Universitat Autònoma de Barcelona

Departament d'Informàtica
Unitat de Combinatòria i
de Comunicació Digital

MÈTODES HEURÍSTICS PER AL PROBLEMA D'STEINER EN GRAFS

Memòria presentada per en
Pere Guitart i Colom per optar al grau de
Doctor Enginyer en Informàtica

Director de la Tesi :
Dr. Josep M. Basart i Muñoz

Bellaterra, setembre 1999

El sotasignant, Dr. Josep M. Basart i Muñoz, Professor del Departament d'Informàtica de la Universitat Autònoma de Barcelona
CERTIFICA:

Que la present memòria ha estat realitzada sota la seva direcció per en Pere Guitart i Colom

Bellaterra, setembre 1999

Signat: Dr. Josep M. Basart i Muñoz

Als meus pares
i al meu germà

Prefaci

Arribat el moment en què, després de l'esforç important i continuat dels darrers anys, es comença a fer realitat aquesta memòria, permeteu-me fer balanç i considerar alguns dels aspectes que, d'una manera moltes vegades fortuïta, han conduït la meva recerca en aquesta direcció, fructificant en la present tesi doctoral.

No he estat mai d'aquella gent privilegiada que segueixen un camí preestablert, clar, decidit. D'aquells que marquen el pas segurs d'arribar a bon port. El meu pas és discontinu, marcat per circumstàncies atzaroses com ara aquella que em va portar novament cap a la Universitat, crec que de manera positiva, després d'una breu experiència professional en l'empresa privada.

Quan ara miro enrere, però, em fa l'efecte com si tots els camins haguessin anat en la mateixa direcció: el meu interès pels grafs, l'optimització combinatòria i l'algorísmica; la dedicació docent centrada bàsicament en les assignatures *Grafs i Complexitat*, i *Algorismes i Programació*; la tesi doctoral *Els arbres de Steiner: obtenció i propietats* presentada pel meu amic i director Josep Maria Basart. És com si la trajectòria que porta cap a aquesta memòria fos d'allò més coherent, directa, sense entrebancs. Com si tots aquells moments de dubte en què un es troba perdut, temorós de que tot aquest esforç acabi en un nores, s'haguessin esvaït de sobte i tot plegat hagués fructificat en la present memòria, d'un caràcter clarament algorísmic i en la qual la teoria de grafs juga un paper fonamental.

Comparteixo aquella dita prou coneguda segons la qual la recerca està formada en un 1% d'inspiració i un 99% de transpiració. Hi afegiria, però, dos elements que crec prou importants: la intuïció i, sobretot, la sort.

La intuïció em va portar a modificar l'algorisme genètic del Dr. Henrik Esbensen

per al problema d'Steiner en Grafs. Encara ara se'm fa difícil d'explicar com vaig poder dedicar tant de temps i esforç per intentar millorar un algorisme els resultats del qual semblaven suficientment bons per a desanimar qualsevol. El propi Esbensen, en mostrar-li els primers experiments que apuntaven cap a una millora del seu algorisme, va preguntar-me sobre els motius que m'havien portat a intentar-ho. La veritat és que tenia indicis d'elements que, aparentment, podien millorar el seu algorisme. Per sobre de tot, però, hi havia la curiositat per arribar a entendre el perquè d'aquests resultats. Com era possible que aquests algorismes anomenats genètics, que intenten emular el model biològic d'evolució de les espècies, produïssin uns resultats que podien competir amb els dels millors mètodes coneguts per al problema.

La sort va fer que alguns d'aquests indicis es veiessin confirmats i, tot i la limitada concepció que en aquell moment teníem d'alguns aspectes de l'algorisme, els resultats experimentals fossin superiors als que havia obtingut l'algorisme d'Esbensen. Havíem posat la primera pedra d'un camí que tot just acabava de començar i que ha fructificat en la present tesi doctoral.

Agraïments

Agrair el suport que en tot moment he tingut d'en Josep M. Basart. Sense el seu ajut inestimable aquesta tesi no hauria estat possible.

El Prof. Henrik Esbensen, a més de donar-me consells valuosos, em va animar a continuar endavant i em va cedir molt amablement les rutines per a reduir la mida dels grafs que ell mateix havia programat.

El Prof. T. Koch va enviar-me un dels seus articles i em va resoldre dubtes. També vam intercanviar un article amb el Prof. S. Voβ, mentre que els professors J. Beasley, N. Kapsalis, J. Petit, B. Waxman i A. Zelikovsky m'han ajudat a resoldre qüestions puntuals que han anat sorgint.

Els suggeriments d'en Joan Serra, amb qui vam començar junts, i els d'en Quim Borges, pacient company de despatx, han estat sempre encertats. En Joan Borrell va reservar-me espai de disc per a fer els experiments després de donar-li força maldecaps. En Julià Minguillón i en Lluís Ribas m'han resolt no pocs dubtes de programació.

La Rosemary Thwaite i en Manel Guitart m'han ajudat en algunes de les presentacions en anglès.

La present memòria no pot ser vista com una cosa aïllada, sinó com el resultat d'un procés que va començar en el moment en què vaig incorporar-me a la Unitat de Combinatòria i Comunicació Digital. El suport i la disponibilitat de tots els seus membres ha estat constant al llarg d'aquests anys. En Josep Rifà, responsable últim de la meva entrada a la Unitat, i en Jaume Pujol, sempre disponible, m'han estimulat a superar-me en la feina. Amb la Mercè Villanueva, l'Albert Puiggené, en Sergi Robles, l'Inma Ortuño, l'Andreu Riera i en Jordi Pons sempre hi he pogut comptar. I també amb en Josep M. Arqués i en Toni Codina als quals, després de

compartir tants dinars, desitjo molta sort en la nova etapa professional que comencen.

Agrair, finalment, el suport incondicional dels meus pares, sobretot en aquests darrers mesos en què he tornat al niu familiar per tal de finalitzar la redacció d'aquesta memòria.

Índex

Prefaci v

Agraïments vii

1 Introducció 1

1.1 Objectius 7

1.2 Continguts 11

2 Precedents i situació actual en el PSG 13

2.1 Definicions, notació i propietats 15

2.2 El PS: origen i versions principals 18

2.2.1 El problema d'Steiner en el pla euclidià 19

2.2.2 El problema d'Steiner en la mètrica rectilínia 22

2.3 El PSG: origen i mètodes heurístics 24

2.3.1 Algorismes d'aproximació 25

2.3.2 Estratègies principals dels mètodes d'execució múltiple 27

2.4 Algorismes d'un pas 32

2.4.1 L'heurística del camí més curt (SPH) 33

2.4.2 L'heurística del graf distància (DNH) 34

2.4.3 L'heurística de Zelikowsky (ZH) 34

2.5 Mètodes competitius 35

2.5.1 Els mètodes repetitius de Winter i Smith 36

2.5.2 Els mètodes iteratius d'Alexander i Robins 37

2.5.3 Els algorismes de Duin i Voβ 38

2.5.4 L'algorisme genètic d'Esbensen (EGA) 39
2.5.5 Branch and Cut 40

3 Un mètode heurístic per al PSG **43**
3.1 Preliminars 44
3.2 *Hill-Climbing* SPH (HCSPH) 45
3.3 GBA: un mètode heurístic per al PSG 46
3.3.1 Reordenar els vèrtexs 46
3.3.2 Eliminar els no-terminals de grau dos 49
3.3.3 Sacsejar l'arbre 50
3.3.4 Donar prioritat als camins de més d'una aresta 52
3.3.5 GBA 53
3.4 Resultats experimentals 54
3.4.1 Anàlisi del rendiment 56
3.5 Conclusions 61

4 Un algorisme genètic per al PSG **63**
4.1 Algorismes genètics (AG) 65
4.1.1 Funcionament dels AG 66
4.2 Els AG per al PSG (AGPSG) 69
4.2.1 Els AG de Kapsalis *et al.* (KRGA) i d'Esbensen (EGA) 70
4.3 Línies estratègiques dels AGPSG 73
4.3.1 Heurístiques per al PSG: l'estratègia de recombinació 73
4.3.2 AGPSG: heurístiques de recombinació de vèrtexs 75
4.4 GBGA: un AG per al PSG 76
4.4.1 El descodificador 77
4.5 Generació de solucions de qualitat 78
4.5.1 El rendiment del descodificador 79
4.5.2 La reducció de l'espai de solucions accessibles 80
4.6 Selecció de vèrtexs de qualitat 85
4.7 El mecanisme de recombinació 86
4.8 Resultats experimentals 87

4.8.1 Anàlisi del rendiment 88
4.9 Conclusions 100

5 La SPH i la DNH **103**
5.1 Les estratègies de la SPH i de la DNH 107
5.1.1 DNH: heurística de camí 108
5.1.2 SPH: heurística d'arbre 109
5.1.3 SPH i DNH: heurístiques de vèrtexs 111
5.1.4 Comparant la SPH amb la DNH 112
5.2 La FRA de la SPH 116
5.3 Reduint la complexitat de la SPH 118
5.3.1 Definicions prèvies 119
5.3.2 L'algorisme 119
5.3.3 Implementació i complexitat 121
5.3.4 L'algorisme és una implementació de la SPH 126
5.3.5 Quasi-SPH (qSPH) 127
5.4 Noves variants de la SPH 129
5.4.1 Mètodes preliminars 131
5.4.2 Alternative-SPH (ASPH) 132
5.4.3 Improved-SPH (ISPH) 133
5.4.4 Generalized-SPH (GSPH) 135
5.4.5 Intercanvis de camins per arestes 136
5.5 Conclusions 138

6 Conclusions **141**
6.1 Futures línies de recerca 146

Bibliografia **149**

Capítol 1

Introducció

El *problema d'Steiner en grafs* (PSG) és una de les tres versions principals del *problema de l'arbre Steiner*, un dels clàssics d'optimització combinatòria amb una popularitat només comparable a la del famós *problema del viatjant*. Tant el PSG com les altres versions principals del problema d'Steiner són *intractables*, és a dir, no només no se'n coneixen algorismes amb complexitat polinomial, també anomenats *eficients*, que permetin obtenir la solució òptima, sinó que tot sembla indicar que no n'hi poden haver. Podríem dir que l'existència d'un algorisme eficient que retornés la solució òptima per a tota instància del PSG representaria una revolució tan gran en la informàtica i en la manera de concebre els algorismes, que tot fa pensar que difícilment pugui existir.

La present memòria cal situar-la dins d'aquest context de manca d'algorismes eficients per al PSG, que obliga a cercar alternatives que permetin donar resposta a les necessitats que es presenten en la pràctica. Les solucions que es proposen són, per tant, limitades i, de manera semblant a les de la majoria de problemes intractables, poden englobar-se en els quatre àmbits següents: els *mètodes exactes*, els *mètodes d'abast restringit*, les *tècniques complementàries* i els *mètodes heurístics*.

Els *mètodes exactes* són aquells que retornen una solució òptima per al problema. Pel que acabem de comentar, aquests mètodes no poden ser eficients i, per tant, només poden servir per a resoldre instàncies de mida relativament petita. La seva utilitat, però, va augmentant progressivament amb l'aparició d'algorismes cada cop

menys "ineficients" i amb l'increment espectacular de la potència dels ordinadors.

Els *mètodes d'abast restringit* obtenen, de manera eficient, la solució òptima per a un subconjunt d'instàncies del problema amb unes característiques determinades. Només seran d'utilitat, per tant, per a resoldre casos particulars. Per exemple, s'han proposat diversos algorismes eficients per a trobar un arbre generador de cost mínim en un graf i, també, per a trobar el camí de cost mínim entre dos vèrtexs. Tots dos problemes són casos particulars del PSG que es resolen de manera eficient. Aquests casos, però, representen un subconjunt molt reduït del total d'instàncies del problema.

L'objectiu de les *tècniques complementàries* és millorar l'efectivitat de la resta de mètodes més que no pas obtenir solucions per si mateixes. Per exemple, en el PSG s'utilitzen tècniques que redueixen prèviament la mida de les instàncies a resoldre. Aquestes *reduccions* [57] permeten millorar el temps d'execució dels diversos mètodes i, per tant, capaciten els mètodes exactes per a resoldre problemes de mida més gran. D'altres tècniques complementàries s'apliquen per a millorar una solució aproximada donada o són modificacions de detalls no especificats en l'algorisme que acostumen a millorar el seu rendiment en la pràctica.

Finalment, els *mètodes heurístics* permeten aconseguir una aproximació a la solució òptima en un temps raonable. Es basen moltes vegades en estratègies que acostumen a complir-se o, com a mínim, a no fallar del tot, en la majoria de casos. Moltes d'aquestes estratègies només poden obtenir-se a partir de l'observació directa, l'experimentació o, fins i tot, la intuïció. Tot plegat dificulta en gran mesura la comparació entre diferents algorismes heurístics.

Algorismes d'aproximació

Dins de les tècniques heurístiques cal destacar els *algorismes d'aproximació*, que són aquells que donen una certa garantia respecte la qualitat de la solució aproximada que retornen. Per exemple, un algorisme d'aproximació assegura que el cost de la solució aproximada no pot ser superior a, posem per cas, el doble del cost de la solució òptima. Aquesta relació entre el cost de la pitjor aproximació que pot retornar l'algorisme i el cost de la solució òptima s'anomena *fita de la raó d'aproximació* (FRA).

No tots els problemes intractables admeten algorismes d'aproximació. Per a alguns d'ells s'ha demostrat que l'existència d'aquests algorismes seria equiparable a l'existència de mètodes exactes amb complexitat polinomial i, per tant, sembla del tot improbable que n'hi hagi. Per a d'altres s'ha demostrat la possibilitat de generar algorismes d'aproximació amb una FRA tan ajustada a 1 com es vulgui. És a dir, per a aquests problemes sempre es pot trobar un algorisme de complexitat polinomial que garanteixi un percentatge d'error tan petit com calgui, sempre que aquest error sigui més gran que zero. Aquests algorismes es generen sistemàticament, formant els anomenats *esquemes d'aproximació en temps polinomial* (en anglès, *polynomial time approximation schemes* (PTAS)). El problema està en què, tot i ser polinomial, l'ordre de complexitat dels algorismes amb una FRA molt propera a 1 és suficientment gran per a què només puguin utilitzar-se, tal i com passa amb els mètodes exactes, per a instàncies de mida relativament petita.

El PSG no es troba en cap d'aquests dos extrems d'aproximabilitat, ja que, tot i admetre algorismes d'aproximació, sembla poc probable que tingui PTAS. L'existència d'aquests esquemes està novament condicionada a la d'algorismes eficients per al PSG, cosa del tot improbable. No resulta estrany, doncs, l'interès que hi ha en determinar el valor de la menor FRA per a la qual és possible trobar un algorisme d'aproximació per al PSG. Aquesta és, sense cap mena de dubtes, la qüestió oberta més important per al problema, i la principal causant d'una mena de competició per a trobar l'algorisme amb menor FRA. Al capdavant d'aquesta competició s'hi troba, de moment, el recent algorisme de Hougardy i Prömel [54] amb una FRA propera a 1,589, seguit de l'algorisme Karpinski i Zelikovsky [65] amb una FRA propera a 1,644, i de l'algorisme de Berman i Ramaiyer [12] amb una FRA propera a 1,734, tal i com han demostrat Brochers i Du [16].

Creiem important remarcar la relació entre la recerca d'algorismes d'aproximació per al PSG i la que es dóna per a d'altres problemes intractables [77]. El PSG forma part d'una classe de problemes anomenada *MAX-SNP*, definida per Papadimitriou i Yannakakis l'any 1989 [76]. A partir d'un algorisme d'aproximació per al PSG se'n pot obtenir un altre per a cadascun dels problemes dins d'aquesta classe, tot i que, malauradament, l'algorisme resultant no té per què conservar la mateixa FRA. De

forma recíproca, l'obtenció d'algorismes d'aproximació per a alguns dels problemes d'aquesta classe, en concret per a tots aquells que com el PSG estan en la categoria dels anomenats *MAX-SNP-complet*, permet obtenir nous algorismes d'aproximació per al PSG. Aquests fets incrementen l'interès que ja de per si desperten els algorismes d'aproximació dins dels mètodes heurístics.

Heurístiques modernes

Quan es parla de mètodes heurístics és habitual distingir els *mètodes clàssics*, fortament dependents del problema a resoldre, del conjunt de mètodes de més recent aparició concebuts per a un propòsit més general i, per tant, menys dependents del problema. Aquests mètodes reben la denominació d'*heurístiques modernes* [85] i, generalment, estan basats en models de processos biològics o físics. En formen part els *mètodes evolutius* [7] —com ara els algorismes genètics—, la *recerca tabú*, la *simulació termodinàmica* (*simulated annealing*, en anglès) i les *xarxes neuronals*.

La concepció inicial de les heurístiques modernes com a mètodes de propòsit general ha anat deixant pas a mètodes cada cop més dependents del problema, única manera que tenen de competir amb aquells mètodes dissenyats específicament per a resoldre un problema [94]. Un exemple clar d'aquesta tendència el formen les anomenades *meta-heurístiques* [75] , les quals poden considerar-se com a heurístiques modernes o, també, com a mètodes híbrids que barregen conceptes d'heurístiques clàssiques amb estratègies d'heurística moderna. En les meta-heurístiques l'estratègia moderna s'utilitza per a controlar un procés iteratiu en el qual les aproximacions són generades mitjançant una heurística clàssica. Aquesta última és la que incorpora a l'algorisme la informació dependent del problema.

En aquest treball, el nostre interès per les heurístiques modernes se centra en els algorismes genètics (AG), precursors dels mètodes evolutius. Aquests algorismes varen ser introduïts per Holland [52] l'any 1975 i es basen en el model darwinià d'evolució de les espècies. A grans trets, els AG parteixen d'un subconjunt de solucions aproximades —*població*— que s'intenta millorar progressivament a través d'un procés

iteratiu —*evolució*— en el qual es generen noves solucions combinant parelles de solucions existents —*reproducció*. El procés s'atura quan, després d'un nombre màxim relativament alt d'iteracions no s'ha produït cap millora. Com que per a obtenir solucions de bona qualitat són generalment necessaris un nombre molt alt d'iteracions, els AG acostumen a ser considerats lents en comparació amb heurístiques clàssiques que generen solucions de qualitat similar.

En les heurístiques clàssiques el comportament d'un mètode es justifica a partir de les línies estratègiques que han servit per a dissenyar-lo, les quals s'obtenen d'apreciacions del problema i, per tant, estan limitades per la seva intractabilitat. Els resultats que obtingui el mètode dependran de l'encert d'aquestes estratègies i, també, del seguiment d'aquestes per part de les instàncies de prova. D'aquí que d'aquests resultats se'n derivi informació que pot ser d'utilitat per a millorar els algorismes.

