Práctica 3: Metaheurísticas basadas en poblaciones

Convocatoria de junio 2014 (curso académico 2013/2014)

Asignatura: Metaheurísticas
3º Grado Ingeniería Informática (Universidad de Córdoba)

2 de abril de 2014

Resumen

Esta práctica sirve para familiarizar al alumno con las metaheurísticas basadas en poblaciones, en concreto, con los algoritmos genéticos y con los operadores asociados a los mismos. Para ello, se introducirán dos variantes de algoritmos genéticos (algoritmo genético generacional y algoritmo genético de estado estacionario) y se aplicarán al problema del *capacited p-hub* (CPH) considerado en la Práctica 1. La entrega se hará utilizando la tarea en Moodle habilitada al efecto. Se deberá subir en un único fichero comprimido todos los entregables indicados en este guión. El día tope para la entrega es el **4 de mayo**. En caso de que dos alumnos entreguen prácticas copiadas, no se puntuarán ninguna de las dos.

1. Introducción

El trabajo a realizar en la práctica consiste en aplicar dos variantes de algoritmos genéticos (Algoritmo Genético generacional, AGg, y Algoritmo Genético de estado estacionario, AGe) al problema del *capacited p-hub* (CPH), considerado en la Práctica 1.

Para ello, se desarrollará un programa capaz de resolver dicho problema aplicando las variantes AGg y AGe y tomando como base la Práctica 1. Este programa se utilizará para el conjunto de instancias disponible en Moodle y se realizará un estudio comparativo del comportamiento de los algoritmos según los resultados obtenidos. Este análisis influirá en gran medida en la calificación de la práctica.

En el enunciado de esta práctica, se proporcionan valores orientativos para todos los parámetros de los algoritmos. Sin embargo, se valorará positivamente si el alumno encuentra otros valores para estos parámetros que le ayuden a mejorar la calidad de las soluciones obtenidas. La única condición es que no se puede modificar el número máximo de evaluaciones (establecido en todo caso a $3500 \cdot n$, donde n es el número de nodos de la instancia del CPH) y que los valores de los parámetros deben ser constantes para todos los problemas/instancias o, en todo caso, depender del citado tamaño. Estos parámetros si que pueden ser distintos para AGg y AGe.

La sección 2 describe como aplicar cada uno de los algoritmos al CPH. La sección 3 describe como proceder para la comparación de los algoritmos. Finalmente, la sección 4 especifica los ficheros a entregar para esta práctica.

2. Consideraciones sobre ambos AGs

Ambas variantes de AG utilizarán codificación entera de las soluciones. Las características comunes que definen como se aplican ambos AGs son las siguientes:

- *Esquema de representación de las soluciones*: como ya se ha comentado, se seguirá la representación en forma de vector de valores enteros $\mathbf{x}$ de tamaño n que representa, al mismo tiempo, el rol de cada uno de los nodos dentro de la red y el esquema de conexión. De esta forma, el valor x_i de $\mathbf{x}$ indica el rol y/o conexión del nodo i-ésimo del siguiente modo:

- Si $x_i < p$: el nodo i-ésimo es un cliente y el valor x_i representa el número de concentrador al que está conectado.
- Si $x_i \geq p$: el nodo i-ésimo es un concentrador y el valor x_i representa el número de concentrador que se le asigna. En concreto, el número de concentrador será igual $x_i - p$ y cualquier cliente que tenga ese número estará conectado a él. Además, todo concentrador está conectado a si mismo.

- *Función objetivo*: se debe **minimizar** la suma total de distancias desde los nodos clientes a los concentradores que tienen asignados, teniendo en cuenta que la capacidad máxima de un concentrador (c) no puede ser superada (un concentrador nunca podrá suministrar mayor carga que c). La instancia del problema también incluye la demanda generada por cada cliente f_i, tal y como se estudió en la Práctica 1. Hay que tener en cuenta que los concentradores también generan una demanda, que siempre será atendida por ellos mismos (por ser el nodo más cercano).

- *Tratamiento de las soluciones no factibles*: cualquier solución en la que se supere la capacidad de un concentrador será considerada como **solución no factible**. La forma de tratar dichas soluciones será asignarle un valor de aptitud muy bajo (como estamos en un problema de minimización, podemos asignar un valor objetivo igual al valor máximo de un `float`).

- *Generación de la población inicial*: cada individuo de la población inicial será una solución aleatoria. Cada solución se generará del mismo modo que se indicó en la Práctica 1 (escoger aleatoriamente qué nodos serán concentradores y conectar cada cliente a un concentrador). Para favorecer la obtención de soluciones factibles, a la hora de decidir el concentrador de cada cliente, elegiremos como concentrador aquel que tenga su capacidad hasta el momento al mínimo.

