# Cerca de Circuits Hamiltonians usant l'Algorisme de Roberts i Flores

Josep Roma Sau

**Resum –** Des de fa més de 150 anys, quan W.R Hamilton va plantejar el concepte dels circuits hamiltonians, fins a dia d'avui, s'han descobert criteris per demostrar si un graf conté circuits hamiltonians, s'han desenvolupat algorismes per tal de trobar-los tots, però l'alta complexitat temporal que comporta resoldre aquestes qüestions en grafs no trivials fa que continuï essent un problema d'actualitat.
Aquest projecte ha recopilat els principals criteris i tècniques per tal de desmostrar si un graf és hamiltonià, i també, ha desenvolupat un *software* per tal de trobar tots els circuits hamiltonians d'un graf utilitzant diferents algorismes, comparant els temps d'execució d'aquests i avaluant quina és la màxima complexitat que toleren utilitzant un ordinador convencional.

**Paraules clau–** Grafs, Circuits Hamiltonians, Roberts i Flores, Força Bruta

**Abstract–** For more than 150 years, When W.R Hamilton suggested the concept of Hamiltonian paths, until now, It has been discovered more judgements have proved that if a graph has Hamilitonian paths, as well as algorithms to get all the path's from the graph. However, because of the high temporal complexity that involves resolving these questions on a non trivial graph, It continues to be a problem still to be resolved.
This project has gathered together the principal criteria and techniques to prove if a graph is Hamiltonian as well as the development of a software to find all Hamiltonian paths in a graph using the suggested algorithms and evaluating how is the highest complexity that they accept.

**Keywords–** Graphs, Hamiltonian Circuits, Roberts i Flores, Brute Force

## 1 INTRODUCCIÓ

Un circuit hamiltonià és aquell que passa un sol cop per cada node d'un graf, exceptuant el primer, que també serà l'últim del circuit. Un graf és hamiltonià quan té un o més circuits hamiltonians. Hi ha molts grafs que, degut a les connexions dels nodes que presenten, no compleixen aquesta característica, per tant, no tots els grafs són hamiltonians.
Cal remarcar que en l'actualitat no es coneixen algorismes per tal de resoldre aquest problema en un temps polinòmic.

E-mail de contacte: Josep.RomaS@e-campus.uab.cat
Menció realitzada: Tecnologies de la Informació
Treball tutoritzat per: Quim Borges Ayats (UAB)

Actualment cercar circuits hamiltonians segueix essent un problema d'interès. A petita escala ho és per a les empreses de transport, suposant que mitjançant un graf es representa la xarxa de poblacions i carreteres a les quals donen servei, els pot ser molt útil per establir rutes de repartiment. Per exemple, utilitzant aquest tipus d'algorismes poden cercar rutes en les que els repartidors passin un cop per cada població/zona, aconseguint que a l'acabar tornin a la seu de l'empresa, i tot, recorrent el mínim de quilòmetres possibles i/o amb el mínim temps possible.
A gran escala es poden trobar treballs i competicions de prestigiosos enginyers i matemàtics per veure qui pot trobar el circuit hamiltonià més òptim en el conjunt de ciutats i carreteres més gran possible.
Per aquesta raó, aquest projecte té com a propòsit fer un petit recopilatori de les tècniques que existeixen

per tal de resoldre aquest problema, així com també desenvolupar un *software* enfocat a usuaris amb coneixements bàsics d'eines informàtiques, amb la finalitat de que puguin dibuixar qualsevol graf i observar com es determinen els circuits hamiltonians (si en conté). A més a més, aquest *software* s'utilitzarà per comparar dos algorismes de càlcul exacte de circuits hamiltonians.

## 1.1 Estructura del document

Al principi, es detallen els objectius del projecte i la metodologia de desenvolupament que s'ha utilitzat.
A continuació, s'expliquen alguns mètodes per identificar o descartar si un graf té un circuit hamiltonià, i seguidament, quins mètodes existeixen per tal de trobar circuits hamiltonians. En aquesta part s'expliquen detalladament els algorsismes implementats en el *software* desenvolupat, a més d'altres tècniques de cerca de circuits hamiltonians. També es fa un incís i breu explicació del problema *The Travelling Salesman Problem*.
Més endavant, es poden veure els resultats obtinguts pels dos algorismes i finalment les conclusions, juntament amb la bibliografia.
Al final del document, s'hi pot trobar un apartat de com millorar el programa desenvolupat, i un apèndix amb més informació sobre el programa i el seu desenvolupament.

## 1.2 Objectius

L'objectiu principal d'aquest treball és comparar els temps d'execució de dos algorismes per tal de trobar tots els circuits hamiltonians que té un graf, així com també trobar el circuit amb el cost més òptim de tots (en el supòsit que n'hi hagi més d'un).
Per acomplir aquest objectiu principal s'han determinat diferents metes o requisits per tal d'acotar el problema a resoldre i la manera de com fer-ho. Aquestes metes són les següents:

1. Desenvolupar (o trobar) un software per tal que un usuari amb coneixements bàsics d'informàtica pugui representar un graf.
2. Donar la possibilitat a l'usuari d'importar grafs així com també de poder-los guardar.
3. Implementar l'algorisme de Roberts i Flores de manera que, aquest, s'executi sobre el graf que l'usuari ha dibuixat en el moment que ell ho esculli.
4. Implementar l'algorisme de Permutacions Total de manera que aquest s'executi sobre el graf que l'usuari ha dibuixat en el moment que ell ho esculli.
5. Diferenciar d'una forma visualment clara el circuit hamiltonià amb el cost més òptim possible.
6. Comparar els temps d'execució dels dos algorismes.

