo las siguientes operaciones:

1. *Reproducción*: Para cada uno de los individuos se aplica el operador evolutivo (cruces y/o mutaciones) de forma que cada individuo es perturbado (en mayor o menor medida) por el entorno. Como entorno se considera el vecindario $B(i)$, del que son tomados, con cierta aleatoriedad, (según el operador) una serie de individuos con los que se aplican los operadores evolutivos. Como el vecindario son los considerados problemas cercanos, es esperable que los cambios entre ellos sean pequeños por lo que los individuos tenderán a ser parecidos.

   En concreto se utilizará el operador EOP1 derivado de Evolución Diferencial + Perturbación Gaussiana, presentado (implementado y evaluado) en la primera parte del trabajo. de forma que el mismo del siguiente modo:

   a) El individuo $\boldsymbol{P_i}$ se genera el vector mutante $\hat{\boldsymbol{y}}$, con el uso de cinco vectores 'noisy', mediante la expresión

   $$\hat{\boldsymbol{y}} = \boldsymbol{x}^{(r1)} + F \cdot (\boldsymbol{x}^{(r2)} - \boldsymbol{x}^{(r3)}) + rand \cdot F \cdot (\boldsymbol{x}^{(r4)} - \boldsymbol{x}^{(r5)})$$

   donde F es elegido aleatoriamente de un pool de valores (entre 0 Y 1) y $rand$ corresponde a un número aleatorio (entre 0 y 1).

   b) Tras esto, se realiza una recombinación utilizando el cruce típico de evolución diferencial, tal que cada componente del vector $\boldsymbol{y}$ se elige aleatoriamente entre $\hat{\boldsymbol{y}}$ y $\boldsymbol{P_i}$, de forma que $\boldsymbol{y}$ contenga al menos una componente del vector mutante. Para ello se sigue la expresión:

   $$y_j = \begin{cases} \hat{y}_{ij} & \text{si } rand \leq CR \text{ o } j = \delta \\ P_{ij} & e.o.c. \end{cases}$$

   donde *CR* corresponde a un parámetro ajustable (entre 0 y 1) que representa la probabilidad de cruce, δ corresponde a una de las componentes del descendiente elegida aleatoriamente y $rand$ corresponde a un número aleatorio (entre 0 y 1).

c) Finalmente cada componente de $\boldsymbol{y}$ es perturbada con probabilidad p_m (parámetro ajustable entre 0 y 1, típicamente establecida en la literatura como $1/p$, con p la dimensionalidad del espacio de búsqueda) mediante una distribución gaussiana ($N(0, \sigma_j = \frac{x_{Uj} - x_{Lj}}{SIG})$ con SIG un parámetro ajustable).

d) Reparación: si alguna de las componentes quedara fuera del espacio de búsqueda según las cotas su valor es establecido al valor de la más cercana.

2. *Evaluación*: El descendiente obtenido es evaluado respecto a todos los objetivos $\boldsymbol{F}(\boldsymbol{y})$ y respecto a todas las restricciones definiéndose $V(\boldsymbol{y}) = \left| \sum_{j=1}^{n} min(0, g_j(\boldsymbol{y})) \right|$ (cuantía da violación de restricciones).

3. *Actualización del punto de referencia* $\boldsymbol{z}$: Recuérdese que z correspondía al vector m-dimensional (m el número de objetivos) cuyas componenetes correspondientes correspondían a los pseudo-óptimos (mejores valores obtenidos hasta el momento) luego hemos de actualizar z en aquellas componentes tal que $F_j(\boldsymbol{y}) < z_j$ (con $j = 1, \ldots, m$). Nótese que el punto de referencia es actualizado de igual forma tanto con soluciones factibles como infactibles.

4. *Actualización del entorno* $B(i)$: El vecindario del subproblema i (en el que se incluye él mismo) es actualizado siguiendo un criterio de selección. En este trabajo vamos a presentar un criterio mixto, basado en el concepto de dominancia pareto (separación de objetivos y restricciones) y en el concepto de función penalti. Este último suele ser más utilizado en la literatura y existen multitud de versiones *Static*, *Multi-staged*, *Dynamic*, *Self-adaptative*, *Three-steps*, ... [1]. Nuestra perspectiva (basada en [2, 3, 4, 5, 6]) utiliza un método adaptativo y basado en umbral de forma que se aplicará una diferente penalización según la cuantía de la violación de restricciones. Además se aplicará según un criterio basado en la técnica de separación de objetivos y restricciones. En concreto el operador de selección toma $\boldsymbol{x_j}$ el individuo considerado en la actualización con valoración $\boldsymbol{F}(\boldsymbol{x_j})$ y cuantía de violación de restricciones $V(\boldsymbol{x_j})$ y $\boldsymbol{y}$) el descendiente con valoración $\boldsymbol{F}(\boldsymbol{y})$ y violación de restricciones $V(\boldsymbol{y})$, entonces:

   - Si $V(\boldsymbol{y}) = 0$ y o $V(\boldsymbol{x}) > 0$ o $g^{te}(\boldsymbol{y}|\lambda_j, \boldsymbol{z}^*) < g^{te}(\boldsymbol{x_j}|\lambda_j, \boldsymbol{z}^*)$, entonces $\boldsymbol{x_j}$ es actualizado con $\boldsymbol{y}$.

   - Si no, entonces $V(\boldsymbol{y}) > 0 \wedge V(\boldsymbol{x_j}) > 0$, entonces entonces se toma $\boldsymbol{\epsilon}$ un véctor de la élite y se toma de entre los tres aquél con mejor (menor) *fitness* definida como:

$$fitness(\boldsymbol{x}) = g^{te}(\boldsymbol{x}|\lambda_j, \boldsymbol{z}^*) + \begin{cases} s * V(\boldsymbol{x})^2 & \text{si } V(\boldsymbol{x}) \leq \tau \\ s * \tau^2 + \eta(V(\boldsymbol{x})^2 - \tau) & \text{si } V(\boldsymbol{x}) > \tau \end{cases}$$

   donde τ es el valor umbral de penalización aceptada y es calculado como la cuantía media de entre las violaciones, η corresponde a la penalización de superación del umbral y s a la penalización de restricciones por debajo del umbral. De forma que,

$$\tau = \frac{\sum_{i \in B(j)} V(\boldsymbol{x_i})}{d_j}; \quad d_j = |\{i \in B(j)/V(\boldsymbol{x_i}) > 0\}|; \quad \eta = 100^{\frac{it}{G}}; \quad s = \begin{cases} 10^{-\frac{G}{it}} & \text{si } it/G < 75\,\% \\ \eta & \text{e.o.c.} \end{cases}$$

Nótese que lo que se pretende con η y s es realizar una búsqueda por la región factible y por las inmediaciones de la misma, de forma que en las primeras iteraciones se permite una búsqueda

más libre mientras que en las últimas se fuerza a que los individuos vayan hacia la zona factible. Además nótese que usualmente los vectores de la élite son factibles (en el momento en el que se encuentre una factible todas las infactibles salen del conjunto NDS) por lo que las soluciones tenderán a moverse a la zona factible.

Hemos de notar también que la actualización de los vecinos puede conllevar una gran pérdida de la diversidad mucho más apreciable que en el caso no restringido. Por ello se limita la capacidad de actualización a un número de actualizaciones muy bajo (normalmente menor del 5 % del tamaño de la población).Además dado el proceso iterativo propuesto, aunque no ha sido especificado, el proceso de actualización de la población (y sus valoraciones) se realiza directamente sobre la población, y no se espera para ser efectiva al fin de la generación. Por ello es conveniente que a la hora de llevar a cabo las actualizaciones los elementos se recorran en un orden aleatorio, para tratar de evitar descompensaciones entre los individuos.

2.5. Final del algoritmo

El proceso iterativo previo se realiza hasta alcanzar el criterio de parada, dado por el número máximo de generaciones G. De forma que al final del proceso en la última población se tienen justamente los mejores individuos encontrados durante el proceso para cada uno de los subproblemas y cuyas valoraciones constituyen, precisamente, la aproximación al frente de Pareto.

También es posible, y casi más fiel, considerar el NSD una aproximación más cercana y más amplia al frente real Pareto-óptimo, dado que en él se encontrarán todas aquellas soluciones no dominadas por ninguna otra de las encontradas durante todo el proceso de búsqueda.

3. Implementación

En este apartado describiremos la implementación concreta realizada de la metodología previamente propuesta, concretando cada una de las etapas y procesos. En el *algoritmo 1* se presenta el pseudocódigo del marco general de desarrollo, del algoritmo. En los puntos esta sección concretaremos cada una de las partes del mismo, dando los detalles de implementación correspondientes.

