Examen Parcial de IA

(9 de noviembre de 2009)
grupo 20
Duración: 1 hora

1. (6 puntos) Después de los incendios del verano se nos plantea el problema de repoblar las áreas afectadas. Dada un área concreta de cierto número de hectáreas decidimos volver a repoblarla, para ello dividimos el área en una cuadrícula de $N \times M$ y nos planteamos cuanto plantar en cada área.

   Podemos decidir un factor de repoblación para cada elemento de la cuadrícula ($Fr(i)$) que va de 0 a 3 (0 significa ninguna repoblación, 1 significa plantar A árboles, ..., 3 significa plantar $3 \times A$ árboles). Disponemos un máximo de $K \times A$ árboles para plantar, pero no queremos plantar menos de $I \times A$ árboles.

   Además queremos minimizar el riesgo de incendio del área. Este es una función del factor de repoblación de un elemento de la cuadrícula y todos los elementos adyacentes, de la siguiente manera:

$$R(i) = \sum_{\forall j \ adyacente \ a \ i} Fr(i) \times Fr(j)$$

   El objetivo es encontrar una solución que plante el máximo número de árboles (entre los límites $I \times A$ y $M \times A$) y con el mínimo riesgo de incendio.

   Una posible solución a éste problema se puede obtener mediante el uso de algoritmos de búsqueda local. En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística,...). Comenta muy brevemente la solución que se propone respecto a si es correcta y si es mejor/peor respecto a otras alternativas posibles. Justifica tus respuestas.

   a) Usar Hill Climbing. Para ello generamos una solución inicial en la que todas las áreas tienen factor de repoblación 0. Los operadores son aumentar o disminuir el factor de repoblación de un área. La función heurística que queremos optimizar es:

$$h'(n) = \sum_{i=1}^{M \times N} R(i) - \sum_{i=1}^{M \times N} Fr(i)$$

   b) Usar Hill Climbing. Para ello generamos una solución inicial en la que asignamos secuencialmente factor de repoblación 1 a cada elemento de la cuadrícula hasta llegar a la cantidad mínima de árboles que hemos de plantar. Los operadores son aumentar el factor de repoblación siempre que este no sea 3 o disminuir el factor de repoblación de un área siempre que no sea 0. La función heurística que queremos optimizar es:

$$h'(n) = (\sum_{i=1}^{M \times N} Fr(i) - K) - \frac{1}{\sum_{i=1}^{M \times N} R(i)}$$

   c) Usar algoritmos genéticos. Para representar una solución generamos una tira de $2 \times M \times N$ bits (con 2 bits codificamos el factor de repoblación). Como solución inicial utilizamos la misma que en el apartado anterior. Como operadores utilizamos los operadores habituales de cruce y mutación. La función heurística que queremos optimizar es:

$$h'(n) = \sum_{i=1}^{M \times N} R(i) \times \sum_{i=1}^{M \times N} Fr(i)$$

2. (4 puntos) Queremos asignar un conjunto de seminarios a una lista de aulas. Para cada seminario tenemos la fecha tentativa en la que debería realizarse, pero sabemos que tenemos un margen de hasta tres días que podemos usar para retrasar su inicio. También disponemos de la duración del seminario (1, 2 o 3 horas).

   Para cada aula sabemos en qué fecha está disponible y durante cuántas horas (1, 2 o 3 horas). Un aula sólo se puede reservar para todo el periodo durante el que está disponible, por lo que si el seminario dura menos estaremos perdiendo horas.

