PROJECTE DE PROGRAMACIÓ

SEGONA ENTREGA

Grup 16

05 de Maig del 2014

Professora:

Alicia Ageno

Alumnes:

Marc Muntanyola Pros

Alessio Linares Bernardini

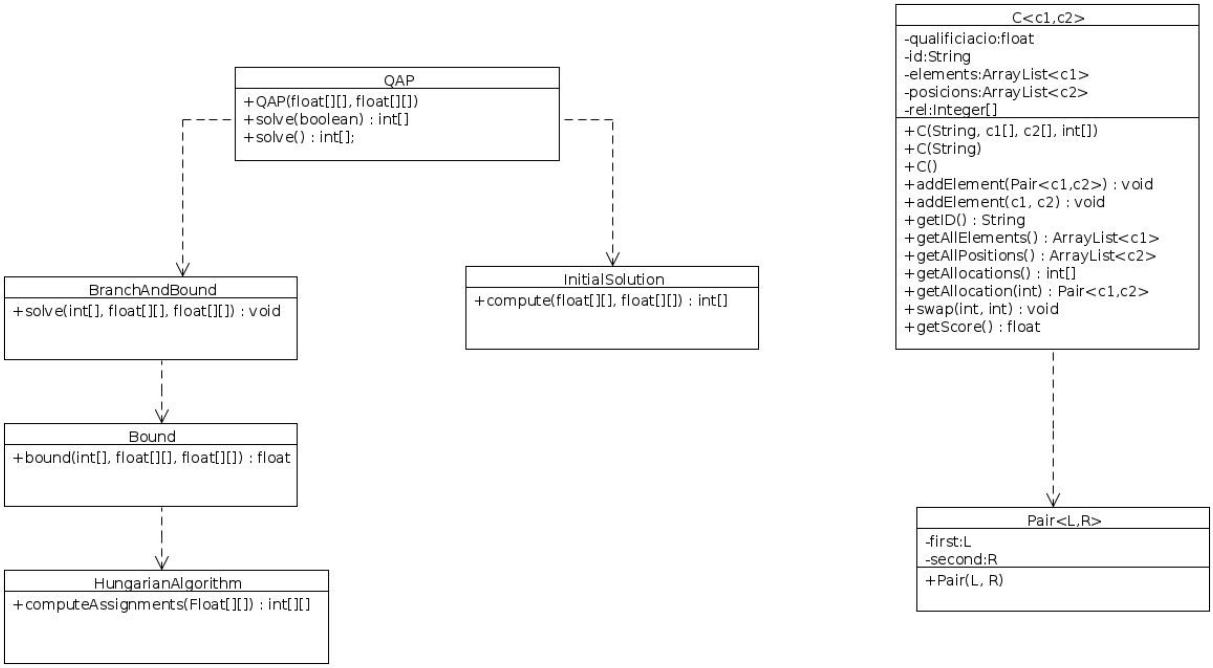
Samuel Bryan Pierno Alfonsín

Albert Trelis Saiz

Índex

1. Part comuna del clauster:

1.1 Diagrama estàtic complet de les classes compartides:



1.2. Normes de programació:

· Nom de les classes: primera lletra en majúscula i, si està composat per diverses paraules, la primera de cada paraula també en majúscula.
· Nom funcions: primera lletra en minúscula i, si està composat per diverses paraules, la primera de cada paraula en majúscula.
· Nom atributs: primera lletra en minúscula i, si està composat per diverses paraules, la primera de cada paraula en majúscula.
· Identació amb dos espais.
· Obrir cortxet a la mateixa linia separat per un espai.
· format de comentaris al codi:

```
/**
* [bound description]
* @param  a [description]
* @param  b [description]
* @param  c [description]
* @return   [description]
*/
```

· Una sentència per linía.
· Límit de caràcters per línia: 80

1.3 Especificació detallada de les classes compartides, de fitxers i sistemes de directoris compartits, de formats d'entrada/sortida compartits:

1.3.1 Classe Bound:

Classe que conté els mètodes necessaris per calcular la cota mínima de la solució parcial d'un QAP(A, B) que se li passa.