En les heurístiques modernes, en canvi, hi ha una certa predisposició a considerar que l'estratègia està associada, de forma implícita, al model emprat i, per tant, té poc a veure amb el problema a resoldre. Per exemple, s'intenta aplicar una estratègia basada en un model d'origen biològic per a resoldre un problema combinatòric. Si els resultats són satisfactoris, el mèrit sol atribuir-se a les suposades virtuts inherents en el model, mentre que, si no ho són, les causes solen atribuir-se a una manca de coneixement del model que en fa difícil la seva correcta utilització. En aquest cas, el dissenyador es pot trobar intentant confuses modificacions —intercanviant algunes funcions, modificant alguns paràmetres, etc.— amb l'esperança cega de poder millorar els resultats [22]. Com es pot veure, la informació relativa al problema que es deriva de tot aquest procés acostuma a ser limitada. És aquesta independència entre el model i el problema la principal responsable, des del nostre punt de vista, de l'eterna controvèrsia associada a les heurístiques modernes.

Valoració dels mètodes heurístics

En els mètodes exactes hi ha una certa unaminitat a l'hora de considerar la complexitat com a criteri determinant per a valorar la qualitat d'un algorisme. Tots els algorismes retornen la solució òptima i, per tant, aquell que tingui la menor complexitat, és a dir, aquell que vagi més ràpid, es considera millor. Aquesta unaminitat

es trenca quan es parla de mètodes heurístics en els quals, a més del factor de la complexitat, cal afegir-hi el de la qualitat de les solucions que retornen i, en el cas dels algorismes d'aproximació, el del valor de la FRA.

La importància que es dóna a cadascun dels factors es veu, a més a més, influenciada per la utilitat que es vulgui donar a l'algorisme. Per exemple, l'absència d'una FRA no sembla, des d'un punt de vista pràctic, un impediment massa gran per un mètode que tendeixi a donar bones aproximacions en la pràctica. I és que aquesta fita, que resulta a vegades molt difícil de demostrar, no té per què assegurar un millor resultat per a les instàncies específiques del problema que cal resoldre. Des d'un punt de vista més teòric, però, no sembla massa prudent la utilització de mètodes sense garanties quan hi ha alternatives que n'ofereixen. Tot plegat permet diferenciar, com passa en la majoria d'àrees, entre aquella recerca que sembla tenir una orientació més teòrica i aquella altra que sembla tenir-ne una de més aplicada.

En l'orientació teòrica s'inclouria, bàsicament, la demostració de FRA, la proposta de nous algorismes d'aproximació i la millora dels existents. Els algorismes solen ser *d'un pas*, en el sentit que només exploren una solució, i no acostumen a utilitzar tècniques complementàries per incrementar el rendiment. Des d'aquesta perspectiva teòrica, per exemple, seria molt ben valorat un algorisme d'aproximació que millorés la FRA de l'algorisme de Hougardy i Prömel, independentment de si la seva complexitat en fes poc viable la seva utilització en la pràctica.

Des d'una orientació més aplicada s'intenta aconseguir mètodes que donin resposta a les necessitats pràctiques, en què una aproximació a la solució és necessària. Aquests mètodes han de tenir una bona relació entre el temps d'execució i la qualitat de les solucions i, per tant, han d'explotar al màxim les possibilitats de millora: incorporant tècniques complementàries, combinant diversos algorismes, explorant un nombre relativament alt de solucions, etc. La principal dificultat des d'aquesta perspectiva es troba en determinar la qualitat d'una aproximació quan no es pot conèixer, per raons òbvies, el cost de la solució òptima. Aquesta dificultat porta a avaluar experimentalment els algorismes utilitzant instàncies de prova, per tal de predir-ne el comportament. Les conclusions a les que s'arriba, però, tenen una validesa relativa, degut especialment a la inevitable manca de representativitat dels grafs de prova.

Aquesta utilització d'instàncies de prova ha generat una mena de competició entre algorismes per veure quin és el que obté millors resultats experimentals per a un conjunt d'instàncies o *joc de proves* donat. Els jocs de prova estan formats per instàncies que passen a ser de domini públic després de ser utilitzades en algun article i faciliten la comparació amb els nous algorismes que puguin sorgir. La forma en què les instàncies han estat generades és diversa —aleatòriament, a partir de casos pràctics, etc.— i pot condicionar els resultats que produeixen els algorismes. Tot i això, els jocs de prova s'han convertit en un element quasi imprescindible per a comparar algorismes, degut especialment a la dificultat de trobar alternatives. En casos extrems, la utilització generalitzada d'un joc de proves, com podria ser la dels grafs de la *OR-library* [10], pot portar a promocionar aquells mètodes que s'ajusten més a les característiques de les instàncies de prova en detriment d'altres mètodes que podrien tenir un millor comportament en un àmbit més general [53].

És per això que la utilització de jocs de prova cal considerar-la com un element més, i no exclusiu, en la valoració dels mètodes heurístics. Aquesta exclusivitat es dóna moltes vegades en les heurístiques modernes, que utilitzen els experiments per ajustar el seu funcionament, i es valoren moltes vegades en funció dels resultats.

1.1 Objectius

La present memòria se centra en els mètodes heurístics per al PSG. Des d'un punt de vista general, té com a objectius l'estudi i la millora de mètodes heurístics existents, la proposta de nous mètodes i, també, l'obtenció de tècniques complementàries que puguin contribuir a una millora del rendiment. Contempla des de la vessant més pràctica, en la qual s'intentarà obtenir algorismes que puguin competir experimentalment amb els millors mètodes coneguts, fins a la més teòrica, en la qual s'estudiaran algorismes d'aproximació i les seves FRA. Abarca des de mètodes d'un pas, fins a mètodes iteratius, incloent els que combinen diversos algorismes. Considera, finalment, tant les heurístiques clàssiques com les modernes.

L'origen d'aquesta tesi cal situar-lo en l'interès per la *Teoria de Grafs* i l'*Algorísmica* i en la tesi doctoral *Els arbres de Steiner: obtenció i propietats* [9] presentada per

J.M. Basart l'any 1988. El seu punt de partida, però, es troba en l'EGA, l'algorisme genètic que va introduir H. Esbensen l'any 1995 [30].

A grans trets, es pot definir l'EGA com una meta-heurística en què un dels algorismes d'aproximació clàssics per al PSG s'encarrega de generar solucions en un procés iteratiu controlat per una estratègia d'algorisme genètic. L'algorisme d'aproximació utilitzat és l'*heurística del graf distància*, (en anglès, *distance network heurístic* (DNH)) proposada per diversos autors (veure secció 2.4.2). El nostre interès per l'EGA parteix dels resultats experimentals que obté aquest algorisme utilitzant els grafs de proves de la *OR-library* [10]. L'EGA se situa al capdavant dels algorismes heurístics pel que fa a la qualitat dels resultats però, a més a més, també resulta competitiu en temps d'execució. D'aquesta manera, l'EGA es converteix en un dels pocs AG capaços de competir empíricament amb mètodes clàssics específics per a un problema i, més encara, com un dels únics que poden competir en temps d'execució, aspecte aquest que és considerat un dels punts més febles dels AG.

Si bé l'EGA és una meta-heurística i, per tant, està dissenyada específicament per al PSG, conserva aspectes inherents a la majoria d'heurístiques modernes les quals en dificulten la seva valoració de manera independent als resultats experimentals que produeix. En altres paraules, no creiem que ningú s'hagués sorprès si els resultats experimentals produïts per l'EGA haguessin estat mediocres. De manera molt semblant, es fa molt difícil de justificar la impressionant millora que representa el rendiment de l'EGA respecte al de l'AG proposat per Kapsalis i Rayward-Smith (KRGA) [63], de característiques molt similars. Són tots aquests aspectes els que marquen els objectius inicials d'aquesta tesi i que podríem resumir en els quatre punts següents:

1. Determinar els elements clau que fan que l'EGA produeixi bons resultats.
2. Aclarir les causes del pobre rendiment del KRGA en comparació al de l'EGA.
3. Implementar un AG que millori els resultats experimentals de l'EGA.
4. Proposar un mètode no evolutiu que pugui competir amb l'EGA.

Quan es defineixen uns objectius de recerca sempre resulta difícil conèixer fins a quin punt aquests objectius poden ser portats a terme. En funció dels resultats,

s'anirà orientant la recerca en aquelles direccions en què s'intueixen més possibilitats i, al mateix temps, s'aniran redefinint els objectius. No és estrany doncs que, un cop desenvolupada part de la recerca, els objectius inicials tinguin una certa tendència a adaptar-se als resultats.

En el nostre cas, més que un canvi d'objectius, el que s'ha anat produint és un canvi d'orientació respecte la manera en què crèiem que aquests objectius podien acomplir-se. Aquest canvi podria resumir-se en el fet d'abandonar la idea que l'EGA obté bons resultats gràcies al model evolutiu, i passar a considerar-lo bàsicament com un mecanisme d'exploració d'un espai de solucions candidates. És a dir, si en un principi volíem superar els resultats de l'EGA modificant algunes de les seves funcions pròpies dels AG —l'encreuament, alguns paràmetres, etc.— més endavant volíem analitzar aquestes funcions des d'una perspectiva diferent per tal d'escollir la funció o la modificació més indicada en cada cas. Els resultats experimentals haurien de perdre pes en favor d'arguments justificant l'elecció de cadascuna de les parts i, a partir d'aquí, permetre deduccions d'un nivell més global. És per això que els objectius, tot i ser bàsicament els mateixos, poden ser reescrits d'una manera, creiem, més ambiciosa:

1. Justificar el funcionament de l'EGA com a mètode d'exploració d'un espai de solucions candidates basat en recombinar parts de solucions prèviament obtingudes.

2. Determinar la funció associada a cadascun dels elements que formen part de l'EGA i valorar-ne la seva utilitat d'una forma independent de la seva interpretació en clau genètica. (A partir d'aquí, haurien de sortir a la llum les causes del rendiment contraposat entre el KRGA i l'EGA.)

3. Proposar un AG que millori el funcionament de l'EGA. Aquestes millores haurien de quedar reflectides, també, en els resultats experimentals.

4. Dissenyar un nou mètode no evolutiu que tingui un bon rendiment.

Com es pot veure, tots aquests objectius estan estretament lligats a una correcta interpretació de la manera en què l'EGA obté els seus resultats. La possibilitat de

millorar l'EGA es desprèn, des del nostre punt de vista, de la dificultat d'argumentar alguns dels elements que utilitza. Cal veure, però, si aquests indicis de millora es confirmen i, per tant, es transformen finalment en millores reals.

L'exemple més clar d'aquesta dificultat el trobem a l'hora de justificar l'elecció de la DNH com a algorisme d'aproximació subordinat a l'estratègia genètica de l'EGA. Des del nostre punt de vista, aquesta elecció juga un paper determinant en els resultats de l'EGA, ja que l'heurística escollida és la màxima responsable d'introduir informació dependent del problema a l'algorisme. Entre d'altres possibles opcions, es podria haver escollit l'*heurística del camí més curt*, (en anglès, *Shortest path heuristic* (SPH)) introduïda per Takahashi i Matsuyama l'any 1980. La SPH, que té una complexitat més gran i una FRA similar a la de la DNH, acostuma a produir segons diversos autors millors resultats experimentals (veure la introducció del capítol 5).

Creiem important destacar el paper que la DNH i, en especial, la SPH tenen en aquest treball. Tots dos algorismes d'aproximació formen part de les *heurístiques de camí i distància*, (en anglès, *path-distance heuristics* [93]) i han estat utilitzades com a base d'un nombre relativament gran d'algorismes iteratius per al PSG (veure el capítol 5). La nostra predilecció per la SPH s'intueix clarament al llarg dels capítols centrals d'aquesta memòria, però resulta difícil de justificar d'una manera definitiva. D'aquí que sorgissin dos nous objectius els quals, no per posteriors, deixen de ser igualment importants:

5. Justificar d'una manera no empírica que el rendiment de la SPH és generalment millor que el de la DNH.

6. Introduir modificacions en la SPH per tal de millorar-ne el rendiment.

La consecució d'aquests dos objectius permetria la millora de força algorismes proposats en la literatura i, per tant, va molt més enllà del fet de justificar la conveniència d'utilitzar la SPH en comptes de la DNH en un AG.

1.2 Continguts

La present memòria està formada per 6 capítols, incloent l'actual capítol introductori i el darrer, on es fa un resum dels principals resultats, conclusions i futures línies de recerca del present treball. La part central de la memòria la formen els capítols 3 i 4, orientats a l'obtenció de mètodes iteratius que produeixin bons resultats experimentals, i el capítol 5, orientat a l'estudi i millora de la SPH i la DNH.

En el capítol 2 s'introdueix el PSG, relacionant-lo amb les altres versions del problema, i es fa un repàs a l'estat actual de la recerca dins de l'àrea, incidint sobretot en els algorismes d'aproximació que tenen millor FRA i en els mètodes iteratius que estan donant millors resultats experimentals.

El capítol 3 pot associar-se directament al quart objectiu formulat abans, el de dissenyar un mètode no evolutiu que tingui un bon rendiment. L'algorisme que es proposa —(GBA) [45]— és un mètode iteratiu en què, a partir de l'aproximació resultant d'una execució de la SPH, s'obté una nova instància per a resoldre en la següent iteració. A més a més, incorpora noves tècniques complementàries que n'incrementen el rendiment, com ara el reetiquetatge dels vèrtexs del graf en cada iteració, les quals podrien ser utilitzades per a millorar d'altres mètodes. Tot plegat permet al GBA superar, tant en temps d'execució com en qualitat, els resultats obtinguts per l'EGA utilitzant els grafs de la OR-library.

En el capítol 4 es proposa un AG per al PSG —(GBGA) [43, 44, 46]— i es tenen en compte els tres primers objectius definits en la secció anterior. El GBGA està basat en la SPH i utilitza moltes de les tècniques complementàries que han estat introduïdes per al GBA. En realitat, els dos algorismes s'han influenciat mútuament, obeint el seu ordre d'aparició en aquesta memòria més a criteris pràctics —el GBA facilita la comprensió del GBGA— que a criteris estrictament cronològics.

Novament, el GBGA millora clarament en temps i qualitat els resultats obtinguts per l'EGA utilitzant els grafs de la OR-library. Des del nostre punt de vista, però, el principal interès d'aquest capítol es troba en la justificació que es dóna a cadascun dels elements que conformen el GBGA. Definida l'estratègia a seguir, el conjunt de peces del trencaclosques que formen el GBGA va encaixant progressivament creant

una, al nostre entendre, sòlida interpretació del funcionament de l'algorisme i dels seus predecessors, el KRGA i l'EGA.

El capítol 5 és el més teòric dels que conformen aquest treball i se n'encarrega dels dos darrers objectius que hem formulat.

En la primera secció s'analitza el funcionament de la SPH i la DNH des de diverses perspectives i es posen de manifest algunes de les diferències que permetran introduir-hi millores. En la secció 5.2 es demostra que la FRA de la SPH és igual a la de la DNH. Tot i ser un resultat ja conegut, la demostració va ser el punt de partida de les variants de la SPH que es proposen més endavant.

La DNH és l'algorisme d'aproximació per al PSG que té menor complexitat computacional, únic aspecte aquest en el qual la DNH es mostra clarament superior a la SPH. En la secció 5.3 proposem una nova implementació de la SPH que en redueix sensiblement la complexitat [47], la qual esdevé similar a la de la DNH per a la majoria d'instàncies. També s'introdueix un algorisme que té la mateixa complexitat que la DNH i un comportament similar al de la SPH.

Finalment, en la secció 5.4 es proposen dues noves variants de la SPH orientades a millorar-ne el rendiment [48]. La primera d'aquestes variants permet garantir un resultat no inferior al de la SPH mentre que la segona és la més interessant des d'un punt de vista estratègic. Tots dos algorismes tenen una complexitat semblant a la de la SPH, i admeten una implementació similar a la que ha estat proposada per a aquest algorisme en la secció anterior.

Capítol 2

Precedents i situació actual en el PSG

En la primera secció d'aquest capítol s'introdueix la notació i la terminologia bàsica que s'utilitzarà al llarg d'aquesta memòria, es defineix el PSG de manera formal i se'n descriuen algunes de les propietats bàsiques. En la secció segona, es fa una breu introducció històrica al problema d'Steiner i es comenten la versió del problema en el pla euclidià, (en anglès, *Euclidean Steiner Problem* (ESP)) i la versió en la mètrica rectilínia, (en anglès, *Rectilinear Steiner Problem* (RSP)), destacant per a cadascuna d'elles alguns dels aspectes que han centrat l'atenció en els darrers anys. Aquestes dues versions són, juntament amb el PSG, les tres versions principals del problema d'Steiner.

A partir de la secció 2.3 la memòria se centra en el PSG. En el primer apartat d'aquesta secció, i de manera semblant al que s'ha fet per a les altres dues versions, es fa un recorregut per l'evolució dels algorismes d'aproximació que han estat proposats per al problema. L'objectiu és situar aquests algorismes en el context adequat, distingint aquells que, de moment, només tenen un interès teòric, d'aquells altres que es poden fer servir per a construir mètodes que poden ser utilitzats en la pràctica. Aquest tipus de mètodes acostumen a executar un nombre considerable de vegades un algorisme heurístic per al PSG per tal de millorar el rendiment. En el segon apartat de la secció es comenten les que, en la nostra opinió, són les principals estratègies que

utilitzen aquests mètodes. A partir d'aquestes estratègies es poden obtenir una gran varietat de mètodes, molts dels quals no han estat encara implementats.

En la secció 2.4 es descriuen en detall tres algorismes d'aproximació: l'heurística del camí més curt (SPH), l'heurística del graf distància (DNH) i l'algorisme de Zelikovsky (ZH). Els dos primers són utilitzats al llarg dels capítols centrals d'aquesta memòria. Hem cregut oportú incloure també l'algorisme de Zelikovsky per la repercussió que aquest mètode ha tingut tant des d'un punt de vista teòric, com des d'un punt de vista pràctic.

Finalment, en la darrera secció del capítol es comenten alguns dels mètodes que, amb una orientació eminentment pràctica, considerem més interessants. Alguns d'aquests mètodes destaquen per la qualitat dels resultats experimentals que produeixen, resultats que seran comparats amb els del algorismes que proposem en els dos capítols següents.

L'origen del problema d'Steiner (PS) sol situar-se a principis del segle XVII, en temps del matemàtic Pierre Fermat. Des d'aleshores, i sobretot en els darrers 30 anys, l'interès per al problema ha anat en augment, com ho demostra l'extensa literatura que li ha estat dedicada:

- Per a un ampli resum dels principals aspectes relacionats amb el PS i les seves diferents versions recomanem el llibre *The Steiner Tree Problem* de Hwang, Richard i Winter [57], editat l'any 1992.

