Silly Season Algorithm. Diseño e implementación de una metaheurística

Antonio José Blánquez Pérez

Práctica final Metaheurísticas - Universidad de Granada

1 Proposición

1.1 Motivación

El concepto de esta metaheurística nace cuando, en medio de la cuarentena, la temporada 2020 de Formula 1 se está desarrollando de manera anómala. Esta comenzará en julio y no en marzo como es habitual[1], provocando que la temporada dure hasta noviembre ininterrumpidamente también durante el mes de agosto(lo que nos dejará ver un Gran Premio de España el 16 de agosto, que se correrá a una temperatura poco convencional), se cancela el Gran Premio de Mónaco por primera vez desde 1955 y, lo que nos ocupa en este caso, debido al cambio en el calendario y a que en este deporte no existe un mercado de fichajes con fechas establecidas como, por ejemplo, en el fútbol, los equipos están dedicando su tiempo a cerrar fichajes para 2021, año en el que la mayoría de pilotos acaba contrato al ser fecha de cambio en el reglamento(aunque se ha retrasado a 2022). El término *Silly Season*, que da nombre a la metaheurística, se acuñó para definir el tiempo, generalmente los meses de verano, en el que a falta de noticias los medios de comunicación comienzan a hablar sobre rumores y, en el contexto del deporte, rumores sobre fichajes; aunque este término es ampliamente usado para referirse al período de fichajes que suele darse en los parones de verano e invierno. Aunque en el momento en el que se escriben estas líneas aún no es verano y deberíamos de estar a la espera del GP de Francia, la realidad es que esta temporada aún no ha comenzado y se ha desarrollado una Silly Season como no se recuerda en años[2].

[1] https://www.formula1.com/en/latest/article.f1-schedule-2020-latest-information.3P0b3hJYdFDm9xFieAYqCS.html

[2] https://www.motorpasion.com/formula1/cascada-fichajes-formula-1-daniel-ricciardo-confirmado-como-sustituto-carlos-sainz-mclaren

De esta situación surgió la idea de plantear una metaheurística basada en la ya comentada Silly Season, donde los pilotos serán individuos soluciones al problema que se organizarán en equipos o conjuntos que determinarán el trato que se le dará a cada individuo.

1.2 Resumen

El funcionamiento de esta metaheurística se basa en unos elementos bien definidos, las soluciones, que serán nuestros pilotos, se generarán y organizarán en equipos inicialmente de manera aleatoria, y se les asignará una puntuación a través de la función objetivo. Los equipos cuyos pilotos tengan mejor puntuación serán los que estén en la zona alta de la tabla y viceversa. Como los equipos de la parte alta tienen más dinero no pueden arriesgarse a cambios relevantes en su organización, por lo que generalmente se centrarán en mejorar al máximo el rendimiento de su coche(y por tanto el de su piloto). Por otra parte los equipos de la parte, puesto que no tienen nada que perder, tienden a realizar grandes cambios tanto en su organización como en su plantilla. Es decir, en los equipos con mayor puntuación los pilotos evolucionarán mediante explotación y los que tengan una puntuación más baja lo harán mediante exploración. Además, tras un cierto período(la temporada o época) los pilotos cambiarán de equipo según su rendimiento, aunque también habrá una posibilidad de que lo hagan entre temporadas si su rendimiento está desproporcionado con el de su equipo.

Este concepto se puede plantear como una buena metaheurística en la medida en la que se consigue equilibrar la exploración y la explotación, equilibrio que podría proporcionar una buena solución. Esto es porque intentaremos conseguir que las soluciones sean buenas mediante explotación en el caso de las que sean ya buenas en los que hemos definido como los mejores equipos y evitaremos óptimos locales mediante la exploración en el entorno de los peores equipos, ya que con ellos terminaremos consiguiendo soluciones potencialmente buenas.

2 Descripción detallada

2.1 Concepto

2.2 Implementación

3 Aplicación

3.1 Descripción del problema

El agrupamiento o clustering busca clasificar objetos de un conjunto en subconjuntos o clusters a través de sus posibles similaridades. Es una técnica de aprendizaje no supervisado que permite descubrir grupos inicialmente desconocidos o agrupar objetos similares, de manera que podemos encontrar patrones en grandes grupos de datos que serían imposibles(o extremadamente difíciles) de encontrar sin esta técnica.