· bound:
Funció principal de la classe. Aquesta rep la solució parcial i les matrius de costos.

· removeDiagonal:
Extreu la diagonal (i,i) de A i la guarda a v com a vector.

· dotProduct:
Producte escalar entre els vectors u i v.

· diagonalsProduct:
Multiplica els vectors u i v, el primer horitzontal i el segon vertical, quedant de resultat una matriu.

· add:
Suma de dos matrius.

· calculate:
Funció que calcula el cost de les assignacions a la solució parcial i guarda a elements i positions les facilitats i localitzacions, respectivament, que no estan assignades.

· removeUnneededRowsAndColumns:
Retorna una matriu que contè de m només les files i columnes que indica v.

1.3.2 Classe Branch and Bound:

Aquesta classe usa l'algorisme "Branch and Bound" que consisteix en interpertar com un àrbre de solucions, on cada branca ens porta a una possible solució. Aquesta classe usa també la classe "Bound" per tal de detectar quines branques no segueixen sent òptimes i per tant, es poden "podar" fent d'aquesta manera que l'algorisme sigui més eficient.

· solve:
Única funció que donades unes matrius amb els costos i les distàncies entre elements, aplica l'algorisme "Branch and Bound" i crea solució que guarda en un vector d'enters.

· class Node:

Aquesta classes represent un Node en l'arbre de cerca de l'algorisme Branch and Bound. És privada ja que el seu ús és únic i intrínsec de la classe.

1.3.3 Classe C:

Aquesta és la classe que s'encarrega de guardar totes les dades necessàries una vegada s'han calculat les distàncies i posicions dels elements.

· C:

Constructora que pot inicialitzar els atributs amb valors buits o amb valors que se li han passat després de fer els càlculs.

· addElement:

Afageix un element i actualitza totes les dades necessàries.

· getID:

Retorna la id de la classe C.

· getAllElements:

Retorna tots els elements.

· getAllPostitions:

Retorna les posicions de tots els elements.

· getAllocations:

Retorna el conjunt de relacions entre elements i posicions que conté C. També pot retornar la relació d'un element en concret.

· getAllocation:

Es retorna una parella amb l'element en qüestió i la posició on està.

· swap:

Intercanvia la posició entre dos elements.

· getScore:

Retorna el cost actual del conjunt.

1.3.4 Classe HungarianAlgorithm:

Implementa una solució pel problema d'assignació en temps polinómic.

· computeAssignments:
Funció principal de la classe. Aquesta espera una matriu nxn de costos i torna les assignacions (fila, columna) la suma de les quals és mínima.

· allAreCovered:
Comprova si ja s'han tractat totes les columnes de la matriu.

· reduceMatrix:
Per cada fila de la matriu, troba el valor mínim i li resta als altres de la mateixa fila, fa el mateix per les columnes.

· initStars:
Guarda les files i columnes que al menys tenen un zero, si una columna/fila té un zero, pero la fila/columna on es troba ja en té un, s'ignora.

· coverColumnsOfStarredZeroes:
Marca, a coveredCols, les columnes en les que ja s'ha trobat algun 0, guardades a starsByCol.

· primeSomeUncoveredZero:
Troba un 0 que no hagi sigut marcat encara.

· incrementSetOfStarredZeroes:
Troba més 0s a les columnes que no han estat marcades encara.

· makeMoreZeroes:
Fa més 0s restant el següent valor més petit de les files/columnes.

1.3.5 Classe InitialSolution:

Aquesta classe només te una funció que ordena els costos de menor a major i assigna els caracters amb costos menors a les tecles amb distancies menors.

· compute:
Funció que retorna una solució inicial amb un resultat no òptim.

1.3.6Classe Pair:

Classe que té com a objectiu contenir una parella de valors. Els components en són dos, “first” que és el primer i “second” que és el segon.

