Algoritmos evolutivos para agrupar información biomédica en un número desconocido de grupos

María Eugenia Curi*, Lucía Carozzi*, Renzo Massobrio*, Sergio Nesmachnow*,
Grégoire Danoy†, Marek Ostaszewski‡, Pascal Bouvry†
*Universidad de la República, Uruguay. Email: {maria.curi, lucia.carozzi, renzom, sergion}@fing.edu.uy
†CSC Research Unit, University of Luxembourg, Luxemburgo. Email:{gregoire.danoy, pascal.bouvry}@uni.lu
‡Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxemburgo. Email: marek.ostaszewski@uni.lu

***Resumen*—Este artículo presenta el diseño e implementación de algoritmos evolutivos para resolver el problema de agrupamiento en un número desconocido de grupos. Se proponen operadores evolutivos simples adaptados al problema con el objetivo de mantener la búsqueda evolutiva tan simple como sea posible, para permitir a los métodos propuestos escalar y resolver problemas de gran dimensión. La evaluación experimental se realiza sobre un conjunto de instancias reales del problema, incluyendo un caso real de análisis y categorización de información biomédica del proyecto que propone construir un mapa de la enfermedad de Parkinson. Los principales resultados muestran que el enfoque evolutivo permite calcular soluciones con buenos niveles de compromiso y es capaz de manejar la complejidad de las instancias que involucran información biomédica.**

I. Introducción

El problema de agrupamiento propone agrupar un conjunto de elementos de manera que elementos en el mismo grupo sean más similares entre sí que con elementos en otros grupos [1]. La similitud entre elementos se evalúa a través de una métrica de similitud previamente definida, que se busca maximizar. El problema de agrupamiento es muy importante en ciencia de datos. De hecho, es considerado como uno de los problemas más importantes de aprendizaje no supervisado, ya que permite modelar otros problemas que tratan sobre descubrir estructuras en un conjunto de datos.

En particular, la investigación biomédica trata con un gran número de conceptos vinculados entre sí por relaciones complejas, que a menudo se representan utilizando grandes grafos. Para procesar y comprender estas bases de conocimiento, los investigadores necesitan herramientas confiables para visualizar y explorar grandes cantidades de datos, organizando apropiadamente los conceptos con características similares.

El problema de agrupamiento es NP–difícil [2] y ha sido estudiado exhaustivamente en los últimos 30 años [3]. Diferentes heurísticas y metaheurísticas [4] se han aplicado para resolver instancias realistas del problema. Entre ellas, los algoritmos evolutivos (AE) han demostrado ser técnicas útiles para resolver el problema de agrupamiento [5], [6].

Este artículo aborda dos formulaciones del problema de agrupación, una en la que el número de grupos se conoce de antemano y una versión multiobjetivo que propone maximizar la similitud entre elementos en el mismo grupo y minimizar el número de agrupaciones, simultáneamente.

Se presentan AE para el problema de agrupamiento con objetivo único y para la versión multiobjetivo, que se comparan contra varios métodos de agrupamiento de la literatura relacionada. En particular, se estudian casos reales del proyecto de mapa para la enfermedad de Parkinson [7], una iniciativa de investigación que propone construir un repositorio de conocimiento para describir los mecanismos moleculares relacionados con ese trastorno neurodegenerativo crónico [8]. El repositorio compila información sobre la enfermedad de Parkinson basada en la literatura y organiza los principales conceptos y contenidos en un mapa fácil de explorar y accesible libremente, incluyendo datos experimentales, información de fármacos y otros conceptos.

El artículo se organiza de la siguiente manera. Las metaheurísticas y los AE se introducen en la Sección II. La Sección III presenta la formulación del problema de agrupamiento en sus versiones de objetivo único y multiobjetivo y una reseña de trabajos relacionados. Los AE propuestos se describen en la Sección IV y la evaluación experimental se reporta en la Sección V. Finalmente, la Sección VI presenta las conclusiones y las principales líneas de trabajo futuro.

II. Metaheurísticas, Computación Evolutiva y Algoritmos Evolutivos Multiobjetivo

Esta sección presenta los métodos aplicados en este artículo para resolver el problema de agrupación: metaheurísticas, computación evolutiva y algoritmos evolutivos multiobjetivo.

II-A. Metaheurísticas

Las metaheurísticas son estrategias para diseñar métodos eficaces y precisos para resolver problemas de optimización. Definen métodos de alto nivel basados en heurísticas, que pueden aplicarse a diferentes problemas instanciando un esquema genérico de resolución [4]. Muchos problemas de optimización que surgen en aplicaciones del mundo real son NP–difíciles, ya que tienen espacios de búsqueda muy grandes o dispersos, como consecuencia de restricciones duras. Este es el caso del problema que se resuelve en este trabajo: el agrupamiento de grandes conjuntos de datos. Las metaheurísticas son métodos eficientes y precisos para resolver casos realistas de problemas de optimización, para los cuales los métodos exactos demandan tiempos poco útiles en la práctica. En este trabajo se aplican AE, los cuales se describen a continuación.

II-B. Algoritmos Evolutivos

Los AE son métodos estocásticos que emulan la evolución natural para resolver problemas de optimización, búsqueda y aprendizaje [9]. En los últimos 30 años, los AE se han aplicado con éxito para resolver problemas de optimización que subyacen a muchas aplicaciones reales y complejas. El Algoritmo 1 presenta el pseudocódigo de un AE.

Algoritmo 1 Pseudocódigo de un Algoritmo Evolutivo

```
1: t ← 0                                          ▷ contador de generaciones
2: inicializar(P(0))                              ▷ inicialización de la población
3: mientras no criterio_parada hacer
4:     evaluar(P(t))                              ▷ evaluación de la población
5:     padres ← selección(P(t))
6:     hijos ← operadores de variación(padres)
7:     P(t+1) ← reemplazo(hijos, P(t))
8:     t = t + 1
9: fin mientras
10: retornar mejor solución hallada
```

Un AE es una técnica iterativa (cada iteración se denomina *generación*) que aplica operadores estocásticos en un conjunto de *individuos* (la población *P*) con el fin de mejorar su aptitud (*fitness*), una medida relacionada con la función objetivo que evalúa cuán buena es una solución para resolver el problema. Cada individuo en la población codifica una solución candidata para el problema. La población inicial se genera mediante un método aleatorio o utilizando una heurística específica para el problema. Una función de evaluación asocia un valor de aptitud a cada individuo, indicando su idoneidad al problema. La búsqueda es guiada por una selección hacia soluciones tentativas de mayor calidad. Iterativamente, las soluciones se modifican aplicando probabilísticamente *operadores de variación*, incluyendo la *recombinación* de individuos o cambios aleatorios (*mutación*) en su contenido, construyendo nuevas soluciones durante la búsqueda. El criterio de parada suele implicar un número fijo de generaciones o tiempo de ejecución, un umbral de calidad sobre el mejor valor de aptitud o la detección de una situación de estancamiento. Se utilizan políticas específicas para seleccionar los individuos a recombinar (*selección*) y para determinar qué nuevos individuos se insertan en la población en cada nueva generación (*reemplazo*). El AE retorna la mejor solución encontrada en el proceso iterativo, teniendo en cuenta la función de fitness.