## 1.3 Metodologia

Per afrontar el projecte s'ha utilitzat una metodologia de desenvolupament del *software* incremental [1], degut a que, al desenvolupar programari, és una metodologia pensada per afrontar els canvis que puguin sorgir durant el desenvolupament.
El projecte s'ha dividit principalment en dos fases, en què cadascuna ha estat formada per diferents iteracions. La primera fase incloïa purament el desenvolupament del software, mentre que la segona incloïa les proves, les conclusions i la documentació que es troba en aquest document.
La primera fase es va dividir en iteracions de llarga durada, en que a cada una s'hi incloïen múltiples punts a completar.
Durant la primera iteració es va desenvolupar la interfície gràfica. La representació gràfica dels nodes, les arestes i part del menú. D'aquesta manera durant les altres iteracions es podien resoldre problemes en l'àmbit gràfic i anar afegint millores, ja que, com es va preveure, aquesta ha estat una part costosa de desenvolupar.
La segona iteració incloïa el desenvolupament de la representació interna dels grafs, així com la possibilitat de guardar-los i importar-los des de l'ordinador.
En la tercera iteració es van desenvolupar els algorismes de cerca de circuits hamiltonians, i es van començar a realitzar proves amb la finalitat de comprovar el correcte funcionament del programa.
Finalment es va donar un llarg temps de contingència per tal d'afrontar les possibles adversitats sorgides, així com també per realitzar modificacions que fessin el software més agradable i senzill per a l'usuari.
Aquesta metodologia s'ha complementat amb l'establiment de dos repositoris remots, a més d'un repositori local, per tal de tenir diferents versions del programa i poder recuperar versions anteriors en cas d'errors de programació durant el desenvolupament.

# 2 Estat de l'Art

L'any 1857 William Rowan Hamilton va proposar l'*Icosian Game* [2]. La finalitat del joc era trobar les arestes d'un dodecaedre tals que es visitessin tots els vèrtexs només una vegada. Un camí com aquest ara es coneix com un circuit hamiltonià. Aquest joc es va arribar a vendre amb el nom *Around the World*, i en l'àmbit comercial, va ser un complet fracàs degut a que era massa senzill de resoldre.
El joc s'assemblava molt al problema proposat anys abans per L. Euler, que consistia en passar per totes les arestes d'un graf una sola vegada, tornant al punt inicial.

Encara que aquest tipus de circuits portin el nom de Hamilton, el concepte havia estat proposat dos anys abans per Thomas Kirkman, però com que no era tant reconegut com Hamilton, no va transcendir la seva aportació [3].
A més a més, Hamilton també va trobar un mètode per tal de resoldre circuits hamiltonians però només funciona pels sòlids platònics. Aquest mètode és una estructura algebraica amb la propietat associativa, però que no compleix la commutativa. Aquesta estructura és coneguda com l'*Icosian Calculus* [4].
Posteriorment es van postular diferents teoremes per tal de determinar de forma matemàtica si un graf és hamiltonià o no. Els més recents, i importants, ja que són molt generalistes i es poden aplicar a qualsevol tipus de graf daten de meitats del *s. XX*. Estan detallats més endavant en el document i van ser postulats per reconeguts matemàtics com G. A. Dirac [5], Ø. Ore [6], V. Chvátal [7] entre d'altres.
En aquella época també va sorgir un dels métodes més coneguts i "senzills"d'aplicar per tal de trobar els circuits hamiltonians d'un graf. Es tracta de l'algorisme de Roberts i Flores [8].
En el mateix periode de temps, es va començar a gestar la tècnica que avui en dia s'utilitza per tal de resoldre aquest tipus de problemes. Es tracta de la programció lineal, que encara que alguns dels fonaments havien estat descoberts durant els *s. XVIII* i *s. XIX*, té com a gran exponent G. Dantzig que l'any 1947 va descobrir el mètode *simplex* [9].
L'any 1979, M. Garey i D.S. Johnson van catalogar en el seu llibre el problema de trobar circuits hamiltonians en un graf com un cas de problema NP-Complet [10]. Des de que l'any 1954, G. Dantzig, R. Fulkerson i S. Johnson van trobar un circuit hamiltonià òptim en un conjunt de 49 ciutats, fins l'any 2006 que D. Espinoza, M. Goycoolea i K. Helsgaun entre d'altres van trobar un circuit hamiltonià òptim en un conjunt de 85 900 ciutats, només han passat 52 anys, però s'ha demostrat l'eficàcia de la programació lineal tenint en compte el creixement de la complexitat [11].

# 3 Cerca de Grafs Hamiltonians

Com s'ha mencionat anteriorment, trobar un circuit hamiltonià en un graf és un problema NP-Complet [10].
A més a més, determinar si un graf té algún circuit hamiltonià (ignorant quin/s) suposadament té la mateixa complexitat. És per això que, per tal de determinar si un graf és hamiltonià, molts cops la millor forma és directament buscar un circuit hamiltonià en el graf.
Tot i això, cal destacar que hi ha diferents criteris per tal d'assegurar o descartar si un graf és hamiltonià o no. Alguns d'aquests criteris estan resumits a continuació.

## 3.1 Grafs Complets

Els grafs complets de més de tres vèrtexs sempre seran hamiltonians, ja que, cada vèrtex té connexió directa amb tota la resta. A més a més, cal destacar que aquests grafs sempre contindran més circuits hamiltonians que els grafs amb el mateix número de vèrtexs que no siguin complets, ja que, totes les permutacions dels vèrtexs menys un vèrtex seran un circuit hamiltonià.

$$\#Circuits = (|V| - 1)! \tag{1}$$

Sigui *G(V,A)* un graf complet podem calcular el número de circuits hamiltonians diferents que obtindrem, partint sempre des de el mateix vèrtex i sempre que $|V| \geq 3$.