3.1. Datos de entrada

Como se propone en la metodología los datos de entrada corresponden a:

- **Funciones objetivo** $f_1, \ldots, f_m$: Dadas como una lista de longitud m de funciones, que reciban como entrada un indivividuo (vector $p - dimensional$) y devuelvan una valor real correspondiente a la valoración del individuo respecto de dicha función.

- **Restricciones** $g_1, \ldots, g_n$: Dadas como una lista de longitud n de funciones, que reciban como entrada un indivividuo (vector $p - dimensional$) y devuelvan una valor real correspondiente a la valoración del individuo respecto de dicha función.

- **El espacio de búsqueda** Ω: Que establece para cada una de las variables los límites de variación (continua) de la variable. De forma que dichos límites se darán como una lista de longitud p de pares (x_{Lj}, x_{Uj}) de valores reales. Dichos pares son ordenados para asegurar que $x_{Lj} \leq x_{Uj}$.

- **Número de subproblemas** N: Un entero que corresponde al número de subproblemas considerados, esto es el número de vectores de pesos $\boldsymbol{\lambda_i}$ y el tamaño considerado para la población.

- **Número de generaciones** G: Un entero que corresponde al número de máximo de iteraciones llevadas a cabo por el algoritmo.

- **Número de vecinos** T: Un entero correspondiente al número de subproblemas, incluyéndose a sí mismo, que serán considerados parte de la vecindad.

3.2. Inicialización

Durante la fase de inicialización se establecen los valores iniciales para:

- **Vectores de pesos** $(\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_N})$, de forma que si el problema es bi-objetivo dichos vectores son generados utilizando la expresión $\boldsymbol{\lambda_i} = \left(\frac{i-1}{N-1}, \frac{N-i}{N-1}\right)$ con $i \in \{1, \ldots, N\}$ para $i = 1, \ldots, N$. Si se desea es posible especificar un fichero en formato *ASCII* del que se leerán las N primeras lineas que deberán contener los vectores de pesos considerados.

Algoritmo 1: Marco gral. del CMOEA/D

Entrada:

- MOP:

$$\boldsymbol{F} = [f_1, \ldots, f_m]$$
$$\boldsymbol{C} = [g_1, \ldots, g_n]$$
$$\Omega = [(x_{L1}, x_{U1}), \ldots, (x_{Lp}, x_{Up})]$$

- N: número de subproblemas
- G: número de generaciones
- T: número de vecinos
- EOP: operador evolutivo
- UN: Updations number
- NDS: Usar/no usar NDS.

Inicialización:

- $\boldsymbol{\lambda}$. Vectores de pesos $(\lambda_1, \ldots, \lambda_N)$
- B. Vecindad de los subproblemas $[[B_{1,1}, \ldots, B_{1,T}], \ldots, [B_{N,1}, \ldots, B_{N,T}]]$
- P. Población $(\boldsymbol{I_1}, \ldots, \boldsymbol{I_N})$
- FP. Valoración de la población $(\boldsymbol{F}(P))$
- VP. Violación de restricciones de la población $(\boldsymbol{V}(P))$
- $\boldsymbol{z}$. Punto de referencia $(z_1, \ldots, z_m)$
- *Conjunto de soluciones no dominadas (NDS Set)*.

Actualización (hasta el límite G):
Para $i = 1, \ldots, N$

- $\boldsymbol{y_i} \leftarrow EOP(\boldsymbol{P_i})$
- Evaluar $\boldsymbol{y}$: $\boldsymbol{F}(\boldsymbol{y})$ y $V(\boldsymbol{y})$
- Actualizar $\boldsymbol{z}$
- Actualizar soluciones vecinas de i.
- Actualizar NDS Set

Salida:

- Aproximación al frente de Pareto. (NDS Set)

- **Vecindad de los subproblemas** B. Corresponde a una lista de listas que contienen los índices considerados dentro de la vecindad de un subproblema, de forma que para cada subproblema i son considerados como vecinos los T cuyos vectores asociados son más cercanos a $\boldsymbol{\lambda_i}$. Para ello se crea (temporalmente) un matriz de distancias $D_{N\times N}$ tal que cuya posición i, j corresponde a la distancia euclídea entre $\boldsymbol{\lambda_i}$ y $\boldsymbol{\lambda_j}$. Una vez calculada dicha matriz, el vecindario de i corresponde a los T j con menor d_{ij}.