- *Tamaño de la población*: el número de individuos de la población será igual a $T = 50$.

- *Condición de parada*: la condición de parada del algoritmo será que hayan producido un total de evaluaciones igual a $3500 \cdot n$, donde n es el tamaño de la instancia (es decir, el número de nodos de la red).

- *Operador de selección*: en cada generación del AG, se seleccionará un conjunto de individuos que actuarán como *padres* a los que aplicar los distintos operadores. Está selección se realizará utilizando **selección por ruleta**, considerando una probabilidad de escoger a un individuo de la generación anterior proporcional a su valor de aptitud. Si el algoritmo es AGg, se seleccionará $T - 1$ individuos y, si es AGe, se seleccionarán 2 individuos (ver subsecciones 2.1 y 2.2). Como estamos en un problema de minimización, la ruleta deberá construirse con el inverso del valor objetivo ($1/f_i$).

- *Operador de cruce*: por cada individuo a cruzar de la población de *padres* generada en el paso anterior, se intentará aplicar el operador de cruce con una probabilidad de aplicación $p_c = 0{,}9$. Es decir, tendremos que obtener un número aleatorio entre 0 y 1, $r = U(0, 1)$, por cada padre y, si $r \leq p_c$, aplicaremos el operador de cruce. En caso contrario, el individuo no se cruzará. Se seleccionará el segundo padre aleatoriamente de entre los restantes. Por cada posición del genotipo que sea cliente en ambos padres, y con una probabilidad de $0{,}5$, intercambiaremos el contenido de dicha posición, es decir, intercambiaremos el concentrador al que se conectan, tal y como se ha introducido en las diapositivas de clase. Los individuos modificados deberán ser marcados para evaluar, pero no los evaluaremos hasta después de que apliquemos el operador de mutación.

- *Operador de mutación*: por cada individuo de la población intermedia generada tras el cruce, se intentará aplicar el operador de mutación con una probabilidad de aplicación $p_m = 0{,}15$. Es decir, tendremos que obtener un número aleatorio entre 0 y 1, $r = U(0, 1)$, por cada individuo y, si $r \leq p_m$, aplicaremos el operador de mutación. En caso contrario, el

individuo no se mutará. La mutación consistirá en intercambiar dos posiciones escogidas aleatoriamente del genotipo de la solución, tal y como se ha introducido en las diapositivas de clase. Los individuos modificados deberán ser marcados para evaluar.

- *Evaluación de los descendientes*: los individuos generados a partir de la mutación (*descendientes*) deberán ser evaluados, siempre y cuando hayan sido modificados en el cruce o en la mutación. Cada vez que se produzca un reinicio o al comenzar la evolución, también deberemos evaluar la población completa.
- *Elitismo*: en cualquier caso, el mejor individuo de la generación anterior siempre deberá pasar a la siguiente generación.
- *Reinicio de la población*: si durante 50 generaciones seguidas (para el AGg) y 500 generaciones seguidas (para el AGe), el mejor individuo de la población no ha mejorado, habrá que provocar un reinicio de la población para mejorar la diversidad de la misma y evitar la convergencia prematura. Este reinicio consistirá en sustituir la población actual por una población de soluciones generadas aleatoriamente, manteniendo siempre el mejor individuo de la población.

2.1. Algoritmo Genético generacional (AGg)

En esta variante de algoritmo, se debe tener en cuenta lo siguiente:

- Durante cada generación, los operadores de cruce y mutación se aplican a un conjunto de tamaño similar al de la población. Por tanto, la selección de padres para aplicar los operadores (que, como ya se ha comentado, será una **selección por ruleta**) seleccionará un total de $T - 1$ individuos a partir de la población actual.
- Aplicaremos los operadores de cruce y mutación con su probabilidad asociada a cada uno de los individuos. Se intentará aplicar $T - 1$ veces el cruce (a cada uno de los individuos seleccionados) y $T - 1$ veces la mutación (a cada uno de los resultantes de los cruces).
- La fase de **reemplazo**, sustituirá la población actual por los descendientes generados mediante los operadores de cruce y mutación, añadiendo siempre el mejor individuo de la generación anterior (elitismo). Será un reemplazo directo por sustitución, sin selección.
- Los reinicios se producirán cuando transcurran 50 generaciones sin mejora.