## 3.2 Grafs No-Complets

Sigui *G(V,A)* un graf. Per a tot vèrtex $x \in V$, denotem $\Gamma(x)$ el conjunt de successors de $x$.

- Si considerem *u* i *v* com a parella de vèrtexs no adjacents, i per a tota parella de vèrtexs no adjacents del graf es verifica:

  $$|\Gamma(u)| + |\Gamma(v)| \geq |V|, \tag{2}$$

  llavors *G* és hamiltonià.

- Sigui *G* connex amb $|V| > 2$. Si es verifica:

  $$|\Gamma(v)| \geq \frac{|V|}{2} \; \forall \, v \in V, \tag{3}$$

  llavors *G* és hamiltonià.

G.A Dirac, l'any 1951, va establir la desigualtat (3), coneguda com el Teorema de Dirac [5].
Uns anys més tard, l'any 1960, Ø. Ore va generalitzar la desigualtat (3) i a partir d'aquesta va postular el Teorma d'Ore (2) [6].
Finalment, l'any 1972, J. A. Bondy i V. Chvátal, a partir del Teorema de Dirac i el Teorema d'Ore van postular que un graf és hamiltonià si i només si el seu *closure graph* és hamiltonià [7].
En grafs dirigits, el *closure graph* és un subgraf induït amb arestes d'entrada des de l'exterior però sense arestes de sortida.

## 3.3 Altres

Sigui *G(V,A)* un graf, si no compleix alguna de les següents condicions no serà hamiltonià. Aquestes condicions es poden trobar, per exemple, a [12].

- *G* ha de ser connex. Si no és connex sempre hi haurà vèrtexs als quals no es podrà arribar i, per tant, no complirà la condició de poder passar per tots els vèrtexs, característica que el farà no hamiltonià.

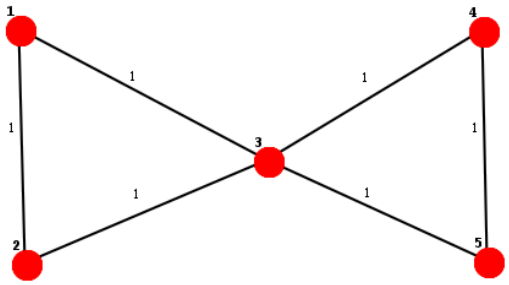


**Figura 1:** Exemple d'un graf amb una articulació al node 3.

- Tots els vèrtexs de $G$ han de ser de grau dos com a mínim. En cas de tenir un grau menor a dos, no podria ésser hamiltonià ja que per tal d'arribar al node i sortir d'ell s'hauria de repetir el node pel qual s'ha arribat.
  Un clar cas d'aquest tipus de graf són els arbres, les fulles dels quals tenen grau u i per tant, cap arbre és hamiltonià.
- $G$ ha de ser 2-connex, és a dir no pot tenir cap articulació (veure *Figura 1*). En cas contrari, com a mínim, s'hauria de passar dos cops per un mateix node diferent a l'inicial.
- Si $G$ és bipartit, considerant que $V1$ és una part i $V2$ l'altra, ha de complir la condició $|V1| = |V2|$.
- Sigui $G$ connex i pla que conté un circuit hamiltonià $H$, i siguin $R_j$ i $R'_j$ el nombre de regions de $G$ que queden dins i fora del circuit $H$, respectivament, i estan delimitades per $j$ arestes. Es verifica:

$$\sum_{j>2}(j-2)(R_j - R'_j) = 0 \qquad (4)$$

  Aquesta condició normalment s'utilitza per demostrar que un cert graf pla no conté circuits hamiltonians, ja que per tal que hi hagi circuits hamiltonians és necessari que es compleixi aquesta condició.

# 4 The Travelling Salesman Problem (TSP)

El TSP és un dels problemes més populars que s'han plantejat sobre grafs i optimització combinatòria. El problema tracta d'un viatjant que surt d'una població $X$ i ha de visitar diverses ciutats per vendre els seus productes. Per tal de fer-ho, el viatjant té dues alternatives:

- Trobar un recorregut amb origen i final a $X$ que visiti un únic cop cada ciutat i tingui un cost total mínim. Dit d'altra manera, trobar un circuit hamiltonià de cost mínim.
- Trobar un recorregut amb origen i final a $X$ que visiti com a mínim un cop cada ciutat i tingui un cost total mínim.

El cost de les arestes és generalment la distància o el temps de desplaçament, encara que pot tenir en compte altres factors com el recorregut més econòmic...
El problema és famós per la seva complexitat, ja que és un problema NP-Hard, això implica que és tant o més difícil de resoldre que trobar un circuit hamiltonià en el graf [11][13].

# 5 Cerca Exacta de Circuits Hamiltonians

En aquest apartat s'expliquen de forma detallada els algorismes que s'han implementat en el *software* desenvolupat. Aquests algorismes són mètodes exactes de cerca de circuits hamiltonians.
Aquests mètodes, en grafs amb molta quantitat de nodes i arestes es poden trobar amb explosions combinatòries tant grans que el temps de càlcul pot arribar a ser d'anys, per tant, per tal que siguin útils s'han d'utilitzar en grafs relativament petits.
Cal mencionar que en grafs molt grans aquests algorismes no són eficients. Per aquest motiu, una de les tècniques més utilitzades per resoldre problemes NP-Complets, com el TSP, és la programació lineal entera (PLE).
Aquesta metodologia es basa en calcular el nombre de casos possibles a tractar i, en eliminar les condicions d'integritat, obtenint en conseqüència un problema de programació lineal que pot ser resolt, per exemple, mitjançant l'algorisme *simplex* [9]. Dins d'aquestes metodologies cal destacar l'algorisme de poda i ramificació (*Branch and Bound*) ja que va ser un dels mètodes pioners.