   El objetivo sería asignar los seminarios a las aulas de manera que se minimice el número de horas desperdiciadas y que los seminarios se retrasen lo mínimo posible. Supondremos que tenemos aulas suficientes para asignar todos los seminarios y que hay muchas más aulas que seminarios. Se nos plantean las siguientes formas de solucionar el problema:

   a) Queremos utilizar A* de manera que ordenamos las aulas según la fecha en las que están disponibles (en caso de estar disponible en la misma fecha decidimos un criterio de ordenación). Procedemos a asignar los seminarios siguiendo el orden establecido utilizando dos operadores: asignar un seminario que quepa en las horas disponibles y que no viole las restricciones de fecha de inicio (el coste sería las horas disponibles del aula) o no asignar nada al aula (el coste sería cero). Como función heurística utilizaremos la suma de horas de seminario que quedan por asignar.

   b) Queremos usar satisfacción de restricciones para resolver el problema, para ello elegimos como variables los seminarios y para cada seminario calculamos su dominio seleccionando todas las aulas en la que puede ubicarse cumpliendo las restricciones de fechas y de horas necesarias. Añadimos dos restricciones adicionales al problema, la primera es que la suma de retrasos de los seminarios ha de ser menor que el valor R y que la diferencia entre las horas que dura cada seminario y las horas libres del aula en el que es ubicado ha de ser menor o igual que uno.

   Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

Examen Parcial de IA

(11 de noviembre de 2009)
grupo 10
Duración: 1 hora

1. (6 puntos) Una compañía de telecomunicaciones debe distribuir un conjunto de antenas de comunicaciones $A = \{a_1, a_2, ..., a_n\}$, para ello tiene que ubicar un conjunto de R arrays de antenas en la ciudad.

   Cada una de las n antenas viene descrita por una potencia $P(a_i) \in \{1Mw, 2Mw, 5Mw\}$, y su respectivo ancho de banda $B(a_i) \in \{5Mbps, 15Mbps, 30Mbps\}$, o sea, tenemos antenas de 3 tipos diferentes. En el conjunto de antenas A tenemos un número no determinado de antenas de cada tipo.

   Cada array de antenas está compuesto por tres mástiles con una distancia de 2 metros entre cada mástil, en cada mástil podemos ubicar secuencialmente 3 antenas, cada antena está separada un metro de la siguiente, o sea, nos caben 9 antenas por array y las posiciones están numeradas del 1 al 9. En un array de antenas no podemos colocar más de 5 antenas del mismo tipo.

   1. La compañía quiere la mayor eficiencia en la gestión de los arrays de antenas. El ancho de banda total necesario AB es menor que el ancho de banda de la suma de las antenas, por lo tanto quiere que el número arrays de antenas realmente usado sea el menor posible, pero superando el ancho de banda mínimo AB.
   2. Queremos que cada array esté a su máxima ocupación respetando la restricción de que no puede haber más de 5 antenas de cada tipo en un array.
   3. El ancho de banda efectivo de un array de antenas depende de las interferencias que hay entre antenas, queremos minimizar las interferencias totales entre las antenas. Podemos calcular el total de interferencias entre antenas de un array como:

      $$I(array_k) = \sum_{\forall a_i, a_j \in array_k} \frac{P(a_i)P(a_j)}{d(a_i, a_j)}$$

      donde $d(a_i, a_j)$ es la distancia euclídea entre dos antenas del mástil.

   Una posible solución a éste problema se puede obtener mediante el uso de algoritmos de búsqueda local. En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística,...). Comenta muy brevemente la solución que se propone respecto a si es correcta y si es mejor/peor respecto a otras alternativas posibles. Justifica tus respuestas.

   a) Usar Hill-climbing. Como solución inicial se colocan al azar todas las antenas que quepan en los R arrays. Como operadores se utilizan `añadir_antena(antena,array,posición)`, aplicable si la posición está libre y `quitar_antena(array,posición)`, aplicable si hay una antena en la posición. Como función heurística optimizamos:

   $$h_1(n) = (\sum_{\forall array_i} \sum_{\forall a_j \in array_i} B(a_j) - AB) + \sum_{\forall array_i} I(array_i)$$

   b) Usar Hill-climbing. Como solución inicial se ordenan las antenas según su ancho de banda, se ordenan arbitrariamente los arrays y se van recorriendo iterativamente llenándolos completamente con antenas respetando que no haya mas de 5 antenas del mismo tipo y hasta sobrepasar el ancho de banda mínimo AB. Como operador se utiliza `intercambiar_antenas(arrayO, arrayD, posO, posD)`, los arrays pueden ser el mismo (cambiamos dos antenas dentro de

un array), para realizar el intercambio debe haber antenas en las posiciones indicadas. Como función heurística optimizamos:

$$h_2(n) = \sum_{\forall array_i} I(array_i) \times |array|$$

siendo $|array|$ el número de arrays que tienen antenas.