- Els progressos en l'àrea dels algorismes d'aproximació [51] han estat espectaculars en els darrers anys [5, 6], essent el PS un dels exemples paradigmàtics d'aquestes millores. Els treballs de Zelikovsky [97], Karpinski i Zelikovsky [65] i Berman i Rammayer [12] han permès obtenir millors algorismes d'aproximació per a les diferents versions del problema. Bern i Eppstein [15] fan un resum d'aquesta evolució al que cal afegir-hi el recent resultat de Hougardy [54]. Arora [4] demostra l'existència d'esquemes d'aproximació en temps polinomial (PTAS) per a l'ESP i l'RSP, deixant obsolets alguns dels resultats anteriors per a aquestes versions.

- Des d'un punt de vista més aplicat, Lucena i Beasley [72] comenten breument alguns dels articles més recents per al PSG.

- Les heurístiques de camí i distància són analitzades per Winter i Smith [93], essent aquests autors els primers a donar una visió de la DNH com a restricció de la SPH. El funcionament de la SPH també és analitzat en detall per Duin i Voß a [28].

- Finalment, destacarem la pàgina d'*internet* dedicada al PS que manté Ganley [37]. A més d'una extensa bibliografia, s'hi poden trobar problemes oberts, i les adreces d'alguns dels investigadors més destacats de l'àrea.

2.1 Definicions, notació i propietats

Un *graf* $G = (V, A)$ consisteix en un conjunt no buit V amb $|V|$ *vèrtexs* i un conjunt A amb $|A|$ *arestes*, $A \subset V \times V$, on denotem per $|X|$ el cardinal d'un conjunt X.

Utilitzem $a_k = (v_i, v_j)$ per indicar que l'aresta a_k de A relaciona els vèrtexs v_i i v_j de V, i ens podem referir a aquesta aresta com a (v_i, v_j) o a_k, indistintament. Sempre i quan no s'especifiqui el contrari, considerarem que una aresta no pot relacionar un vèrtex amb ell mateix.

Un graf és *dirigit* si la relació que estableixen les arestes no és simètrica i, per tant, (v_i, v_j) i (v_j, v_i) són dues arestes diferents. En cas contrari, direm que el graf és *simètric*. Com que la majoria de grafs que s'utilitzen al llarg d'aquesta memòria són simètrics, considerarem que un graf ho és a menys que s'especifiqui el contrari. En un graf simètric, el vèrtex v_i és *veí* del vèrtex v_j si el graf conté l'aresta (v_i, v_j). El *grau* d'un vèrtex $v \in V$ és el nombre nombre de veïns que té v, i es representa per $grau(v)$. Una *fulla* és un vèrtex de grau 1.

Considerarem que $G = (V, A, c)$ és el graf $G = (V, A)$ al qual s'ha associat una funció de *cost* $c : A \to \mathcal{R}^+ \cup \{0\}$ a les arestes. El cost de l'aresta $a_k = (v_i, v_j)$ en G es representa per $c_G(a_k)$ o $c_G(v_i, v_j)$, tot i que, quan no hi pot haver confusió respecte al graf, utilitzarem $c(a_k)$ o $c(v_i, v_j)$ i, simplificant aquest darrer, c_{ij}. El *cost* d'un graf G

és la suma de costos de les seves arestes i es representa per $|G|$. El context permetrà distingir fàcilment $|G|$ del cardinal d'un conjunt, en el cas que G fos un conjunt.

Sigui $G = (V, A, c)$ un graf:

Un *camí* $C(s,t)$ de s a t és una seqüència d'arestes $((s, v_{i_1}), (v_{i_1}, v_{i_2}), \cdots, (v_{i_{k-1}}, t))$, de manera que s és el primer vèrtex de la primera aresta, t és el segon vèrtex de l'última aresta i, el vèrtex segon de l'aresta en la posició j és el primer de l'aresta en la posició $j+1, 1 \leq j \leq k-1$. La representació del camí s'acostuma a simplificar indicant només una vegada cada vèrtex, és a dir, $C(s,t) = (s, v_{i_1}, \cdots, v_{i_{k-1}}, t)$. Un *circuit* és un camí que comença i acaba en el mateix vèrtex i no repeteix arestes. La *distància* $d_G(s,t)$ entre els vèrtexs s i t, representada per $d(s,t)$ quan no hi ha confusió respecte el graf, és el cost del camí de cost mínim, o *camí més curt*, que va de s a t. El vèrtex t és *accessible* des del vèrtex s si hi ha algun camí de s a t.

Un graf és *connex* si té, pel cap baix, un camí entre cada parella de vèrtexs. El graf $G_S = (V_S, A_S, c_S)$ és un *subgraf* de G si $V_S \subseteq V$, $A_S \subseteq A$ i $c_{G_S}(a) = c_G(a) \, \forall a \in A_S$. El subgraf G_S de G està *induït* per V_S si $A_S = \{(v_i, v_j) | v_i, v_j \in V_S, (v_i, v_j) \in A)\}$. És a dir, les arestes de G_S són les arestes de G que relacionen vèrtexs de V_S.

Un *arbre* és un graf connex que deixa de ser-ho si s'elimina una aresta qualsevol. Per tant, els arbres no tenen circuits. Un *arbre generador* $T = (V_T, A_T, c_T)$ de G és un subgraf arbre o *subarbre* de G amb $V_T = V$. Per a trobar un arbre generador de cost mínim, (en anglès, *minimum spanning tree*), es pot fer servir l'algorisme de Kruskal [69], amb complexitat $O(|A| \log |A|)$, o l'algorisme de Prim [81], amb complexitat $O(|V|^2)$ o $O(|A| + |V| \log |V|)$, depenent de si s'utilitza o no l'estructura de dades proposada per Fredman i Tarjan a [36]. Representem per $MSpT(G,S)$ un arbre generador de cost mínim per a S en el subgraf de G induït per S, $S \subseteq V$. És a dir, $MSpT(G,S)$ és un subarbre de cost mínim de G amb conjunt de vèrtexs S.

Un graf és *complet* si té una aresta entre cada parella de vèrtexs. La *desigualtat triangular* en un graf complet $G = (V, A, c)$ es compleix si $c_{ij} \leq c_{ik} + c_{kj} \forall v_i, v_j, v_k \in V$. Per a complir la desigualtat triangular un graf ha de ser complet.

El *graf distància* $D = (V, E_D, c_D)$ de G és complet i amb costos $c_D(v_i, v_j) = d_G(v_i, v_j) \forall v_i, v_j \in V$. Els grafs distància sempre compleixen la desigualtat triangular.

Per a obtenir D és necessari calcular els camins més curts entre cada parella de vèrtexs utilitzant, per exemple, l'algorisme de Floyd [33], amb complexitat $O(|V|^3)$, o executant $|V|$ vegades l'algorisme de Dijkstra [23] per a trobar el camí de cost mínim entre dos vèrtexs. En aquest cas, la complexitat és $O(|V|(|A| + |V| \log |V|))$ quan s'utilitza l'estructura de dades proposada per Fredman i Tarjan [36].

Donats un graf $G(V, A, c)$ i un subconjunt no buit S de vèrtexs:

- Un *arbre d'Steiner* $StT(G, S)$ per a S en G és un subarbre $T = (V_T, E_T, c_T)$ de G amb $S \subseteq V_T$. Els vèrtexs de S són anomenats *terminals.* Per contraposició, els vèrtexs a $V - S$ s'anomenen *no-terminals.* Els no-terminals que formen part de V_T s'anomenen *vèrtexs d'Steiner.* Fem servir S' per a referir-nos al conjunt de vèrtexs d'Steiner.

- El *problema d'Steiner en grafs* consisteix en trobar un arbre d'Steiner de cost mínim $MStT(G, S)$ per a S en G.

- Quan $S = V$ el problema es pot resoldre de manera eficient calculant un arbre generador de cost mínim de G. Quan $|S| = 2$, el problema és equivalent a trobar el camí de cost mínim entre dos vèrtexs, la qual cosa es pot portar a terme mitjançant l'algorisme de Dijkstra [23] amb complexitat $O(|A| + |V| \log |V|)$. En el cas general, però, el PSG és *NP-complet* [39].

- Si $T = (V_T, E_T, c_T)$ és un $MStT(G, S)$ aleshores T també és un $MSpT(G, V_T)$. Per tant, des d'un punt de vista combinatori, el problema d'Steiner consisteix en localitzar un subconjunt de vèrtexs d'Steiner S' que formi part d'un $MStT(G, S)$. Localitzat aquest conjunt, la solució $MSpT(G, S \cup S')$ es pot obtenir de manera senzilla mitjançant l'algorisme de Kruskal o el de Prim. Notem que hi ha $2^{|V|-|S|}$ maneres diferents d'escollir S'.

- Tot $MStT(G, S)$ és un $MStT(D, S)$, essent D el graf distància de G. A més a més, tot $MStT(D, S)$ es pot transformar en un $MStT(G, S)$ a base de substituir les arestes del primer a D pels corresponents camins més curts a G. Està ja demostrat que hi ha un $MStT(D, S)$ amb, com a molt, $|S| - 2$ vèrtexs d'Steiner.

És per això que, en alguns casos, la utilització de D permet disminuir de manera sensible el nombre de combinacions a considerar per a l'elecció del conjunt S'.

- Els mètodes heurístics per al PSG aproximen un $MStT(G,S)$ mitjançant un $StT(G,S)$. Si aquest arbre conté vèrtexs d'Steiner que són fulles es poden eliminar per tal de millorar el cost de l'aproximació. Aquests vèrtexs seran anomenats *corbates*.

En la figura 1 es proposa un exemple del PSG i es destaquen alguns dels aspectes de la notació que utilitzarem d'ara endavant.

$G = (V, E, c)$
$S = \{v_1, v_2, v_3\}$ terminals
$V - S = \{v_4, v_5\}$ no-terminals

$T = MStT(G, N)$, i té cost $|T| = 4$
$S' = \{v_5\}$ i v_5 és un vèrtex d'Steiner
T és també un $MSpT(G, S \cup \{v_5\})$

Figura 1: Exemple del PSG. L'arbre de la dreta és una solució al problema d'Steiner plantejat a l'esquerra. El problema consisteix en trobar un arbre de cost mínim que connecti els tres vèrtexs de G situats a l'interior d'un requadre.

2.2 El PS: origen i versions principals

A principis del segle XVII, Fermat va proposar en el seu *Treatise on Minima and Maxima* el problema següent: *trobar el punt en el pla que minimitza la suma de distàncies des d'ell a tres punts donats.* Aquest punt va ser anomenat *punt de Torricelli* en honor al matemàtic que va proposar un mètode geomètric per a localitzar-lo, cap a l'any 1640.

Dos segles més tard, Jacob Steiner va plantejar un problema similar: *trobar la manera de connectar tres ciutats mitjançant una xarxa de carreteres amb distància total mínima* —se suposa que la superfície en què es troben les ciutats és completament plana. Tret del cas trivial en què les tres ciutats estan alineades, si es connecten dues a dues les ciutats mitjançant rectes es forma un triangle. No resulta difícil veure que la xarxa solució estarà formada per tres carreteres, cadascuna d'elles partint d'una de les ciutats, que convergeixen en un punt interior d'aquest triangle. La xarxa amb distància mínima s'obtindrà quan aquest punt interior coincideixi amb el punt de Torricelli per a les tres ciutats donades, d'aquí que els problemes proposats per Fermat i Steiner puguin ser considerats equivalents.

2.2.1 El problema d'Steiner en el pla euclidià

El problema d'Steiner en el pla euclidià va ser formulat per primera vegada per Jarník i Kössler [60] l'any 1934 i és una generalització del plantejament anterior per a un nombre qualsevol de ciutats, substituïdes per punts:

> *Trobar la manera de connectar mitjançant segments un conjunt P de punts en el pla de manera que la longitud total dels segments usats sigui mínima.*

Segons sembla, varen ser Courant i Robins en el llibre *What is Mathematics* [21] els qui l'any 1941 varen utilitzar per primera vegada el nom d'Steiner per a referir-se al problema. Steiner, però, va plantejar l'anterior problema per a tres ciutats quan d'altres autors, com ara Torricelli, ja havien considerat segles abans elements per a solucionar-lo. Curiosament, és el nom d'Steiner el que ha quedat associat al problema, tot i que no es coneix pas que aquest autor fes cap altra contribució que la de plantejar el cas per a tres ciutats [57, 20].

L'ESP té aplicació en una gran varietat de problemes d'interconnexió [57] que van des de minimitzar la longitud dels conductes per a transportar gas, a establir les connexions entre autopistes de manera òptima [27].

La dificultat principal de l'ESP consisteix en localitzar aquells punts addicionals en què convergeixen més de dos segments. Són l'equivalent al punt de Torricelli per

al cas general i s'anomenen *punts d'Steiner*. Se sap que en cada punt d'Steiner hi incideixen exactament tres segments, que aquests segments formen entre ells un angle de 120 graus, i que el nombre de punts d'Steiner no pot ser superior al nombre total de punts que cal connectar menys dos.

Un cop localitzats els punts d'Steiner faltarà trobar una forma de connectar aquests punts, conjuntament amb els inicials, de manera òptima. No és difícil veure que aquesta operació és equivalent a la de trobar un *arbre generador de cost mínim* en el graf simètric i complet $G = (P, A, c)$ en què cada vèrtex representa un dels punts, i cada aresta té un cost igual a la distància euclidiana entre els dos punts que relaciona. Més encara, si sabem localitzar un conjunt finit de punts que inclogui els vèrtexs d'Steiner d'un arbre d'Steiner òptim, l'ESP es pot plantejar de manera equivalent al problema d'Steiner en grafs.

La dificultat de localització dels punts d'Steiner fa que l'ESP sigui *NP-hard* [39] fins i tot en la seva versió de decisió[1] i no NP-complet, com les altres dues versions principals[2]. En canvi, des del punt de vista de la seva aproximabilitat, ja fa anys que es coneixen algorismes d'aproximació per al problema:

- El primer algorisme proposat aproxima l'arbre d'Steiner òptim mitjançant un arbre generador de cost mínim per a P en G, $MSpT(G, P)$, on el graf G està definit com a dalt. Per tant, aquest algorisme no considera la possibilitat d'incloure cap punt d'Steiner en l'arbre resultant. Emprant algunes propietats geomètriques en la disposició dels punts en el pla, la complexitat de l'algorisme esdevé $O(|P| \log |P|)$ [17].

- L'any 1968, Gilbert i Pollak [41] van conjecturar que aquest algorisme tenia una FRA igual a $2/\sqrt{3}$. Aquesta fita va passar a anomenar-se la *raó d'Steiner*, (en anglès, *Steiner ratio* (SR)), i assegurava un percentatge d'error inferior al 15,5%. El valor exacte d'aquesta fita s'assoleix en el cas en què només hi ha tres punts situats en els extrems d'un triangle equilàter.

[1] En aquest cas utilitzem la denominació NP-hard de forma estricta i no, com és habitual, per a referir-se a les versions no de decisió de problemes NP.

[2] Tot i que a [54] s'esmenta que totes tres versions són NP-complet no tenim constància de que s'hagi pogut demostrar per a l'ESP [15].

- L'any 1990, després de diverses demostracions per a un nombre reduït de punts [15], Du i Hwang [26] van demostrar la conjectura anterior de Gilbert i Pollak. Des d'un punt de vista teòric, l'interès passaria a centrar-se en veure si l'SR era la menor fita que un algorisme d'aproximació per a l'ESP podia obtenir. En canvi, des d'un punt de vista aplicat, es feia difícil de pensar que l'excel·lent relació entre la complexitat i la FRA d'aquest algorisme pogués ser millorada de forma significativa. És per això que l'interès es dirigiria cap a l'obtenció de tècniques complementàries i mètodes heurístics que permetessin millorar la qualitat de les aproximacions retornades per aquest algorisme.

- Els primers a proposar un algorisme d'aproximació amb fita inferior a l'SR van ser Du, Zhang i Feng [25], basant-se en els treballs de Zelikovsky [97] per al PSG. Tot i que la millora era insignificant i que la complexitat de l'algorisme en feia inviable la seva utilització en la pràctica, es posava punt i final a la conjectura de que l'SR és la menor FRA assolible per un algorisme d'aproximació.

- L'any 1996, Zelikovsky [99] proposa una nova generalització de la seva estratègia que permet obtenir un algorisme amb FRA igual a $1 + \ln(2/\sqrt{3}) \approx 1,144$.

- El mateix any, Arora [4] demostra l'existència de PTAS per a l'ESP, així com per a diversos problemes geomètrics relacionats. Com que aquests esquemes permeten obtenir algorismes d'aproximació amb FRA tant propera a 1 com es vulgui, l'obtenció d'algorismes amb millor fita deixa de tenir interès.

L'existència de PTAS té unes connotacions teòriques molt importants per a l'ESP que, a la llarga, podrien acabar traduint-se en l'obtenció d'algorismes d'aproximació més eficients per al problema. Contràriament al que es preveia, l'ESP no és MAX-SNP-hard [76] a menys que la classe P sigui igual a la classe NP. El problema d'Steiner s'ha convertit en un dels exemples paradigmàtics del que pot donar de si aquesta àrea emergent en què, combinant la teoria de la complexitat i l'algorísmica, els resultats teòrics van acompanyats de mètodes utilitzables en la pràctica.

D'altra banda, des d'un punt de vista més aplicat, Zachariasen i Winter [95] han presentat darrerament un mètode heurístic que combina diverses tècniques i té

complexitat $O(|P| \log |P|)$. Utilitzant jocs de prova, l'algorisme supera en qualitat els resultats dels mètodes amb complexitat similar coneguts i, també, els de la gran majoria de mètodes amb complexitat superior. Per acabar, Warme [91] ha proposat un mètode per a la versió rectilínia que combina, per a trobar una solució òptima, programació lineal i mètodes heurístics (*branch and cut*). Adaptant aquest mètode al cas euclidià, i incorporant algunes de les tècniques del mètode de Zachariasen i Winter que acabem d'esmentar, l'algorisme de Warme troba la solució òptima per a problemes on hi ha fins a 2000 punts.

2.2.2 El problema d'Steiner en la mètrica rectilínia

El problema d'Steiner en la mètrica rectilínia va ser proposat per Hanan [50] l'any 1965:

> *Trobar la manera de connectar mitjançant segments verticals i horitzontals un conjunt P de punts en el pla de manera que la longitud total dels segments usats sigui mínima.*

Aplicacions immediates d'aquest plantejament es troben en alguns problemes d'interconnexió, com ara el disseny de circuits impresos i, amb algunes restriccions addicionals, el disseny de circuits integrats [70, 62, 2], màxim responsable de l'alt interès que ha despertat aquesta versió del PS.