## 5.1 Algorisme de Permutacions Total

Aquest algorisme és purament de força bruta, sense cap tipus "d'intel·ligència".
Consisteix a trobar totes les permutacions possibles dels vèrtexs del graf i, a continuació, comprova si les adjacències de cada parell de vèrtexs de la permutació existeixen en el graf i són directes, o sigui, que no s'ha de passar per cap altre vèrtex per anar d'un cap a l'altre.
El procediment és el següent:

1. Es guarda el primer vèrtex.
2. Es generen totes les permutacions possibles dels altres vèrtexs.
3. Per a cada permutació generada, s'afegeix el primer vèrtex al principi i al final de la permutació.

4. S'analitza cada permutació comprovant si correspon a un circuit hamiltonià.

La implementació de l'algorisme que genera totes les permutacions dels vèrtexs és recursiva, el seu funcionament és el següent. En primer lloc, guarda el primer node de la llista de nodes i seguidament l'elimina. A continuació, la funció fa el mateix per tots els elements de la llista a través de crides a ella mateixa.
Un cop la llista ha quedat buida, crea una llista de llistes de nodes i ho retorna a l'anterior crida.
La anterior crida, per cada subllista de la llista que ha estat retornada inserirà l'element que s'ha borrat en aquella crida a totes les posicions possibles.
Finalment, s'afegiràn aquestes subllistes a la llista principal i es retornarà a la anterior crida per tal d'inserir el següent element. Successivament es toranrà enrere recuperant l'element que s'ha guardat a cada crida i comletant les permutacions [14].

## 5.2 Algorisme de Roberts i Flores

L'algorisme de Roberts i Flores es basa en el *backtracking* retornant com a resultat tots els circuits hamiltonians del graf [8].
El que fa és analitzar les adjacències d'un vèrtex i escollir-ne una, i així va movent-se de vèrtex en vèrtex. En cas que en algun moment es trobi en un cul de sac, torna al vèrtex anterior (*backtracking*) i escull una adjacència que no hagi escollit previament.
Un altre aspecte important a destacar és que a mesura que va visitant els vèrtexs els guarda com a visitats, i així, s'aconsegueix que no se'n repeteixi cap excepte el primer.
Pel desenvolupament del software s'ha realitzat una implementació recursiva de l'algorisme (veure *Figura 3*) seguint les premisses essencials mencionades anteriorment.
En primer lloc se li especifica quin és el node inicial amb el qual ha de començar a treballar. El node inicial, també conté els seus veïns.
En segon lloc, l'algorisme processa la primera adjacència del node inicial afegint-lo a una llista. Després, es comprova que aquesta llista no tingui tots els vèrtexs, ja que en aquest cas ja hauríem trobat el circuit hamiltonià.
Seguidament, es comprova que aquest node no estigui visitat, en cas que ho estigui es treu de la llista i es comprova una altre adjacència (*backtracking*).
Finalment, si el node no s'ha visitat, s'afegeix a la llista temporal i es comproven les seves adjacències tornant a executar l'algorisme (recursivitat) [15].

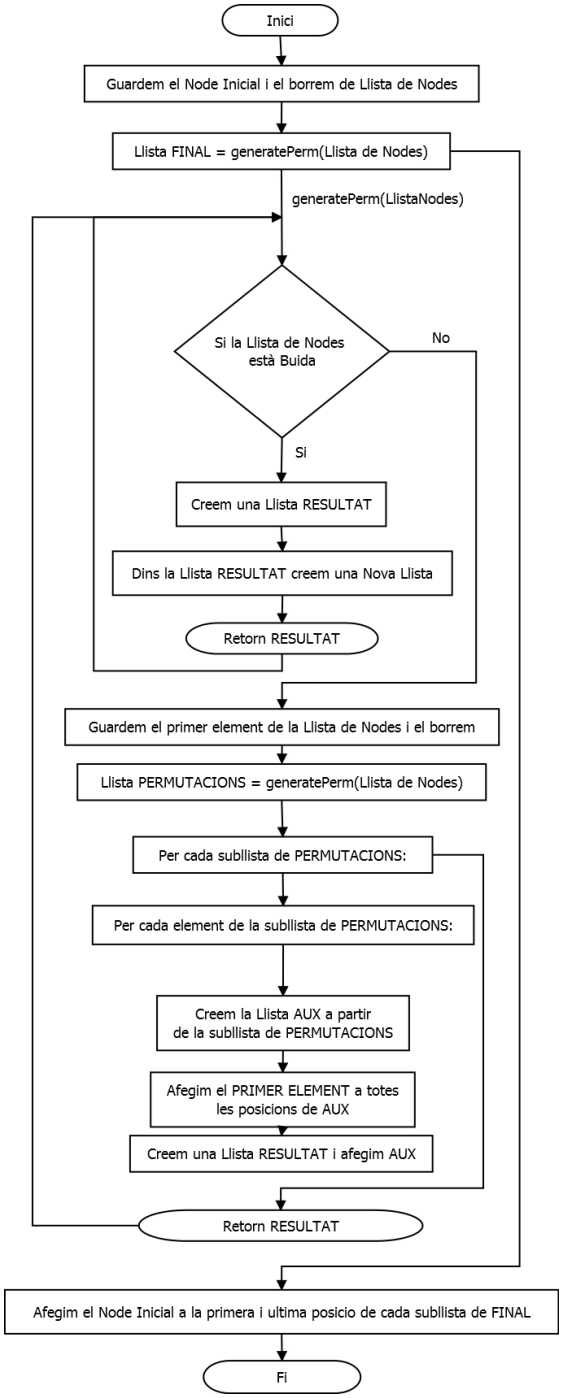


**Figura 2:** Implementació de l'algorisme de Permutacions Total utilitzant la recursivitat.

# 6 Cerca Aproximada de Circuits Hamiltonians

En grafs amb molta quantitat de nodes i arestes, en els que es pot prescindir d'una solució exacta, es po-