Un ejemplo cotidiano del problema del agrupamiento sería dividir una cantidad de frutas en verdes, poco maduras y muy maduras que, aunque trivial, nos permite entender el concepto que hay detrás de la técnica que nos ocupa. Con ésto podemos comprender otras aplicaciones del clustering, como pueden ser el análisis de caracteres escritos a mano, muestras de diálogo, huellas dactilares o imágenes, la clasificación de especies en subespecies o el agrupamiento para moléculas o proteínas. Para poder aplicar esta técnica debemos medir d características de nuestro grupo de n objetos, por ejemplo el color o la textura en el caso de las frutas o la simetría o la intensidad en el caso de caracteres escritos a mano, obteniendo una conjunto de datos de longitud n con d dimensiones.

En nuestro caso concreto abordaremos una variación del clustering clásico, el Problema de Agrupamiento con Restricciones(PAR), es decir, además de nuestro conjunto de datos tenemos cierta información a la que llamaremos restricciones o, más concretamente, restricciones de instancia. Ésto quiere decir que tenemos alguna información sobre objetos que tienen que pertenecer al mismo cluster(ML, Must-Link) y sobre objetos que no pertenecen al mismo cluster(CL, Cannot-Link), siendo éstas restricciones débiles, o lo que es lo mismo, podemos incumplir restricciones pero debemos minimizar el número de incumplidas al máximo. El planteamiento combinatorio de este problema es NP-Completo, por tanto a continuación propondremos algunas alternativas de algoritmos para intentar resolver el problema de forma aproximada en un tiempo razonable. El problema se abordará con k(nº de clusters) conocida, ya que su búsqueda es otro problema complejo.

3.2 Formalización de conceptos del problema

Vamos a definir distintos conceptos que se nombrarán a lo largo del documento y que son necesarios para entender los procedimientos en cada algoritmo. Primeramente podemos formalizar la definición de nuestro conjunto de datos como una matriz X de $n \cdot d$, n objetos en un espacio de d dimensiones(el número de medidas que tenemos sobre cada objeto).

$$\vec{x_i} = \{x_{i,1}, ..., x_{i,d}\} \parallel \ \in \mathbb{R} \ \forall j \in \{1, ..., d\}$$

Llamaremos $C = \{c_1, ..., c_k\}$ al conjunto de los k clusters, de manera que cada c_i será un subconjunto de X y podrá asociada una etiqueta l_i que lo nombre Para cada cluster es posible calcular su centroide asociado $\vec{\mu_i}$, siendo el vector promedio de sus instancias.

$$\vec{\mu_i} = \frac{1}{|c_i|} \sum_{\vec{\mu_j} \in c_i} \vec{x_j}$$

También debemos definir la distancia media intra-cluster, c_i , como la media de las distancias entre cada instancia del cluster y su centroide asociado, usando en este caso la distancia euclídea, aunque es equivalente a usar, por ejemplo, la distancia Manhattan.

$$\overline{c_i} = \frac{1}{|c_i|} \sum_{\vec{\mu_j} \in c_i} ||\vec{x_j} - \vec{\mu_i}||$$

A través de estas, podemos llegar a la desviación general de C, la media de las desviaciones intracluster, que nos será de gran ayuda para minimizar la solución y para su posterior análisis.

$$\overline{C} = \frac{1}{k} \sum_{c_i \in C} \overline{c_i}$$

Por otro lado tenemos las restricciones, $R = ML \cup CL$, donde $ML(\vec{x_i}, \vec{x_j})$ indica que las instancias $\vec{x_i}$ y $\vec{x_j}$ deben estar asignadas al mismo cluster y $CL(\vec{x_i}, \vec{x_j})$ que las instancias $\vec{x_i}$ y $\vec{x_j}$ no pueden ser asignadas al mismo cluster. Con ello, definimos la infactibilidad o infeasibility, que es un indicador de las restricciones que incumple nuestra solución.

$$infeasibility = \sum_{i=0}^{|ML|} 1(h_c(\vec{ML}_{[i,1]}) \neq h_c(\vec{ML}_{[i,2]})) + \sum_{i=0}^{|CL|} 1(h_c(\vec{CL}_{[i,1]}) = h_c(\vec{CL}_{[i,2]}))$$

Siendo 1 la función booleana que devolverá 1 si la expresión que toma como argumento es verdadera y 0 en otro caso.