El fet que en l'RSP es restringeixi, en comparació amb la versió euclidiana, l'orientació dels segments, limita la localització dels punts addicionals en els quals hi convergeixen dos o més segments, també anomenats *punts d'Steiner*. Aquests punts han de formar part de la intersecció de dues rectes perpendiculars obtingudes com a prolongació vertical i horitzontal de, com a mínim, dos dels punts inicials. Localitzats aquests punts, i de manera molt semblant al cas euclidià, és pot transformar el problema en el de trobar un arbre generador de cost mínim en un graf. Més encara, com que el nombre de punts candidats a ser punts d'Steiner és finit, tota instància per a l'RSP pot transformar-se en una instància per al PSG. Diverses consideracions geomètriques i estructurals permeten reduir el nombre de punts d'Steiner candidats a formar part d'un arbre d'Steiner òptim [57].

Garey i Johnson [40] van demostrar l'any 1977 que l'RSP és, al igual que el PSG, NP-complet. En canvi, des del punt de vista de la seva aproximabilitat, la trajectòria de l'RSP ha estat sempre més relacionada amb la de la versió euclidiana:

- El primer algorisme que va ser proposat aproxima l'arbre d'Steiner òptim mitjançant un $MSpT(G, P)$, on $G = (P, A, c)$ és un graf complet en què el cost associat a cada aresta és igual a la distància rectilínia entre els punts que relaciona. Hwang [55] va demostrar l'any 1976 que aquest arbre garanteix una FRA igual a 1,5 i va proposar un algorisme [56] amb complexitat $O(|P| \log |P|)$ per a calcular-lo.

- Basant-se en els seus resultats per al PSG [97], Zelikovsky [96] va proposar un algorisme d'aproximació amb una FRA inferior a $11/8 = 1,375$ per a l'RSP l'any 1992.

- Uns anys més tard, Berman, Föβmeier, Karpinski, Kaufmann i Zelikovsky [13] fan una millor anàlisi que a [96, 12, 34] de l'algorisme i obtenen una FRA igual a $61/48 \approx 1,271$ amb complexitat $O(|P|^{1,5})$. Una versió paramètrica del mateix algorisme [13] aproxima molt aquesta fita en $O(|P| \log^2 |P|)$.

- L'any 1997, Karpinski i Zelikovsky [65] publiquen una nova millora de l'algorisme en què la FRA es redueix fins a $19/15 \approx 1,267$.

- Arora [4] demostra l'any 1996 l'existència de PTAS per a l'RSP.

És important remarcar que, a diferència del que passava en la versió euclidiana, l'obtenció de millors fites per a l'RSP no ha anat estretament associada a increments significatius de complexitat dels algorismes. Així doncs, si els resultats d'Arora tenen una gran importància des d'un punt de vista teòric, però no semblen tenir de moment una repercussió en la pràctica [95], alguns dels algorismes mencionats proporcionen garanties de fins a un 27% amb una complexitat molt raonable. És per això que aquests algorismes poden ser utilitzats com a base de mètodes orientats a obtenir bons resultats experimentals per a problemes de mida cada cop més gran. Cal destacar en aquest aspecte els impressionants resultats obtinguts per Warme [91] aconseguint

trobar la solució òptima per instàncies de fins a 1000 punts. Les últimes notícies dels competidors directes del seu mètode exacte, basat en el *branch and cut*, arriben a trobar l'òptim per a instàncies que no arriben als 60 punts [35], [67]. El mètode exacte proposat per Föβmeier i Kaufmann [35] té complexitat $O(2^{|P|})$, la menor complexitat que ha estat demostrada per a un mètode que retorna l'òptim. Finalment, el recent article de Ganley [38] fa una recopilació dels darrers mètodes aplicats a l'ESP.

2.3 El PSG: origen i mètodes heurístics

El problema d'Steiner en grafs va ser proposat de forma independent per Hakimi [49] i Levin [71] l'any 1971:

> *Trobar un arbre que connecti amb cost total mínim un subconjunt S de vèrtexs en un graf $G = (V, A, c)$ simètric i connex, amb costos no negatius associats a les arestes.*

Tal i com hem constatat anteriorment, la principal dificultat del PSG consisteix en seleccionar aquells vèrtexs que, tot i no ser terminals, formaran part de la solució òptima. Són l'equivalent als punts d'Steiner per a les altres versions del problema.

Tot i que la versió rectilínia és anterior a la versió en grafs, l'RSP pot ser plantejat en termes del PSG [57] i, per tant, els mètodes per al PSG poden també ser utilitzats per a l'RSP. En aquest cas, si bé no es tindran en compte algunes de les propietats geomètriques específiques del cas rectilini, si que es podran utilitzar la gran varietat de rutines que han estat proposades per a reduir la mida dels grafs inicials en el PSG [57]. D'aquesta manera, la no utilització d'algorismes específics per a l'RSP [66], molts d'ells amb menor complexitat i FRA, pot quedar compensada per la reducció de la mida del problema. Queda pendent, tal i com es proposa a [67], l'elaboració d'un conjunt de rutines de reducció específiques per al cas rectilini.

D'altra banda, tal i com hem vist en la secció anterior, també alguns dels mètodes heurístics per a l'ESP estan basats en mètodes heurístics per al PSG. Per exemple, Beasley i Goffinet [11] es basen en les regions de Voronoi per a seleccionar un subconjunt de vèrtexs no-terminals i transformar el problema en la versió en grafs.

El PSG és NP-complet [64] fins i tot en el cas d'un graf complet en què les arestes tenen costos 1 o 2, cas particular que ha permès demostrar que el PSG és MAX-SNP-complet [14]. Conseqüentment, el PSG no pot tenir PTAS a menys que la classe de problemes P sigui igual a la classe NP. Tot i que s'han proposat diversos algorismes d'aproximació, encara no s'ha pogut trobar la menor FRA que pot ser garantida per un d'aquests algorismes.

2.3.1 Algorismes d'aproximació

- Takahashi i Matsuyama [88] van proposar el primer algorisme d'aproximació[3] l'any 1980, conegut com a *heurística del camí més curt* (SPH) (veure l'apartat 2.4.1). L'algorisme està basat en una generalització de l'algorisme de Prim [81], i té complexitat $O(|S|(|A| + |V|\log|V|))$ quan s'utilitza l'estructura de dades de [36], i una FRA igual a $2(1 - 1/|S|)$.

- L'any 1981, Kou, Markowsky i Berman [68] demostren que l'*heurística del graf distància* (DNH) (veure l'apartat 2.4.2) és un algorisme d'aproximació, i té una FRA igual a $2(1 - 1/f)$, on f és el mínim nombre de fulles en una solució òptima. Com que $f \leq |S|$, es va estendre la idea que la FRA de la DNH era lleugerament inferior a la de la SPH [84, 73, 93, 57]. En el capítol 5 es demostra que aquesta idea no és certa. La mateixa conclusió, però, es dedueix de la demostració alternativa proposada per Duin i Voß a [28]. D'altra banda, l'any 1988 Mehlhorn [73] va presentar una implementació de la DNH en què la complexitat es redueix fins a $O(|A| + |V|\log|V|)$ (veure la secció 5.3.2).

- En els anys següents, es proposen diversos algorismes d'aproximació [57], d'entre els quals destaquem l'*heurística de distància mitjana*, (en anglès, *average distance heuristic* (ADH)), proposada per Rayward-Smith i Clare [83] l'any 1983 i l'*heurística de contracció*, (en anglès, *contraction heuristic* (CH)), proposada per Plesník [78] l'any 1981. Tots dos algorismes tenen la mateixa FRA que

[3]En realitat, els autors van ser els primers a demostrar una FRA per a un mètode heurístic per al PSG. La DNH és anterior a la SPH, tot i que no estava demostrat que fos un algorisme d'aproximació.

la DNH i complexitat $O(|V|^3)$. L'ADH acostuma a produir millors resultats experimentals que la SPH i la DNH [84, 93]. La seva FRA va ser demostrada per Waxman i Imase [92] l'any 1988. A [79], Plesník compara teòricament fins a 9 mètodes heurístics per al PSG, entre els que s'inclouen la SPH, la DNH, l'ADH i la CH. Per a cada parella d'algorismes es poden trobar exemples en què un algorisme retorna millor solució que l'altre i, també, exemples del contrari. L'única excepció es dóna quan es compara la CH amb la DNH, ja que no es troba cap exemple en què aquesta última obtingui una millor aproximació que la primera.

- L'any 1990, Zelikovsky [97] proposa un algorisme (ZH) amb FRA $\leq 11/6$ (veure apartat 2.4.3), que té complexitat $O(|S|(|S||V| + |V|\log|V| + |A|))$ [98]. És el primer algorisme d'aproximació amb una FRA constant per sota de 2 i desencadena un seguit de millores que, tal i com hem vist en els apartats anteriors, també afecten a les altres versions del problema. Recentment, Duin i Voß [28] han reduït la complexitat de l'algorisme fins a $O(|S|(|V|\log|V| + |A|))$.

- Berman i Rammayer [12] generalitzen la idea de Zelikovsky proposant un mètode (BRH) per a generar una seqüència d'algorismes d'aproximació polinomial per al problema. Es tracta d'un PTAS per a una versió simplificada del PSG, el resultat de la qual és usat com a arbre aproximat per al PSG. El primer algorisme de la seqüència és equivalent a la DNH, el segon és igual al ZH, el tercer té una FRA no inferior a 16/9 amb complexitat $O(|S|^3|V|^2)$. Cada nou algorisme de la seqüència millora la FRA de l'anterior, però té una complexitat més gran. L'any 1997, Borchers i Du [16] aconsegueixen demostrar que la FRA dels algorismes d'aquesta seqüència tendeix a 1,734.

- També l'any 1997, Karpinsky i Zelikovsky [65] introdueixen algunes modificacions al ZH, obtenint una nova seqüència d'algorismes (KZH) de característiques similars a l'anterior, en la que la FRA tendeix a 1,644.

- L'any 1998 Prömel i Steger [82] proposen un algorisme d'aproximació (PSH) dels anomenats *RNC*, (en anglès, *randomized parallel polylog time*[4]), amb FRA igual a 5/3. La complexitat de l'algorisme és molt alta si s'executa amb una màquina seqüencial, però és $O(\log^2 |V|)$ si es disposa d'$O(|V|^3)$ processadors.

- Finalment, l'any 1999 Hougardy i Prömel [54] proposen un algorisme d'aproximació (HPH) amb FRA demostrada propera a 1,598 que, segons alguns experiments, podria arribar a ser de 1,588. Es tracta d'un algorisme *randomized* en què s'executen de forma repetida algunes de les generalitzacions del ZH.

En la taula 1 es presenta un resum de l'evolució dels algorismes d'aproximació per al problema.

Any	Nom	FRA	Complexitat	Referències
1980	SPH	$2(1-1/f)$	$O(\|S\|(\|A\|+\|V\|\log\|V\|))$	[88]
1981	DNH	$2(1-1/f)$	$O(\|A\|+\|V\|\log\|V\|)$	[68] [73]
1981	CH	$2(1-1/f)$	$O(\|V\|^3)$	[78],[79]
1983	ADH	$2(1-1/f)$	$O(\|V\|^3)$	[83, 84][92]
1990	ZH	$11/6 \approx 1,83$	$O(\|S\|(\|A\|+\|V\|\log\|V\|))$	[97] [28]
1992	BRH	$16/9 \approx 1,78$	$O(\|V\|^2\|S\|^3)$	[12] [16]
1997	KZH	$1+\ln 2 \approx 1,64$		[65]
1998	PSH	$5/3 \approx 1,66$		[82]
1999	HPH	1,598		[54]

Taula 1: Algorismes d'aproximació per al PSG. $f \leq |S|$ és el mínim nombre de fulles en un $MStT(G,S)$.

2.3.2 Estratègies principals dels mètodes d'execució múltiple

En el llibre de Hwang, Richards i Winter [57], hi ha un capítol dedicat a mètodes heurístics per al PSG en què es consideren fins a 25 algorismes diferents, molts d'ells d'aproximació, entre els que s'inclouen els sis primers de la taula 1. Setze d'aquests algorismes tenen una FRA igual a $2(1-1/f)$, on f és el mínim nombre de fulles

[4] *Polynomial time in* $\log|V|$

en un arbre d'Steiner òptim. Tots aquests mètodes retornen una aproximació amb cost no superior al d'un $MSpT(D,S)$, on D és el graf distància de G. Aleshores, el valor de la FRA es dedueix directament de la relació següent, demostrada per Kou, Markovsky i Berman[5] [68]:

$$\frac{|MSpT(D,S)|}{|MStT(G,S)|} \leq 2(1-\frac{1}{f}).$$

Per part nostra, fem una distinció entre els *algorismes d'un pas*, és a dir, aquells algorismes heurístics que calculen un únic arbre d'Steiner, i els *mètodes d'execució múltiple*, els quals, després de diverses execucions d'un algorisme d'un pas, retornen com a solució la millor aproximació obtinguda. Per tant, els primers s'utilitzen com a base per a construir els segons, que solen estar més orientats a tenir un bon rendiment en la pràctica.

Presentem ara les que, en la nostra opinió, són les principals estratègies en què es basen els mètodes d'execució múltiple. Aquestes estratègies són independents de l'algorisme d'un pas que s'utilitza com a base i, per tant, substituint l'algorisme es construeix un nou mètode. A més, tota millora en un algorisme d'un pas té una influència positiva en tots aquells mètodes que l'utilitzen. Les estratègies que analitzem admeten un nombre important de modificacions que n'incrementen les possibilitats. En la nostra opinió, els mètodes que obtindran millors resultats combinaran diversos algorismes heurístics basats en estratègies diferents. Les possibilitats són molt grans i, tal i com es pot veure, encara hi ha molt per fer.

En el que queda d'aquest apartat escriurem $H(G,V_s)$ per indicar l'arbre resultant d'executar l'algorisme d'un pas H per al subconjunt de vèrtexs V_s del graf G.

Estratègia repetitiva

Consisteix en generar un nombre relativament alt d'aproximacions a base d'executar diverses vegades un algorisme d'un pas, sense modificar la instància del problema a resoldre. En cadascuna de les execucions s'escull una opció diferent en aquelles situacions en què hi ha més d'una opció vàlida possible —situacions que es donen,

[5] Veure demostració 5.2

per exemple, quan s'ha de seleccionar un vèrtex, una aresta o un camí. Al final, el mètode retorna la millor aproximació obtinguda.

Els mètodes *repetitius* proposats per Winter i Smith [93] (veure l'apartat 2.5.1) estan basats en la SPH i són l'exemple més clar d'aquesta estratègia. També Koch i Martin [67], en el seu mètode basat en el *branch and bound* que descrivim en l'apartat 2.5.5, utilitzen aquesta estratègia d'una manera restringida. En concret, executen repetidament unes 10 o 11 vegades l'algorisme d'un pas en cada iteració. Per part nostra, la influència de l'estratègia repetitiva és notòria en el disseny de la funció *reordenar* (veure pàgina 46), la qual forma part dels algorismes que proposem en els dos capítols que venen a continuació.

Els mètodes basats en una estratègia repetitiva redueixen el risc d'aconseguir una solució marginal, poc representativa de les que acostuma a produir l'algorisme d'un pas. L'interès d'aquests mètodes és més gran per aquells algorismes que, amb petites variacions, poden generar aproximacions molt diferents. Finalment, és important remarcar que la complexitat d'aquests mètodes es pot reduir en molts casos a base d'aprofitar càlculs comuns en totes les execucions.

Estratègia iterativa *greedy*

L'objectiu d'aquesta estratègia és millorar el cost d'una aproximació donada a base d'incorporar un nou vèrtex d'Steiner en cada iteració. El vèrtex escollit és aquell que produeix una reducció més gran del cost de la solució actual, avaluada mitjançant l'algorisme d'un pas que s'utilitza com a base. És a dir, en cada iteració s'escull aquell vèrtex v que minimitza el cost de $H(G, V_T \cup \{v\})$, on V_T és el conjunt de vèrtexs en l'arbre a l'inici de la iteració. El procés finalitza quan cap vèrtex permet aconseguir una millora en el cost. Tal i com es pot veure, l'arbre inicial no té per què haver estat generat per l'algorisme de base.

El primer mètode basat en una estratègia iterativa *greedy* del que tenim constància va ser proposat per Minoux [74] l'any 1990. L'algorisme d'un pas seleccionat era una variant de l'arbre generador de cost mínim de G, d'aquí que els resultats fossin discrets. A [57] es proposa millorar el mètode substituint el graf G pel seu graf distància i, d'aquesta manera, garantir una FRA inferior a 2. Aquesta nova versió de

l'algorisme és la que anomenem *greedy* de Minoux i, tal com s'ha vist més endavant [28], té molts punts en comú amb el ZH [98].

Alguns altres mètodes basats en aquesta estratègia s'han utilitzat en treballs orientats al disseny de circuits integrats. Per exemple, Kahng i Robins [61] van proposar un mètode basat en la DNH mentre que, Alexander i Robins [2] (veure l'apartat 2.5.2) van proposar l'estratègia iterativa *greedy* tal i com ha estat descrita en aquest apartat.

Tot i que el nombre d'arbres que es generen en cada iteració acostuma a ser de l'ordre del nombre de no-terminals, el nombre d'execucions de l'algorisme pot ser inferior. En alguns casos, els arbres es poden obtenir d'una forma més eficient a partir de la solució actual, la qual cosa fa difícil la distinció entre els mètodes basats en una estratègia d'execució múltiple i els algorismes d'un pas. Per exemple, en el *greedy* de Minoux només executa un únic cop l'algorisme d'un pas per tal de calcular l'arbre inicial. De tota manera, deixant de banda la complexitat, no hi ha cap inconvenient que impedeixi la utilització d'un mètode d'execució múltiple com a algorisme de base d'un altre mètode.

En els mètodes basats en l'estratègia iterativa *greedy* es defineix un *veïnatge* [85], consistent en tots aquells arbres que poden ser obtinguts mitjançant l'algorisme en afegir un nou vèrtex terminal. Aquest veïnatge permet introduir moltes variants en l'estratègia. Per exemple, es poden incrementar el nombre de veïns a base de considerar la possibilitat d'afegir més d'un vèrtex, la qual cosa repercuteix negativament en la complexitat de l'algorisme. Una altra possibilitat és seleccionar el primer vèrtex que produeixi una millora i, d'aquesta manera, reduir la complexitat. El mètode resultant és un *hill-climbing*, ja que el cost de la solució es redueix en cada iteració. Finalment, es possible incrementar la capacitat d'exploració de l'estratègia a base de seleccionar, ocasionalment, un vèrtex que no produeixi millora. Aquesta és una de les bases de l'anomenada *simulació termodinàmica*, (en anglès, *simulated annealing*), de la qual n'hi ha un gran nombre de variants.

