Examen Parcial de IA

(7 de noviembre de 2011)
grupo 10
Duración: 1 hora

1. (6 puntos) Una empresa de transporte marítimo desea decidir la configuración de la carga de su próximo barco. La empresa recibe un conjunto de peticiones de envío de entre las que escoger. Cada petición va dentro de un tipo contenedor que está fijado por el tamaño, el peso del envío y las características de la carga. Existen solamente K tipos de contenedor.

   Los contenedores son propiedad de la empresa y su uso tiene un coste asociado que depende del tipo de contenedor. Cada petición tiene asociada un precio de transporte.

   Existen diferentes restricciones para la carga: la suma total de pesos de los contenedores no ha de sobrepasar la capacidad de carga del barco P_{max}, ni ha de ser inferior a cierto valor P_{min}. Cada contenedor tiene un peso asociado que depende de la carga que contiene. Las posibilidades de colocar los contenedores en el barco también imponen que haya un mínimo de C_{min} contenedores y un máximo de C_{max} contenedores de cada tipo.

   El objetivo es encontrar la combinación de peticiones que den el máximo precio de transporte y tengan el coste más pequeño dentro de las restricciones impuestas.

   En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

   a) Se plantea solucionarlo mediante Hill-climbing utilizando como solución inicial el barco vacío y como operadores añadir y quitar contenedores del barco. Queremos usar como función de evaluación de las soluciones lo siguiente:

   $$h(n) = \sum_{i=0}^{Ncont} Precio_i \times \sum_{i=0}^{Ncont} Coste_i$$

   Donde $Ncont$ es el número de contenedores que hay en una solución, $Precio_i$ es el precio de transporte del contenedor i de la solución y $Coste_i$ es el coste de un contenedor i de la solución.

   b) Se plantea solucionarlo mediante Hill-climbing utilizando como solución inicial escoger al azar C_{min} contenedores de cada tipo. Como operador utilizamos intercambiar un contenedor del barco por otro que no esté en él. Queremos usar como función de evaluación de las soluciones lo siguiente:

   $$h(n) = \sum_{i=0}^{Ncont} \frac{Precio_i}{Coste_i}$$

   c) Se plantea utilizar algoritmos genéticos donde la representación de la solución es una tira de bits con tantos bits como peticiones haya, donde cada bit significa si la petición está o no en el barco. Para generar la población inicial usamos la misma técnica que en el apartado anterior. Usamos los operadores habituales de cruce y mutación y como función de evaluación usamos la del apartado anterior pero haciendo que valga cero si la solución incumple las restricciones del problema.

2. (4 puntos) Una fábrica de coches quiere optimizar su proceso de fabricación integrando el montaje de los k diferentes modelos que fabrica en un solo proceso. Cada modelo de coche necesita utilizar un conjunto de máquinas en un orden específico que se puede describir mediante una secuencia de proceso. Para simplificar supondremos que cada paso del proceso tiene la misma duración. Hay

que tener en cuenta que un coche puede necesitar un mismo tipo de máquina en diferentes pasos de su proceso de fabricación.

Disponemos de un número fijo de máquinas de cada tipo (M). Supondremos que el proceso lo diseñamos para fabricar una unidad de cada modelo de coche y solo podemos volver a iniciar el proceso una vez que hemos acabado, es decir, en el proceso integrado solo se fabricará una unidad de cada modelo a la vez.

Se nos plantean las siguientes alternativas:

a) Queremos utilizar A* para minimizar el número total de pasos del proceso integrado. El estado lo forman los pasos de los procesos individuales que hemos encajado en cada paso del proceso integrado.

Los operadores de cambio de estado consisten en colocar un paso de un proceso individual en un paso del proceso integrado ya existente o en crear un nuevo paso del proceso integrado con uno de los pasos de un proceso individual, la restricción que tenemos para aplicar los operadores es que no podemos asignar un paso individual si los pasos anteriores no han sido colocados. El coste de ambos operadores es uno. La función heurística es la suma de pasos de los procesos individuales que nos quedan por integrar.

b) Queremos utilizar satisfacción de restricciones para minimizar el número total de pasos. Para ello elegimos tener como variables la asignación de un paso de fabricación de un modelo a una máquina en un paso del proceso integrado. Suponemos que el proceso integrado tendrá como máximo tantos pasos como la suma de pasos de todos los procesos individuales, por lo que necesitaremos como variables ese valor multiplicado por el número de máquinas. El dominio de las variables son los modelos de coche. Con esto estamos representando para cada paso del proceso integrado qué máquinas tienen asignado un paso de los procesos individuales. La restriccion que mantenemos es que ninguna asignación viole el orden de las secuencias individuales, es decir, que para un paso concreto no puede haber un modelo asignado a más de una máquina y que entre pasos se cumpla el orden de las secuencias.