c) Usar Algoritmos Genéticos. Asignamos números consecutivos a todas las posiciones de los arrays de antenas, de manera que tenemos las posiciones numeradas de 1 al $R \times 9$. Con la representación en binario de este número construimos una tira de bits concatenando las posiciones donde está ubicada cada una de las n antenas de las que disponemos. Como solución inicial colocamos tres antenas de cada tipo en cada array hasta superar el ancho de banda AB. Como operadores utilizamos los operadores habituales de cruce y mutación. Como función heurística optimizamos:

$$h_3(n) = \sum_{\forall array_i} I(array_i) \times \sum_{\forall array_i} \sum_{\forall a_j \in array_i} B(a_j)$$

2. (4 puntos) Deseamos construir un circuito integrado de manera que la intensidad de corriente necesaria por los diferentes elementos esté equilibrada en todo el circuito. Para simplificar el problema hemos supuesto que el circuito es una cuadrícula de $N \times M$. Los diferentes elementos que queremos colocar en el circuito son de K tipos diferentes y tenemos que colocar un número e_k de cada uno de ellos. Cada tipo de elemento necesita una intensidad de corriente específica.

   Cada posición del circuito puede albergar P elementos de cualquier tipo y hemos de colocar todos los elementos.

   Lo que hemos de conseguir es que la suma de las intensidades que necesitan los elementos para cada fila y cada columna no supere un valor I y que la diferencia de intensidad de corriente entre una celda y cada una de sus cuatro vecinas contiguas no sea mayor que un valor V.

   Se nos plantean las siguientes alternativas:

   a) Queremos utilizar satisfacción de restricciones donde las variables son los diferentes elementos a colocar y los valores son las coordenadas (i, j) de la cuadrícula. Las restricciones aparecerán entre los elementos de manera que la suma de las intensidades de los elementos que compartan la misma fila y los que compartan la misma columna sea menor que I y que la diferencia entre la suma de las intensidades de los elementos de una celda con la suma de las intensidades de los elementos de las celdas vecinas contiguas no sea superior en V.

   b) Queremos utilizar A*, para ello recorremos secuencialmente la cuadrícula desde la esquina superior izquierda hasta la esquina inferior derecha. Usamos como operador el colocar de 0 a P elementos en la casilla actual, el coste del operador es el número de elementos colocados. La función heurística es el número de elementos por colocar o infinito si la suma de intensidades de los elementos de alguna fila o columna supera el valor I o la diferencia entre la suma de las intensidades de los elementos de una celda con la suma de las intensidades de los elementos de las celdas vecinas contiguas no sea superior en V.

   Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

Examen Parcial de IA

(11 de noviembre de 2009)
grupo 30
Duración: 1 hora

1. (6 puntos) Una empresa del mundo del espectáculo gestiona los contratos de N grupos musicales que se dedican a actuar en las fiestas de los pueblos. Cada grupo musical tiene una tarifa por actuación, de la que una parte se la queda el grupo y otra la empresa. Por otro lado, los municipios celebran sus fiestas patronales en unas ciertas fechas y dedican una parte de su presupuesto festivo, X euros, a pagar las actuaciones de los grupos musicales. El problema es organizar las giras de los grupos de modo que se cubran las peticiones de los pueblos, en total P actuaciones (puede haber más de una actuación en un pueblo en días consecutivos), sin pasarse del presupuesto municipal, se maximice la recaudación de la empresa y se minimicen los kilómetros que al cabo de la temporada ha de realizar cada grupo. Además, el mismo grupo no puede actuar más de un día en el mismo pueblo y cuando acaban en un pueblo, se van al siguiente en el que tienen actuación.