- **Población** $P = [\boldsymbol{I_1}, \ldots, \boldsymbol{I_N}]$. Se inicializa con valores aleatorios (dentro del espacio de búsqueda) para cada una de las variables de cada uno de los individuos. Para ello se genera una matriz $\hat{P}_{N\times p}$ de números aleatorios (entre 0 y 1). De forma que ahora P es obtenida aplicando a cada elemento de cada columna la expresión $P_{ij} = \hat{P}_{ij}\,(x_{Uj} - x_{Lj}) + x_{Lj}$, obteniendo un número aleatorio entre x_{Lj} y x_{Uj}. De forma que la población P es almacenada como una matriz $P_{N\times p}$ en la que cada fila corresponde al individuo $\boldsymbol{P_i}$ asociado al problema i; y cada columna corresponde al valor de las características de dicho individuo (genotipo).

- **Valoración de la población** $FP_{N\times m} = \boldsymbol{F}(P)$. Es inicializada con la valoración de los individuos de P. Para ello a cada una de los individuos (filas) de P son valorados según las funciones objetivo $f_1, \ldots . f_m$, dando lugar a la matriz FP, en la que cada fila i corresponde a las valoraciones del individuo $\boldsymbol{P_i}$ (fenotipo) y cada columna j a la valoración según f_j

- **Violación de restricciones de la población** $VP^{(N)} = V(P)$. Es inicializada con la violación de restricciones de cada uno de los individuos de P. Para ello a cada una de los individuos (filas) de P se les aplica $V(\boldsymbol{x}) = -\sum_{j=1}^{n} \text{mín}(g_j(\boldsymbol{x}), 0)$ dando lugar a un vector (o lista) de longitud N con la cuantía de violación (nótese que es positivo, ya que todos los sumandos son no positivos, esto es, negativos o nulos) de cada uno de los individuos.

- **Punto de referencia** $\boldsymbol{z} = (z_1, \ldots, z_m)$. Corresponde al vector de los mejores valores encontrados para cada una de las funciones objetivo $f_1, \ldots . f_m$. Teniendo en cuenta que inicialmente solo se tienen los individuos de P, cuyas valoraciones corresponden a VP, para la inicialización de z basta coger el mínimo de cada columna de FP que será el mejor (menor) valor encontrado hasta el momento) indistintamente de si es factible o infactible.

- **Conjunto de soluciones no dominadas *NDS Set***. Corresponde al conjunto de vectores en el espacio de objetivos (valoraciones) que no son dominada por ninguna encontrada hasta el momento. Teniendo en cuenta que inicialmente solo se han encontrado las soluciones de VP basta recorrer VP y añadir *NDS Set* aquellas que no son dominadas por ninguna otra, utilizando el concepto de Pareto-dominancia extendido. Esto es equivalente a tomar el inicialmente *NDS Set* y realizar una actualización de *NDS Set* con cada solución (vector fila) de VP. Dicha actualización se realiza según los pseudocódigos presentados en el algoritmo 2 y el algoritmo 3

Algoritmo 2: Actualización de *NDS*

Entrada:

- *NDS*: (NDS_P, NDS_F, NDS_V)
- $\boldsymbol{x}, F^{(x)} = F(\boldsymbol{x}), V^{(x)} = V(\boldsymbol{x})$: vector de actualización

Actualización del NDS:

- $NDS' \leftarrow copy(NDS_P) \implies NDS' = (NDS'_P, NDS'_F, NDS'_V)$;
- *Para* $i = 1, \ldots, |NDS_P|$: /*$|NDS_P| = |NDS_F| = |NDS_V|$*/
  - $F^{(j)} \leftarrow NDS_F[j]$; $V_j \leftarrow NDS_V[j]$
  - Si $(F^{(j)}, V^{(j)}) \succ (F^{(x)}, V^{(x)})$: STOP
  - Si $(F^{(x)}, V^{(x)}) \succ (F^{(j)}, V^{(j)})$: $NDS'[j] \leftarrow (x, F_x, V_x)$
- Devolver $NDS \leftarrow unique(NDS')$

Salida:

- *NDS* Actualizado.

