Trabajo practico

Programación 3

Guillermo Tomas Cesario Provenzano

T.U.D.A.I U.N.I.C.E.N

Greedy/Back Tracking

Consigna:

Diseñar:

1. Un algoritmo Greedy que obtenga una solución válida al problema planteado, resultando en dos grupos de empleados con fuerzas de trabajo similares.

2. Un algoritmo Back tracking que obtenga la solución óptima al problema planteado, resultando en dos grupos de empleados con fuerzas de trabajo tan similares como sea posible.

Escribir un informe plasmando:

1. Los detalles del diseño Greedy del punto anterior:
a) ¿Qué son los candidatos?
b) ¿Cuál es el criterio Greedy para seleccionar el próximo candidato (incluyendo aquellos criterios que se pensaron y descartaron durante el desarrollo del trabajo)
c) ¿Cuándo es factible agregar un candidato a la solución?
d) ¿Cuándo alcanzamos una solución al problema?

2. Los detalles del diseño Backtracking del punto anterior:
a) ¿Qué es un estado?
b) ¿Qué datos contiene?
c) ¿Cuándo un estado es final y cuando es solución?
d) ¿Cuáles son los hijos de un estado?
e) ¿Qué modificaciones debo aplicar desde un estado para ir a un estado hijo?
f) ¿Qué poda se le puede aplicar al problema? Incluir un diagrama del árbol de exploración del algoritmo, indicando estado inicial, grado de ramificación y profundidad del árbol.

1

a) Son todos los empleados que la empresa quiere dividir en 2 grupos.

b) Probé ordenando los candidatos por su fuerza de trabajo de menor a mayor, no funcionó ya que noté que al final quedaban los de mayor peso, lo que hacía que solución óptima opte por uno y por otro y quedaba una solución grande de diferencia de trabajo.

Entonces plantee la estrategia a la inversa de mayor a menor, de modo tal que agrega los candidatos con mayor valor al principio dejando la posibilidad de repartir las que tienen menor valor al final.

Debido a que seleccionar verifica si uno de los dos grupos es mayor al otro, al empezar desde el mayor y terminar en los menores, los candidatos al terminar todas las iteraciones con mayor valor de fuerza de trabajo se dividen al principio y se llega a una solución más óptima.

Inicialmente se ordena de mayor a menor los candidatos y luego se utiliza un bucle,while(el cual itera todos los candidatos de principio a fin),el método seleccionar tiene el candidato correspondiente en cada iteración que al ordenarse de mayor a menor siempre va hacer el más grande, identifica su lugar en los grupos dependiendo cual sea el grupo más pequeño según su fuerza de trabajo y lo agrega.

c) En este caso el isFactible no es necesario, ya que todos los candidatos van a estar agregados a un grupo, porque la solución es iterar hasta el final de los candidatos.

d) Cuando se recorre toda la lista de candidatos, más simplificado que el “while” llegue a su fin, y todos los candidatos fueron asignados a un grupo, en este caso al grupo 1 o al grupo2.

2

A) Es la situación actual de la solución. Esta misma se va creando, cada instancia va avanzando en tu solución. Las instancias parciales antes de llegara la solución.

B)

- ArrayList<Empleado> candidatos
- int contador
- Grupo g1(clase donde se tiene empleados y su fuerza de trabajo)
- Grupo g2(clase donde se tiene empleados y su fuerza de trabajo)
- Solución solu1(aquí se almacena la solución más cercana)
- int contadorfuerzamaxima (se usa para la poda, es la cantidad total de fuerza de trabajo de los candidatos).

C) Cuando tengo mi primera solución, llego a la primera hoja. No significa que sea la correcta. Es un estado donde podría llegar a una posible solución si cumpliera con el método de solución vigente.

Un estado final es cuando ya corto la iteración y en el grupo 1, grupo 2 se contiene la totalidad de los candidatos seleccionados pasados por parámetro al inicio del algoritmo. No significa que sea un estado solución. Eso depende de

las condiciones del enunciado para que lo sea.

Ej.: Si tengo 6 personas y una condición es que no pueda tener más de 4 en un grupo, cuando llego a tener los 6 en un solo grupo, será un estado final pero no un estado solución. Para no generar ese estado está la poda.