II-C. Algoritmos Evolutivos Mutiobjetivo

Los AE mutiobjetivo (Multi-Objective Evolutionary Algorithm, MOEA) [10], [11] son AE específicamente concebidos para resolver problemas con dos o más objetivos en conflicto. Los MOEA han demostrado ser métodos eficientes para la resolución de problemas de optimización complejos en muchas áreas de investigación.

A diferencia de otros métodos tradicionales de optimización multiobjetivo, los MOEA permiten encontrar un conjunto con varias soluciones en una única ejecución, ya que trabajan con una población de soluciones tentativas en cada generación.

Los MOEA se diseñan para cumplir dos objetivos simultáneamente: *i*) aproximar el frente de Pareto, usando una búsqueda evolutiva basada en dominancia, y *ii*) mantener la diversidad, en lugar de converger a una sección particular del frente de Pareto, usando técnicas de optimización de funciones multimodales (por ejemplo, sharing, crowding).

Este artículo propone resolver la versión multiobjetivo del problema de agrupación utilizando Non–dominated Sorting Genetic Algorithm, versión II (NSGA-II) [11]. El Algoritmo 2 presenta el pseudocódigo de NSGA-II. El cálculo de fitness se basa en dominancia de Pareto, construyendo frentes de soluciones no dominadas. NSGA-II aplica: *i*) un ordenamiento no dominado elitista que reduce la complejidad de los chequeos de dominancia; *ii*) una técnica de crowding para preservar la diversidad; y *iii*) una asignación de aptitud que considera valores de distancia de crowding.

Algoritmo 2 Pseudocódigo del algoritmo NSGA-II

```
Entrada: N, tamaño de la población
1: t ← 0                                          ▷ contador de generaciones
2: hijos ← ∅
3: inicializar(P(0))                              ▷ inicialización de la población
4: mientras no criterio_parada hacer
5:     evaluar(P(t))                              ▷ evaluación de la población
6:     R ← P(t) ∪ hijos
7:     frentes ← ordenamiento no dominado(R))
8:     P(t+1) ← ∅; i ← 1
9:     mientras |P(t + 1)| + |frentes(i)| ≤ N hacer
10:        distancia de crowding(frentes(i))
11:        P(t+1) ← P(t+1) ∪ frentes(i)
12:        i ← i+1
13:    fin mientras
14:    ordenar por distancia (frentes(i))
15:    P(t+1) ← P(t+1) ∪ frentes(i)[1:(N - |P(t+1)|)]
16:    seleccionados ← selección(P(t+1))
17:    hijos ← operadores de variación(seleccionados)
18:    t ← t + 1
19: fin mientras
20: retornar frente de Pareto calculado
```

III. El problema de agrupamiento

Esta sección presenta el problema de agrupamiento en sus versiones de objetivo único y multiobjetivo y una revisión de trabajos relacionados.

III-A. Formulación del problema

Dados:

- Un conjunto $E = \{e_1, e_2, ..., e_n\}$ de elementos a agrupar.
- Una función $s : E \times E \rightarrow [0, 1]$, donde $s(e_i, e_j)$ indica la similitud entre e_i y e_j. Se cumplen las siguientes condiciones: $\forall e_i, e_j, s(e_i, e_j) = s(e_j, e_i)$ y $s(e_i, e_i) = 1$.
- (*solo para la versión de objetivo único*) Un entero $k > 0$, que indica el número de grupos considerados.

El problema consiste en asignar los elementos de E a un conjunto de grupos $G = \{G_1, ..., G_k\}$, donde $G_i = \{c_i\} \cup \{e_m / s(e_m, c_i) \leq s(e_m, c_j) \forall e_m \in E, c_j, c_i \in C, i \neq j\}$; $C \subseteq E$, $|C| = k$ es el conjunto de centros de los grupos.

Se cumplen las siguientes propiedades:
(a – *índice de grupo en* $[1,k]$) $\forall(i,j), i \neq j : 1 \leq i,j \leq k$,
(b – *los grupos son disjuntos*) $G_i \cap G_j = \emptyset$.

Considerando la métrica de *similitud total* (TS) definida en la Ecuación 1, la versión del problema con objetivo único propone maximizar el valor de *TS* (Ecuación 2).

$$TS = \sum_{e_i \in E} \max_{c_i \in C} \; s(e_i, c_i) \tag{1}$$

$$\text{[versión de objetivo único del problema]} \quad \max \quad TS \tag{2}$$

La versión multiobjectivo propone maximizar el valor de *TS* y minimizar el número de grupos k, simultáneamente (Ecuación 3).

$$\text{[versión multiobjetivo del problema]} \quad \begin{cases} \max TS \\ \min k \end{cases} \tag{3}$$

La métrica de similitud fue definida de acuerdo al conocimiento del problema de agrupamiento de información biomédica en el contexto del proyecto para la creación de un mapa de la enfermedad de Parkinson.

III-B. Trabajos relacionados

Varios artículos han presentado métodos heurísticos y metaheurísticos aplicados al problema de agrupación. Trabajos iniciales consideraron la versión con objetivo único, basándose en la minimización de la distancia o la maximización de la similitud. Las versiones multiobjetivo del problema han sido abordadas más recientemente.

La heurística ávida k-means, propuesta por MacQueen en 1967 [12], se basa en definir k centros aleatorios y seleccionar elementos de acuerdo a la distancia a los grupos ya construidos. Los grupos se recalculan después de incorporar cada nuevo elemento. K-means no es el mejor algoritmo de agrupamiento pero es muy utilizado debido a su simplicidad, escalabilidad y velocidad de convergencia [13]. Los principales inconvenientes de k-means están relacionados con los criterios utilizados para construir grupos, que dependen en gran medida de la selección inicial de k grupos al azar, y del método de construcción que no permite construir soluciones intermedias que no mejoren los criterios de optimización y, por lo tanto, es propenso a quedar atrapado en óptimos locales.

Otro algoritmo clásico de agrupamiento es k-medoids [1], que funciona seleccionando elementos (*medoids*) como centros y explorando vecindarios de la solución inicial. Los métodos de particionamiento (*Partitioning Around Methods*, PAM) implementan k-medoids aplicando dos fases para construir y mejorar grupos: *i*) (*build*) construye k grupos, según un procedimiento basado en distancia y *ii*) (*swap*) intercambia pares de elementos (i,h), donde i es un centro y h no lo es.

Park y Jun [14] presentaron una implementación iterativa de k-medoids usando métodos basados en distancia para encontrar nuevos medoids en cada iteración. Varios métodos se propusieron para realizar la selección inicial de los medoids: selección aleatoria, selección ordenada, selección de un determinado porcentaje de elementos, selección de los elementos de mayor distancia y selección basada en la distancia entre parejas. Los métodos propuestos calcularon soluciones competitivas y redujeron significativamente el tiempo de ejecución en comparación con los métodos basados en PAM para instancias de problemas reales y artificiales con hasta 360 elementos y 3 grupos. Das et al. [15] estudiaron la aplicación de metaheurísticas para diferentes versiones del problema de agrupación, concluyendo que las metaheurísticas basadas en trayectoria tienen limitaciones para resolver el problema, principalmente debido a su baja escalalabilidad al resolver grandes instancias. Deng y Bard [16] aplicaron GRASP para el problema capacitado, que propone agrupar elementos en grupos que tienen restricciones de capacidad (número mínimo y máximo de elementos). El GRASP propuesto opera en cuatro etapas: i) *construcción inicial* aplicando dos estrategias para identificar aquellos elementos que no pertenecen al mismo grupo en una partición hipotética óptima; ii) *búsqueda local* en el vecindario de las soluciones construidas; iii) *selección de soluciones élite* para aplicar la cuarta fase; y iv) *path relinking* para mejorar soluciones élite. El GRASP se evaluó en tres instancias pequeñas con hasta 50 elementos y 5 agrupaciones, y los resultados se compararon con soluciones exactas calculadas con CPLEX. GRASP fue capaz de encontrar soluciones óptimas para las instancias de problemas con 30 y 40 nodos, y superó la solución encontrada usando CPLEX cuando se utilizó un límite de tiempo de ejecución de una hora.