Estratègia iterativa recurrent

En aquesta estratègia es modifica en cada iteració la instància del problema a resoldre en funció de l'arbre resultant de la iteració anterior. El més habitual és que els

vèrtexs d'aquest arbre T siguin considerats els terminals de la nova instància i, per tant, el resultat de la següent iteració és $T' = H(G, vertexs(T))$ —d'aquí que l'hàgim anomenat estratègia recurrent. La majoria d'algorismes d'un pas garanteixen una solució amb cost no superior a la d'un arbre generador de cost mínim per al conjunt de terminals. Per tant, els costos de la seqüència d'arbres obtinguda mitjançant aquesta estratègia acostuma a ser decreixent. El procés finalitza quan en una iteració T' és igual a T o, en alguns casos, quan el cost de T' és el mateix que el de T.

El primer mètode integrant d'aquesta estratègia del que tenim constància va se proposat per Duin i Voβ [28], i és anomenat *Rebuilt*. En realitat, és l'arbre resultant de cada iteració el que s'utilitza com a argument d'entrada en l'algorisme, que és una variant de la SPH. Aquest arbre permet resoldre de manera més eficient la nova instància que s'obté de considerar els seus vèrtexs terminals. El segon mètode integrant d'aquesta estratègia utilitza la SPH com a base i va ser proposat per Guitart i Basart [45] (veure la pàgina 45). No tenim constància de que s'hagi proposat anteriorment la possibilitat de substituir l'algorisme d'un pas, tal i com l'acabem de descriure.

L'estratègia iterativa *recurrent* pot ser considerada com un cas particular d'una estratègia en què diversos algorismes són concatenats per a generar un mètode més poderós. En concret, la sortida d'un algorisme genera una nova instància que es resol utilitzant el mateix algorisme, en el cas recurrent, o utilitzant un altre mètode, en el cas general. Tal i com veurem en el capítol següent, aquesta estratègia és molt versàtil, ja que permet afegir funcions de millora, incrementa les possibilitats de l'estratègia repetitiva i, a més, necessita calcular molts menys arbres que en l'estratègia iterativa *greedy*.

Finalment, la informació que s'obté de la concatenació d'algorismes és una bona eina per a entendre i millorar el comportament dels algorismes, permetent, tal i com veurem en els capítols següents, actuacions en dos sentits complementaris: adaptar els algorismes de manera que la solució resultant no pugui ser millorada via execució concatenada, o incorporar noves funcions als algorismes de manera que, sense incidir negativament en la solució, la facin susceptible de ser millorada per concatenació.

Estratègies híbrides

Els mètodes basats en aquestes estratègies combinen metodologies genèriques de resolució de problemes amb mètodes heurístics per al PSG. L'algorisme d'un pas acostuma a estar està supeditat a una estratègia de nivell superior. Distingim, bàsicament, dos tipus de mètodes diferents que tenen cabuda en aquest apartat: els que utilitzen programació lineal i les meta-heurístiques basades en una estratègia d'heurística moderna.

Els mètodes basats en la programació lineal van ser ideats per a obtenir fites inferiors de la solució òptima via una *relaxació lineal* [57] i han anat evolucionant cap a mètodes exactes que, a grans trets, combinen aquestes tècniques amb mètodes de recerca en arbre. Tot plegat porta cap a l'anomenat *branch and cut* [18, 72, 67] que, en el cas del *problema del viatjant*, és el mètode que ha aconseguit resoldre instàncies per a un nombre més gran de ciutats [3]. En la secció 2.5.5 es descriuen els dos mètodes basats en aquesta estratègia que han estat proposats recentment [72, 67]. Tots dos mètodes utilitzen la SPH com a algorisme de base i poden també ser usats per a obtenir aproximacions quan la mida de la instància impedeix aconseguir una solució òptima en un temps raonable.

Les meta-heurístiques basades en una estratègia d'heurística moderna han estat comentades en el capítol introductori (veure la pàgina 4). Els exemples més abundants són els algorismes genètics: el de Kapsalis i Rayward-Smith [63] basat en l'arbre generador de cost mínim, el d'Esbensen [30] (veure l'apartat 2.5.4) i el de Voβ i Gutenschwager [90] basats en la DNH, i l'algorisme basat en la SPH de Guitart i Basart [44]. De tots ells en parlarem amb més detall en el capítol 4.

2.4 Algorismes d'un pas

En aquesta secció descriurem els tres algorismes d'un pas que són més freqüentment utilitzats per a construir els mètodes que competeixen per obtenir els millors resultats experimentals: la SPH, la DNH i el ZH. Les versions que es donen de la SPH i de la DNH estan basades en la seva descripció clàssica. Més endavant, en el capítol 5,

es proposen versions diferents que permeten, d'una manera més clara, ressaltar les similituds i diferències entre tots dos algorismes.

2.4.1 L'heurística del camí més curt (SPH)

La SPH va ser proposada per Takahashi i Matsuyama [88] l'any 1980:

Pas 1. Començar amb $G_1 = (V_1 = \{s_0\}, A_1 = \emptyset, c)$, $s_0 \in S$;
Pas 2. Seleccionar $s, t \in V$ i P_{st} de manera que
$d(s,t) = min\{d(v,v')\}\, \forall v \in S - V_1, \forall v' \in V_1$, i
P_{st} és un camí de distància $d(s,t)$ de s a t;
Pas 3. Afegir P_{st} a G_1;
Pas 4. Si V_1 no conté tots els terminals
Tornar al *Pas 2*; ($|S| - 1$ vegades)
Pas 5. Retornar $T = trim(MSpT(G, V_1))$.

Algorisme 1: L'heurística del camí més curt (SPH)

La SPH està basada en l'algorisme de Prim [81] per a calcular l'arbre generador de cost mínim d'un graf. A diferència d'aquest, però, utilitza el camí de cost mínim, i no l'aresta de cost mínim, quan afegeix un nou vèrtex a l'arbre en construcció. El procés iteratiu de la SPH comença amb un arbre inicial format per un únic vèrtex. En cada iteració, el terminal a menor distància s'hi connecta mitjançant el camí més curt possible. El procés finalitza quan l'arbre conté tots els terminals.

El *Pas 5* va ser incorporat més tard a l'algorisme inicial de Takahashi i Matsuyama, tal i com varen proposar Rayward-Smith i Clare [84] l'any 1986, i té com a objectiu assegurar que l'arbre resultant connecta de manera òptima els seus vèrtexs. La funció *trim* elimina les corbates ja que, al ser vèrtexs no-terminals fulla, són innecessaries per a connectar els terminals. La complexitat de l'algorisme és $O(|S|(|A| + |V| \log |V|))$, quan no es tenen en compte les millores que proposem en el capítol 5.

2.4.2 L'heurística del graf distància (DNH)

Segons [57], la DNH va ser proposada de forma independent per Choukhmane [19], Kou, Markowsky i Berman [68], Plesník [78] i Iwainsky, Canuto, Taraszow i Villa [59].

Pas 1. Construir el subgraf D_S del graf distància D de G induït per S;
Pas 2. $T_0 := MSpT(D_S, S)$;
Pas 3. Construir G_1 a base de substituir les arestes de T_0 pel camí de cost mínim corresponent en G. Sigui V_1 el conjunt de vèrtexs de G_1;
Pas 4. Retornar $T = trim(MSpT(G, V_1))$.

Algorisme 2: L'heurística del graf distància (SPH)

L'objectiu dels dos primers passos de l'algorisme és construir un $MSpT(D, S)$, on D és el graf distància de G. Les arestes d'aquest arbre són substituïdes per camins de cost mínim en el *Pas 3*. Tot i que G_1 no té per què ser un arbre, no resulta difícil seleccionar els camins de cost mínim de manera que ho sigui [93]. En qualsevol cas, el *Pas 4*, que és igual al *Pas 5* de la SPH, assegura que els vèrtexs de la solució final estaran connectats de manera òptima.

L'any 1988, Mehlhorn [73] va proposar una implementació de la DNH amb complexitat $O(|A| + |V| \log |V|)$, la qual cosa el converteix en l'algorisme d'aproximació amb menor complexitat.

2.4.3 L'heurística de Zelikowsky (ZH)

El ZH [98] és famós per ser el primer algorisme d'aproximació per al PSG amb FRA constant per sota de 2. Es basa en trobar una aproximació per a una versió restringida del problema en la qual els vèrtexs d'Steiner poden tenir com a molt grau tres.

Denotem per $MSpT_3(\overline{G}, V_S, v)$ l'arbre generador de cost mínim per a $V_S \cup \{v\}$ del graf $\overline{G}$ restringit a què el grau de v és no superior a 3.

En cada iteració es selecciona un no-terminal que maximitza la millora de cost, o guany, respecte l'arbre d'Steiner que s'està construint, en el qual tots els vèrtexs d'Steiner tenen grau tres. Aquest arbre es pot obtenir substituint les dues arestes de cost zero que s'introdueixen en el *Pas 3* pel vèrtex no-terminal que ha estat seleccionat

Pas 1. $\overline{D}$:= D, $V_1 := \emptyset$;
Pas 2. Seleccionar $v \in V - (S \cup V_1)$ tal que
$guany(v) = |MSpT(\overline{D}, S)| - |MSpT_3(\overline{D}, S, v)|$ és màxim;
Pas 3. Si $guany(v) > 0$ aleshores
$V_1 := V_1 \cup \{v\}$;
Posar a zero els costos de les arestes (v_i, v_j) i (v_j, v_k) a $\overline{D}$,
on v_i, v_j i v_k són els tres veïns de v en el $MSpT_3(\overline{D}, S, v)$;
Tornar al *Pas 2*;
Pas 4. Retornar $T = trim(DNH(G, S \cup V_1))$.

Algorisme 3: L'heurística de Zelikowsky (ZH)

en aquest pas i les tres arestes incidents a aquest vèrtex. L'arbre resultant del *Pas 5* s'obté a partir d'una execució de la DNH en què els vèrtexs que han estat inclosos a V_1 són considerats com si fossin terminals.

Amb l'excepció de l'arbre generador de cost mínim que es calcula en la primera iteració, la resta d'arbres poden ser obtinguts a partir dels anteriors. És per això que, en la implementació que proposen Duin i Voß [28], la complexitat de l'algorisme és $O(|S|(|A| + |V| \log |V|)$ i està dominada pel càlcul dels camins de cost mínim entre els vèrtexs terminals i la resta de vèrtexs del graf.

2.5 Mètodes competitius

Una de les possibles maneres de comparar els mètodes heurístics és mitjançant la utilització de jocs de proves i l'experimentació. Un cop implementat l'algorisme, el rendiment es prova utilitzant instàncies generades de manera controlada, o jocs de proves normalment accessibles via *internet*. En el primer cas, es pot tenir més cura dels experiments decidint, per exemple, el tipus i mida dels grafs, la probabilitat d'existència d'una aresta, la distribució dels costos de les arestes, etc. En el segon, però, es poden comparar els resultats que s'obtenen amb els dels altres mètodes que han utilitzat els mateixos grafs de prova.

Els grafs de proves més utilitzat per al PSG van ser generats per Beasley [10] i

són els que utilitzarem en els capítols 3 i 4. Els 78 grafs que en formen part són tots poc densos (*sparse*) i han estat generats de forma aleatòria. Tot i estar reconegut l'inconvenient que representa que tots els grafs tinguin unes característiques similars, segueixen sent utilitzats en la gran majoria de treballs de recerca.

Els mètodes que presentem en aquesta secció són d'execució múltiple i utilitzen algunes de les estratègies que hem comentat en l'apartat 2.3.2. Alguns d'aquests mètodes són els que estan aconseguint millors resultats experimentals en jocs de prova. Posteriorment, aquests resultats seran comparats amb els que obtenen els mètodes que proposem en els dos capítols següents.

2.5.1 Els mètodes repetitius de Winter i Smith

Els quatre mètodes següents van ser proposats per Winter i Smith [93] i estan basats en una estratègia repetitiva que utilitza la SPH:

- SPH-S: en cadascuna de les execucions es selecciona un vèrtex terminal diferent com a vèrtex inicial s_0 del *Pas 1* (veure l'algorisme en la secció anterior). En total, la SPH és executada $|S|$ vegades, tot i que, calculant els camins necessaris una única vegada, la complexitat esdevé $O(|S|(|A| + |V|\log|V| + |S||V|))$.

- SPH-V: en cadascuna de les execucions es selecciona un vèrtex diferent, que no té per què ser terminal, com a vèrtex inicial s_0 del *Pas 1*. En total, la SPH és executada $|V|$ vegades tot i que, calculant els camins necessaris una única vegada, la complexitat esdevé $O(|V|(|A| + |V|\log|V| + |S||V|))$.

- SPH-zS: en cadascuna de les execucions és parteix d'un subarbre inicial format per un camí diferent: el camí més curt que va des d'un vèrtex terminal z, que és el mateix en totes les execucions, a un altre vèrtex terminal, que va canviant en cada execució. En total, la SPH és executada $|S| - 1$ vegades i, per tant, la complexitat és igual a $O(|S|(|A| + |V|\log|V| + |S||V|))$.

- SPH-SS: es tracta d'executar un cop la SPH-zS per a cada terminal z. En total s'executa la SPH $|S|(|S| - 1)/2$ vegades. La complexitat del mètode és $O(|S|(|S| - 1)(|A| + |V|\log|V| + |S||V|))$, si es calculen els camins un únic cop.

En l'article [93] es descriuen els experiments realitzats per a comparar el rendiment d'aquests mètodes, i també el d'alguns algorismes d'un pas com la SPH, la DNH i l'ADH. Els grafs utilitzats tenen característiques diverses i es generen de forma aleatòria, variant la densitat de les arestes i la distribució dels costos associats a aquestes.

Quan es compara la qualitat de les solucions obtingudes, la SPH-V i la SPH-SS superen clarament la resta de mètodes. A certa distància hi ha la SPH-S, la SPH-zS i l'ADH. Els resultats de la SPH són inferiors als d'aquests mètodes, però superiors als de la DNH. Comparant aquesta última amb la SPH-V, resulta que després de 4507 proves diferents, la SPH-V obté millor resultat que la DNH 1378 vegades, mentre que la DNH supera la SPH-V una única vegada. D'altra banda la SPH supera la DNH 394 vegades, mentre que la DNH supera la SPH 53 vegades. En l'article no es considera la possibilitat de reduir la complexitat dels mètodes repetitius a base de calcular un únic cop els camins necessaris.

2.5.2 Els mètodes iteratius d'Alexander i Robins

L'article d'Alexander i Robins [2] és el darrer d'una sèrie de treballs [61, 1] que han anat conformant l'estratègia iterativa *greedy*, tal i com ha estat descrita en l'apartat 2.3.2. Aquests treballs estan orientats a minimitzar el retard en la propagació del senyal a l'hora de dissenyar circuits integrats (VLSI), problema que pot ser plantejat com a variant del conegut *routing* [86], també intractable. Alguns dels mètodes heurístics que s'han proposat per a aquest problema es basen en la obtenció de diversos arbres d'Steiner [62]. Sense entrar en detall en aquests aspectes, anem a destacar alguns dels elements de l'article d'Alexander i Robins que fan referència al PSG.

En concret, en aquest article es comparen experimentalment dos mètodes basats en una estratègia iterativa *greedy*: un d'ells fa servir la DNH (IDNH) i l'altre utilitza el ZH (IZH). Tots dos algorismes es comparen a partir dels resultats que obtenen utilitzant instàncies de prova típiques en el VLSI. Els resultats també són comparats amb els de les versions no iteratives de tots dos algorismes. L'IZH és el que obté millors resultats, seguit a poca distància per L'IDNH. Força més lluny queden els

resultats del ZH, mentre que els de la DNH són molt inferiors als de la resta de mètodes. Com es pot veure, la qualitat dels resultats està directament relacionada amb la complexitat computacional dels mètodes.

D'altra banda, Esbensen [30] mostra els resultats que obté l'IDNH utilitzant els grafs de Beasley. Els resultats d'aquest mètode són inferiors, en temps i en qualitat, als de l'algorisme genètic que proposa Esbensen en el mateix treball.

2.5.3 Els algorismes de Duin i Voβ

A grans trets, els algorismes que proposen Duin i Voβ [28] s'obtenen de la concatenació de dos mètodes d'execució múltiple: el primer basat en una estratègia iterativa *greedy* i el segon basat en una estratègia iterativa recurrent. L'arbre resultant d'aplicar el primer mètode és utilitzat per a generar una nova instància inicial del problema per al segon.

Per al primer dels dos mètodes es consideren dues possibilitats diferents: una variant del *greedy* de Minoux que anomenem l'*heurística de Duin i Vos* (DVH), i el ZH. A diferència del *greedy* de Minoux, el DVH evita recalcular alguns dels costos per tal de reduir la complexitat. De manera similar, en l'article es descriu el ZH a partir d'una restricció del DVH, aspecte aquest doblement interessant ja que, d'una banda, possibilita una reducció de la complexitat del ZH i, de l'altra, permet conjecturar que la FRA de la DVH, així com la del *greedy* de Minoux, és no superior a la FRA de valor 11/6 demostrada per Zelikovsky. Els resultats experimentals utilitzant el DVH acostumen a ser millors que els que produeix el ZH, la qual cosa sembla apuntar cap la veracitat de la conjectura. La complexitat del DVH és $O(|V|(|A| + |V|\log|V|))$ i la del ZH $O(|S|(|A| + |V|\log|V|))$, en tots dos casos dominada pel càlcul de camins de cost mínim entre els vèrtexs.

Per al segon dels mètodes s'utilitza un algorisme que proposen els mateixos autors i que té aspectes en comú amb les variants de la SPH que proposem en el capítol 5. L'algorisme és descriu més en detall en l'apartat 5.4.5, on es compara amb les variants esmentades.