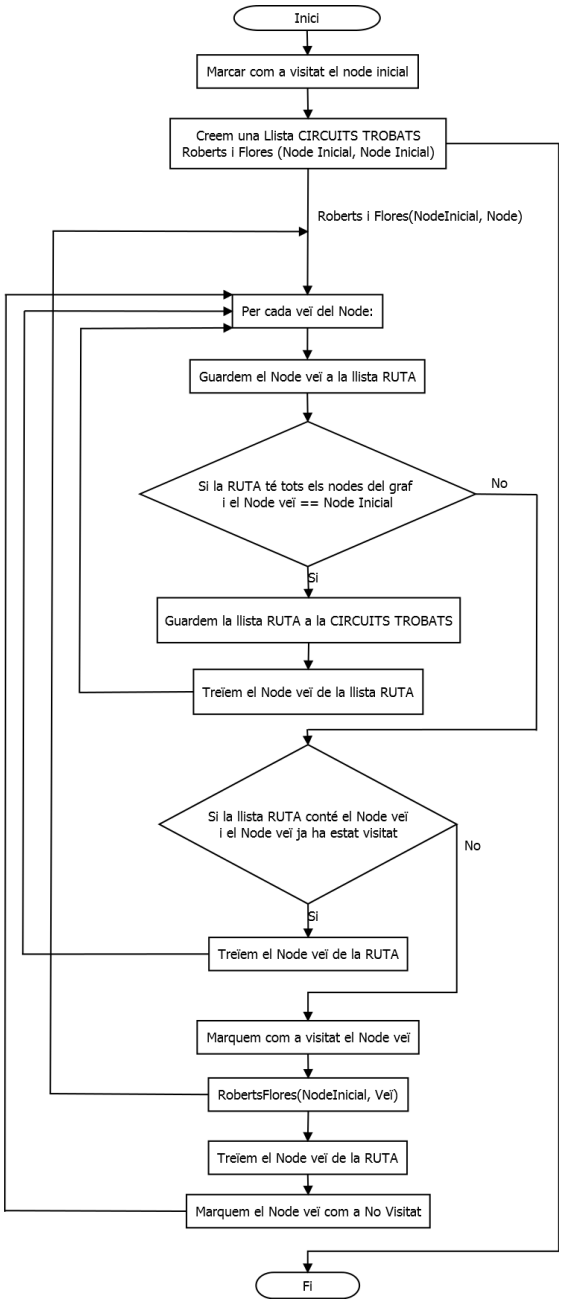


**Figura 3:** Implementació de l'algorisme de Roberts i Flores utilitzant la recursivitat.

den utilitzar mètodes heurístics per tal de trobar o calcular aproximadament el cost del circuit hamiltonià òptim.
Podem diferenciar dos tipus diferents d'heurístiques:

## 6.1 Heurístiques Específiques

Són heurístiques d'un únic pròposit, pensades per resoldre un problema concret, per exemple, trobar un circuit hamiltonià, aproximar el cost mínim d'un circuit hamiltonià... Dins d'aquest grup cal destacar els algorismes d'aproximació. Els algorsimes d'aproximació troben el cost aproximat del circuit hamiltonià òptim que conté el graf, però, a més a més, fiten l'error màxim que té aquesta aproximació [13].

## 6.2 Metaheurístiques

Són heurístiques de pròposit general, serveixen per intentar optimitzar multitud de problemes a partir d'un o més estats d'aquell problema.
S'en poden trobar una gran quantitat però solen ser menys òptimes que les heurístiques específiques.
Cal destacar:

- Optimtització Aleatòria
- Algorisme Voraç
- Algorismes Genètics
- Refredament Simulat
- Cerca Tabú

# 7 Conclusions

Utilitzant el programari desenvolupat, que compta amb les implementacions dels algorismes explicades anteriorment, s'han realitzat proves per determinar quina és la màxima complexitat que accepten i per comparar els temps d'execució dels dos algorsimes.

## 7.1 Metodologia de les Proves

Per fer les proves s'han generat grafs simètrics i s'han dividit en tres tipus diferents segons el seu nombre d'arestes.
El primer grup està format per grafs complets amb diferent número de vèrtexs. Mitjançant aquest grup s'ha determinat la màxima complexitat que toleren els algorsimes, ja que com s'ha mencionat anteriorment, els grafs complets de $n$ vèrtexs és el pitjor cas possible de qualsevol graf amb el mateix nombre $n$ de vèrtexs.
El segon grup està format per grafs amb el mateix nombre de vèrtexs que el primer grup, però amb un nombre molt més reduït d'arestes (els anomenarem Grafs Semi-Complets). El nombre d'arestes d'aquests grafs oscil·la entre el 55% i el 80% respecte