Algoritmo 3: Dominancia Pareto extendida: $(F^{(a)}, V^{(a)}) \succ (F^{(b)}, V^{(b)})$

Entrada:

- $s_a = (F^{(a)}, V^{(a)})$, $s_b = (F^{(b)}, V^{(b)})$
- m: número de objetivos

Desarrollo:

- Si $V^{(a)} < V^{(b)}$: devolver *True*
- Si $V^{(a)} > V^{(b)}$: devolver *False*
- Si $V^{(a)} = V^{(b)}$:
  - Si $\exists j = 1, \ldots, m / F_i^{(b)} < F_i^{(a)}$: devolver *False*
  - Si no, $\exists j = 1, \ldots, m / F_i^{(a)} < F_i^{(b)}$: devolver *True*
  - Si no: devolver *False*

Salida: Si s_a domina a s_b: $s_a \succ s_b$

- *NDS* Actualizado.

3.3. Desarrollo evolutivo (iterativo)

Una vez inicializado el algoritmo se lleva a cabo un proceso iterativo de evolución durante $G - 1$ iteraciones (de forma que el algoritmo genera en total G generaciones y realiza $G \cdot N$ evaluaciones. Durante cada una de las iteraciones del proceso evolutivo se lleva a cabo:

3.3.1. Reproducción

Consistente en la generación de los nuevos descendientes. Para ello, se pueden utilizar distintos operadores evolutivos con la condición de que devuelvan un único descendiente. Para que el algoritmo tenga un carácter más general, se propone que la función generadora de los descendientes EOP sea dada al algoritmo también como entrada, aunque en la experimentación utilizaremos el operador EOP1 descrito a continuación.

Algoritmo 4: Pseudocódigo EOP1

Entrada:

- i: índice del individuo actual
- P: Población (genotipos)
- VP: Valoración de la población (fenotipos)
- B_i: Índices de vecinos de i
- Ω: Espacio de búsqueda.
- Parámetros :
  - Fs : Pool de valores para F
  - CR : Probabilidad de cruze
  - PM : Probabilidad de mutación
  - SIG : Apertura de la mutación

(1) Mutación de $\boldsymbol{P_i}$: Vector mutante $\hat{\boldsymbol{y}}$.

1. $r_1, r_2, r_3, r_4, r_5 \in B_i$.
2. $\hat{\boldsymbol{y}} \leftarrow \boldsymbol{P_{r1}} + F \cdot (\boldsymbol{P_{r2}} - \boldsymbol{P_{r3}}) + rand \cdot (\boldsymbol{P_{r2}} - \boldsymbol{P_{r3}})$

(2) Cruce de P_i con $\hat{\boldsymbol{y}}$:

1. $Cp \leftarrow (\dots, Cp_\delta = 1, \dots) \in \{0,1\}^p$.
2. $y_j \leftarrow Cp \odot \hat{\boldsymbol{y}} + (\mathbf{1}^{(p)} - Cp) \odot \boldsymbol{P_i}$

(3) Mutación de $\boldsymbol{y}$:

Para $j = 1, \dots, dim(\boldsymbol{P_i})$

1. Si $rand \in [0,1] < PM$:
   a) $\sigma_j \leftarrow |x_{Uj} - x_{Lj}|/SIG$
   b) $y_j \leftarrow y_j + rand \in \mathcal{N}(0, \sigma_j)$

(4) Reparación de $\boldsymbol{y}$:

Para $j = 1, \dots, dim(\boldsymbol{P_i})$

1. $y_j \leftarrow \text{mín}(\text{máx}(y_j, x_{Lj}), x_{Uj})$

Salida: Descendiente $y \in \Omega$

EOP1. Basado en DE + GP. En el *algoritmo 4* se presenta el pseudocódigo de la función que describimos a continuación. El operador evolutivo EOP1 toma un individuo P_i y genera un desciendente $\boldsymbol{y}$ a partir de él, de forma que:

1. Realiza una primera mutación de $\boldsymbol{P_i}$ (procendente de DE) de forma que se genera un vector mutante a partir de cinco vectores escogidos del entorno de P_i. En concreto se escogen al azar cinco elementos del vecindario de i (correspondientes a los subproblemas $r1$, $r2$. $r3$, $r4$ y $r5$, con reemplazamiento si es necesario) y se genera el vector mutante aplicando la expresión descrita a partir de los vectores asociados a los subproblemas que son justamente las filas correspondientes a los índices $r1$, $r2$ y $r3$ de la población (matriz) P, de forma que el vector mutante corresponde a $\hat{\boldsymbol{y}} = P_{r1} + F(P_{r2} - P_{r3})$. Con F un parámetro de la función establecido por el usuario (típicamente 0,5).