Las metaheurísticas basadas en población, y especialmente los AE, también se han aplicado para resolver problemas de agrupamiento. Sheng y Liu [6] compararon k-medoids, una búsqueda local basada en trayectoria y un algoritmo k-means híbrido (HKA). La búsqueda local opera en dos bucles: el bucle externo asigna cada elemento al centro más cercano y el bucle interno realiza modificaciones considerando un conjunto de p vecinos más cercanos para actualizar los k centros, tratando de minimizar la distancia. HKA es un AE que utiliza una codificación entera, inicialización aleatoria, selección por torneo, el operador de recombinación de subconjunto de mezcla y una mutación de inversión de bits. Los tres métodos fueron evaluados sobre dos conjuntos de datos con 517 elementos y 10 grupos, y 2945 elementos y 30 grupos. HKA obtuvo los mejores resultados en la instancia más grande y resultados ligeramente mejores para la instancia pequeña.

Trabajos iniciales propusieron AE sin utilizar un enfoque multiobjetivo explícito. Cowgill et al. [17] optimizaron métricas de agrupación definidas en términos de aislamiento de grupos y homogeneidad interna de grupos. El AE mejoró los resultados de algoritmos de agrupación jerárquica teniendo en cuenta el criterio interno.

Maulik et al. [18] revisaron varios MOEA para el problema de agrupación. La mayoría de las propuestas se centran en la optimización de dos métricas de similitud, estudiando así diferentes características de los datos a analizar. Considerar varias métricas de agrupación, y optimizar la compacidad y la separación de elementos como criterios diferentes, es una mejor opción que combinar los objetivos del problema en una única métrica (artificial) para optimizar.

Ripon et al. [19] propusieron un MOEA considerando la variación intragrupos y la distancia intergrupos, sin asumir un número fijo de grupos. El análisis experimental realizado sobre problemas con hasta 3000 elementos, nueve clases y dos características mostró que el MOEA fue capaz de mejorar a un NSGA–II personalizado, implementado por los autores.

Handl y Knowles propusieron MOCK, un MOEA para agrupamiento con determinación automática del número de grupos, considerando funciones objetivo basadas en la compacidad y en la conectividad de los grupos. Estos objetivos son contradictorios ya que la compacidad mejora cuando se utilizan más grupos, pero la conectividad disminuye. MOCK se comparó con algoritmos de agrupamiento de un solo objetivo, logrando buenos resultados y escalabilidad. Korkmaz et al. [20] presentaron un MOEA basado en Pareto para hallar agrupamientos no dominados considerando la variación intragrupos y el número de grupos. La evaluación experimental se realizó sobre dos pequeños conjuntos de datos estándar (150 y 75 elementos, con sólo dos atributos), pero no se reportaron resultados numéricos o análisis de optimización multiobjetivo.

Otros trabajos han aplicado enfoques multiobjetivo al problema de clasificación difusa, donde el agrupamiento también es importante. Bandyopadhyay et al. [21] optimizaron una métrica de compacidad y una medida global de la variación intragrupos dividida por una medida local (la distancia entre los dos grupos más cercanos), para la clasificación de píxeles en imágenes de teledetección. Banerjee [22] aplicó un estimador difuso para optimizar el agrupamiento y el número de grupos para datos ruidosos, obteniendo mejores resultados que los algoritmos tradicionales de agrupación difusa.

La mayoría de los trabajos revisados han propuesto AE personalizados para el problema de agrupación y pocos de ellos han resuelto versiones multiobjetivo. Este artículo contribuye con AE simples y con un MOEA explícito, diseñados para escalar y resolver grandes instancias del problema. El estudio se centra en una instancia realista del problema que involucra la organización de información biomédica en el contexto del proyecto de mapa de la enfermedad de Parkinson.

IV. Algoritmos evolutivos para agrupamiento

Esta sección presenta los AE monoobjetivo y el MOEA propuesto para resolver el problema de agrupación.

IV-A. AE monoobjetivo

Función de aptitud. La función de aptitud es sencilla y consiste en calcular la suma de las similitudes entre cada elemento y su centro más similar, tal como se presenta en la formulación matemática en la Sección III-A.

Representación de soluciones. Se proponen dos representaciones de soluciones, que posteriormente se evalúan en el análisis experimental.

La representación *binaria* codifica cada solución como un vector binario de longitud N (el número de elementos a agrupar). Cada posición i en el vector representa si el elemento es un centro de grupo (1) o no (0). El número de 1s en una codificación debe coincidir con el número de grupos k.

La representación *entera* codifica cada solución como un vector de k enteros en el rango $1 \dots N$ que representan a los centros de los grupos. Cada entero figura una vez, ya que los grupos deben tener diferentes elementos como centros.

Operadores de recombinación. Se implementaron dos operadores de recombinación diferentes para el AE con codificación binaria: cruzamiento de un punto (SPX) y cruzamiento de dos puntos (2PX). En SPX, una posición de cruce se selecciona al azar para ambos padres y los genes después de este punto de cruce se intercambian entre ambos padres. 2PX selecciona al azar dos posiciones de cruce para ambos padres, e intercambia los genes situados entre estos dos puntos.

Para la codificación entera se implementaron y evaluaron tres operadores de recombinación: SPX, Generalized Cut and Splice (GenC&S), y la Recombinación Híbrida (SPX-GenC&S). GenC&S es una extensión del operador Cut&Splice (C&S) [23], que fue concebido para ser sensible a las propiedades semánticas de una solución, lo que permite una recombinación significativa de los padres. En este artículo se propone una versión del operador GenC&S para el problema de agrupación para preservar características útiles de la información en ambos padres. La versión del operador GenC&S implementada se describe en el Algoritmo 3.

Algoritmo 3 Recombinación GenC&S para agrupamiento

Entrada: padre1, padre2, ε
Salida: hijo1
1: cp = rand(0,k)
2: s = rand(0,k)
3: cp_elemento = padre1[cp]
4: hijo1.agregar(cp_elemento)
5: LP1 = ordenarAscendente(padre1,cp_elemento)
6: LP2 = ordenarAscendente(padre2,cp_elemento)
7: **para** $i = 0$ to $s - 1$ **hacer** ▷ Copiar los primeros s elementos de LP1 a los hijos
8: hijo1.agregar(LP1[i])
9: **fin para**
10: **para** $j = 0$ a $k - s$ **hacer** ▷ Copiar los primeros $N-s$ elementos de LP2 a hijo1
11: **si** similitud(LP2[j],hijo1)$< \varepsilon$ **entonces** ▷ no tan cercano
12: hijo1.agregar(LP2[j]) ▷ ya en hijo1
13: **fin si**
14: **fin para**
15: **mientras** hijo1.largo() $< k$ **hacer** ▷ Completar hijo1 con elementos aleatorios
16: nuevo_centro = rand(0,N)
17: hijo1.agregar(nuevo_centro)
18: **fin mientras**

GenC&S selecciona un punto de corte (cp) en un padre (línea 1) y un entero S, $0 \leq S \leq k$ (línea 2). Luego se crean dos listas ordenadas por similitud con el elemento en la posición cp en el padre1: LP1 con los elementos del padre1 (línea 5) y LP2 con los elementos en el padre2 (línea 6). Los primeros s elementos en LP1 se copian al hijo1 (líneas 7–9). Los $k - s$ elementos restantes en hijo1 se copian desde LP2 en caso que su similitud con los elementos ya copiados al hijo1 sea menor que el parámetro de entrada ε. En caso que menos de k centros se copien al hijo1 la solución se completa con centros seleccionados aleatoriamente (líneas 15–18).