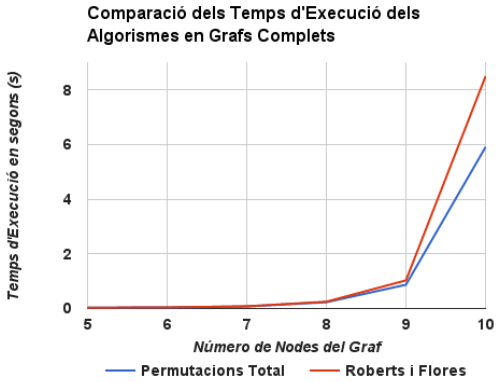


**Figura 4:** Comparació dels temps d'execució en Grafs Complets dels dos algorismes en funció del número de nodes.

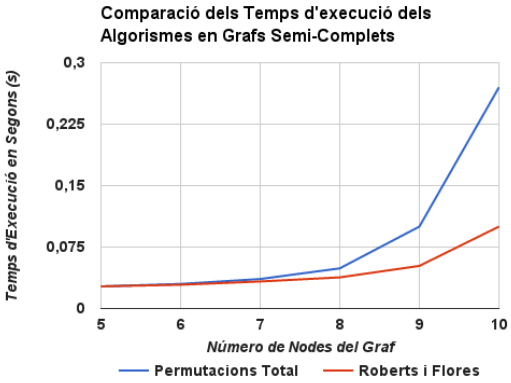


**Figura 5:** Comparació dels temps d'execució en Grafs Semi-complets dels dos algorismes en funció del número de nodes.

els grafs complets.
El tercer grup està format per grafs amb el mateix nombre de vèrtexs que el primer i el segon, però amb el número d'arestes mínim per tal que els grafs siguin hamiltonians.
Per arribar als resultats que es mostren a continuació, s'han realitzat fins a cinc execucions de cada graf amb cada algorisme.
Una vegada s'han obtingut els resultats de les cinc execucions s'ha fet una mitjana aritmètica i s'han confeccionat els gràfics (veure *Figura 4, Figura 5, Figura 6*).
També s'ha realitzat un altre conjunt de proves en grafs simètrics i dirigits per tal de comprovar el correcte funcionament del programa sense tenir en compte el temps d'execució[12].

## 7.2 Primer Grup de Proves: Grafs Complets

Com s'ha mencionat anteriorment aquest és el pitjor cas d'un graf de $n$ vèrtexs, ja que qualsevol altre graf amb la mateixa quantitat $n$ de vèrtexs tindrà menys circuits hamiltonians que un graf complet.
Es van començar a realitzar les proves amb aquest tipus de graf per tal de trobar el límit de nodes que accepten els algorismes, amb la finalitat de trobar tots els circuits hamiltonians en un temps raonable.
S'ha conclòs que el número màxim de nodes que accepten els algorismes és 10. Encara que el temps d'execució pugui semblar baix (veure *Figura 4*), el número de circuits trobats és 362 880. En el cas d'afegir un node més, el número de circuits a trobar seria de 3 628 800, 10 cops més. Aquest càlcul es pot realitzar en un temps que continua essent relativament baix, però és completament impossible escriure tots els circuits resultants en una sola finestra ja que la pila (*heap*) no té prou espaï per tal quantitat de dades.
També s'ha provat l'execució d'un graf complet de 12 nodes. Degut a la implementació recursiva dels algorismes, el número de crides és molt elevat i es dóna una sobrecarrega de la memòria (*GC Overhead*).
El fet que l'algorisme de Permutacions Total sigui més eficient, en aquest tipus de grafs, que el de Roberts i Flores és normal. Ja que el nombre de comparacions que ha de realitzar l'algorisme de Permutacions Total és menor que les que ha de realitzar el de Roberts i Flores.
En general, els temps d'execució són molt baixos però es pot apreciar clarament la corba exponencial a mesura que s'augmenta el número de nodes.

## 7.3 Segon Grup de Proves: Grafs Semi-Complets

Com es pot apreciar al gràfic (veure *Figura 5*), en general, el temps d'execució dels dos algorismes és molt més baix que en el cas de proves anterior, aquest fet és degut a que la quantitat de circuits hamiltonians continua essent elevada però és molt menor que en el cas dels grafs complets.
A més a més, al haver-hi menys circuits, s'han de fer menys comprovacions per trobar el de cost òptim.
En el cas de l'algorisme de Permutacions Total, també s'ha de tenir en compte que per cada permutació comprova les adjacències mentre existeixin. En cas que detecti que una adjacència no existeix, no comprovarà les restants de la mateixa permutació, ja que aquella permutació segur que no serà un circuit hamiltonià. Per tant és normal que el temps d'execució sigui més baix respecte els casos de prova de Grafs Complets.
També es pot observar l'eficiència de l'algorisme de Roberts i Flores en casos en que els grafs no són complets. Mentre l'algorisme de Permutacions Total comprova cadascuna de les Permutacions possibles del circuit de la forma descrita anteriorment, el de Roberts i Flores busca directament els circuits des-

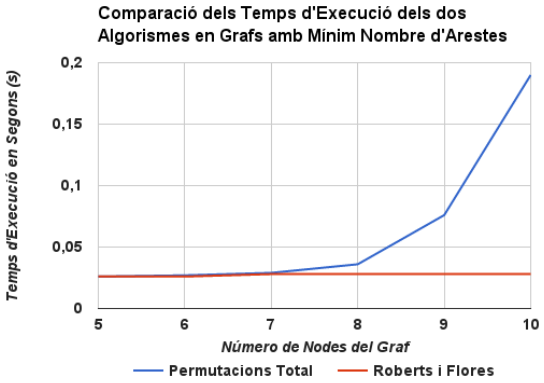


**Figura 6:** Comparació dels temps d'execució dels dos algorismes en funció del número de nodes en grafs amb el mínim número d'arestes.

cartant amb antelació totes les permutacions que no contenen cap circuit hamiltonià.
En cas de provar-ho amb grafs amb major nombre de vèrtexs, la diferència entre els dos temps d'execució s'anirà ampliant.