· pair:
Crea una estructura “pair” donats dos valors.

1.3.7 Classe QAP:

Aquesta classe s'encarrega de resoldre el problema QAP (Problema de l'Assignació Quadràtica) usant l'algorisme Branch and Bound.

· QAP:
Inicialitza la classe actualitzant les matrius de costos i distàncies.

· solve:
Crida a un dels dos algorismes per resoldre el problema: Branch and Bound o a Taboo Search.

· callBranch:
Donada una solució inicial, crida l'algorisme Branch and Bound.

· callAprox:
Donada una solució inicial, crida l'algorisme Taboo Search.

1.3.8 Classe TS:

Aquesta classe usa l'algorisme Taboo Search que és un mètode d'optimització matemàtica que consisteix en augmentar el rendiment de la cerca local a través de l'ús d'estructures de memòria.

· solve:
Retorna en un vector el conjunt dels elements en ordre.

· funcionObjetivo:
Retorna el cost de la permutació.

· costeMovimiento:
Retorna el cost d'un moviment. Com més petit sigui el cost, millor haurà estat el moviment.

· valorarMovimiento:
Donat dos valors, determina quin és el moviment més òptim.

1.4 Especificació de les classes compartides:

1.4.1 Classe “Bound”:

```
class Bound {
/**
         * Funció principal de la classe. Aquesta rep la solució parcial i les
         * matrius de costos.
         * @param  partial_solution solució parcial per la qual s'ha de calcular
         *                          la cota mínima.
         * @param  distances        matriu de distancies entre localitzacions.
         * @param  costs             matriu de pes entre facilitats.
         * @return                  cota mínima del cost de la solució parcial donada.
         */

        public static float bound (int[ ] partial_solution, float[ ][ ] distances, float[ ][ ] costs)
}
```

1.4.2 Classe “Solució Inicial”:

```
public class InitialSolution{
        Pre: costs.length == distances.length
        Post: Aquest mètode retorna la llista d'assignaments inicial
        public static int[ ] compute(float[ ][ ] costs, float[ ][ ] distances)
}
```

1.4.3 Classe “QAP”:

```
public class QAP {
        float[ ][ ] costs;
        float[ ][ ] distances;

        /**
        * @param costs
         * @param distances
         */
        public QAP(float[ ][ ] costs, float[ ][ ] distances)

         /**
         * @param force_branch_and_bound
         * @return
         */
        public int[ ] solve(boolean force_branch_and_bound)

        /**
           * @return
        */
        public int[ ] solve()
}
```

1.4.4 Classe “Branch And Bound”:

```
class BranchAndBound {
  /**
  * @param solution
  * @param distances
  * @param costs
  */
public static void solve(int[ ] solution, float[ ][ ] distances, float[ ][ ] costs)
}
```

1.4.5 Classe “Pair”:

```
public class Pair<L, R> {
        public L first
        public R second

        /**
         * @param first
         * @param second
        */
        public Pair(L first, R second)
}
```

1.4.6 Classe “C”:

```
public class C<c1, c2> {
        protected float qualificacio;
        protected String id;
        protected ArrayList<c1> elements;
        protected ArrayList<c2> posicions;
        protected Integer[ ] rel;


        /**
         * @param elements
         * @param posicions
         * @param rel
        */
        public C(String id, c1[ ] elements, c2[ ] posicions, int[ ] rel)

        /**
         * @param id
        */
        public C(String id)

        public C()
```

```
        /**
         * @param element
         * @param posicio
        */
        public void addElement(c1 element, c2 posicio)

        /**
         * @param p
        */
        public void addElement(Pair<c1, c2> a)

        /**
         * @return elements
        */
        public E[ ] getAllElements()

        /**
         * @return id
        */
        public getID()


        /**
         * @return posicions
        */
        public c1[ ] getAllPositions()

        /**
         * @return rel
        */
        public int[ ] getAllocations()

        /**
         * @param id
        * @return new Pair<>(element, posicio)
        */
        public Pair<E, P> getAllocation(int id)

        /**
         * @return qualificacio
        */
        public float getScore()
}
```