3.3 Representación de la información

Ahora que tenemos algunos conceptos claros y formalizados, pasamos a ver como representaremos la información necesaria en el proceso de resolución del problema. Primero aclarar que los datos de entrada, nuestro conjunto de datos, estará organizado en una matriz implementada como un vector<vector<float>> (ambos vectores de la librería STL), es decir, un vector de longitud n de vectores de reales en coma flotante de longitud d. Para las restricciones usaremos dos formas de representación, una para facilitar el acceso cuando conocemos la restricción que queremos comprobar y otra para cuando queramos recorrer todas las restricciones. Para el primer caso se usará la misma representación que para el dataframe, mientras que para el segundo se construirá una lista con elementos del tipo $[x, y, 1, -1]$, siendo x e y los dos elementos que tiene la restricción y el último elemento 1 si conforman una restricción ML y -1 si es CL. Esta lista se ha implementado usando un vector<vector<int>>, también de la STL. De ésta manera, al recorrer esa lista evadiremos todos los pares que no tienen una restricción y nuestra búsqueda será mucho más eficiente que en una matriz. Por último, representaremos la solución con un vector<int> de longitud n, de manera que la posición i contendrá el número del cluster asignado al objeto i.

3.4 Función objetivo

La función objetivo es la función que debemos minimizar, es decir, la función que dice como de buena es la solución que le pasamos como parámetro. De todas las definiciones que hemos hecho, desviación general($\overline{C}$) e infactibilidad(infeasibility) nos dan información acerca de la bondad de la solución, la primera a través de la distancia promedio de cada dato a su centroide correspondiente y la segunda a través de la medida del incumplimiento de las restricciones de la solución dada. Ambas han de ser minimizadas, pero hay que darle más o menos importancia a una y a otra, de manera que debemos introducir un parámetro con el que controlar ésta importancia que debatiremos posteriormente.

$$f = \overline{C} + infeasibility \cdot \lambda$$

Dicho ésto queda definir el parámetro λ, el cual se ha decidido proponer como el entero superior a la distancia máxima D dividida por el número de restricciones totales R.

$$\lambda = \frac{\lceil D \rceil}{|R|}$$

A continuación se expone la descripción en pseudocódigo de todos los operadores y de la función objetivo.

3.5 Conjuntos de datos usados

Los conjuntos de datos usados para testear la bondad de los algoritmos son los siguientes:

- Iris: Contiene información sobre las características de tres tipos de flores de Iris. Tiene 3 clases(k=3) y 6 dimensiones.
- Ecoli: Contiene medidas sobre las ciertas características de diferentes tipos de células que pueden ser empleadas para predecir la localización de ciertas proteínas. Tiene 8 clases(k=8) y 7 dimesiones.
- Rand: Conjunto de datos artificial formado por tres agrupamientos bien diferenciados generados en base a distribuciones normales. Tiene 3 clases(k=3) y 2 dimensiones.
- Newthyroid: Contiene medidas cuantitativas tomadas sobre la glándula tiroides de 215 pacientes. Tiene 3 clases(k=3) y 5 dimensiones.

4 Manual de usuario

Compilación: Se incluye un makefile, por lo que se compila tan solo ejecutando la orden 'make' desde la terminal en la carpeta que contiene el código.

Ejecución: Para ejecutar el algoritmo sobre un conjunto de datos concreto se hará mediante la siguiente orden en la terminal.

./silly_season <fichero_datos> <fichero_restricciones> < k > <seed>

También es posible llamarlo sin argumentos, entonces pedirá por teclado la información necesaria, o usar el script incluído, que replicará todos el experimento realizado para el análisis de rendimiento.

5 Análisis de rendimiento

5.1 Descripción de los casos del problema

5.2 Resultados obtenidos

5.3 Análisis de resultados

5.4 Posibles mejoras