### 7.4 Tercer Grup de Proves: Grafs amb el Mínim Nombre d'Arestes

Com es pot apreciar a la gràfica (veure *Figura 6*), els temps d'execució dels dos algorismes són més baixos. Tot i així, el temps d'execució de l'algorisme de Permutacions Total ha descendit molt lleugerament en comparació el segon grup de proves, però en canvi, el temps d'execució de l'algorisme de Roberts i Flores ha baixat de forma dràstica.
Aquest fet és degut a que, mentre l'algorisme de Permutacions Total ha de realitzar totes les permutacions i comprovar-les per detectar si són un circuit hamiltonià, l'algorisme de Roberts i Flores, a partir de les adjacències troba els circuits hamiltonians directament i, en aquest cas només n'hi ha un, per tant el troba molt ràpidament.

En resum, es pot constatar que el límit del *software* desenvolupat és de 11 vèrtexs, però a la pràctica, si es volen visualitzar tots els circuits hamiltonians del graf és de 10 vèrtexs.
A partir del conjunts de proves executats, també es pot constatar que en cas que el graf no sigui complet l'algorisme de Roberts i Flores és molt més eficient que el de Permutacions Total.
És normal que com menys arestes i menys circuits tinguin els grafs el temps d'execució dels dos algorsimes es redueixi, ja que s'han de realitzar menys càlculs i comprovacions per tal de trobar tots els circuits i també per trobar el circuit òptim.
En el cas de l'algorisme de Permutacions Total, com menys arestes o circuits tingui el graf més es redueix el temps d'execució de l'algorisme ja que en el moment que comprova que una adjacència no compleix ja passa a comprovar la següent permutació.
També s'ha de destacar que, segurament, utilitzant implementacions iteratives molt òptimes dels dos algorismes, en comptes de recursives, els temps d'execució podrien arribar a ser lleugerament més baixos, i potser, es podria ampliar el límit que s'ha marcat en els grafs complets, però de forma molt lleugera també, ja que per cada node de més que té el graf la complexitat augmenta de forma exponencial.

## 8 Propostes de Millora

Es proposen les següents opcions de millora en futures implementacions o modificacions, ja que la part més costosa que és la representació gràfica i interna del graf entre d'altres ja està completada.

1. Es podria afegir algorismes senzills basats en programació lineal (com el *Branch and Bound*) per tal de comparar els resultats obtinguts amb aquests.

2. Es podrien afegir més algorismes que treballin sobre grafs de manera molt senzilla ja que la representació dels grafs, tant gràfica com interna està feta i separada de la implementació dels algorismes que ja s'ha fet i per tant només faltaria afegir una classe amb el nou algorisme.

3. Es podria fer una representació gràfica dels grafs utilitzant un framework (com *PaperJS*, *Raphaël*, *ProcessingJS*...) de manera que la interfície seria més fàcil de programar que l'actual i milloraria molt l'estètica de cara l'usuari. El còmput es podria seguir fent des de l'actual programa. A més a més, aquest fet té l'avantatge afegida que es podria posar a disposició de tothom des de qualsevol lloc.

4. Es podrien implementar versions no recursives dels algorsimes per tal de determinar si són més ràpides (i per tant més òptimes temporalment parlant).

## Referències

[1] Procesos Software [Online] https://procesosoftware.wikispaces.com/Modelo+Incremental [31/10/2016]

[2] Icosian Game [Online] http://mathworld.wolfram.com/IcosianGame.html [31/10/2016]

[3] N. L. Biggs, *T. P. Kirkman, Mathematician*, Bulletin of the London Mathematical Society, vol. 13, p. 97-120, 1981.

[4] W. R. Hamilton, *Account of the Icosian Calculus*, Proceedings of the Royal Irish Academy, vol. 6, p. 415-416, 1858.

[5] G. A. Dirac, *Some Theorems on Abstract Graphs*, Proceedings of the London Mathematical Society, vol. s3-2, p. 69–81, 1951.

[6] Ø. Ore, *Note on Hamilton Circuits*, The American Mathematical Monthly, vol. 67, p. 55, 1960.

[7] J. A. Bondy, V. Chvátal, *A method in graph theory*, Discrete Mathematics, vol. 15, p. 111-135, 1976.

[8] S. M. Roberts, B. Flores, *Systematic Generation of Hamiltonian Circuits*, Communications of the ACM, vol. 9, p. 690-694, 1966.

[9] G. Dantzig, *The Simplex Method for Linear Programming*, Computing in Science & Engineering, vol. 2, p. 29-31, 1947.

[10] M. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979.

[11] D. L. Applegate, R. E. Bixby, V. Chvátal *et al. The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2006.

[12] J. M. Basart, *Grafs: Fonaments i Algorismes*, Publicacions de la Universitat Autònoma de Barcelona, 1993.

[13] *Diapositives de Matemàtica Discreta*, Departament d'Enginyeria de la Informació i de les Comunicacions, Universitat Autònoma de Barcelona, 2014.

[14] Generating all possible Permutations [Online] http://stackoverflow.com/questions/10305153 [02/11/2016]

[15] Algoritmo Roberts e Flores [Online] http://respostas.guj.com.br/47865-algoritmo-roberts-e-flores---teoria-dos-grafos [15/10/2016]

# APÈNDIX

## 1 REQUISITS DEL SOFTWARE

Just a l'inici del projecte es va realitzar una reunió amb el tutor per tal de determinar els requisits i especificacions que havia de complir el *software* a desenvolupar.
Al final d'aquesta reunió es va concloure que el *software* havia de complir els següents requisits.