2.2. Algoritmo Genético de estado estacionario (AGe)

En esta variante de algoritmo, se debe tener en cuenta lo siguiente:

- Durante cada generación, los operadores de cruce y mutación se aplican a dos individuos de la población (lo mínimo para poder generar descendencia). Por tanto, la selección de padres para aplicar los operadores (que, como ya se ha comentado, será una **selección por ruleta**) seleccionará un total de 2 individuos a partir de la población actual.
- Aplicaremos los operadores de cruce y mutación con su probabilidad asociada a los dos individuos. Primero se intentará el cruce (una sola vez) sobre los dos individuos escogidos y posteriormente se intentará aplicar la mutación (dos veces) a los individuos resultantes del cruce.
- Durante la fase de **reemplazo**, uniremos la descendencia (que estará formada por tan solo dos individuos) con toda la población original (formada por T individuos). De entre los $T + 2$ individuos, escogeremos un total de $T - 1$, utilizando para ello la **selección por torneo**.El tamaño del torneo será $K = 2$. A los individuos seleccionados añadiremos siempre el mejor individuo de la generación anterior para mantener así el elitismo.

- Los reinicios se producirán cuando transcurran 500 generaciones sin mejora (el número de generaciones total es siempre mayor en este tipo de algoritmos).

3. Valoración de la calidad de los algoritmos

La valoración de la calidad de un algoritmo se da cuando queremos elegir el mejor algoritmo entre un conjunto de algoritmos disponibles, o cuando queremos valorar, para un solo algoritmo, el mejor valor de los parámetros o la mejor estrategia. En general, la calidad de los algoritmos se puede medir en función de su **eficacia** (valor objetivo de la solución obtenida) o de su **eficiencia** (el tiempo computacional necesario para su obtención). Como ya hemos visto, el coste computacional se suele limitar, ya que se establece un número máximo de evaluaciones de la solución. Sin embargo, dependiendo de cómo se haya implementado el algoritmo, el coste necesario para realizar un número de evaluaciones puede ser mayor o menor.

Además, a diferencia de los algoritmos deterministas, las metaheurísticas son algoritmos estocásticos, es decir, su ejecución depende de decisiones aleatorias y cada vez que se ejecutan pueden resultar en soluciones distintas. El papel de la aleatoriedad es especialmente importante en los AGs. Cuando se analiza su comportamiento, se intenta que el resultado no esté sesgado por una secuencia aleatoria concreta que pueda influir positiva o negativamente en las decisiones tomadas durante su ejecución. De otro modo, las conclusiones obtenidas pueden no ser válidas en general y estar determinadas por dicha secuencia. Por tanto, resulta necesario efectuar varias ejecuciones con distintas secuencias aleatorias (caracterizadas normalmente por una semilla inicial) y calcular el resultado sobre todas las ejecuciones para representar con mayor fidelidad su comportamiento.

Para mostrar un resumen de los resultados obtenidos por un algoritmo a lo largo de varias ejecuciones en las que se han usado distintas semillas, se deben construir tablas que recojan los valores correspondientes a estadísticos como el mejor y peor resultado para cada instancia del problema, así como la media y la desviación típica de todas las ejecuciones. Alternativamente, se pueden emplear descripciones más representativas como los *boxplots*, que proporcionan información de todas las ejecuciones realizadas mostrando mínimo, máximo, mediana y primer y tercer cuartil de forma gráfica [1].

Para valorar cómo funcionan los distintos algoritmos comparados en esta práctica, se emplearán las instancias del CPH disponibles en Moodle. De esta forma, tendremos un total de 20 instancias (10 con 50 nodos entre los que hay que escoger 5 concentradores y 10 con 100 nodos de los que vamos a escoger 10 concentradores). Cada algoritmo se repetirá tres veces, con tres semillas aleatorias (10, 20 y 30) y, partir de los resultados obtenidos, se obtendrán las tablas de comparación. Se van a considerar dos medidas, $Desv$ y $Tiempo$:

- $Desv$ es la desviación, en porcentaje, del valor obtenido por cada método con respecto al mejor valor conocido para esa instancia del problema,

$$Desv_{ij} = 100 \cdot \frac{valorObtenido_{ij} - mejorValor_i}{mejorValor_i} \tag{1}$$

  donde i representa el índice de cada una de las instancias del problema, j representa cada una de las repeticiones del problema, $valorObtenido_{ij}$ es el valor obtenido por el algoritmo para la instancia i y repetición j y $mejorValor_i$ es el mejor valor conocido para esa instancia.

- $Tiempo_{ij}$ es el tiempo total empleado por el algoritmo para resolver la instancia i durante la repetición j.