2. Realiza el cruce del individuo $\boldsymbol{P_i}$ con el vector mutante $\hat{\boldsymbol{y}}$. Para ello se genera un vector de números aleatorios booleanos $\boldsymbol{CP}$ (crossing points) (con probabilidad CR, dada por el usuario, típicamente

$0{,}5$), esto es un vector de valores 0 y 1, de forma que para garantizar que el vector resultante contiene al menos un elemento del vector mutante, una posición aleatoria de CP es fijada a 1. Ahora para el vector resultante (descendiente) se toman de $\hat{\boldsymbol{y}}$ aquellas componentes en las que $CP_j = 1$ y de $\boldsymbol{P_i}$ aquellas que son 0, esto es justamente (en operaciones vectoriales) $\boldsymbol{y} = \boldsymbol{CP} \odot \hat{\boldsymbol{y}} + (\mathbf{1}^{(p)} - \boldsymbol{CP}) \odot \boldsymbol{P_i}$ (con $\mathbf{1}^{(p)}$ el vector p-dimensional con todas sus componentes 1).

3. Realiza una perturbación gaussiana en algunas de las componentes del descendiente. De forma que para cada una de las componentes se sortea aleatoriamente con probabilidad PM (dada por el usuario o en su ausencia $1/p$, con p el número de componentes de los individuos) y en caso favorable dicha componente y_j es perturbada añadiéndole una cantidad aleatoria obtenida de una distribución normal (gaussiana) de media 0 y desviación estándar $\frac{|x_{Uj} - x_{Lj}|}{SIG}$ (donde SIG corresponde a un parámetro ajustable por el usuario, típicamente 20).

4. Finalmente cada una de las componentes de y que se encuentren fuera del espacio de búsqueda son ajustadas a la cota correspondiente, equivalente a aplicar la función $y_j = min(max(y_j, x_{Lj}), x_{Uj})$ sobre cada componente.

3.3.2. Evaluación del descendiente

Una vez generado el sucesor éste es evaluado respecto a cada uno de los objetivos almacenándose en un vector $\boldsymbol{f_y}$ las valoraciones tal que la posición k de dicho vector corresponde a $f_k(\boldsymbol{y})$.

De igual forma el sucesor es evaluado respecto de las restricciones, calculándose y almacenándose $V_y = V(y)$.

3.3.3. Actualización del punto de referencia

Como $\boldsymbol{z}$ debe mantener en todo momento los mejores valores obtenidos por alguno de los individuos, es necesario actualizar aquellas componentes en las que $\boldsymbol{f_y}$ sea mejor (menor) que $\boldsymbol{z}$, de forma que para cada componente $j = 1, \ldots, m$, se actualiza $\boldsymbol{z}$ con $z_j = \min(z_j, (f_y)_j)$.

3.3.4. Actualización de los vecinos

A continuación se aplicará sobre el vecindario el operador de selección descrito en la metodología. Para ello se opera siguiendo el pseudocódigo presentado en el algoritmo 5.

Como se ha indicado previamente la actualización de muchos de los vecinos puede llevar al algoritmo a una convergencia prematura (estancamiento) por ello se controla mediante el parámetro UN el número máximo de actualizaciones.

Algoritmo 5: Actualización de la vecindad del subproblema i

Entrada:

- i: el subproblema considerado.
- B: la lista de vecindades.
- P, FP, VP: la población y su valoración de objetivos y restricciones.
- $\boldsymbol{\lambda}$: la lista de vectores de pesos.
- z: el vector de mejores valores objetivo.
- NDS: conjunto de soluciones no dominadas.
- $\boldsymbol{y}, \boldsymbol{f_y}, V_y$: el descendiente y su valoración de objetivos y restricciones.
- it: el número de generación actual.
- G: el número máximo de generaciones.