1.4.7 Classe “Taboo Search”:

```
public class TS {
        /**
         * @param solution
        * @param distances
        * @param costs
        * @return solució el conjunt dels elements en ordre
        */
        public static int[] solve(int[ ] solution, float[ ][ ] distances, float[ ][ ] costs)
}
```

1.5 Repartiment de les responsabilitats de les classes compartides entre el clúster

Grup Keyboard (grup 6.1):

- Classe Bound
- Classe InitialSolution

Grup Planetes (grup 6.2):

- Classe QAP
- Classe BranchAndBound

Grup Llibres (grup 6.3):

- Classe Pair
- Classe C
- Classe TS

2. Part específica:

2.1 Estructures de dades

2.1.1 Alphabet

Aquesta classe s'encarrega de gestionar els caràcters que conté l'alfabet que s'usarà per calcular la solució final. Per això, a part d'un string que serà el nom d'aquest, usa un ArrayList per emmagatzemar els caràcters.

2.1.2 Character

Aquesta classe només conté un string amb el nom del caràcter. La seva única funció és la creació de caràcters individuals.

2.1.3 CharacterSet

La funció de la classe CharacterSet és per una banda, guardar tots els caràcters i les possibles combinacions entre ells. I per altra banda, calcular la freqüència d'aparició d'aquests en un text o donat un fitxer amb un cert format.

2.1.4 Keyboard

Aquesta classe serà l'encarregada d'emmagatzemar tota l'informació i càlculs que han fet les altes classes per poder crear correctament el teclat. A més a més, a part dels típics atributs com el nom, la mida o les opcions de configuració que ha escollit l'usuari (mode d'ús i forma), també guarda en un vector les rutes dels fitxer de text que l'usuari ha introduït per calcular la freqüència dels caràcters.

2.1.5 Position

Aquesta classe s'encarrega de guardar la x i la y d'una posició donada dos floats.

2.1.6 PositionSet

La funció principal de PositionSet és distribuir totes les tecles en un teclat depenent de la forma que l'usuari ha escollit. D'aquesta manera, pot calcular no només la posició que ocuparà cada tecla, sinó també la distància que tindrà cada tecla amb totes les altres, usant vectors com a mètode d'emmagatzament.

2.1.7 Relation

S'encarrega de gestionar les relacions entre tecles, és a dir, la distància que té una tecla amb totes les altres, i les relacions entre caràcters, és a dir, la freqüència amb la que estan un al costat de l'atre.

2.2 Algorismes

2.2.1 Branch&Bound:

L'objectiu principal és trobar el valor mínim d'una funció f(x) on fixem x rangs sobre un determinat conjunt S de possibles solucions.
L'idea clau de l'algorisme Branch and Bound és que si la menor branca per node (conjunt de candidats) A és major que la branca pare per un altre node B, llavors A ha de ser descartada de la cerca. Això és el que anomenem poda que s'encarrega de guardar un valor que permet fer el tall, tal que qualsevol node que tingui com a fill un valor més gran que la cota pot ser descartat.

Pseudocodi:

Sent:

· G(x) és la funció d'estimació de l'algorisme.

· P és la pila de possibles solucions.

· esFactible és la funció que es considera si la proposta és vàlida.

· esSolucio és la funció que comprova si es compleix l'objectiu.

· optim és el valor de la funció a optimitzar evaluat sobre la millor solució trobada fins el moment.

```
Funcion B&B {
 P = Fills(x,k)
 mentre( no buit(P) )
   x(k) = Extreure(P)
   if esFactible(x,k) y G(x,k) < optim
      si esSolucio(x)
          Guardar(x)
      else
          B&B(x,k+1)
```

2.2.2 Bound

Classe que conté els mètodes necessaris per calcular la cota mínima de la solució parcial d'un QAP(A, B) que se li passa, mitjançant el mètode de Gilmore-Lawler.