### 1.1 Requisits Funcionals

- S'han de poder representar grafs simètrics.
- S'han de poder representar grafs dirigits.
- S'han de poder importar grafs mitjaçant fitxers de text.
- S'han de poder calcular tots els circuits hamiltonians del graf quan l'usuari ho esculli. Un cop fet, s'han de mostrar tots els circuits hamiltonians existents.
- S'ha de mostrar el cost del circuit hamiltonià de cost mínim un cop s'hagin trobat els circuits hamiltonians del graf.
- S'ha de mostrar el temps que ha tardat el programa en trobar tots els circuits hamiltonians del graf i mostrar-lo a l'usuari.

### 1.2 Requisits No Funcionals

- El programa ha de comptar amb una interfície gràfica per tal de que l'usuari pugui realitzar totes les accions descrites.
- L'interfície gràfica ha de ser fàcilment entendible
- El programa ha de donar el màxim *feedback* possible a l'usuari de les accions que pot realitzar o dels errors que puguin succeir.
- Els grafs que l'usuari vol representar per pantalla s'han de mostrar a mesura que els va generant d'una forma visualment atractiva.
- S'ha d'implementar l'algorisme de Roberts i Flores per tal de cercar tots els circuits hamiltonians del graf.
- S'ha d'implementar un algorisme que generi totes les permutacions existents dels vèrtexs del graf i comprovi quines corresponen a un circuit hamiltonià (Algorisme de Permutacions Total).

## 2 CANVIS EN ELS REQUISITS A MIG DESENVOLUPAMENT

Un dels requisits és poder importar grafs en fitxers de text. Aquest s'ha hagut de modificar (arribant a un acord amb el tutor) degut a la següent problemàtica.
Si l'usuari el volgués modificar o crear un graf a partir d'un fitxer de text hauria de tenir en compte les coordenades en que posa cada vèrtex, que els vèrtexs estiguessin dins el rang representat pel panell de dibuix.
A més a més la quantitat de paràmetres a escriure per representar un graf seria molt gran.
En cas que el posicionament dels nodes l'hagués de fer el *software* automàticament el cost i el temps de desenvolupament seria extremadament elevat.
Per aquests motius, s'ha acordat amb el tutor donar la possibilitat a l'usuari de guardar els grafs que crea, i poder-los importar, però en un format que no fos editable o que es pogués crear sense el *software*. Un altre característica que no estava als requisits és la de poder esborrar vèrtexs. Per comoditat alhora de treballar amb el programa s'ha afegit aquesta funcionalitat, ja que en cas que no hi fos i l'usuari cometés un error hauria de tornar a dibuixar el graf sencer.

## 3 DISSENY DEL SOFTWARE

Per tal de desenvolupar el programari amb interfície gràfica s'ha utilitzat el patró de disseny *Model-View-Controller* (MVC).
S'ha considerat que aquest paradigma és el més òptim per tal de desenvolupar aquest *software* degut a la separació de dades que s'aconsegueix. La funcionalitat del programa es troba al model, mentre que el que veu l'usuari es troba a la vista. El controlador fa d'intermediari entre el model i la vista aixi com també comprovacions per a la detecció d'errors.

### 3.1 Decisions de Disseny

És important destacar algunes decisions de disseny per entendre millor i més fàcilment el funcionament del programa.

- S'ha decidit que el llengüatge de programació sigui Java per diferents motius. En primer lloc, Java gestiona la memòria de forma automàtica amb el *Garbage Collector*, el programador no s'ha de preocupar per reservar o alliberar memòria.
  En segon lloc, perquè el tutor, en una de les reunions em va donar un exemple d'un projecte d'anys anteriors en el que es podien dibuixar grafs. Aquest programa estava fet en Java, i per tant, com que era una bona referència es va decidir seguir el mateix camí.

- Tant els nodes com les arestes s'han representat de dos formes. Internament i de forma gràfica. S'ha portat a terme d'aquesta manera per separar les dades que es mostren a l'usuari de les que els algorismes treballen.
- La correspondècia entre les arestes gràfiques i les internes (ja siguin dirigides o simètriques) es duu a terme mitjançant els identificadors que es troben a ambdós objectes. Com que la creació es realitza en el mateix moment la correspondència és 1 a 1.
- Un cop s'hagi calculat els pitjors casos que toleren els algorismes del programa, es limitarà el nombre de nodes del graf a aquella quantitat de nodes per tal d'evitar errors o realitzar només còmput de circuits hamiltonians de forma "immediata".

## 4 Eines Utilitzades

En aquest apartat es citen les eines utilitzades per desenvolupar el *software*.

- NetBeans IDE (v8.1).
- easyUML (*plug-in* per NetBeans).
- Open Icon Library Standard (v0.11).
- Dia Diagram Editor.
- SourceTree + Git i Google Drive.

## 5 Material Adjunt

Juntament amb aquest document es pot trobar un CD amb els següents fitxers:

- Versió executable (*.jar*; per Windows 10) del *software* desenvolupat.
- Codi complet del *software* desenvolupat.
- Diagrama de Classes Complet del *software* desenvolupat.
- Manual d'Usuari del *software* desenvolupat.
- Conjunt de proves *Grafs Complets*.
- Conjunt de proves *Grafs Semi-Complets*.
- Conjunt de proves *Grafs Mínims*.
- Conjunt de proves *Grafs Diversos - Proves Cost*.

## 6 Proves

Malgrat no s'hagin pogut realitzar proves de caixa negra ni de caixa blanca s'ha realitzat *exploratory testing* en busca d'errors a l'aplicació.
L'última versió, que és l'única que s'adjunta, no ha presentat cap problemàtica.

## 7 Especificacions de l'Ordinador de Proves

L'ordinador que s'ha utilitzat per les proves és un Asus-A55V amb la memòria secundaria modificada.

- CPU: Intel i7 - 3036QM (Ivy Bridge; 22nm)
- RAM: 6GB (DDR3; 1600MHz)
- SSD Kingston V300 (250GB)