La Recombinación Híbrida combina SPX y GenC&S. En lugar de seleccionar dos números aleatorios cp y s para copiar elementos desde diferentes padres (como en GenC&S), SPX-GenC&S usa un único número aleatorio p. Los elementos ubicados antes de p en el padre1 son copiados desde el hijo1 (como en SPX), y los $k - p$ elementos restantes en el hijo1 se copian del padre2, si su similitud con los elementos ya copiados al hijo1 es menor que el parámetro de entrada ε. En caso que menos de k centros se copien al hijo1 la solución se completa con centros seleccionados aleatoriamente.

Operadores de mutación. Se implementaron cinco operadores de mutación. Para representación binaria:

- *Inversión de bit*. Los valores en una solución se cambian probabilisticamente por el valor binario opuesto.
- *Mutación de incorporación de centro*. Los puntos dato en una solución se cambian probabilisticamente a centros (valor binario = 1).
- *Mutación de eliminación de centro*. Centros en una solución se cambian probabilisticamente a puntos dato (valor binario = 0).

Para representación entera:

- *Mutación de un gen*. Los elementos en una solución se cambian probabilisticamente por otro elemento que no está incluido en la solución. El nuevo elemento se selecciona aleatoriamente de acuerdo con una distribución uniforme en el conjunto E.
- *Mutación de un Gen Adaptada*. Los elementos en una solución se cambian probabilisticamente a un elemento similar al que se quiere mutar. El elemento más similar se encuentra aplicando la siguiente búsqueda: se procesan todos los elementos de la solución y se evalúa la similitud con el elemento que se está mutando. El mejor valor de similitud (γ) se almacena y el nuevo centro se selecciona para tener una similitud menor que γ.

Función correctiva. Algunos de los operadores evolutivos implementados no garantizan la preservación del número de centros en una solución. Por esta razón, es necesaria una función correctiva para mantener la factibilidad de las soluciones. Una función correctiva simple se aplica tanto para codificaciones binarias como enteras. Cuando se utiliza la codificación binaria, en caso de que el número de 1s en la solución (i.e., el número de centros) no sea k después de aplicar un operador evolutivo, la función correctiva agrega o elimina aleatoriamente centros hasta que la solución sea factible. Para la codificación entera, si el mismo elemento aparece más de una vez (i.e., el mismo elemento es centro de más de un grupo) después de aplicar un operador evolutivo, cada elemento repetido se reemplaza con un elemento diferente elegido aleatoriamente según una distribución uniforme sobre el conjunto de elementos.

IV-A1. Inicialización de la población: Los individuos de la población se generan aleatoriamente con una distribución uniforme en $\{0, 1\}$ (codificación binaria) y una distribución uniforme en el conjunto de centros C (codificación entera). De ser necesario se aplica la función correctiva a cada individuo en la población inicial para generar soluciones factibles.

IV-B. AE multiobjetivo

El algoritmo NSGA-II se implementó para la versión multiobjetivo del problema de agrupamiento, como se describe en la Sección II-C. Se aplicó un enfoque incremental para resolver el problema, abordando primero la versión de objetivo único. Luego, se eligió para NSGA-II la codificación y los operadores evolutivos que lograron los mejores resultados en un análisis comparativo realizado utilizando el AE de objetivo único. Como se reportó en la Sección V, los mejores resultados se obtuvieron utilizando *codificación binaria*, *SPX* y *Mutación de eliminación de centros*.

En el problema multiobjetivo la solución con todos los genes en 0 no es factible, ya que no representa ningún agrupamiento. Si se detecta esta situación, la función correctiva añade aleatoriamente un centro a la solución para evitar esta solución. La población inicial se genera aleatoriamente siguiendo una distribución uniforme en $\{0, 1\}$ y la función correctiva se aplica a los individuos generados.

V. Evaluación experimental

Esta sección describe la evaluación experimental de los AE propuestos para el problema de agrupamiento.

V-A. Instancias del problema

Se utilizaron un total de 13 instancias para evaluar los AE propuestos, correspondientes a problemas de agrupamiento que surgen en diferentes campos de estudio. Se incluyeron dos instancias que modelan el mapa de la enfermedad de Parkinson. A continuación, se describen las principales características de cada instancia:

- La instancia #1 se compone de datos hidrométricos correspondientes a 46 cuencas ubicadas en Uruguay.
- Las instancias #2 a #8 y #10 a #12, tienen entre 80 y 846 elementos, y se obtuvieron de Knowledge Extraction based on Evolutionary Learning [24], un repositorio de datos para problemas de clasificación.
- Las instancias #9 y #13 contienen datos del mapa de la enfermedad de Parkinson, que representa visualmente todas las vías moleculares principales implicadas en la patogénesis de la enfermedad de Parkinson. La instancia #9 tiene 801 elementos. La instancia # 13 tiene 3056 elementos y se usa para probar el desempeño del enfoque multiobjetivo en una instancia de gran dimensión que contiene información biomédica.

V-B. Configuración experimental y metodología

Esta subsección describe la configuración utilizada para la evaluación experimental de los algoritmos propuestos y la metodología para analizar los resultados obtenidos.

Implementación de los algoritmos. Los algoritmos se desarrollaron en ECJ [25], un entorno de código libre para computación evolutiva implementado en Java.

Plataforma computacional. La evaluación experimental se realizó en un Intel Core i5 a 2.7GHz y 8 GB de RAM.

Análisis estadístico. Debido a la naturaleza estocástica de los AE, se realizaron 30 ejecuciones independientes de cada algoritmo sobre cada instancia del problema para tener confianza estadística en los resultados. Para cada instancia se reporta el mejor valor y el promedio del fitness obtenido a lo largo de las 30 ejecuciones independientes (para la versión monoobjetivo) y el promedio de las métricas multiobjetivo calculadas sobre las 30 ejecuciones independientes (para la versión multiobjetivo). Para analizar las distribuciones de resultados se aplicaron tests estadísticos de normalidad y tests no paramétricos.

Comparación contra otros algoritmos. La comparación de los resultados de los AE propuestos contra otros algoritmos de la literatura se enfoca en la calidad de las soluciones logradas. Los resultados calculados se comparan en términos de la función objetivo (similitud total) en el problema monoobjetivo y en términos de la métrica relativa de hipervolumen (véase la Sección V) para la versión multiobjetivo del problema de agrupamiento. En ambos casos, el test de Kolmogorov-Smirnov se aplica a cada conjunto de resultados para evaluar si los valores siguen o no una distribución normal. Posteriormente, se aplica el test no paramétrico de Kruskal-Wallis para comparar las distribuciones de resultados obtenidas por diferentes algoritmos. La prueba de Kruskal-Wallis no requiere normalidad en las muestras para comparar y la prueba de Kolmogorov-Smirnov rechaza la normalidad como hipótesis nula. Para todas las pruebas estadísticas se utilizó un nivel de confianza del 95 %.