Es solución cuando la totalidad de los integrantes del grupo 1 y grupo2 es igual a la totalidad de los candidatos. Y la diferencia entre la suma de trabajo de los integrantes del grupo1 y del grupo 2 es la menor de todas.

D) Si yo parto desde un estado donde no asigné ningún candidato mis 2 posibles hijos es agarrar el primer candidato y meterlo en el grupo 1 o en el grupo 2.

El estado padre con la incorporación de un nuevo candidato al grupo 1 o la incorporación al grupo 2.

E)Agregar el candidato alguno de los 2 grupos y volver a llamarse a sí mismo (método back).

F) La primera poda fue:

F)a)
```
if(g1.getIntegrantes().size()!=candidatos.size()&&g2.getIntegrantes().size()!=candidatos.size()) {//puede ser poda menor 2 variantes menos depende la cantidad de grupos
```

En este caso verifico que los candidatos no estén todos en un solo grupo. Esto me fue inútil ya que al guardar las soluciones por diferencia de fuerza de trabajo ya se descartaban las opciones que esta “poda” podía llegar ayudar.

F)b)
```
if(!(candidatos.get(contador).getFuerzaDeTrabajo()+g2.getFuerzaTotal()>contadorFuerzaMaxima/3)){
```
No funcionó porque en este caso puntual no llega a ninguna solución viable.

F).c)
```
if(!(Math.abs(g1.getFuerzaTotal()-g2.getFuerzaTotal())>100)){
```

Luego probé usar 100 (la fuerza máxima que habían establecido), entonces de este modo estoy evitando poner a 2 personas con un valor alto sin saber que voy a tener 2 personas altas para equilibrar en el grupo, por lo que me está generando resultados distintos. Para solucionar esto aumenté el valor de la suma de toda la fuerza, es un valor suficientemente alto para no podar estados validos que me llegan a la solución con este cambio:

f.d)
```
if(!(Math.abs(g1.getFuerzaTotal()-
g2.getFuerzaTotal())>contadorFuerzaMaxima/2)){
```

La fuerza total de trabajo que tiene la totalidad de candidatos la dividí a la mitad, ahí tengo el valor ideal que tendría que tener cada grupo de trabajo, ese número de diferencia, la mitad de mi valor máximo que puedo tener de valor máximo entre los grupos.

En la poda lo que se va haciendo a medida que se va agregando, se chequea que la diferencia entre ellos no sea mayor a la mitad de la máxima carga de trabajo posible que hay entre todos.

¿Cuál es la diferencia de hacerlo así con la poda que hice en la b? Que se puede tener personas con fuerza de trabajo alta y eso fue lo que pasó. Al tener valor de carga alta, ej. 253/3, Delfina no quedaba en ningún lado ya que ella ya era más de 1/3 de trabajo total.

Esto es un límite que yo tolero, que tengan de diferencia en carga de trabajos entre las personas, elegí esta porque supuse que ninguno de los empleados va a tener la mitad de carga de trabajo entre todos los candidatos seleccionados, en este caso nos dan de 0 a 100. Sin embargo, si tengo una persona que tenga un valor de fuerza más alto que el que elegí puede dar erróneo.

Si tengo 4 personas, todas tienen carga baja, ej. 10 30 5 5, la carga máxima total de trabajo es =50.

Si 50/2 es 25, entonces mi límite es 25 y ya no es 100.

Cuando yo agarré a la de 30, me supera la de 25 y nunca la voy a meter.

Justo en todos los ejemplos que dan no pasa, se podría solucionar si llegara a pasar este caso, no hacer la poda, daría bien el resultado, pero sin ser más eficiente ya que la poda no se realizaría en este caso en particular.