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

(9 de noviembre de 2011)
grupo 30
Duración: 1 hora

1. (6 puntos) El ministerio de ciencia y tecnología desea saber como hacer la asignación de ayudas a proyectos de investigación de las C convocatorias de este año. Cada convocatoria de ayudas tiene un presupuesto a repartir entre las peticiones aceptadas. A estas convocatorias los grupos de investigación pueden presentar peticiones (la misma petición puede presentarse a hasta tres convocatorias distintas, pero solo se puede conceder una vez). Una petición tiene un presupuesto solicitado y el ministerio le asigna una prioridad entre 1 y 3 dependiendo del historial del grupo investigador. Hay un total de P peticiones.

   El ministerio quiere asignar la mayor cantidad de dinero posible, pero haciendo que globalmente la proporción de grupos con concesiones de ayuda cumpla las siguientes restricciones: como máximo el 30 % de concesiones ha de ser para los grupos de prioridad 2 y como máximo el 10 % para los de prioridad 3. Obviamente la suma de los presupuestos solicitados de las peticiones asignadas a una convocatoria no puede superar el presupuesto de la convocatoria.

   En los siguientes apartados se proponen diferentes alternativas para algunos de los elementos necesarios para plantear la búsqueda (solución inicial, operadores, función heurística, ...). El objetivo es comentar la solución que se propone respecto a si es correcta, es eficiente, o es mejor o peor respecto a otras alternativas posibles. Justifica tu respuesta.

   a) Usar Hill-climbing, tomando como solución inicial la solución vacía, como operadores tenemos asignar una petición a una convocatoria y desasignar una petición de una convocatoria. Como función heurística usamos:

   $$h(n) = \sum_{\forall conv_j} \sum_{\forall pet_i \in conv_j} Presupuesto(pet_i) - \sum_{\forall conv_j} Presupuesto(conv_j)$$

   donde $pet_i \in conv_j$ significa que la petición i ha sido concedida en la convocatoria j.

   b) Usar Hill-climbing. Para hallar la solución inicial recorremos todas las convocatorias y asignamos tantas peticiones de prioridad 1 solicitadas a esa convocatoria como podamos, sin exceder su presupuesto. Como operadores utilizamos asignar una petición a una convocatoria, respetando las restricciones de máximos de peticiones concedidas por prioridad, y mover una petición de una convocatoria a otra. Como función heurística usamos:

   $$h(n) = \frac{\sum_{\forall conv_j} \sum_{\forall pet_i \in conv_j} Presupuesto(pet_i)}{\sum_{\forall conv_j} Presupuesto(conv_j)}$$

   c) Usar algoritmos genéticos. Codificamos el problema con $C \times P$ bits donde cada grupo de P bits corresponde a una convocatoria de manera que un uno significa que la petición ha sido concedida en esa convocatoria. Las soluciones iniciales las generamos utilizando el método del apartado anterior. Como operadores genéticos usamos los operadores habituales de cruce y mutación. Como función heurística usamos:

   $$h(n) = \sum_{\forall conv_j} \left(Presupuesto(conv_j) - \sum_{\forall pet_i \in conv_j} Presupuesto(pet_i) \right)$$

2. (4 puntos) En todo proyecto de desarrollo de una aplicaciones informáticas es necesario determinar cuanto tiempo y recursos va a ser necesario invertir. Una vez decididas las tareas, su duración y

la dependencia entre ellas podemos determinar automáticamente cuando se van a desarrollar y qué personas van a hacer cada tarea.

Supondremos que hemos dividido el problema en tareas que tienen todas la misma duración y hemos determinado su grafo de dependencias, de manera que para cada una de ellas sabemos qué tareas han de estar terminadas para poder empezarla. Disponemos también de un conjunto de programadores (prog_1 a prog_n) que podemos asignar a cada una de las tareas. Obviamente si asignamos el mismo programador a dos tareas que pueden realizarse simultáneamente tardaremos más que si asignamos programadores diferentes, pero siempre nos saldrá más barato usar un número menor de programadores.