V-C. Algoritmos de base para la comparación

Se utilizó una serie de métodos clásicos y específicos de agrupamiento para comparar los resultados calculados por los enfoques evolutivos propuestos. Los algoritmos de base se describen a continuación.

V-C1. k-medoids: es un método particional clásico relacionado con k-means (ambos se basan en particionar el conjunto de puntos a agrupar). Los grupos se construyen con el objetivo de minimizar la distancia entre puntos y el centro del grupo correspondiente, de acuerdo con una métrica de distancia dada. k-medoids es más genérico que k-means, debido a dos características: *i*) en lugar de usar centroides (i.e., el punto medio como el centro de un grupo), k-medoids selecciona puntos dato como centros (*medoids*); y *ii*) k-medoids considera cualquier función de distancia arbitraria en lugar de la norma euclidiana. Estas características permiten a k-medoids proporcionar un esquema de agrupación más robusto, que es menos sensible a la presencia de valores atípicos.

Se han propuesto varios enfoques para implementar el algoritmo k-medoids [26]. Dos de las estrategias más utilizadas son PAM y conjuntos de Voronoi. PAM aplica una búsqueda ávida basada en k puntos iniciales seleccionados al azar para ser medoids. Los grupos iniciales se crean agrupando los puntos datos al medoid más cercano. Posteriormente, la búsqueda ávida realiza un conjunto de intercambios, considerando cada medoid y cada punto dato. Las distancias entre los puntos dato y el medoid correspondiente se reevaluan. En el caso de que la suma de las distancias para todos los puntos dato se reduzca, el intercambio es aceptado; de lo contrario, se descarta. El proceso se repite hasta que no se encuentren cambios en el conjunto de grupos. La implementación basada en conjuntos de Voronoi construye una solución aleatoria inicial (seleccionando k medoids al azar) y realiza una búsqueda iterativa: el punto que minimiza la suma de las distancias dentro de cada grupo se selecciona como nuevo medoid, los grupos se reasignan y las distancias se recalculan. La búsqueda finaliza cuando no se cambia ningún grupo (i.e., la suma de distancias no disminuye).

V-C2. Linkage: Linkage es una técnica de agrupación jerárquica basada en la creación de grupos mediante la combinación de elementos de grupos previamente definidos. El agrupamiento jerárquico se basa en agrupar datos para crear un árbol, que representa una jerarquía multinivel de grupos que se utilizarán para la agregación de datos. Linkage funciona como se describe a continuación. Inicialmente, cada elemento individual se considera un grupo. Iterativamente, el método selecciona los dos grupos que tienen la distancia más corta entre ellos para combinar. La función de distancia evalúa una métrica de similitud relevante para el problema y distintas implementaciones de Linkage utilizan diferentes funciones de distancia: *single linkage* (vecino más cercano) utiliza la menor distancia entre los objetos de los dos grupos; *complete linkage* (vecino más alejado) utiliza la mayor distancia entre los objetos de los dos grupos; *medium linkage* utiliza la distancia media entre todos los pares de objetos en dos grupos; *centroid linkage* utiliza la distancia euclídea entre los centroides de los dos grupos; y *median linkage* utiliza la distancia euclídea entre los centroides ponderados de los dos grupos. En este artículo se utiliza la implementación *single linkage* de Matlab para comparar los resultados de los AE propuestos.

V-C3. Búsqueda local: El algoritmo de búsqueda local fue propuesto por Sheng y Liu [6], combinando k-medoids y una búsqueda exhaustiva realizada para cada grupo. El Algoritmo 4 presenta el pseudocódigo de la búsqueda local. El algoritmo es estocástico, ya que parte de un conjunto de centros seleccionados aleatoriamente. Luego, se determina el conjunto de p vecinos más cercanos para cada centro y se realiza una búsqueda local sobre estos conjuntos para encontrar un nuevo centro que minimice la distancia con todos los elementos. El algoritmo finaliza cuando no se cambia ningún centro en dos iteraciones consecutivas.

Algoritmo 4 Algoritmo de búsqueda local

```
1: sortear k centros
2: seleccionar p vecinos más cercanos para cada centro
3: hacer
4:     asignar elementos a un grupo
5:     actualizar centros
6:     para cada grupo hacer
7:         encontrar un nuevo centro que minimice la distancia
8:         hacia todos los elementos del grupo
9:     fin para
10: mientras centros no cambien
```

V-C4. Algoritmo ávido: El algoritmo ávido construye grupos iterativamente, tomando una decisión localmente óptima en cada paso. Parte de un centro seleccionado al azar y, en cada paso de iteración, busca el elemento con la menor similitud con la solución ya construida. Este elemento se incluye en la solución como un nuevo centro. Todos los grupos se recalculan y el procedimiento se aplica hasta crear k grupos.

V-C5. AE híbrido: Una metaheurística híbrida incluye conocimiento dependiente del problema (híbrido fuerte) o combina varios métodos para resolver el mismo problema (híbrido débil). El objetivo principal es aprovechar las características de cada método para obtener un mejor patrón de búsqueda [4]. El AE híbrido propuesto en este artículo combina una búsqueda evolutiva y una búsqueda local para resolver el problema de agrupación. El algoritmo sigue el esquema genérico presentado en el Algoritmo 5, tal como lo proponen Sheng y Liu [6].

Algoritmo 5 Esquema del AE híbrido para el problema de agrupamiento

```
1: Inicializar k centros aleatoriamente
2: mientras no criterio_parada hacer
3:     [padre1, padre2] = SeleccionTorneo(P)
4:     si rand(0,1) > p_C entonces
5:         [hijo1, hijo2] = Mix Subset (padre1, padre2)
6:     fin si
7:     [hijo1, hijo2] = InversionBit(p_M)
8:     si rand(0,1) > p_LS entonces
9:         [hijo1, hijo2] = BusquedaLocal()
10:    fin si
11: fin mientras
12: retornar mejor solución hallada
```

El AE híbrido utiliza la codificación binaria para representar soluciones, inicialización aleatoria, selección por torneo, el operador de recombinación Mix Subset y la mutación de inversión de bits. El operador de recombinación Mix Subset funciona de la siguiente manera. Dadas dos soluciones padre1 y padre2: *i*) concatena padre1 y padre2 para obtener X_{mix}; *ii*) ordena aleatoriamente los elementos de X_{mix}; *iii*) aplica mutación de inversión de bits (con probabilidad p_{MIX}); y *iv*) obtiene hijo1 e hijo2, dividiendo X_{mix} en dos vectores de igual tamaño.

El AE híbrido utiliza dos criterios de parada: esfuerzo prefijado (la búsqueda se detiene después de un número dado de iteraciones) y detección de una situación de estancamiento (la mejor solución calculada permanece sin cambios por un número dado de generaciones).

V-D. Versión monoobjetivo

El análisis experimental de los algoritmos propuestos para la versión monoobjetivo del problema de agrupamiento se presenta a continuación.

Configuración paramétrica. Los valores de los parámetros de cada AE se definieron en base a resultados de experimentos preliminares y a valores reportados en trabajos relacionados.