Para resumir el comportamiento de los algoritmos, recogeremos los valores medios de $Desv$ y $Tiempo$ agrupando las instancias, de forma que se deberán construir una tabla similar a la

[1] Para obtener información adicional sobre la técnica de representación basada en *boxplots*, consultar el siguiente enlace `http://en.wikipedia.org/wiki/Box_plot`. La dirección `http://www.vertex42.com/ExcelTemplates/box-whisker-plot.html` proporciona una plantilla para generar los *boxplots* en Excel.

recogida en la Tabla 1. Cada una de las filas de dicha tabla, representa la media obtenida a partir de las tres repeticiones. La tabla también incluye valores medios, desviación típica, máximo y mínimo obtenidos a partir de los valores medios de todas las instancias. Es obligatorio incluir esta tabla en la memoria de la práctica y analizar los resultados.

Para facilitar la obtención de esta tabla se ha subido a Moodle un hoja Excel preparada para realizar todos los cálculos. Los valores reflejados en esa hoja son los obtenidos al ejecutar los algoritmos en la máquina `ts.uco.es`. Además, se ha subido también un *script* que aplica los dos algoritmos a todos los ficheros de instancias y genera una salida en formato `.csv` que puede ser incorporada fácilmente a la hoja Excel.

	AGg		AGe	
Instancia	$Desv(\%)$	$Tiempo$	$Desv(\%)$	$Tiempo$
phub_100_10_10.txt	12,94	7,969	21,53	6,845
phub_100_10_1.txt	12,77	8,113	23,55	6,965
phub_100_10_2.txt	12,48	7,973	23,98	6,910
phub_100_10_3.txt	12,50	8,069	19,10	6,880
phub_100_10_4.txt	11,18	8,048	16,40	6,911
phub_100_10_5.txt	8,06	8,036	16,51	6,814
phub_100_10_6.txt	6,25	7,993	15,92	6,872
phub_100_10_7.txt	9,62	8,038	16,25	6,789
phub_100_10_8.txt	17,03	8,021	18,36	6,873
phub_100_10_9.txt	11,05	8,052	14,56	6,857
phub_50_5_10.txt	9,51	0,972	18,07	1,367
phub_50_5_1.txt	7,87	0,975	18,87	1,382
phub_50_5_2.txt	5,59	0,979	9,48	1,367
phub_50_5_3.txt	11,42	0,988	24,83	1,365
phub_50_5_4.txt	2,69	0,964	8,76	1,348
phub_50_5_5.txt	13,20	0,970	23,87	1,365
phub_50_5_6.txt	8,68	0,980	9,47	1,373
phub_50_5_7.txt	6,18	0,968	13,93	1,365
phub_50_5_8.txt	5,23	0,976	7,96	1,353
phub_50_5_9.txt	5,75	0,968	21,49	1,356
Media	9,50	4,503	17,14	4,118
Desviación Típica	3,56	3,620	5,29	2,825
Máximo	17,03	8,113	24,83	6,965
Mínimo	2,69	0,964	7,96	1,348

Tabla 1: Resultados obtenidos para el CPH

4. Entregables

Los ficheros a entregar serán los siguientes:

- Memoria de la práctica en un fichero `pdf` que describa el programa generado, las tablas de resultados y analice los mismos.
- Fichero ejecutable de la práctica y código fuente.

4.1. Memoria de la práctica

La memoria de la práctica deberá incluir, al menos, el siguiente contenido:

- Portada con el número de práctica, título de la práctica, asignatura, titulación, escuela, universidad, curso académico, nombre, DNI y correo electrónico del alumno.