   En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística). Hay que comentar la solución que se propone respecto a si es correcta, es eficiente, y es mejor o peor en comparación con otras alternativas. Y hay que justificar todas las respuestas.

   a) Se propone usar Hill-Climbing. La solución de partida coloca, en orden creciente de fecha, un grupo diferente en cada una de las fechas previstas de actuación y cada grupo se asigna una sola vez. El operador disponible es `poner_grupo` y la función heurística es:

$$h'(n) = \frac{\sum_{\forall grupo_i} Kilometros\_recorridos(grupo_i)}{\sum_{\forall grupo_i} Dinero\_recaudado\_empresa(grupo_i)}$$

   b) Se propone usar Hill-Climbing. La solución de partida se calcula de la siguiente manera: se ordenan los grupos en orden decreciente de beneficio para la empresa y se empiezan a asignar siguiendo ese orden a cada una de las fechas disponibles. Si cuando ya se han colocado todos los grupos una vez todavía quedan fechas sin cubrir, se repite el proceso las veces que sea necesario. Se respeta que el mismo grupo no actúe más de una vez en el mismo pueblo. El operador es `intercambiar_grupo` siempre y cuando no suponga que se supera el presupuesto del municipio y se respete una única actuación por grupo en el mismo pueblo. La función heurística es:

$$h'(n) = \sum_{\forall grupo_i} Kilometros\_recorridos(grupo_i) - \sum_{\forall grupo_i} Dinero\_recaudado\_empresa(grupo_i) * 100$$

   c) Usar Algoritmos Genéticos. Se asigna un identificador binario de longitud b a cada uno de los grupos, la solución se representa como una tira de $P \times b$ bits. La solución de partida se calcula de la siguiente forma: Se ordenan los grupos en orden decreciente de beneficio, se toma el primer grupo y se le asigna la primera fecha libre sin pasarse del presupuesto del municipio, luego se busca la siguiente fecha libre en la que pueda actuar el mismo grupo, y tal que suponga un desplazamiento mínimo para el grupo y que no supere el presupuesto del municipio, se le asigna el grupo y se repite el proceso hasta que no queden más fechas posibles para ese primer grupo. Se repite exactamente el mismo proceso para los grupos sucesivos hasta que ya no queden fechas por cubrir o no haya más grupos.

   Como operadores genéticos se utiliza solamente el operador de cruce, donde los puntos de cruce no pueden ser dentro de la identificación de un grupo. El heurístico es el siguiente:

$$h'(n) = \sum_{\forall grupo_i} Kilometros\_recorridos(grupo_i) \times \frac{Dinero\_cobrado(grupo_i)}{Dinero\_recaudado\_empresa(grupo_i)}$$

2. (4 puntos) Queremos conectar N ciudades con B autobuses. Cada autobús ha de comenzar su trayecto en la ciudad C_1 y finalizarlo en la ciudad C_N. Conocemos las distancias entre cada par de ciudades y queremos que el camino recorrido por cada autobús no exceda K kilómetros y que el número de paradas que haga cada autobús no sea mayor que P.

   Buscamos la solución que conecte todas las ciudades sin que dos autobuses pasen por la misma ciudad (exceptuando C_1 y C_N) recorriendo el mínimo número de kilómetros.

   a) Queremos utilizar A*. Definimos el estado como la asignación de ciudades a autobuses. El estado inicial consiste en asignar las ciudades C_1 y C_N a cada autobús. Tenemos un operador que asigna una ciudad a un autobús y cuyo coste es la suma de las distancias entre la ciudad asignada con C_1 y C_N, sólo podemos asignar una ciudad a un autobús si no excedemos el valor P. Como función heurística usamos la suma de las distancias de todas las ciudades por asignar con C_1 y C_N.

   b) Aplicar un algoritmo de satisfacción de restricciones. Las variables son las ciudades, los dominios son los autobuses donde podemos asignarlas. Supondremos que no tenemos en cuenta las ciudades C_1 y C_N ya que estarán asignadas a todos los autobuses. Como restricciones imponemos que un autobús no esté asignado a más de P ciudades y que la longitud del camino mínimo que recorre las ciudades asignadas a un autobús no sea mayor que K.

   Comenta cada una de las posibilidades indicando si resuelven o no el problema, qué errores te parece que tiene cada solución y cómo se podrían corregir, y qué ventajas e inconvenientes tienen cada una de ellas. Justifica la respuesta.