Las configuraciones paramétricas corresponden a:

- *AE monoobjetivo*: $pop = 100$, $p_C = 0,75$, $p_M = 0,01$, tamaño del torneo $= 2$ (dos individuos son elegidos al azar y el más apto gana el torneo) y criterio de parada de 10000 generaciones.
- *k-medoids*. El algoritmo se detiene cuando los centros de los grupos no cambian en dos iteraciones consecutivas.
- *Búsqueda local*. El tamaño de la vecindad (p) se estableció en 3 y el criterio de parada es el mismo que para k-medoids, según lo recomendado por Sheng y Liu [6].
- *AE híbrido*. Los parámetros del algoritmo híbrido también se establecieron de acuerdo con las recomendaciones de Sheng y Liu [6]: $pop = 30$, $p_C = 0{,}95$, $p_M = 0{,}02$, $p_{MIX} = 0{,}05$, $p_{LS} = 0{,}2$, tamaño de la vecindad $p = 3$, tamaño del torneo $= 2$ y criterio de parada de 10000 generaciones.

Comparación de operadores evolutivos. Para los AE monoobjetivo se estudiaron las dos representaciones propuestas: binaria y entera.

Para la representación entera, se estudiaron los tres operadores de recombinación y las dos mutaciones propuestas, generando seis posibles combinaciones de operadores evolutivos. Las combinaciones son:

- *SPX-One.* SPX y mutación de un gen;
- *SPX-Adapt.* SPX y mutación de un gen adaptada;
- *SPXGCS-One.* SPX-GenC&S y mutación de un gen;
- *SPXGCS-Adapt.* SPX-GenC&S y mutación de un gen adaptada;
- *GCS-One.* GenC&S y mutación de un gen;
- *GCS-Adapt.* GenC&S y mutación de un gen adaptada.

Cada una de las seis posibles combinaciones de operadores evolutivos fueron evaluadas experimentalmente realizando 30 ejecuciones independientes en cada instancia del problema. Se aplicó el test de Kolmogorov-Smirnov para evaluar la normalidad de las distribuciones de resultados. Usando un nivel de confianza del 95 %, la hipótesis de la normalidad fue rechazada en la mayoría de los casos. Por lo tanto, se utilizó el test no paramétrico de Kruskal-Wallis para comparar los resultados, ya que no asume normalidad de las distribuciones. Los resultados muestran que la combinación *SPX-One* permite calcular los mejores resultados en 7 instancias de problema y *GCS-One* en 5 instancias de problema, superando a las otras combinaciones de operadores evolutivos. Por lo tanto, el resto del análisis experimental del AE monoobjetivo utilizando la codificación de enteros se realizó sobre estas dos combinaciones de operadores evolutivos.

Para la representación binaria, se estudiaron los dos operadores de recombinación y las tres mutaciones propuestas, generando seis posibles combinaciones de operadores evolutivos. Las combinaciones son:

- *SPX-bit.* SPX y mutación de inversión de bit;
- *SPX-add.* SPX y mutación de incorporación de centro;
- *SPX-del.* SPX y mutación de eliminación de centro;
- *2PX-bit.* 2PX y mutación de inversión de bit;
- *2PX-add.* 2PX y mutación de incorporación de centro;
- *2PX-del.* 2PX y mutación de eliminación de centro.

Cada combinación de operadores evolutivos para la codificación binaria fue evaluada sobre el conjunto de instancias realizando 30 ejecuciones independientes. Los resultados del test de Kolmogorov-Smirnov permitieron rechazar con 95 % de confianza la hipótesis nula de que los resultados siguen una distribución normal. Por lo tanto, se aplicó el test de Kruskal-Wallis para comparar los resultados de cada combinación de operadores evolutivos. Los resultados experimentales mostraron que *SPX-del* funcionó mejor en instancias pequeñas, superando al resto de las combinaciones. Sin embargo, en las instancias medianas #5 y #6, *SPX-bit* tuvo los mejores resultados, mientras que en las instancias grandes #11 y #12 *2PX-del* obtuvo los mejores resultados. Por lo tanto, el resto del análisis experimental del AE monoobjetivo utilizando la codificación binaria se centró en estas tres combinaciones de operadores evolutivos.

Comparación de representaciones. La Tabla I reporta los resultados promedio calculados en 30 ejecuciones independientes de el AE utilizando codificación binaria y entera, considerando sólo los operadores evolutivos que lograron los mejores resultados en el análisis experimental anterior.

Tabla I
SIMILITUD MEDIA UTILIZANDO DIFERENTES CODIFICACIONES Y OPERADORES EVOLUTIVOS (PROBLEMA MONOOBJETIVO).

#I	codificación entera		codificación binaria		
	SPX-One	*GCS-One*	*SPX-bit*	*SPX-del*	*2PX-del*
#1	**18.66**	**18.66**	**18.66**	**18.66**	**18.66**
#2	**1.96**	**1.96**	**1.96**	**1.96**	**1.96**
#3	12.42	12.44	12.27	**12.46**	**12.46**
#4	16.43	16.41	15.93	**16.50**	**16.50**
#5	78.35	78.16	**78.61**	78.51	78.42
#6	116.18	116.39	**116.45**	115.69	115.34
#7	54.71	54.68	54.80	**54.98**	**54.98**
#8	63.27	63.30	61.10	63.42	**63.43**
#9	673.57	656.56	633.91	**675.20**	**675.20**
#10	37.77	36.49	35.88	**38.22**	**38.22**
#11	235.33	229.58	221.17	**236.11**	**236.11**
#12	32.89	32.08	31.20	**33.23**	**33.23**

Las combinaciones de operadores evolutivos que hallaron los mejores resultados fueron *SPX-del* y *2PX-del*, usando codificación binaria. Si bien ambas combinaciones superan significativamente los resultados calculados utilizando la codificación entera y utilizando la configuración *SPX-bit* con codificación binaria, no existe una diferencia significativa en los resultados entre ambas configuraciones. Por motivos de simplicidad del AE, el resto de la evaluación experimental se realizó utilizando los operadores evolutivos *SPX-del*.

Comparación contra otros algoritmos. El AE con codificación binaria, SPX y mutación de eliminación de centros se comparó con los métodos de agrupamiento de la literatura relevados en la Sección V-C. La Tabla II reporta la similitud promedio calculada en 30 ejecuciones independientes de cada algoritmo en las 12 instancias del problema. Los mejores resultados promedio están marcados en negrita. Se aplicó el test de Kolmogorov-Smirnov para estudiar las distribuciones de resultados. En la mayoría de los casos, la prueba permitió rechazar (con 95 % de confianza) la hipótesis nula de que los resultados siguen una distribución normal. Por lo tanto, se aplicó el test de Kruskal-Wallis para comparar las distribuciones de resultados de cada algoritmo. El p-valor correspondiente al test de Kruskal-Wallis se reporta en la última columna de la Tabla II. Los resultados de test de Kruskal-Wallis permiten rechazar la hipótesis nula de que los resultados calculados por los diferentes algoritmos siguen la misma distribución.