Actualización de la vecindad:

- $c = 0$, $\eta = 20^{\frac{it}{G}}$, $s = it < \frac{3*G}{4} ? 10^{-\frac{G}{it}} : \eta$;
- *Para* $j = 1, \ldots, B(i)$:
  - $\tau = mean(\{V_x \in VP[B(j)] / V_x > 0\})$
  - Si $V_y = 0 \wedge (V_x > 0 \vee g^{te}(\boldsymbol{y}|\lambda_j, \boldsymbol{z})$:
    - $P[j] = \boldsymbol{y}$, $FP[j] = \boldsymbol{f_y}$, $VP[j] = V_y$
    - $c \leftarrow c + 1$
  - Si no:
    - $\boldsymbol{x_e} = rand \in NDS_P$, $\boldsymbol{f_e} = (NDS_F)_e$, $V(e) = (NDS_V)_e = V(e)$
    - $winner \leftarrow argmin[fitness_y, fitness_j, fitness_e]$
    - Si $winner = 1$:
      - $P[j] = \boldsymbol{y}$, $FP[j] = \boldsymbol{f_y}$, $VP[j] = V_y$
      - $c \leftarrow c + 1$
    - Si $winner = 3$:
      - $P[j] = \boldsymbol{x_e}$, $FP[j] = \boldsymbol{f_e}$, $VP[j] = V_e$
- Devolver $NDS \leftarrow unique(NDS')$

Salida:

- *NDS* Actualizado.

3.3.5. Actualización de *NDS Set*

Para mantener en todo el momento la consistencia del conjunto cada vez que es generado un descendiente se actualiza el conjunto añadiendo la valoración asociada y eliminando las soluciones dominadas por ella, si procede. Para ello se recorre el conjunto de soluciones dominadas, comparando cada solución con $\boldsymbol{F}(\boldsymbol{y})$ en el momento en que se encuentre una que la domine, el proceso es abortado. En caso de que no

se encuentre ninguna, se habrán ido eliminando, al mismo tiempo que se recorren, aquellas soluciones dominadas por $\boldsymbol{F}(\boldsymbol{y})$, por lo que basta añadir $\boldsymbol{F}(\boldsymbol{y})$ al conjunto para completar la actualización.

3.3.6. Otras cuestiones

Nótese que el proceso de evolución (actualización de P y VP) se realiza de forma síncrona, esto es la actualización de P_i se realiza directamente sobre P y no en una matriz auxiliar que luego sea volcada. Por ello, los individuos son recorridos en orden aleatorio en cada iteración (mediante una lista con los índices que es reordenada aleatoriamente en cada iteración) de manera que las actualizaciones influyan (probabilísticamente) igual en todos los individuos.

A fin de que se pueda testear el funcionamiento del algoritmo al final de cada iteración, la población (su valoración) es volcada a un fichero *ASCII* en el que cada fila corresponde a un individuo y sus columnas reflejan las valoraciones del individuo y una última columna con el número de restricciones violadas. También es generado un fichero con el contenido de todas las generaciones.

Referencias

[1] F. Vaz, Y. Lavinas, C. Aranha, and M. Ladeira, "Exploring Constraint Handling Techniques in Real-world Problems on MOEA/D with Limited Budget of Evaluations," Tech. Rep.

[2] Z. Fan, H. Li, C. Wei, W. Li, H. Huang, X. Cai, and Z. Cai, "An improved epsilon constraint handling method embedded in MOEA/D for constrained multi-objective optimization problems," in *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*. Institute of Electrical and Electronics Engineers Inc., feb 2017, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7850224/

[3] M. Asafuddoula, T. Ray, R. Sarker, and K. Alam, "An adaptive constraint handling approach embedded MOEA/D," in *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 2012.

[4] Y. Yang, J. Liu, and S. Tan, "A constrained multi-objective evolutionary algorithm based on decomposition and dynamic constraint-handling mechanism," *Applied Soft Computing Journal*, vol. 89, p. 106104, apr 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1568494620300442

[5] M. A. Jan and Q. Zhang, "Moea/d for constrained multiobjective optimization: Some preliminary experimental results," in *2010 UK Workshop on Computational Intelligence (UKCI)*, 2010, pp. 1–6.

[6] Q. Zhu, Q. Zhang, Q. Lin, and J. Sun, "MOEA/D with Two Types of Weight Vectors for Handling Constraints," in *2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., jun 2019, pp. 1359–1365. [Online]. Available: https://ieeexplore.ieee.org/document/8790336/

[7] B. Liu, F. V. Fernandez, Q. Zhang, M. Pak, S. Sipahi, and G. Gielen, "An enhanced moea/d-de and its application to multiobjective analog cell sizing," 08 2010, pp. 1 – 7.