IA Midterm Exam

(November 11th 2009)

Time: 1 hour

1. (6 points) A company that organizes musical events manages the contracts of N musical groups that perform in small cities festivals. Each musical group charges certain quantity of money for a concert, part of the money is for the group and part for the company. All small cities have festivals in certain dates and save part of the budget (X euros) to pay for the concerts of the musical groups.

   The problem is to organize the tours of the groups in order to fulfill all the petitions of the small towns, a total of P concerts (there could be more than a concert in a town in consecutive days), not exceeding the budget, maximizing the money that the company gets and minimizing the kilometers that at the end of the year each group travel. Additionally, the same group can not perform more than a day in the same town and when they finish in a town then they go to the next town.

   We want to solve this problem as a local search problem and, in the following sections, different alternatives are proposed for some of the necessary elements to start the search (initial state, operators, evaluation function...). You have to comment the proposed solution with respect to wether it is correct, it is efficient, or it is better or worse than other alternatives. Justify your comments.

   a) Using Hill climbing search. The initial solution assigns in ascendant order of date a different group for each one of the concert dates and each group is only assigned once. As search operator we use `put_group` and the heuristic function is:

   $$h'(n) = \frac{\sum_{\forall group_i} Kilometers\_traveled(group_i)}{\sum_{\forall group_i} Money\_for\_the\_company(group_i)}$$

   b) Using Hill climbing search. The initial solution is computed in the following way: We obtain a list of groups sorted by the benefit for the company in descending order and we use this list to assign the groups to the dates of the concerts. If when we finish assigning all the groups there are still dates not covered then we iterate the process until all concerts have a group. We respect the constraint that a group can not give more than a concert in a town. As search operator we use `swap_groups` that can be applied if the cost of the group is covered by the town budget and if a group only performs once in the same town. The heuristic function is:

   $$h'(n) = \sum_{\forall group_i} Kilometers\_traveled(group_i) - \sum_{\forall group_i} Money\_for\_the\_company(group_i) * 100$$

   c) Using Genetic Algorithms. We assign an binary identifier of length b to each one of the groups, the solution is represented as an string of $P \times b$ bits. The initial solution is computed in the following way: Groups are ordered descending by benefit, we get the first group and we assign it to the first free date if we do not exceed the budget of the town, then we look for the next free date where this group can perform with the shortest travel and not exceeding the budget of the town, we assign the group again and we go on until we can not assign this group anymore. We repeat this process for the rest of the groups until all dates are covered or there are no more groups.

   As genetic operators we use only the crossover operator, but we can not use as crossover point positions in the middle of the code of a group. The heuristic is as following:

   $$h'(n) = \sum_{\forall group_i} Kilometers\_traveled(group_i) \times \frac{Money\_for\_the\_group(group_i)}{Money\_for\_the\_company(group_i)}$$

2. (4 points) We want to connect C cities with B buses. Each bus begins its route in city C_1 and ends in city C_N. We know the distances between each pair of cities and we want the length of the route of each bus to be less than K kilometers and the number of stops for each bus to be less than P.

   We want a solution that connects all the cities with routes that do not share cities among them (except for C_1 and C_N) and that the total length of the routes is the minimum.

   a) We want to use A*. We define a state as the sets of cities assigned to the buses. The initial state is to assign cities C_1 and C_N to each bus. We have an action that assigns a city to a bus with cost the sum of distances from this city to C_1 and C_N, a city can be assigned to a bus only if we do not exceed the limit P. As heuristic function we use the sum of distances from all the cities not yet assigned to C_1 and C_N.

   b) We want to use constraints satisfaction. The variables are the cities, the domains are the buses. The cities C_1 and C_N are not used because all the buses will have these cities. As constrains we impose that a bus can not be assigned to more than P cities and that the length of the minimal path that connects all the cities assigned to the same bus has to be less than K.

   Comment on each one of the possibilities about if they are or are not solutions to the problem, what errors you can find in each solution and how they should be corrected and the advantages or disadvantages for each one. Justify your answers.