El AE propuesto superó a todos los demás algoritmos, calculando los mejores resultados promedio en 10 de 12 instancias. El AE híbrido fue el segundo mejor algoritmo, calculando los mejores resultados promedio en 5 de las 12 instancias. Las mejoras sobre k-medoids fueron de hasta 9.5 % en el mejor de los casos. Se logró mejorar hasta 156.2 % sobre el algoritmo ávido, para una de las instancias pequeñas. Al comparar con el algoritmo Linkage, el AE propuesto mejora las soluciones calculadas hasta en 29.1 %. La mejora sobre el algoritmo de búsqueda local fue de 31.9 %. Finalmente, las mejoras sobre el AE híbrido son más pequeñas. En el mejor de los casos (instancia #10), el AE propuesto superó al híbrido en 4.0 % (2.3 % en promedio).

V-E. Versión multiobjetivo

Esta sección describe el análisis experimental correspondiente a los algoritmos para la versión multiobjetivo del problema de agrupamiento.

Operadores evolutivos y configuración paramétrica. Los parámetros del MOEA propuesto fueron configurados mediante experimentos preliminares y corresponden a:

- Probabilidad de recombinación $p_C = 0{,}75$
- Probabilidad de mutación $p_M = 0{,}01$
- Selección por torneo (tamaño del torneo = 2)
- Criterio de parada: 1000 generaciones
- Tamaño de la población $pop = 100$

Resultados numéricos. Se utilizaron los mejores algoritmos para la versión monoobjetivo del problema de agrupamiento (el AE propuesto y k-medoids) para comparar los resultados del MOEA propuesto. Para ello se realizaron 30 ejecuciones independientes de cada algoritmo, variando la cantidad de grupos a formar en el caso de los algoritmos monoobjetivo.

Las Figuras 1, 2, y 3 muestran los frentes de Pareto alcanzados al combinar las mejores soluciones alcanzadas en las 30 ejecuciones independientes del MOEA propuesto. Adicionalmente, para cada solución no dominada encontrada por el MOEA, se realizaron 30 ejecuciones independientes del AE monoobjetivo propuesto y del algoritmo k-medoids. Las mejores soluciones encontradas se presentan en las Figuras 1 a 3, que se corresponden con las instancias #4, #6 y #12 del problema, respectivamente. Estos resultados son representativos del conjunto de instancias del problema resuelto.

Los resultados muestran que al utilizar pocos grupos no existen diferencias significativas en los resultados alcanzados por los distintos algoritmos. Sin embargo, a medida que la cantidad de grupos aumenta, el AE monoobjetivo propuesto

Tabla II
COMPARATIVA DE SIMILITUD PROMEDIO DEL AE PROPUESTO CONTRA ALGORITMOS DE LA LITERATURA.

instancia	*ávido*	*linkage*	*k-medoids*	*búsqueda local*	*AE híbrido*	*AE SPX-del*	*p-valor K-W*
#1	7.28	17.01	17.03	15.49	**18.66**	**18.66**	$< 10^{-15}$
#2	1.12	1.65	1.95	1.70	**1.96**	**1.96**	$< 10^{-15}$
#3	5.77	10.18	12.14	10.50	12.45	**12.46**	$< 10^{-15}$
#4	7.41	14.04	16.00	13.23	16.22	**16.50**	$< 10^{-15}$
#5	47.69	76.08	76.47	69.11	**78.62**	78.51	$< 10^{-15}$
#6	83.61	109.68	116.30	108.86	**116.45**	115.69	$< 10^{-15}$
#7	29.31	50.77	54.98	41.68	**54.98**	**54.98**	$< 10^{-15}$
#8	31.81	62.25	62.51	52.99	63.24	**63.42**	$< 10^{-15}$
#9	499.54	523.19	667.94	615.64	661.48	**675.20**	$< 10^{-15}$
#10	22.90	30.61	37.09	32.94	36.73	**38.22**	$< 10^{-15}$
#11	170.65	198.75	236.10	205.96	229.56	**236.11**	$< 10^{-15}$
#12	22.80	27.02	32.85	28.56	33.10	**33.23**	$< 10^{-15}$

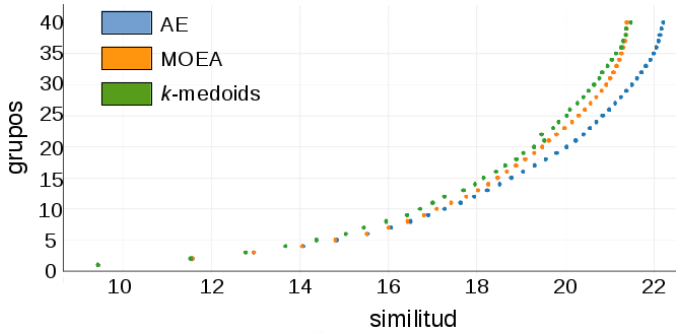

Figura 1. Frentes de Pareto para la instancia #4

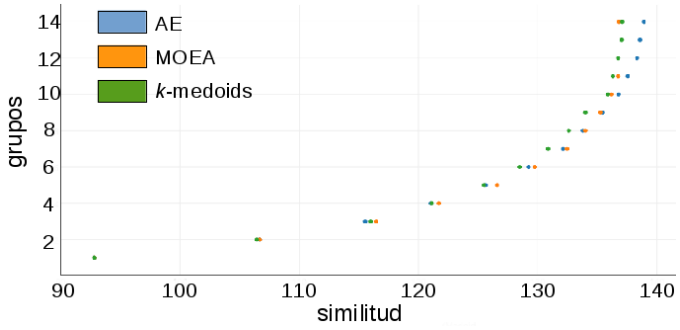

Figura 2. Frentes de Pareto para la instancia #6

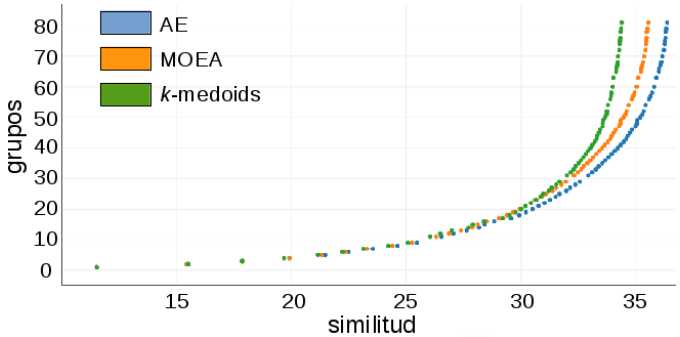

Figura 3. Frentes de Pareto para la instancia #12

supera los resultados alcanzados por el MOEA y por k-medoids. Es importante destacar que el MOEA propuesto es capaz de obtener un frente de Pareto con soluciones con diferentes niveles de compromiso entre las funciones objetivos en una única ejecución, mientras que son necesarias muchas ejecuciones (variando el número de grupos) al utilizar el AE monoobjetivo. Por lo tanto, el MOEA propuesto es igualmente útil para que un tomador de decisiones pueda visualizar agrupamientos con diferentes niveles de compromiso entre las funciones objetivo y pueda seleccionar aquél que mejor capture las características del problema.

Para evaluar los resultados desde el punto de vista de la optimización multiobjetivo, se utilizó la métrica de hipervolumen relativo (Relative HyperVolume, RHV) [11]. RHV es el cociente entre los volúmenes (en el espacio de las funciones objetivo) cubiertos por el frente de Pareto calculado y por el frente de Pareto real. El valor ideal para el RHV es 1. En nuestro análisis, el frente de Pareto real es desconocido para las instancias del problema estudiadas, por lo que se aproxima por el conjunto de todas las soluciones no dominadas encontradas en el total de ejecuciones independientes.