- Índice del contenido de la memoria con numeración de las páginas.
- Breve descripción del problema CPH (**máximo 1 carilla**).
- Descripción de las consideraciones generales de los AGs, incluyendo descripción del esquema de representación de soluciones, descripción en pseudocódigo de la función objetivo, etc... (**máximo 2 carillas**).
- Descripción en pseudocódigo de los pasos de cada algoritmo y de todas aquellas operaciones relevantes. El pseudocódigo deberá forzosamente reflejar la implementación/el desarrollo realizados y no ser una descripción genérica extraída de las diapositivas de clase o de cualquier otra fuente. Para esta práctica, además de la descripción del esquema general de ambos AGs, se deberá incluir la descripción en pseudocódigo del método de selección por ruleta y del método de selección por torneo. Describir también la implementación de los operadores de cruce y mutación (**máximo total 3 carillas**).
- Experimentos y análisis de resultados:
  - Descripción de las instancias de los problemas empleadas y de los valores de los parámetros considerados en las ejecuciones de cada algoritmo.
  - Resultados obtenidos según el formato especificado en la sección anterior.
  - Análisis de resultados. El análisis deberá estar orientado a justificar (según el comportamiento de cada algoritmo) los resultados obtenidos, en lugar de realizar un análisis meramente descriptivo de las tablas. Tener en cuenta que esta parte es decisiva en la nota de la práctica. Se valorará la inclusión de los siguientes elementos de comparación:
    - Gráficas de convergencia: reflejan, en el eje x, el número de evaluación del algoritmo y, en el eje y, el valor de la función objetivo para la mejor solución encontrada hasta el momento. Para el caso de los AGs, es una buena idea incluir también, en la misma gráfica, el valor medio de la función objetivo en toda la población (de esta forma, será fácil observar el efecto de los reinicios en la población y comprobar si éstos ayudan a evitar la convergencia prematura).
    - *boxplots*: representan de manera más fiel la distribución de las soluciones obtenidas para cada algoritmo. Consulta la sección 3 para ampliar la información sobre los mismos.
- Referencias bibliográficas u otro tipo de material distinto del proporcionado en la asignatura que se haya consultado para realizar la práctica (en caso de haberlo hecho).

Aunque lo importante es el contenido, se valorará también la presentación, incluyendo formato, estilo y estructuración del documento. La presencia de demasiadas faltas ortográficas puede disminuir la nota obtenida.

4.2. Ejecutable y código fuente

Junto con la memoria, se deberá incluir el fichero ejecutable preparado para funcionar en las máquinas de la UCO (en concreto, probar por `ssh` en `ts.uco.es`). Además se incluirá todo el código fuente necesario. No incluir los ficheros correspondientes a las instancias de los problemas. El fichero ejecutable deberá tener las siguientes características:

- Su nombre será `practica3`.
- El programa a desarrollar recibe cuatro argumentos por la línea de comandos (que pueden aparecer en cualquier orden)[2]:

[2]Para procesar la secuencia de entrada, se recomienda utilizar la función `getopt()` de la librería `libc`

- Argumento `f`: Indica el nombre del fichero que contiene la instancia del problema a cargar. Sin este argumento, el programa no puede funcionar.
- Argumento `p`: indica el tipo problema al que corresponde la instancia a cargar (en este caso, siempre será `CPH`).
- Argumento `s`: Indica la semilla para los números aleatorios que se debe utilizar. Si el usuario no especifica semilla, se extraerá a partir de la fecha.
- Argumento `a`: Indica el algoritmo a aplicar (`AGg` o `AGe`).

- Para que la salida generada por el programa sea compatible con el script de análisis, es necesario que se genere una línea como la siguiente:

```
i02gupep@NEWTS:~/practica3MH/Release$ practica3 -s 10 -f "../instancias/CPH/
    phub_100_10_6.txt" -p CPH -a AGg
...
Función objetivo final: XXX
```

  donde `XXX` es el valor objetivo de la solución obtenida para esa instancia por el algoritmo aplicado. Esa línea "`Función objetivo final: XXX`" debe por tanto generarse tal cual, incluyendo la tilde en `Función` y el espacio tras los dos puntos.

El *script* `bash` de análisis (disponible en Moodle) aplica este programa al conjunto de instancias considerado. Las instancias para esta práctica son las mismas que las que se utilizaron en la Práctica 1 para el CPH.

Para ejecutar el *script*, éste deberá copiarse (junto con el fichero de mejores valores) a la carpeta donde esté el ejecutable de la práctica. El *script* espera recibir como argumentos en la línea de comandos la carpeta donde están todas las instancias del problema y el nombre del informe a generar. El informe generado, será un fichero `.csv` que contendrá, separados por comas, los valores obtenidos para cada uno de los dos algoritmos en cada una de las tres repeticiones consideradas por cada instancia del problema. También se incluyen los mejores valores obtenidos en la literatura con algoritmos específicos. Se incluye a continuación un ejemplo de ejecución del *script*:

```
i02gupep@NEWTS:~/practica3MH/Release$ ./procesaTodasInstancias.sh ../instancias/ informe
    .csv
...
```

Para cargar el informe con OpenOffice, al abrirlo, hay que asegurarse que se considera la coma como carácter separador, que la codificación de caracteres escogida es correcta y que el idioma escogido admite el punto como separador decimal (por ejemplo, idioma inglés). El contenido del informe puede incorporarse a la hoja Excel, para así generar la tabla necesaria.