2.2.3 Hungarian Algorithm

L'algorisme modela un problema d'assignació com una matriu de costos nxm, on cada element reperesenta el cost d'assignar l'enèssima n al emèssim m. Per defecte, l'algorisme realitza la minimització dels elements de la matriu. Per això, com que usem un algorisme de minimització de costos, apliquem una eliminació de Gauss-Jordan per fer zeros. A més a més, a la matriu se li aplica un conjunt d'operacions que permetran conèixer amb millor eficacia el resultat final.

Algorisme:

1. Donada una matriu de costos C, es construeix C' trobant el valor mínim de cada fila i restant aquest valor a cada element de la fila.
2. Es troba el valor mínim de cada columna i es resta a cada element de la columna.
3. Determinem si per a totes les files, existeix una columna amb cost 0 que no ha sigut assignada a una altra fila.
4. Si totes les files tenen com a mínim una intersecció amb cost 0 que no ha sigut ocupada per una altra fila, estem en l'òptim. Per tant, l'algorisme acabarà.

Si no acaba es repeteix el següent bucle:

a. Considerant C', s'etiqueten les files que no han sigut acoplades o assignades per l'algorisme de trobar el màxim.

b. S'etiqueten a C' les columnes que tenen els zeros en correspondencia o assignades a les files etiquetades amb un * (o caràcter especial).

c. S'etiqueten les files que no han sigut etiquetades i acoplades o assignades per l'algorisme de cerca del màxim amb les columnes ja etiquetades amb un * (o caràcter especial).

d. Repetir els passos (b) i (c) fins que no es trobin més files o columnes per etiquetar.

e. Borrar les files no etiquetades i les columnes etiquetades. Per fer això es pot traçar una línia recta en les columnes i files borrades.

f. Sigui z un element de C' de valor mínim entre aquells costos no borrats en el pass anterior.

g. Restar z a cada element no borrat i sumar-lo als elements doblement borrats (on hi hagi intersseció o creuaments entre les línies marcades en el pas (e)).

h. Tornar al pas (c).

2.2.4 InitialSol

La InitialSolution calcula una solucó ràpida no òptima, però suficientment bona del QAP, per tal de començar el branch and bound amb una cota suficientment alta per podar.

2.2.5 TabuSearch

La cerca Taboo és un mètode d'optimització matemàtica que augmenta el rendiment del mètode d'una cerca local amb l'ús d'estructures de memoria: cada vegada que una potencial solució és determindada, es marca com a "taboo" de manera que l'algorisme no torni a visitar aquella possible solució.

Pseudocodi:

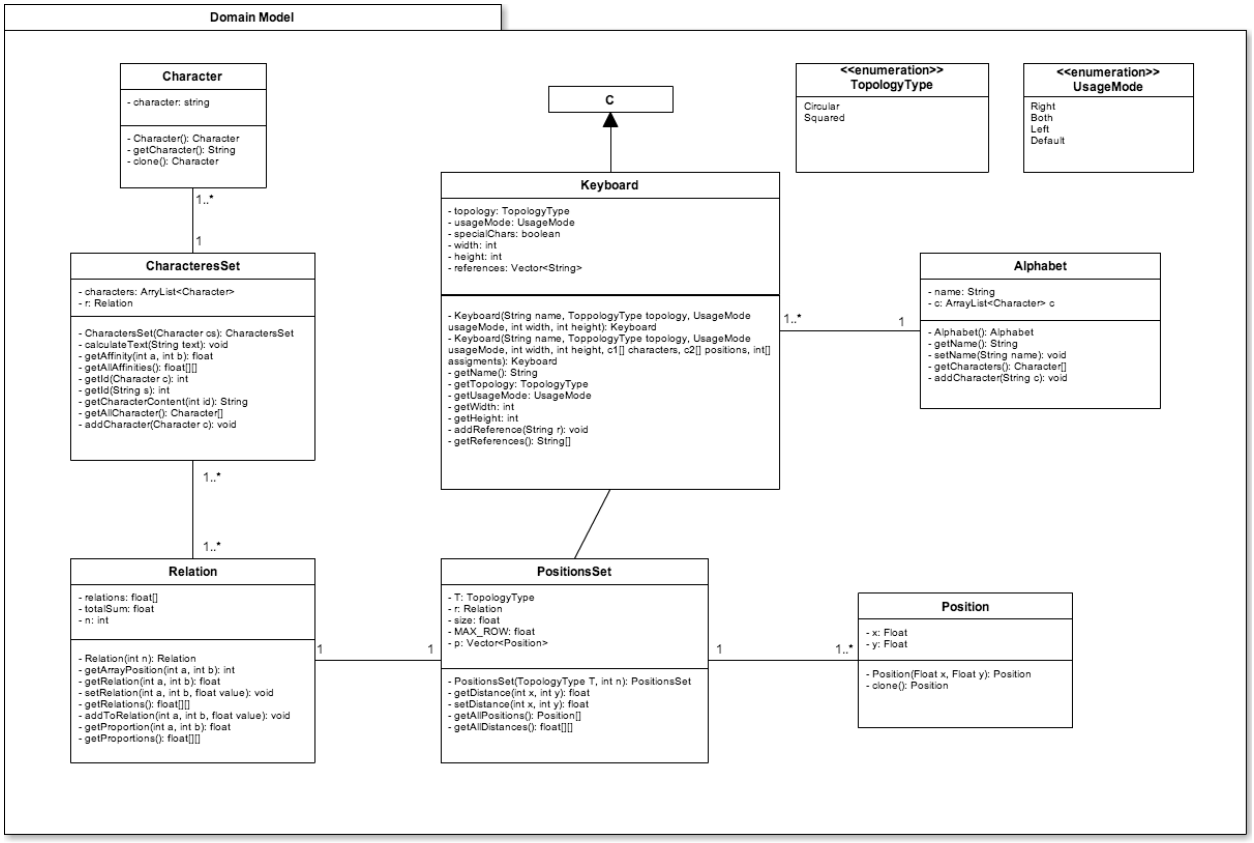
```
Procés Tabu Search
        c-sol-actual = Inicialitzacio
        llista-tabu = buida
        mentres (no es compleixi el criteri de parada) fer
                Seleccionar la millor c-solucio no tabu entre els veïns i guardar-la com a
                c-sol-actual
                Si (c-sol-actual és millor que millor-c-sol)
                        millor-c-sol = c-sol-actual
                llista-tabu = Actualitzar-llista-tabu
        Fi mentres
Fi del procés Tabu Search
```

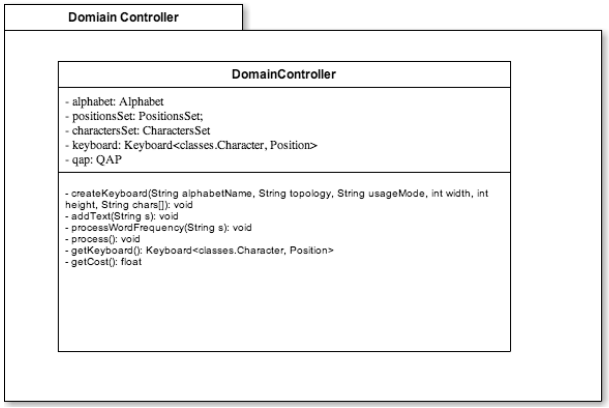
2.3 Relació de les classes implementades per cada membre del grup

Albert	Alessio	Marc	Samuel

Classes + Driver	Assignat
Alphabet	Albert
Character	Marc
CharactersSet	Marc
Keyboard	Samuel
Position	Albert
PositionsSet	Albert
Relation	Marc
InitialSolution	Samuel
Bound	Alessio
HungarianAlgorithm	Alessio
DomainController	Alessio
Main	Samuel
Enumerations	Samuel

2.4 Diagrama estàtic complet de les classes del domini





2.5 Disseny de totes les pantalles o llistats