De manera semblant a les propostes que es fan en el capítol 5, l'article de Duin

i Voβ té com a origen l'estudi del funcionament de la SPH i, tot i que no ha tingut una influència directa en els continguts d'aquesta tesi, si que ens ha obligat a matisar alguns dels aspectes de la memòria. Per exemple, l'article proposa una demostració segons la qual la SPH té la mateixa FRA que la DNH i, per tant, la demostració que proposem en l'apartat 5.2 ha passat a ser una forma alternativa, posterior, de demostrar el mateix. També, la supressió de vèrtexs de grau 2 és utilitzada per Duin i Voβ en el DVH i, per tant, no és exclusiva dels mètodes que proposem en el capítol 3. Finalment, hem classificat un dels mètodes que s'utilitzen en l'article dins del grup de les estratègies iteratives recurrents. D'aquesta manera, l'algorisme que proposem en l'apartat 3.2 deixa de ser el primer mètode del que tenim constància que forma part d'aquesta categoria. Creiem important aclarir, però, que la utilització que fan Duin i Voβ d'aquesta estratègia no és, en cap cas, un element central en els algorismes que proposen.

2.5.4 L'algorisme genètic d'Esbensen (EGA)

L'algorisme genètic d'Esbensen (EGA) [30, 31] és el punt de partida del capítol 4 i, per tant, serà considerat de manera detallada més endavant. A més de l'algorisme genètic pròpiament dit, aquest algorisme inclou dues tècniques de complement amb l'objectiu de millorar el rendiment. D'una banda, fa servir rutines de reducció per tal de reduir la mida dels grafs de prova i, d'aquesta manera, millorar el rendiment del mètode. De l'altra, incorpora un *hill-climbing* basat en l'arbre generador de cost mínim, per tal de millorar el cost de l'aproximació resultant de l'algorisme genètic.

Utilitzant les instàncies del joc de proves proposat per Beasley, el rendiment de l'EGA es compara amb el de diversos mètodes heurístics ja coneguts. El procediment que segueix ha servit de guia per a les proves que realitzarem en els capítols 3 i 4, i és similar el que havien utilitzat anteriorment Kapsalis i Rayward-Smith [63] per a fer les proves amb el seu algorisme genètic per al PSG.

2.5.5 Branch and Cut

Recentment s'han presentat dues noves versions de *branch and cut* per al PSG: la de Lucena i Beasley [72] (LBBC), i la de Koch i Martin [67] (KMBC). Sembla ser que els primers a aplicar aquest mètode al PSG van ser Chopra, Gorres i Rao [18]. Els dos mètodes han estat publicats l'any 1998 (en números consecutius de la revista *Networks*), utilitzen la SPH, i es diferencien bàsicament en els aspectes referents a la formulació del problema en termes de programació lineal. Per la nostra part, no entrarem en detall en aquests aspectes, ni tampoc considerarem la manera en què es recorre l'arbre de recerca per tal d'explorar totes les opcions possibles i assegurar, quan no hi ha limitacions de temps, la localització d'una solució òptima. Anem només a destacar breument el paper que tenen la programació lineal i l'algorisme heurístic per al PSG en tots dos mètodes.

La programació lineal permet descriure la instància del problema com un conjunt d'equacions i inequacions que cal optimitzar. La majoria de les incògnites de les equacions prenen un valor en funció de la presència o no d'una aresta en la solució. Per exemple, la variable x_a, associada a l'aresta a, tindrà valor 1 o 0 en funció de si a forma part o no de l'arbre òptim. Com que normalment no és possible trobar una solució entera òptima en temps raonable, es fan servir *relaxacions* per tal d'aconseguir una fita inferior del problema d'optimització plantejat. Tot i que la solució derivada d'aquestes relaxacions no és entera, es considera que és més probable que l'aresta a formi part d'un arbre òptim en els casos en que x_a té un valor proper a 1, que en els casos en què el valor d'aquesta variable s'acosta a 0.

El mètode heurístic és l'encarregat de generar les aproximacions a l'arbre òptim. La informació obtinguda de la relaxació del problema s'utilitza per alterar les probabilitats de que una aresta formi part o no de l'arbre òptim. En concret, en tots dos mètodes el cost associat a l'aresta a es multiplica per $1 - x_a$. Cada aproximació proporciona una fita superior del cost de la solució òptima la qual, conjuntament amb la fita inferior, dóna una relació dinàmica de l'error comès, permetent aturar el procés.

En els mètodes basats en el *branch and cut* les rutines de reducció tenen un paper

molt important, ja que permeten reduir el nombre d'incògnites en les equacions. Per això, tant l'LBBC com el KMBC utilitzen un ampli conjunt d'aquestes rutines per tal de millorar els resultats.

En tos dos articles es dóna una extensa informació dels resultats experimentals que produeixen tots dos algorismes. L'LBBC utilitza únicament els grafs de Beasley. En canvi, el KMBC utilitza, a més d'aquests grafs, un ampli conjunt de jocs de prova de diversa procedència i característiques variades. A diferència de l'LBBC, el KMBC obté la solució òptima per a tots els grafs de Beasley. Tot i això, resulta difícil comparar els dos mètodes, sobretot si tenim en compte la dependència del temps d'execució en l'obtenció dels resultats, i el fet que s'han utilitzat condicions molt diferents per a portar a terme les proves.

D'altra banda, la intenció dels autors d'obtenir una solució òptima dificulta la comparació d'aquests mètodes amb altres mètodes orientats a produir bones aproximacions (veure l'apartat 4.8.1).

Capítol 3

Un mètode heurístic per al PSG

El principal objectiu d'aquest capítol és dissenyar un algorisme no evolutiu per al PSG que pugui competir experimentalment amb l'algorisme genètic proposat per Esbensen (EGA) [30] i, per tant, amb els mètodes heurístics que estan produint actualment millors resultats en la pràctica. L'algorisme que presentem (GBA) [45] és d'execució múltiple, està basat en una estratègia iterativa recurrent (veure l'apartat 2.3.2) que utilitza la SPH [88] com a algorisme de base, i incorpora diverses funcions per tal de millorar el rendiment. El propòsit del capítol se centra tant en la utilització d'aquesta estratègia, de la qual el mètode proposat n'és un dels pioners, com en les funcions, algunes de les quals poden ser utilitzades per a millorar els resultats que aconsegueixen altres mètodes.

En la primera secció del capítol s'introdueixen alguns elements previs necessaris per al seu seguiment. En la secció 3.2 es proposa un nou mètode iteratiu (HCSPH), basat en la SPH, que segueix fidelment la que hem anomenat estratègia iterativa recurrent. El GBA és presentat en la secció 3.3 a base d'incorporar diverses funcions al mètode anterior. Cadascuna d'aquestes funcions és analitzada, posant especial èmfasi en descriure la seva influència en el comportament global de l'algorisme. En la secció 3.4 es mostren els resultats experimentals obtinguts per l'algorisme usant els grafs de Beasley [10] de la *OR library*, i es comparen amb els d'EGA. Finalment, en la darrera secció, es treuen conclusions de l'algorisme que ha estat presentat.

3.1 Preliminars

En l'estratègia iterativa recurrent, proposada en el capítol anterior, un algorisme d'un pas és executat de manera iterativa fent servir els vèrtexs de l'arbre resultant d'una iteració com a conjunt de vèrtexs terminals de la següent. Per tant, en cada iteració hi ha una nova instància del problema a resoldre, que té un nou conjunt de terminals que inclou el conjunt inicial. Utilitzem el terme *falsos terminals* per a referir-nos als vèrtexs que estan sent considerats com a terminals en la iteració, però que en realitat no ho són. D'aquesta manera, l'objectiu en cada iteració és trobar una aproximació a un $MStT(G, S \cup F)$ per al graf $G(V, A, c)$, on S és el conjunt de terminals i F és el conjunt de falsos terminals.

La presència de falsos terminals obliga a modificar lleugerament la SPH en relació a la descripció que s'ha donat en l'apartat 2.4.1. En concret, el problema es troba en el tractament que han de rebre aquests vèrtexs quan són fulla ja que, si són eliminats, s'obté una millor aproximació per al problema inicial. Les modificacions que introduïm en la SPH, permeten que els falsos terminals siguin considerats corbates i, per tant, puguin ser eliminats. En l'algorisme resultant, que es mostra a continuació, $TM(G, S \cup F)$ indica l'execució de la versió originària de la SPH proposada per Takahashi i Matsuyama [88] (TM), que es correspon als 4 primers passos de la SPH. La funció $trim(T)$, que traduiríem per esporgar, se n'encarrega d'eliminar els vèrtexs corbata de l'arbre, que inclouen no-terminals i falsos terminals. Finalment, el *Pas 2* es correspon al darrer pas de la SPH.

Pas 1. $T_1 := TM(G, S \cup F)$, $T_2 := trim(T_1)$;
Pas 2. $T_3 := MSpT(G, vertexs(T_2))$, $T := trim(T_3)$;

Algorisme 4: Versió de la SPH en què s'eliminen els falsos terminals

Podem veure que T_1 i T_2 seran iguals sempre que no hi hagi falsos terminals i, per tant, aquesta versió de l'algorisme es comporta de manera idèntica a la SPH en aquests casos.

3.2 *Hill-Climbing* SPH (HCSPH)

L'HCSPH és un mètode d'execució múltiple, basat en una estratègia iterativa recurrent, que utilitza la SPH com a algorisme d'un pas:

Pas 0. $V_S := S$;
Pas 1. $T_1 := TM(G, V_S)$, $T_2 := trim(T_1)$;
Pas 2. $T_3 := MSpT(G, vertexs(T_2))$, $T_4 := trim(T_3)$;
Pas 3. **si** $|T_1| > |T_4|$ **aleshores**
$V_S := vertexs(T_4)$, **anar al** Pas 1
si no retornar T_1.

Algorisme 5: HCSPH

L'algorisme "hereta" la FRA de la SPH en la primera iteració. El nom *hill-climbing* ha estat escollit pel fet que $|T_1| \geq |T_2| \geq |T_3| \geq |T_4|$ i, també, perquè l'algorisme pot ser usat amb l'objectiu de millorar un arbre aproximat proporcionat per un mètode heurístic qualsevol. En aquest cas, el conjunt de vèrtexs d'aquest arbre haurà de ser utilitzat en comptes de S en el *Pas 0.*

Anàlisi de la complexitat

La complexitat[1] en el pitjor cas d'una iteració de l'algorisme és $O(|V_S|(|A| + |V|\log|V|))$, i és deguda al càlcul dels camins de cost mínim necessaris. Com que V_S pot tenir tots els vèrtexs, la complexitat en el pitjor cas d'una única iteració esdevé $O(|V|(|A|+|V|\log|V|))$ i, per tant, és la mateixa que la del càlcul dels camins òptims entre cada parella de vèrtexs. És per això que aquest càlcul es pot fer una única vegada abans de començar l'execució del procés iteratiu. Si es disposa dels camins, la complexitat en el pitjor cas del càlcul de TM en el *Pas 1* esdevé $O(|V_S||V|)$. Cal afegir-hi la complexitat del càlcul de MSpT en el *Pas 2* que és, utilitzant l'algorisme de Prim, $O(|A_2| + |V_2|\log|V_2|)$, essent V_2 el conjunt de vèrtexs de T_2, i A_2 el conjunt d'arestes del subgraf de G induït per V_2.

[1] Al llarg de tot el capítol, els càlculs de la complexitat es realitzen considerant que s'utilitza l'estructura de dades proposada per Fredman i Tarjan [36], i sense tenir en compte la reducció de la complexitat de la SPH que proposem en el capítol 5.

La complexitat en el pitjor cas de l'HCSPH és $O((|V|+k)(|A|+|V|\log|V|)+k|V|^2)$, on k és el nombre total d'iteracions. El primer producte inclou el càlcul dels camins, i les execucions de la MSpT en el procés iteratiu, considerant la situació extremadament desfavorable en què V_2 conté sempre tots els vèrtexs. El segon terme és degut al càlcul de TM en el *Pas 1*, i també considera la situació especialment desfavorable en què en totes les iteracions $|V_S|$ és igual a $|V|$. Com que en cada iteració es millora el cost de la solució, podem assumir que k no serà mai més gran que $|V|$ —o $c|V|$ per a una constant c petita— i, per tant, la complexitat de l'HCSPH és $O(|V|(|A|+|V|\log|V|+|V|^2))$ en el pitjor cas, i $O(|V|(|A|+|V|\log|V|+|S|^2))$ en el millor.

3.3 GBA: un mètode heurístic per al PSG

Només hi ha dues maneres a partir de les quals es pot obtenir un arbre amb cost menor després d'una iteració de l'HCSPH: mitjançant la inclusió de nous vèrtexs d'Steiner durant l'execució de TM, o mitjançant l'eliminació de corbates amb la funció *trim*. Totes dues maneres depenen de la implementació, de l'ordre dels vèrtexs i, també, del mètode utilitzat per a calcular els camins de cost mínim. El nostre objectiu és introduir noves funcions a l'algorisme per tal de disminuir aquestes dependències, reduir l'espai de recerca de solucions i incrementar les possibilitats de que sorgeixin nous vèrtexs d'Steiner i corbates.

En els apartats següents comentem les funcions que han estat incorporades a l'HCSPH fins a obtenir el GBA. La versió final de l'algorisme, on queda reflectida la disposició exacta d'aquestes funcions, es presenta en el darrer d'aquests apartats.

3.3.1 Reordenar els vèrtexs

Normalment, a partir d'un subconjunt U de vèrtexs, es poden trobar molts arbres que són $TM(G,U)$ i també uns quants que són $MSpT(G,U)$. En canvi, com que la implementació d'aquests algorismes acostuma a ser determinista, només un d'aquests arbres serà retornat com a solució. Fins i tot, el més probable és que implementacions diferents d'aquestes funcions retornin arbres diferents, tot i ser executades per a una

mateixa instància. I és que, tal i com hem apuntat en la secció 2.3.2, algunes de les decisions que es prenen no estan especificades de forma explícita en l'algorisme i, per tant, quan hi ha més d'una possibilitat es resolen de forma arbitrària. Des d'aquesta perspectiva, podem veure la funció TM —o MSpT— com un mecanisme que selecciona arbitràriament un arbre del conjunt d'arbres que poden ser considerats $TM(G,U)$ —o $MSpT(G,U)$.

Tot i que l'arbitrarietat en la selecció d'un arbre no sembla tenir massa importància quan la funció és executada una única vegada —al cap i a la fi, no es disposa de cap criteri que permeti establir quina és la millor decisió— sí que en té en els mètodes d'execució múltiple. Per exemple, depenent de la implementació concreta de la funció, un vèrtex d'etiqueta menor pot ser sempre seleccionat, en detriment d'un vèrtex d'etiqueta més gran que compleixi les mateixes condicions. Tenint en compte que els mètodes d'execució múltiple exploren diverses solucions dins d'un espai de solucions accessibles, podem considerar que aquesta exploració es fa d'una forma esbiaixada, donant sempre prioritat a uns vèrtexs respecte d'altres, sense que hi hagi cap criteri definit. Amb el propòsit de disminuir aquest biaix difícil de controlar i, a la vegada, incrementar la capacitat d'exploració de l'HCSPH, hem incorporat la funció següent al GBA:

- *reordenar*(G): retorna un graf isomorf a G. Les etiquetes associades als vèrtexs de G es tornen a assignar de forma aleatòria entre tots els vèrtexs.

Aquesta funció és executada just abans de cada execució de TM i de MSpT. L'efecte produït equival a la introducció d'indeterminisme en aquestes funcions i, per tant, poden retornar arbres diferents en dues execucions per a la mateixa instància. Amb la funció *reordenar*, la prioritat arbitrària en l'elecció dels vèrtexs desapareix, ja que el sentit d'aquesta prioritat es va alternant de forma aleatòria en cada iteració. A més a més:

- Com que arbres diferents poden ser obtinguts a partir d'un mateix conjunt de terminals, augmenta el nombre d'arbres que poden ser accedits de forma efectiva pel GBA.

- Es redueix la tendència a retornar arbres semblants en iteracions consecutives que, generalment, acostumen a tenir conjunts de terminals molt similars. Per tant, s'incrementen les possibilitats d'aparició de noves corbates i de nous vèrtexs d'Steiner.

- Com que dues iteracions per a la mateixa instància poden produir solucions diferents, les condicions de parada del GBA poden ser millorades en relació a les de l'HCSPH. El GBA es para quan no es produeix cap millora després d'un nombre màxim d'iteracions fixat ($nMaxIter$). D'aquesta manera, el nombre de solucions candidates que és explorat és generalment més gran i, per tant, augmenten les possibilitats d'aconseguir una solució millor.

- S'elimina parcialment la dependència que té l'algorisme de les etiquetes dels vèrtexs. (Tal i com veurem més endavant, aquesta dependència no s'evita en el càlcul dels camins). Per tant, els resultats experimentals que obté el GBA per a uns grafs de prova, son vàlids per als seus grafs isomorfs.

Creiem important remarcar l'efecte positiu que una simple reordenació de vèrtexs pot produir en l'HCSPH, el qual segur que pot ser traslladat a altres algorismes.

Com que $HCSPH(G,S)$ és igual a $HCSPH(G, vertexs(HCSPH(G,S))$, no té sentit la utilització de l'HCSPH com a algorisme de base en una estratègia iterativa recurrent. En canvi, introduint una simple reordenació de vèrtexs, el mètode recurrent pot produir millores. És a dir,

$$|HCSPH(G,S)| \geq |HCSPH(reordenar(G), vertexs(HCSPH(G,S)))|.$$

Per tant, la solució que retorna el nou mètode derivat d'incorporar la funció *reordenar* a l'HCSPH és, en el pitjor dels casos, la mateixa que la de l'HCSPH. Aquest nou mètode pot accedir, si més no com a solucions intermèdies, a tots els arbres que retorna l'HCSPH. En canvi, hi ha un subconjunt de solucions privilegiades, en el sentit que milloren el resultat de l'HCSPH, que només poden ser retornades per la nova versió recurrent.

Anem a veure què passa quan modifiquem l'HCSPH mitjançant la inclusió de la funció *reordenar* després de cada iteració (HCSPHr), tal i com s'ha fet en el GBA. En

principi, resulta difícil de comparar els costos de $HCSPH(G,S)$ i $HCSPHr(G,S)$, tot i que, sobretot en les fases finals de l'execució, tot sembla indicar que l'HCSPHr incrementa les possibilitats de que sorgeixin noves corbates i vèrtexs d'Steiner. Deixant de banda aquest fet, podem apreciar que, a diferència de l'HCSPH, l'HCSPHr pot accedir a totes les solucions de la nova versió recurrent que hem proposat a dalt i, per tant, pot retornar també aquelles solucions privilegiades que hem esmentat. Conseqüentment, tot sembla indicar que, a més dels efectes abans mencionats, la funció *reordenar* produeix per si sola una millora en el rendiment del GBA.

