Examen Parcial de GRAU-IA

(15 de noviembre de 2017)
grupo 10
Duración: 55 min

1. (6 puntos) Una de las labores de los equipos de comunicaciones en una red es distribuir el flujo de paquetes que le llegan a un equipo entre los diferentes equipos con los que está conectado, optimizando la calidad de la distribución de paquetes de forma local.

   Una posible estrategia de distribución de paquetes sería decidir a priori cuantos de los paquetes que se reciben en cierto segundo se envían por cada conexión, sin preocuparnos exactamente de a donde deben ir.

   Un equipo de comunicaciones recibe cierto número de paquetes por segundo (P) que tiene que distribuir entre sus conexiones de salida. Para cada una de las N conexiones de salida del equipo se conocen tres informaciones: su capacidad (en número de paquetes por segundo), el tiempo de retraso que introduce la conexión (tiempo medio adicional que añade el nodo de salida al tiempo de llegada a destino de cada paquete) y el número medio de saltos que hará cada paquete hasta llegar a su destino.

   Deseamos calcular el número de paquetes que debemos enviar por cada conexión de salida para optimizar la distribución de paquetes de manera que el retraso y número medio de saltos de los paquetes sea el mínimo posible. Evidentemente se han de enviar todos los paquetes que llegan.

   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 usando como solución inicial el repartir todos los paquetes a partes iguales entre todas las conexiones de salida del equipo. Como operador utilizamos mover cierta cantidad de paquetes de una conexión a otra. Como función heurística usamos el sumatorio, para todas las conexiones con paquetes asignados, del producto entre el retraso de la conexión y el número de saltos.

   b) Se plantea solucionarlo mediante Hill-Climbing tomando como solución inicial el repartir los paquetes entre las conexiones de salida del equipo asignando aleatoriamente a una conexión el máximo de su capacidad, repitiendo el proceso hasta haber repartido todos los paquetes. Como operador se utilizaría mover cualquiera de los paquetes de una conexión a otra siempre que la operación sea válida. Como función heurística usamos el sumatorio para todas las conexiones con paquetes asignados del producto entre el número de paquetes y el retraso de la conexión más el sumatorio, para todas las conexiones con paquetes asignados, del producto entre el número de paquetes y el número de saltos.

   c) Se plantea solucionarlo mediante Algoritmos Genéticos, donde la representación del problema es una tira de bits compuesta por la concatenación de la representación en binario del número de paquetes asignados a cada conexión (evidentemente usando el mismo número de bits para cada conexión). Como solución inicial ordenamos las conexiones por su retraso (de menor a mayor) y asignamos paquetes en ese orden llenando la capacidad de cada conexión hasta haber asignado todos los paquetes. Usamos los operadores de cruce y mutación habituales. Usamos como función heurística el sumatorio, para todas las conexiones, del producto entre el número de paquetes asignados a cada conexión y el retraso de la conexión.

2. (4 puntos) En todo proyecto de desarrollo de una aplicacion informática 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 (t_1 a t_m) 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 operadores usaremos dos: asignar un programador cualquiera a una tarea cualquiera, el coste del operador sería una unidad de duración; y desasignarle a un programador una tarea, el coste de este operador seria una unidad negativa de duración . 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, estableciendo que un programador no puede estar asignado a 2 tareas entre las que haya una dependiencia. 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.