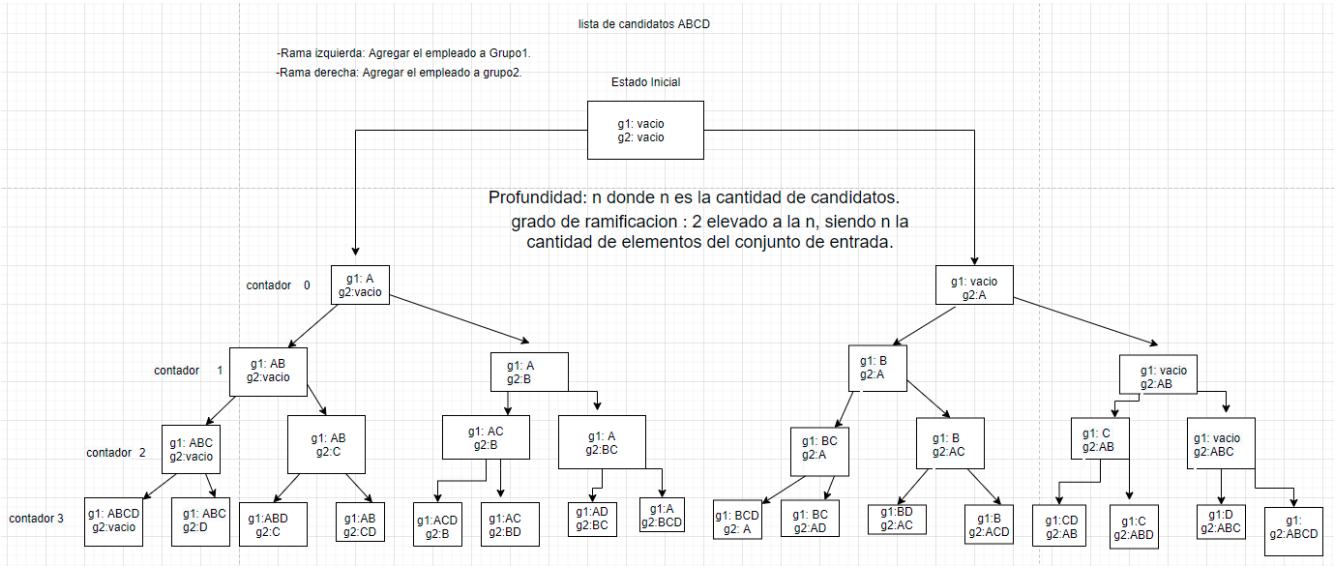
Otra es que el valor máximo entre el valor que calculamos y 100, si da bajo se va subiendo la vara a medida que sea necesario.

F)e) Ultima poda implementada:

```
if(!(Math.abs(g1.getFuerzaTotal()-g2.getFuerzaTotal())-
fuerzaDeTrabajoEmpleadoActual>contadorFuerzaMaxima/2)){
```

En este caso utilizo la diferencia de trabajo total entre los grupos menos la fuerza del empleado a agregar, si esto es mayor al máximodefuerzatotal de todos los candidatos dividido 2, podo, este caso es más eficiente que el anterior ya que contempla valores que podrían llegar a romperse en el otro a pesar de que tenga más iteraciones, por lo que es la mejor opción encontrada.

C) Árbol de exploración:



Resultados sin poda, primer poda y poda final(mas efectiva):

SIN PODA

----------=Inicio= ----------
INTERACICONES:63
Grupo1:[Juan, Roberto, Mateo] Grupo2:[Camila, Francisco, Benjamín]Diferencia de fuerza de trabajo:1

INTERACICONES:78
Grupo1:[Delfina, Emilia] Grupo2:[Francisco, Catalina]Diferencia de fuerza de trabajo:36

INTERACICONES:93
Grupo1:[Andrea, Joaquín] Grupo2:[Martina, Olivia]Diferencia de fuerza de trabajo:2

INTERACICONES:124
Grupo1:[Delfina, Elena] Grupo2:[Catalina, Camila, Francesca]Diferencia de fuerza de trabajo:16

INTERACICONES:187
Grupo1:[Mateo, Emilia, Francesca, Valentino] Grupo2:[Santino, Joaquín]Diferencia de fuerza de trabajo:1

INTERACICONES:314
Grupo1:[Santi no, Roberto, Francisco] Grupo2:[Andrea, Elena, Martina,

Diferencia de fuerza de trabajo:1

INTERACICONES:569
Grupo1:[Martina, Bautista, Ignacio, Benicio] Grupo2:[Mateo, Roberto,Valentino, Francesca]
Diferencia de fuerza de trabajo:0