3.3.2 Eliminar els no-terminals de grau dos

La incorporació de nous vèrtexs no-terminals com a vèrtexs d'Steiner és un dels aspectes clau per a millorar el cost d'una aproximació donada. Tot i això, no tots els vèrtexs d'Steiner tenen per què contribuir a millorar el cost de la solució. Aquest és el cas, per exemple, dels vèrtexs d'Steiner que tenen grau dos i que formen part d'un arbre al final d'una iteració de l'algorisme. La funció següent se n'encarrega d'eliminar aquests vèrtexs:

- $eGrau2(T)$: retorna el subconjunt de vèrtexs de l'arbre de Steiner T un cop els no-terminals de grau dos han estat eliminats.

Aquesta funció és executada un únic cop al final de cada iteració de l'algorisme i, per tant, el conjunt de vèrtexs que retorna és el conjunt de terminals de la següent iteració. Aquests són alguns dels efectes que la funció produeix en el GBA:

- Si $T' = MSpT(D, eGrau2(T))$, on D és el graf distància de G, aleshores

$$|T| \geq |T'| \geq |TM(G, vertexs(T'))|.$$

  Per tant, la funció introdueix noves possibilitats per a millorar el cost de l'arbre en la següent iteració.

- Els conjunts de terminals a l'inici de cada iteració passen a tenir, com a molt, $|S| - 2$ vèrtexs[2]. D'aquesta manera, es produeix una reducció important del nombre de solucions possibles que cal considerar.

[2] $\exists S' \in V$ amb $|S'| \leq |S| - 2$ tal que $SPH(G, S \cup S')$ és $MStT(G,S)$.

- Com que la complexitat en el cas pitjor de T_M depèn del nombre de vèrtexs terminals, la velocitat de l'algorisme es veu incrementada de forma significativa. (Veure l'anàlisi de la complexitat en la pàgina 53).

A més d'aquests avantatges, *eGrau2* és també un mètode selectiu de "sacsejar" l'arbre, tal i com és comenta en l'apartat següent.

3.3.3 Sacsejar l'arbre

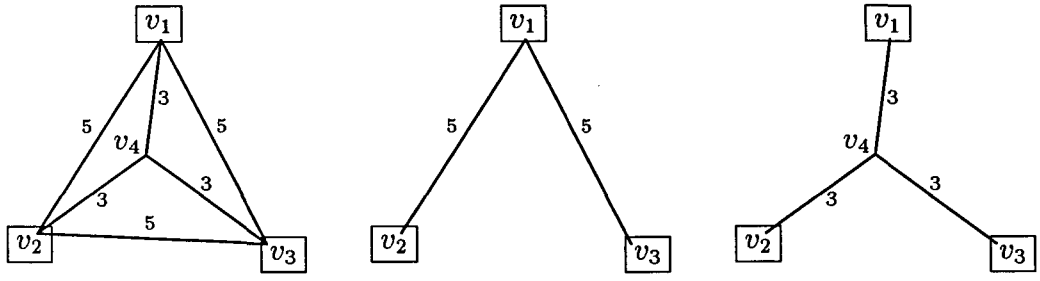
La inclusió en el GBA d'una funció que introdueix petites modificacions de forma aleatòria en el conjunt de terminals sembla, a primera vista, una opció discutible. Hem anomenat a aquesta funció *sacsejar* per tal de fer referència a l'objectiu que li atribuïm: la de "sacsejar" l'arbre per tal de que alguns vèrtexs "caiguin" i, d'aquesta manera, "deixin espai" per a incorporar-ne de nous:

- $sacsejar(vertexs(T))$: retorna el conjunt de vèrtexs de T després d'introduir-hi o eliminar-ne, de forma aleatòria, alguns no-terminals. En concret, cada vèrtex no-terminal té una probabilitat petita de passar a formar part —o deixar de formar part— del conjunt resultant.

Es evident que el nombre d'arbres que poden ser accedits pel GBA gràcies a aquesta funció es veu clarament incrementat, tot i que aquest fet no té per què representar cap avantatge en aquest cas concret. De tota manera, hi ha una propietat important que es introduïda a l'algorisme mitjançant aquesta funció:

- Tal i com es pot veure en l'exemple de la figura 2, hi ha alguns vèrtexs que, tot i no ser intermedis de cap camí de cost mínim, poden formar part d'un arbre d'Steiner òptim. Aquests vèrtexs no són tinguts en compte en el procés d'exploració de l'HCSPH, ja que no poden ser incorporats a cap arbre mitjançant la SPH. (En l'apartat següent veurem que fins i tot vèrtexs intermedis de camins de cost mínim també són descartats, d'una forma totalment arbitrària, del procés d'exploració.) La funció *sacsejar* fa que aquests vèrtexs siguin tinguts

en compte durant aquest procés i, per tant, puguin formar part d'una solució. A més, permet garantir que l'algorisme pot retornar una solució òptima[3].



$G = (V, E, c)$, $S = \{v_1, v_2, v_3\}$.

T_1 és $SPH(G, S)$ i, també, $HCSPH(G, S)$.

T_2 és $MStT(G, S)$.

Figura 2: Tot i formar part de l'arbre de Steiner òptim T_2, ni la SPH ni tampoc l'HCSPH consideren la possibilitat d'incorporar v_4 a una solució. El motiu és que v_4 no és vèrtex intermedi de cap camí de cost mínim.

La funció *sacsejar* no hauria de modificar l'estructura bàsica de l'arbre de manera important. És per això que la probabilitat de que un no-terminal passi a formar part —o deixi de formar part— del conjunt de vèrtexs de l'arbre és calculada a partir d'un nombre esperat de *moviments* ($nEspMov$). D'aquesta manera, la probabilitat $P_m(v)$ de "moure" un vèrtex és calcula de la forma següent:

$$P_m(v) := \frac{nEspMov}{|V| - |S|} \quad \forall v \in V - S.$$

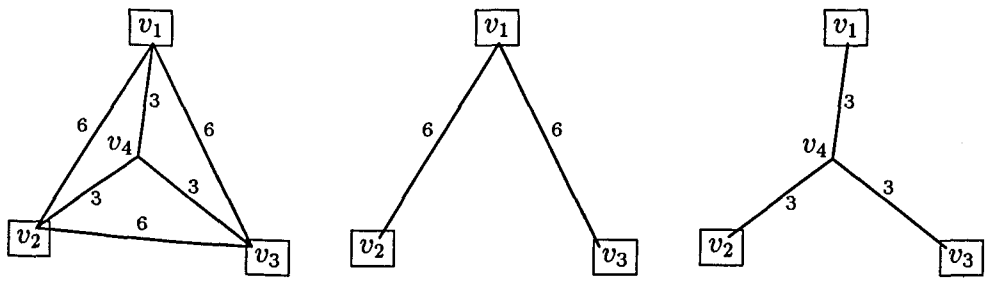
La funció *sacsejar* és la més inestable de totes les que formen part del GBA, ja que permet explorar solucions que tenen un cost superior al de la solució anterior i, per tant, l'algorisme deixa de ser un *hill-climbing*. Hem contemplat la possibilitat de només sacsejar l'arbre en aquells casos en què no s'ha produït cap millora respecte l'arbre de l'anterior iteració. Els resultats experimentals, però, posen de manifest una lleugera millora quan la funció s'executa un cop en cada iteració.

Per a concloure aquest apartat, noteu que és molt possible que la utilització de mecanismes més selectius per a decidir quins vèrtexs han de ser inclosos —o exclosos—

[3]La probabilitat d'aconseguir una solució òptima és més gran que 0.

del conjunt de terminals, com ara la funció *eGrau2*, permetin millorar l'algorisme. Aquest mecanismes podrien ser incorporats en comptes de la funció *sacsejar* o, també, podrien ser utilitzats com a complement d'aquesta. En la nostra opinió, una funció basada en una estratègia iterativa *greedy*, o en alguna variant més eficient d'aquesta, resultaria molt indicada per aquest propòsit.

3.3.4 Donar prioritat als camins de més d'una aresta



$G = (V, E, c)$, $S = \{v_1, v_2, v_3\}$.

T_1 és $SPH(G, S)$ i, també, $HCSPH(G, S)$

T_2 és $MStT(G, S)$

Figura 3: Tot i formar part de l'arbre de Steiner òptim T_2, i d'un camí de cost mínim entre cada parella de terminals, el vèrtex v_4 no serà tingut en compte si el mètode per a calcular els camins selecciona, en cas d'igualtat, el camí amb menys arestes.

El rendiment de la SPH, així com el de la majoria d'algorismes heurístics en què s'utilitzen camins, depèn en gran mesura de la manera en què aquests camins són calculats. Els algorismes que s'utilitzen normalment —Floyd [33], Dijkstra [23], etc.— són deterministes i, per tant, retornen sempre un mateix camí quan n'hi ha més d'un amb el mateix cost. D'aquesta manera, tal i com es mostra en la figura 3, els vèrtexs intermedis dels camins que no són mai seleccionats deixen de ser tinguts en compte, d'una forma arbitrària, durant la fase d'exploració de l'HCSPH.

A diferència del que ha estat esmentat en l'apartat 3.3.1, l'impacte negatiu que

tindria en la complexitat[4] de l'algorisme ens ha fet descartar la possibilitat de calcular cada vegada els camins i, d'aquesta manera, aprofitar els avantatges de la funció *reordenar*. Per tant, assumint l'efecte negatiu que poden tenir aquestes arbitrarietats durant el procés d'exploració, hem aplicat un criteri que, si bé no soluciona el problema, en la pràctica tendeix a millorar els resultats de l'algorisme:

- Donar prioritat als camins que tenen vèrtexs intermedis respecte als camins del mateix cost formats per una única aresta.

Aquest criteri, que s'inclou en les implementacions dels mètodes que calculen els camins, no altera la complexitat d'aquests algorismes i, per tant, sembla recomanable per a molts dels mètodes heurístics per al PSG en què s'utilitzen camins [57].

3.3.5 GBA

L'algorisme següent mostra l'aspecte final del GBA.

Pas 0. $V_S := S$;
Pas 1. $T_1 := TM(reordenar(G), V_S)$, $T_2 := trim(T_1)$;
Pas 2. $T_3 := MSpT(reordenar(G), vertexs(T_2))$, $T_4 := trim(T_3)$;
Pas 3. **si** $noParar(nMaxIter)$ **aleshores**
$V_S := sacsejar(eGrau2(T_4))$, **anar al** Pas 1
si no retornar l'arbre de cost menor trobat.

Algorisme 6: GBA

Anàlisis de la complexitat

Mitjançant la funció *eGrau2* el nombre de vèrtexs de V_S és sempre inferior a $2|S|$. Per tant, la complexitat en el pitjor cas de cada execució de TM esdevé $O(|S||V|)$ (veure la complexitat de l'HCSPH en la pàgina 45). A més, el més probable és que el nombre de vèrtexs que contindrà T_2 sigui menor i, per tant, el temps d'execució del *Pas 2* també disminuirà.

[4]No s'han tingut en compte les millores en la complexitat de la SPH que proposem en el capítol 5.

Hem tingut en compte la possibilitat de suprimir el *Pas 2* per tal de millorar el temps d'execució de l'algorisme. En aquest cas, T_4 hauria de ser substituït per T_2 en el *Pas 3*. Hi ha dos motius principals que ens han decidit a descartar aquesta possibilitat. D'una banda, quan s'aplica *eGrau2* a un arbre que, a l'igual que T_2, pot no ser generador de cost mínim, alguns dels vèrtexs que produeixen millora poden ser eliminats. Aquests vèrtexs poden tenir difícil la seva incorporació al conjunt de terminals si aquesta funció s'executa cada cop que es genera un nou arbre. D'altra banda, la incidència global que aquest pas té en el temps d'execució és generalment molt inferior a la del *Pas 1*, sobretot en el cas de grafs poc densos[5].

Finalment, limitant el nombre d'iteracions a $O(|V|)$, la complexitat en el pitjor cas del GBA és $O(|V|(|A| + |V|\log|V| + |S||V|))$ i inclou un únic càlcul dels camins entre cada parella de vèrtexs. Val a dir que $|V|$ iteracions s'han mostrat suficients per a produir els resultats que es presenten a continuació.

3.4 Resultats experimentals

El GBA ha estat provat utilitzant les 60 instàncies més grans dels grafs proposats per Beasley [10] en la OR-library [58], que denotem per c1,...,c20, d1,...,d20, e1,...,e20. Cadascun dels graf de tipus *c*, *d* i *e* té, respectivament, 500, 1000 i 2500 vèrtexs. Tots els grafs han estat generats de forma aleatòria i són poc densos. Els costos de les arestes són enters de l'1 al 10. Amb l'objectiu de comparar el rendiment del GBA amb el de l'algorisme genètic EGA proposat per Esbensen [30, 31], les proves realitzades s'han adaptat a les que es van portar a terme per a aquest algorisme.

Les taules 2, 3 i 4 mostren els resultats aconseguits pel GBA amb els grafs *c*, *d* i *e*, respectivament. La taula 5 resumeix aquests resultats i els compara amb els obtinguts per l'EGA en les dues versions que han estat presentades de l'algorisme d'Esbensen, EGA1 [30] i EGA2 [31]. A grans trets, els canvis introduïts en l'EGA2 en relació a l'EGA1 consisteixen en una selecció més acurada d'algunes funcions estàndard dels algorismes genètics i, també, en un augment significatiu del nombre d'iteracions que, si

[5]Un cop calculats els camins, la majoria de comparacions en el *Pas 1* es realitzen en el subgraf distància de G, que és complet. En canvi, les del *Pas 2* es realitzen en G.

bé permet millorar els resultats, té una repercussió important en els temps d'execució.

Els resultats de l'EGA1 van ser comparats en condicions similars amb els de cinc algorismes diferents, dos d'ells mètodes exactes, que es mostraren incapaços, per limitacions de temps, de trobar la solució per a algunes de les instàncies. Entre els mètodes heurístics cal esmentar un dels mètodes repetitius proposats per Winter i Smith [93] (SPH-SS) (veure l'apartat 2.5.1) i, també, el mètode heurístic basat en una estratègia iterativa *greedy* proposat per Alexander i Robins (IDNH) [2] (veure l'apartat 2.5.2). En la taula 5 es mostren els resultats de l'IDNH, els quals són superiors als que obté l'SPH-SS, i que serveixen de referència per a comparar el rendiment dels diferents mètodes. Aquests resultats han estat extrets de [30] i foren obtinguts fent servir, per a la majoria d'instàncies, un temps superior al que va utilitzar l'EGA1.

Abans que el GBA sigui executat, es fan servir diverses rutines per tal de reduir la mida dels grafs de prova. Les rutines que hem utilitzat van ser programades pel propi Esbensen [30], que molt amablement ens les va cedir. Per tant, les proves s'han portat a terme fent servir exactament les mateixes instàncies reduïdes. En canvi, a diferència de l'EGA, el GBA no utilitza cap procés de *hill-climbing* per tal de millorar la solució que retorna l'algorisme.

El GBA ha estat implementat en llenguatge C i totes les proves s'han realitzat en una *SUN Ultra 30*, 250 MHz, 256Mb RAM. En cap moment no s'ha utilitzat l'estructura de dades proposada per Fredman i Tarjan [36] i, per tant, els temps d'execució que es donen podrien millorar sensiblement. El càlcul dels camins s'ha portat a terme en $O(|V|^3)$ passos mitjançant l'algorisme de Floyd, mentre que cada execució de la SPH s'ha realitzat utilitzant una implementació de l'algorisme de Prim amb complexitat $O((|S| + |S_i'|)^2)$, on $|S_i'|$ és el nombre de vèrtexs que són incorporats en la iteració i-èssima, el qual acostuma a ser de l'ordre de $|S|$. La funció per a generar els números aleatoris s'ha extret de [80] on és anomenada *ran1*.

El paràmetre relatiu al nombre esperat de moviments deguts a la funció *sacsejar* s'ha fixat en 3 ($nEspMov = 3$), mentre que el nombre màxim d'iteracions sense millora $nMaxIter$ ha estat inhibit[6], i substituït per un nombre fix d'iteracions (nIt),

[6] A [45] es poden veure els resultat de GBA quan $nMaxIter = 1000$.

que depèn del nombre de vèrtexs del graf. D'aquesta manera, es facilita la comparació entre els diferents algorismes.

L'EGA està basat en la DNH (veure l'apartat 2.4.2), algorisme que té complexitat molt semblant a la de la SPH quan es disposa dels camins. A més, l'EGA també utilitza una funció que limita el nombre màxim de falsos terminals a $|S| - 2$, la qual cosa permet comparar els temps dels dos mètodes en funció del nombre d'execucions de l'algorisme de base, o *nombre d'arbres avaluats.* D'aquesta manera s'evita la influència en els temps de les característiques específiques del programari i del maquinari que ha estat utilitzat per a provar els algorismes. Cal remarcar, però, que aquest tipus de comparació beneficia l'EGA, per al qual només coneixem una fita inferior del nombre d'arbres avaluats, calculada en funció de les condicions de parada.

Per tal d'evitar duplicitats, a més de la informació que ofereixen les taules 2, 3, 4 i 5, en les taules 9, 10, 11, 12, 13 i 14 del capítol següent també s'inclouen dades referents al GBA. En concret, els resultats del GBA són comparats amb els del l'algorisme que proposem en el proper capítol (GBGA) en la pàgina 98. A més, a partir de les consideracions que s'exposen en l'apartat 4.8.1 i de les taules esmentades, no resulta difícil treure algunes conclusions relatives al rendiment del GBA en comparació al d'un dels mètodes proposats per Duin i Voβ [28] (DVA) i, també, amb el del *branch and cut* proposat per Koch i Martin [67] (KMBC).

3.4.1 Anàlisi del rendiment

A partir les taules 2, 3, 4 i 5 podem concloure que el GBA aconsegueix trobar una solució òptima en el 78,6% del total d'execucions quan el nombre d'iteracions és igual $|V|$, i arriba fins al 89,2% quan aquest nombre s'incrementa fins a $10|V|$. Tenint en compte que l'objectiu principal de l'algorisme és aconseguir bones aproximacions, cal remarcar el fet que l'error comès es troba en totes les execucions per sota de l'1%. Destaca també el poc temps que utilitza l'algorisme en el procés iteratiu. Per exemple, quan $nIt = |V|$, aquest temps és generalment inferior al necessari per a reduir la mida dels grafs, el qual ja hem dit que era similar al del càlcul dels camins de cost mínim mitjançant l'algorisme de Floyd [33].

A l'hora de comparar els resultats del GBA amb els de l'EGA, cal tenir molt present la incidència que tenen les condicions de parada que han estat utilitzades en cadascun dels algorismes. D'una banda, un increment molt alt en el nombre d'iteracions pot repercutir d'una manera molt negativa en el temps, sense que es produeixi una millora en la qualitat dels resultats. De l'altra, el nombre d'execucions de la DNH que es consideren per a l'EGA és una fita inferior que, en el cas de l'EGA1, està molt per sota del nombre real d'execucions.