Los resultados del RHV sobre las 30 ejecuciones independientes del MOEA propuesto indican que el mismo es robusto y es capaz de calcular con precisión frentes de Pareto para las instancias estudiadas. La diferencia máxima con respecto al valor ideal de RHV es 0,02 (instancias #6 y #12), con un RHV promedio de 0.99, y un valor óptimo de 1.00 en tres instancias del problema.

Con respecto a las instancias del problema correspondientes al mapa de la enfermedad de Parkinson, los AE propuestos permiten calcular configuraciones apropiadas con diferentes valores de compromiso entre las funciones objetivo del problema. Las soluciones calculadas por los enfoques evolutivos ofrecen información nueva y promisoria en comparación con las soluciones generadas manualmente en el marco del proyecto (ver la web del proyecto, http://wwwen.uni.lu/lcsb/research/parkinson_s_disease_map).

En resumen, considerando el conjunto total de instancias del problema, el AE monoobjetivo es capaz de mejorar al algoritmo k-medoids en hasta un 31.42 % en el mejor caso.

La mejora sobre el algoritmo multiobjetivo es de 8.76 % en el mejor caso.

VI. Conclusiones y trabajo futuro

Este articulo presentó algoritmos evolutivos aplicados al problema de agrupamiento en sus versiones monoobjetivo y multiobjetivo, con un número desconocido de grupos. Este problema es de relevancia en muchas áreas de investigación que involucran grandes volúmenes de información a ser categorizada y agrupada.

Los algoritmos evolutivos propuestos fueron diseñados para aplicar operadores simples y específicos al problema, con el fin de mantener la búsqueda lo más sencilla posible. De esta forma, se buscó que el algoritmo escale adecuadamente y sea capaz de resolver grandes instancias del problema de agrupamiento.

La evaluación experimental se realizó sobre un conjunto de instancias reales del problema de agrupamiento, incluyendo una instancia real correspondiente a información biomédica en el contexto de un proyecto para generar un mapa de la enfermedad de Parkinson.

Los principales resultados indican que los algoritmos evolutivos propuestos logran calcular soluciones precisas para las instancias del problema estudiadas. Los enfoques evolutivos mejoran los resultados alcanzados por diversos algoritmos de la literatura relacionada. En la versión monoooobjetivo del problema de agrupamiento, el algoritmo evolutivo propuesto permite calcular el mejor resultado promedio en 10 de 12 instancias de prueba. Para la versión multiobjetivo del problema de agrupamiento, el algoritmo evolutivo propuesto logra calcular frentes de Pareto adecuados, que ofrecen a los tomadores de decisión diferentes niveles de compromiso entre las funciones objetivo del problema.

El enfoque evolutivo es especialmente útil para organizar información biomédica en el contexto del proyecto para generar un mapa de la enfermedad de Parkinson. Los AE propuestos permiten encontrar agrupamientos de los datos que proveen diferentes niveles de compromiso entre los objetivos en conflicto y permiten capturar diferentes características de la información agrupada. Los agrupamientos hallados ofrecen soluciones promisorias en comparación a las existentes que fueron construidas manualmente por expertos.

Las principales líneas de trabajo futuro incluyen extender el análisis experimental considerando otros conjuntos de datos provenientes de otras áreas de investigación. Adicionalmente, un modelo paralelo de AE debería ser estudiado para reducir los tiempos de ejecución y manejar conjuntos de datos de mayores dimensiones. Finalmente, la posibilidad de combinar los resultados calculados por los algoritmos evolutivos con herramientas de visualización debe ser explorada, con el fin de ayudar a los investigadores a analizar la información de una forma amigable e intuitiva.

Referencias

[1] L. Kaufman and P. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, ser. Wiley Series in Probability and Mathematical Statistics. New York: Wiley, 1990.

[2] W. Welch, "Algorithmic complexity: threeNP- hard problems in computational statistics," *Journal of Statistical Computation and Simulation*, vol. 15, no. 1, pp. 17–25, 1982.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer New York, 2009.

[4] S. Nesmachnow, "An overview of metaheuristics: accurate and efficient methods for optimisation," *International Journal of Metaheuristics*, vol. 3, no. 4, pp. 320–347, 2014.

[5] E. Hruschka, R. Campello, A. Freitas, and A. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 2, pp. 133–155, 2009.

[6] W. Sheng and X. Liu, "A hybrid algorithm for k-medoid clustering of large data sets," in *IEEE Congress on Evolutionary Computation*, 2004, pp. 77–82.

[7] University of Luxembourg, "Parkinson's disease map project," [Online], http://wwwen.uni.lu/lcsb/research/parkinson_s_disease_map, accessed March 2017.

[8] K. Fujita et al., "Integrating pathways of Parkinson's disease in a molecular interaction map," *Molecular Neurobiology*, vol. 49, no. 1, pp. 88–102, 2014.

[9] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

[10] C. Coello, D. Van Veldhuizen, and G. Lamont, *Evolutionary algorithms for solving multi-objective problems*. Kluwer, 2002.

[11] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[13] J. Karimov and M. Ozbayoglu, "High quality clustering of big data and solving empty-clustering problem with an evolutionary hybrid algorithm," in *IEEE International Conference on Big Data*, 2015, pp. 1473–1478.

[14] H. Park and C. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, 2009.

[15] S. Das, A. Abraham, and A. Konar, *Metaheuristic Clustering*, ser. Studies in Computational Intelligence. Springer, 2009, vol. 178.

[16] Y. Deng and J. Bard, "A reactive GRASP with path relinking for capacitated clustering," *Journal of Heuristics*, vol. 17, no. 2, pp. 119–152, 2011.

[17] M. Cowgill, R. Harvey, and L. Watson, "A genetic algorithm approach to cluster analysis," *Computers & Mathematics with Applications*, vol. 37, no. 7, pp. 99–108, 1999.

[18] U. Maulik, S. Bandyopadhyay, and A. Mukhopadhyay, *Multiobjective Genetic Algorithms for Clustering*. Springer Nature, 2011.

[19] K. Ripon, C.-H. Tsang, S. Kwong, and M.-K. Ip, "Multi-objective evolutionary clustering using variable-length real jumping genes genetic algorithm," in *18^{th} International Conference on Pattern Recognition*, 2006, pp. 3609–3616.

[20] E. Korkmaz, J. Du, R. Alhajj, and K. Barker, "Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering," *Intelligent Data Analysis*, vol. 10, no. 2, pp. 163–182, 2006.

[21] S. Bandyopadhyay, U. Maulik, and A. Mukhopadhyay, "Multiobjective genetic clustering for pixel classification in remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1506–1511, 2007.

[22] A. Banerjee, "Multi-objective genetic algorithm for robust clustering with unknown number of clusters," *International Journal of Applied Evolutionary Computation*, vol. 3, no. 1, pp. 1–20, 2012.

[23] D. Deaven and K. Ho, "Molecular geometry optimization with a genetic algorithm," *Physical Review Letters*, vol. 75, pp. 288–291, 1995.

[24] J. Alcalá et al., "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255–287, 2010.

[25] Sean Luke et al., "ECJ 23: a Java-based Evolutionary Computation Research System," [Online], https://cs.gmu.edu/ eclab/projects/ecj, accessed March 2017.

[26] X. Jin and J. Han, "K-medoids clustering," in *Encyclopedia of Machine Learning*. Springer US, 2010, pp. 564–565.