INTERACICONES:1049144
Grupo1:[Juan, Roberto, Camila, Francisco, Benjamín, Mateo, Delfina, Catalina,Emilia]
Grupo2:[Benicio, Valentino, Olivia, Martina, Joaquín, Bautista, Francesca, Santino, Ignacio, Andrea, Elena]
Diferencia de fuerza de trabajo:1
----------=Fin=----------

CON PRIMER PODA

if(!(Math.abs(g1.getFuerzaTotal()-g2.getFuerzaTotal())>contadorFuerzaMaxima/2)){

----------=Inicio= ----------
INTERACICONES:45
Grupo1:[Juan, Roberto, Mateo] Grupo2:[Camila, Francisco, Benjamín]Diferencia de fuerza de trabajo:1

INTERACICONES:54
Grupo1:[Delfina, Emilia] Grupo2:[Francisco, Catalina]Diferencia de fuerza de trabajo:36

INTERACICONES:63
Grupo1:[Andrea, Joaquín] Grupo2:[Martina, Olivia]Diferencia de fuerza de trabajo:2

INTERACICONES:80
Grupo1:[Delfina, Elena] Grupo2:[Catalina, Camila, Francesca]Diferencia de fuerza de trabajo:16

INTERACICONES:113
Grupo1:[Mateo, Emilia, Francesca, Valentino] Grupo2:[Santino, Joaquín]Diferencia de fuerza de trabajo:1

INTERACICONES:200
Grupo1:[Santino, Roberto, Francisco] Grupo2:[Andrea, Elena, Martina, Emilia]Diferencia de fuerza de trabajo:1

INTERACICONES:381
Grupo1:[Martina, Bautista, Ignacio, Benicio] Grupo2:[Mateo, Roberto,Valentino, Francesca]
Diferencia de fuerza de trabajo:0

INTERACICONES:1005460
Grupo1:[Juan, Roberto, Camila, Francisco, Benjamín, Mateo, Delfina, Catalina,Emilia]
Grupo2:[Benicio, Valentino, Olivia, Martina, Joaquín, Bautista, Francesca, Santino, Ignacio, Andrea, Elena]
Diferencia de fuerza de trabajo:1
----------=Fin=----------

PODA FINAL:

if(!(Math.*abs*(g1.getFuerzaTotal()-g2.getFuerzaTotal())-fuerzaDeTrabajoEmpleadoActual>contadorFuerzaMaxima/2)){

---------=Inicio=----------
ITERACIONES:57
Grupo1:[Juan, Roberto, Mateo] Grupo2:[Camila, Francisco, Benjamín]
Diferencia de fuerza de trabajo:1

ITERACIONES:70
Grupo1:[Delfina, Emilia] Grupo2:[Francisco, Catalina]
Diferencia de fuerza de trabajo:36

ITERACIONES:83
Grupo1:[Andrea, Joaquín] Grupo2:[Martina, Olivia]
Diferencia de fuerza de trabajo:2

ITERACIONES:108
Grupo1:[Delfina, Elena] Grupo2:[Catalina, Camila, Francesca]
Diferencia de fuerza de trabajo:16

ITERACIONES:155
Grupo1:[Mateo, Emilia, Francesca, Valentino] Grupo2:[Santino, Joaquín]
Diferencia de fuerza de trabajo:1

ITERACIONES:266
Grupo1:[Santino, Roberto, Francisco] Grupo2:[Andrea, Elena, Martina, Emilia]
Diferencia de fuerza de trabajo:1

ITERACIONES:489
Grupo1:[Martina, Bautista, Ignacio, Benicio] Grupo2:[Mateo, Roberto, Valentino, Francesca]
Diferencia de fuerza de trabajo:0

ITERACIONES:1027826
Grupo1:[Juan, Roberto, Camila, Francisco, Benjamín, Mateo, Delfina, Catalina, Emilia] Grupo2:[Benicio, Valentino, Olivia, Martina, Joaquín, Bautista, Francesca, Santino, Ignacio, Andrea, Elena]
Diferencia de fuerza de trabajo:1
----------=Fin=----------

Esto reafirma lo mencionado en el punto anterior, que si bien, tiene un mayor cantidad de iteraciones y los resultados son los mismos esta poda contempla las variaciones que la poda anterior no incluía.