El objetivo del problema es encontrar la menor asignación de programadores a tareas que respete el orden de dependencia de estas.

a) Queremos usar A* definiendo el estado como la asignación de programadores a tareas. El estado inicial sería la asignación vacía y el estado final sería la asignación completa de programadores a tareas. Como operador utilizamos asignar un programador cualquiera a una tarea cualquiera, el coste del operador sería una unidad. La función heurística es el número de tareas que quedan por asignar que tengan alguna tarea de la que dependan que no esté ya asignada.

b) Queremos usar satisfacción de restricciones, donde las variables son las tareas a asignar y los programadores los dominios de las variables. Las restricciones son las relaciones de dependencia entre las diferentes tareas. Sobre estas restricciones imponemos además que no pueda haber más de tres programadores diferentes en tareas que pueden realizarse en paralelo.

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 8th 2011)

Time: 1 hour

1. (6 points) The major of Megacity thinks that in some decades he will have enough money to build a new subway line that could reach areas that the current network does not cover. To design the project, the city engineers have determined P possible places for building subway stations and have determined, from statistical projections, how many daily users will have each potential station.

   The subway line will only have S stations chosen from the P potential places. The goal is to decide which S stations to pick and the order they have to be connected in so the line serves the largest possible number of people and the length of the line is the shortest possible.

   We want to solve this problem as a local search problem. 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 each proposed solutions 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. As initial solution we use a solution without stations. As search operators we use `add-station`, that adds a station at the end of the line and `swap-stations`, that swaps two stations of the line. The heuristic function is:

   $$h(n) = \sum_{i=1}^{S-1} dist(S_i, S_{i+1}) + \sum_{i=1}^{S} People(S_i)$$

   b) Using Hill climbing. We pick a station at random, we use this station as reference and we pick the next station as the nearest station to this one, then this new station is the reference station and we look for the nearest one not in the solution. The same procedure is repeated until we have S stations. As search operators we use `change-station`, that changes a station in the solution with one that is not in the solution, and `swap-stations`, that swaps two stations of the line. The heuristic function is:

   $$h(n) = \frac{\sum_{i=1}^{S-1} dist(S_i, S_{i+1})}{\sum_{i=1}^{S} People(S_i)}$$

   c) Using genetic algorithms. We assign to each place a number from 1 to P; the coding of the solution is a string of bits of length $S \times \log_2(P)$ that is the concatenation of the identifiers of the places in the solution; the order of the places in the string is the order of the stations in the line. In order to generate the initial population, S stations are picked at random from the P places and they are put at random in the solution. As genetic operators we use the usual crossover and mutation operators. The heuristic function is:

   $$h(n) = \alpha \times \sum_{i=1}^{S-1} dist(S_i, S_{i+1}) + (1 - \alpha) \times (-\sum_{i=1}^{S} People(E_i))$$

   Where α is a value between 0 and 1 that allows to give more or less weight to each criteria.

2. (4 points) We want to distribute different crops in a grid of size $N \times M$. The crops that we want to grow are from T different types and we have k_t plants of each crop. Each crop has different daily needs of water (w_t) and nutrients (n_t).

Each position in the grid can have a maximum of P plants of any crop. The irrigation system can distribute L liters of water daily to each one of the M columns of the grid and from each cell of the grid the plants can consume G daily grams of nutrients.

We want to distribute all the plants we have with the constraints that the daily consumption of water and nutrients can not exceed the daily amounts and that, in order to uniformly consume the nutrients in the grid, the difference in consumption of nutrients in a cell and any of the adjacent ones can not be larger that C.

We have different alternative solutions:

a) Using A*, considering that the state is an assignment of plants to cells of the grid. We use as operator to add a plant of a specific type to a cell of the grid as long it does not exceed P plants in the cell and it does not exceed the limits of consumption of water and nutrients. The cost of the operator is the sum of needs of water and nutrients of the plant that has been placed. The heuristic function is the sum of needs of water and nutrients of the plants still to be placed or infinite if the current solution has any cell where the difference of nutrients consumption between the cell and the adjacent ones is larger than C.

b) We want to use constraint satisfaction using the cells of the grid as variables and the identifier of each plant we have to place as values(obviously a variable will have a set of values). We have constraints among all cells of a column so the water needs can not exceed L liters per day, also we have constraints for each cell so the daily consumption of nutrients is not larger than G and constraints between each cell and their adjacent ones, so the difference of nutrients consumption is not larger than C.

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, how they should be corrected and the advantages or disadvantages for each one. Justify your answers.