Observant la taula 5, els resultats que obté el GBA quan $nIt = |V|$ són superiors en tots els casos als que obté l'EGA1. Tot i això, el nombre d'arbres avaluats de l'EGA1 és com a mínim el quàdruple per als graf c, el doble per als grafs d i similar per als grafs e, en relació al nombre avaluat pel GBA. En realitat, els resultats del GBA serien comparables als de l'EGA2. En aquest cas, però, la relació entre el nombre d'arbres avaluats per l'EGA2 no deixa cap marge de dubte. L'EGA2 avalua, pel cap baix, 20 vegades més arbres que el GBA per als grafs e, 50 per als grafs d i 100 per als grafs c. Tot indica, però, que les condicions de parada de l'EGA2 per als grafs tipus c i d són poc apropiades. Finalment, el GBA millora clarament els resultats de l'EGA2 quan $nIt = 10|V|$ tot avaluar, com a màxim, la meitat d'individus.

Per acabar, creiem important remarcar que els resultats obtinguts per tots dos algorismes poder ser considerats sorprenentment bons, si tenim en compte que el problema és intractable i que alguns d'aquests grafs tenen una mida relativament gran. En bona part, aquests resultats cal atribuir-los a la feblesa d'algunes de les instàncies, especialment de totes aquelles que tenen un nombre de terminals no superior a 10, i que representen el 40% del total. Descomptant aquests grafs, el percentatge de solucions òptimes passa a ser del 64%, quan $nIt = |V|$, i del 82%, quan $nIt = 10|V|$. De cara al futur, seria interessant veure el comportament de l'algorisme quan s'utilitzen altres tipus d'instàncies.

Taula 2: Resultats experimentals del GBA amb els grafs c

	Mida Reduïda			Cost				Temps de CPU	
G	$\|V\|$	$\|S\|$	$\|A\|$	opt.	$nIt = 500$		$nIt = 10\|V\|$	red	$nIt = 500$
c1	145	5	263	85	—		—	5	0
c2	130	8	239	144	—		—	5	0
c3	120	35	232	754	0,5	(0-2)	—	5	1
c4	109	38	221	1079	—		—	5	1
c5	37	17	91	1579	—		—	5	0
c6	369	5	847	55	—		—	5	0
c7	382	9	869	102	—		—	5	0
c8	336	54	818	509	—		—	5	2
c9	349	78	832	707	0,7	(0-1)	—	6	3
c10	213	76	624	1093	—		—	6	2
c11	499	5	2184	32	—		—	5	0
c12	498	9	2236	46	—		—	5	1
c13	463	65	2108	258	—		—	5	2
c14	427	81	1961	323	—		—	6	2
c15	299	92	1471	556	—		—	6	2
c16	500	5	4740	11	—		—	5	0
c17	499	9	4698	18	—		—	5	0
c18	486	70	4668	113	—		—	6	2
c19	473	98	4490	146	—		—	6	3
c20	386	143	3850	267	—		—	6	4

- Mida Reduïda: característiques dels grafs un cop s'han executat les rutines de reducció.
- Cost: cost d'una solució òptima (opt.) i la diferència mitjana respecte aquest cost després de 10 execucions fent servir nIt iteracions per a cada graf. Un guió indica que una solució òptima s'ha obtingut en totes 10 execucions. En cas contrari, entre parèntesis i separats per un guió, s'indica la diferència entre els costos de la millor i la pitjor aproximació, respectivament, i el cost de l'òptim.
- Temps de CPU: indica els segons de CPU utilitzats per les rutines de reducció (red), i pel procés iteratiu quan $nIt = |V|$. El temps de reducció és semblant al d'una execució de l'algorisme de Floyd i, per tant, es pot fer servir de referència per a futures comparacions.
- Per exemple, per al graf $c3$, l'error absolut mitjà de 10 execucions de GBA amb exactament 500 ($= |V|$) iteracions cadascuna val 0,5. El (0-2) a la dreta d'aquest valor indica que la millor aproximació obtinguda en aquestes 10 execucions té cost 754+0, mentre que la pitjor 754+2, essent 754 el valor de l'òptim per al graf.

Taula 3: Resultats experimentals del GBA amb els grafs d (indicacions a la taula 2)

	Mida Reduïda			Cost					Temps de CPU	
G	$\|V\|$	$\|S\|$	$\|A\|$	opt.	$nIt = 1000$		$nIt = 10\|V\|$		red	$nIt = \|V\|$
d1	274	5	510	106	—		—		50	1
d2	285	10	523	220	—		—		50	1
d3	224	67	441	1565	0,2	(0-2)	—		51	4
d4	159	66	339	1935	—		—		51	3
d5	97	48	246	3250	—		—		52	2
d6	761	5	1741	67	—		—		50	1
d7	754	10	1735	103	—		—		50	1
d8	731	124	1708	1072	1,3	(0-3)	—		51	21
d9	654	155	1613	1448	0,3	(0-2)	—		52	25
d10	418	146	1317	2110	0,8	(0,2)	—		53	13
d11	993	5	4674	29	—		—		50	1
d12	1000	10	4671	42	—		—		50	2
d13	922	122	4433	500	—		—		51	18
d14	853	160	4173	667	—		—		52	23
d15	550	157	2925	1116	—		—		54	14
d16	1000	5	10595	13	—		—		50	1
d17	999	9	10531	23	—		—		51	1
d18	978	145	10140	223	1,4	(1-2)	1,1	(1-2)	50	18
d19	938	193	9676	310	1,0	(1-1)	1,0	(1-1)	51	26
d20	814	324	8907	537	—		—		52	53

Per exemple, per al graf $d19$, l'error absolut mitjà de 10 execucions de GBA amb exactament 1000 (= $|V|$) iteracions cadascuna val 1,0. El (1-1) a la dreta d'aquest valor indica que en totes les execucions s'ha obtingut una aproximació amb cost 310+1, on 310 és el valor de l'òptim. Tot i incrementar el nombre d'iteracions fins a 10000 (= $10|V|$), els resultats per a aquest graf no milloren.

Taula 4: Resultats experimentals del GBA amb els grafs e (indicacions a la taula 2)

	Mida Reduïda			Cost					Temps de CPU	
G	$\|V\|$	$\|S\|$	$\|A\|$	opt.	$nIt = 2500$		$nIt = 10\|V\|$		red	$nIt = \|V\|$
e1	680	5	1286	111	—		—		773	3
e2	710	9	1328	214	—		—		773	3
e3	637	199	1233	4013	5,6	(3-8)	2,5	(2-3)	779	97
e4	435	164	964	5101	1,9	(1-3)	0,3	(0-1)	783	40
e5	222	108	649	8128	—		—		800	14
e6	1845	5	4318	73	—		—		756	6
e7	1891	10	3388	145	—		—		775	8
e8	1723	286	4193	2640	1,7	(0-4)	0,3	(0-1)	791	330
e9	1608	358	4069	3604	3,7	(2-6)	1,8	(0-3)	799	406
e10	1046	366	3388	5600	4,1	(1-5)	2,0	(1-3)	827	264
e11	2498	5	12093	34	—		—		780	8
e12	2500	10	12123	67	—		—		777	9
e13	2341	321	11760	1280	1,5	(1-2)	—		788	415
e14	2139	388	11325	1732	1,2	(0-2)	—		802	442
e15	1461	443	8514	2784	—		—		837	390
e16	2500	5	29332	15	—		—		781	7
e17	2500	10	29090	25	—		—		779	8
e18	2429	355	28437	564	2,4	(2-3)	1,9	(1-2)	781	325
e19	2351	485	27779	758	0,1	(0-1)	—		787	384
e20	1988	758	24423	1342	—		—		801	918

Per exemple, per al graf $e4$, l'error absolut mitjà de 10 execucions de GBA amb exactament 2.500 ($= |V|$) iteracions cadascuna val 1,9. El (1-3) a la dreta d'aquest valor indica que la millor aproximació obtinguda en aquestes 10 execucions té cost 5.101+1, mentre que la pitjor 5.101+3, essent 5.101 el valor de l'òptim per al graf. Es pot veure que, quan s'incrementa en nombre d'iteracions fins a 25.000 ($= 10|V|$) s'obtenen diverses solucions òptimes.

3.5 Conclusions

En aquest capítol hem presentat el GBA, un algorisme basat en una estratègia iterativa recurrent que utilitza la SPH com a algorisme de base. El GBA millora de manera significativa, en temps i qualitat, els resultats aconseguits per l'EGA, l'algorisme genètic d'Esbensen [30], quan s'utilitzen els grafs de Beasley [10] com a instàncies de prova. Restringint el nombre d'iteracions a $O(|V|)$, la complexitat en el pitjor cas del GBA és $O(|V|(|A|+|V|\log|V|+|S||V|))$, terme que inclou un càlcul dels camins entre cada parella de vèrtexs. Fent servir exactament $|V|$ iteracions, el GBA aconsegueix trobar la solució òptima en el 78,6% de totes les execucions, arribant fins al 89,2% quan el nombre d'iteracions s'incrementa fins a $10|V|$. Tenint en compte, però, que el veritable objectiu del mètode és aconseguir bones aproximacions, el fet més rellevant dels resultats és que en totes les execucions el percentatge d'error està per sota de l'1%.

El rendiment de l'algorisme es veu millorat gràcies a tres funcions que poden ser utilitzades per altres mètodes. La més efectiva de totes és la funció *reordenar* ja que, amb una simple reordenació dels vèrtexs millora els resultats de l'algorisme i, a més, fa que aquests resultats puguin ser considerats vàlids per a instàncies de prova isomorfes. La funció *eGrau2* elimina en cada iteració els vèrtexs d'Steiner de grau dos i, d'aquesta manera, redueix l'espai de solucions i incrementa la velocitat de l'algorisme. La funció *sacsejar* produeix petites modificacions en el conjunt de terminals, donant opcions a aquells vèrtexs que són deixats de banda per l'algorisme, moltes vegades d'una forma arbitrària.

Finalment, després de remarcar la importància que té el càlcul dels camins en el rendiment global de tots aquells mètodes per al PSG que els utilitzen, proposem una estratègia molt simple que, sense incidir en la complexitat dels algorismes que calculen els camins, acostuma a produir millores en la qualitat dels resultats del GBA.

Taula 5: Comparació dels resultats amb els d'altres mètodes

Grafs	IDNH		EGA1 $nIt > 2000$		EGA2 $nIt > 50000$		GBA $nIt = \|V\|$		GBA $nIt = 10\|V\|$	
tipus	0%	$< 1\%$	0%	$< 1\%$	0%	$< 1\%$	0%	$< 1\%$	0%	$< 1\%$
c	55,0	80,0	78,0	93,5	90,5	94,0	94,5	100	100	100
d	35,0	80,0	77,5	92,5	83,5	95,5	80,5	100	90,0	100
e	36,8	68,4	55,0	85,0	70,0	90,0	61,0	100	77,5	100
Total	42,3	76,1	70,2	90,3	81,3	93,2	78,6	100	89,2	100

Percentatge de solucions òptimes (0%) i d'aproximacions amb error inferior a l'1% ($< 1\%$) per a cadascun dels algorismes. Els resultats de l'IDNH [2] i de l'EGA1 han estat extrets de [30] i els de l'EGA2 de [31]. Les altres dues columnes resumeixen els resultats del GBA que es mostren en les taules 2, 3 i 4, quan el nombre d'iteracions es correspon a $|V|$ i a $10|V|$, respectivament.

Capítol 4

Un algorisme genètic per al PSG

L'algorisme genètic (AG) per al PSG que proposem en aquest capítol (GBGA) [43, 44, 46] és una versió millorada de l'AG presentat l'any 1995 per Esbensen (EGA) [30, 31] el qual, per la seva part, millorava l'AG introduït l'any 1993 per Kapsalis i Rayward-Smith (KRGA)[1] [63]. En la nostra opinió, les diferències entre els tres algorismes, les quals seran descrites en detall al llarg del capítol, tenen el seu origen en l'objectiu principal que persegueixen els seus respectius autors.

El principal objectiu de Kapsalis i Rayward-Smith és provar que els AG són també d'utilitat per al PSG. El seu article [63], per tant, centra més l'atenció en aspectes genèrics dels AG —quin mètode de selecció és millor, quin efecte produeix la inclusió d'un individu privilegiat en la població inicial, etc.—, que no pas en les propietats del problema que li permetrien millorar-ne el rendiment. En la nostra opinió, el KRGA forma part d'una concepció, ja superada, dels AG com a mètode universal capaç de resoldre, amb petites modificacions, una gran varietat de problemes. Potser per això, els resultats aparentment prometedors del KRGA resulten ser, tal i com veurem, molt limitats.

El principal objectiu d'Esbensen és aconseguir un AG per al PSG que pugui competir experimentalment amb els millors mètodes heurístics per al problema. Des del seu punt de vista [32], només tenen interès aquells mètodes que milloren el rendiment

[1] La primera referència que tenim de l'EGA és de l'any 1994 [29] i, tot i que ja fa esment del KRGA, és molt probable que es desenvolupés de manera independent.

dels ja existents, independentment de si les estratègies en què es basen són modernes o clàssiques. I és que la proliferació de treballs presentant un AG que millora, parcialment o global, els també pobres resultats que obtenen altres AG, pot haver incidit negativament en els mètodes evolutius. Quan els resultats parlen per si sols, les veus discordants, que tenen el seu origen en la dificultat de justificar el funcionament d'aquests mètodes, són fàcils de rebatre.

Esbensen vol competir amb els millor mètodes i, per tant, explota tots els recursos que té a l'abast: rutines per a reduir la mida dels grafs de prova, un *hill-climbing* per tal d'optimitzar el cost de l'aproximació que retorna l'AG, etc. Llegint el seu article, la desproporció entre els resultats de l'EGA i els del KRGA és tant impressionant, que resulta difícil d'assumir. D'una banda, els autors del KRGA no treuen prou profit, tal i com es pot veure en l'apartat 4.2.1, del potencial real del mètode que proposen. De l'altra, Esbensen [30] no compara els dos mètodes en igualtat de condicions, tenint en compte que el KRGA no redueix la mida dels grafs ni utilitza cap *hill-climbing*.

El nostre principal objectiu és interpretar el comportament dels AG per al PSG, prescindint de les suposades virtuts inherents als mètodes evolutius [22, 94]. Ens fa la impressió que els resultats de l'EGA haurien estat més àmpliament valorats si haguessin estat obtinguts a partir d'un mètode heurístic tradicional. A més, les estratègies evolutives dels AG es mostren clarament insuficients per a justificar el pobre rendiment del KRGA en comparació al de l'EGA, i obliguen a realitzar un nombre considerable d'experiments per tal de decidir entre una o altra funció. El nostre objectiu, per tant, és justificar el funcionament d'aquests algorismes per al PSG de manera similar a com es fa en la resta de mètodes heurístics: a partir d'unes línies estratègiques versemblants i d'una sèrie de funcions que afavoreixen l'explotació d'aquestes línies. No estem interessats en la utilitat dels AG com a mètodes generals de resolució de problemes.

Un cop definides unes línies estratègiques per al GBGA, es mostra que un AG és una eina vàlida per a desenvolupar-les. A partir d'aquestes línies es poden justificar les importants diferències de rendiment entre el KRGA i l'EGA i es pot interpretar l'efecte que produeix cadascuna de les funcions que integren aquests algorismes per, a continuació, adaptar-les i incorporar-ne de noves per tal de millorar el rendiment.

L'objectiu marcat és ambiciós, tenint en compte el que és habitual en l'àrea, i cal situar-lo dins del context de les estratègies heurístiques.

Voß and Gutenschwager [90] (VGGA) també han presentat un AG per al PSG basat en idees similars als anteriors. Concretament, el seu algorisme intenta disminuir el temps de convergència de l'EGA a base de fixar de manera heurística alguns dels vèrtexs de l'arbre i, d'aquesta manera, reduir l'àmbit de recerca. En la nostra opinió, la pèrdua de qualitat dels resultats del VGGA, inferiors als de la primera versió de l'EGA (EGA1 en la taula 10), no es veu compensada per la possible millora en temps. A més, sense cap mena de justificació visible, el VGGA no utilitza l'operador *filtre* (veure la pàgina 71), l'efecte del qual és fonamental per a incrementar la velocitat de l'EGA. Atribuïm a aquest fet algunes de les conclusions confuses dels autors [90], després d'intentar reproduir els experiments de l'EGA.

En la primera secció del capítol es descriu el funcionament bàsic dels AG. La secció 4.2 presenta el funcionament dels AG que han estat proposats per al PSG, i mostra les principals diferències entre el KRGA i l'EGA. En la secció 4.3, es defineixen les línies estratègiques que haurà de seguir el GBGA, i es justifica la utilització d'un AG per a implementar-les en la pràctica. Després de descriure en detall el GBGA en la secció 4.4, el comportament i la utilitat de les diferents funcions que formen part de l'algorisme és analitzat, en funció de les línies estratègies, en les tres seccions següents. La secció 4.8 mostra els resultats experimentals i, finalment, en la darrera secció s'obtenen algunes conclusions.

4.1 Algorismes genètics (AG)

Fent servir la definició de Goldberg [42], els AG són algorismes de recerca basats en els mecanismes de la selecció natural i la genètica. Està fora del nostre àmbit analitzar fins a quin punt aquest model computacional proposat per Holland [52] emula correctament la manera en què les espècies evolucionen. En qualsevol cas, considerem molt interessant el paral·lelisme amb el model biològic, des d'un punt de vista pedagògic, per tal d'explicar el seu funcionament. D'altra banda, tampoc estem interessats en els AG com a mètode general de resolució de problemes [22], més enllà

de la seva utilitat per al PSG.

En un problema d'optimització combinatòria l'objectiu consisteix en trobar una solució òptima dins d'un espai generalment finit de solucions candidates. Per a alguns d'aquests problemes hi ha algorismes eficients capaços d'anar directe cap a una solució òptima —o quasi-òptima— en un temps raonable. Cas contrari, quan una recerca exhaustiva no és possible, es poden generar un nombre relativament gran de solucions candidates per tal de trobar una aproximació. Algunes vegades, aquestes solucions candidates són generades de forma totalment aleatòria en el que, en anglès, és anomenat *random walk*. Com a alternativa, en els mètodes anomenats *moderns* [85], com ara la *simulació termodinàmica*, la *recerca tabú* i els AG, les solucions candidates són generades d'una forma "esbiaixada", depenent de la *història* recent. Els resultats d'aquests mètodes dependran en gran mesura de l'efectivitat d'aquest biaix a l'hora de produir solucions de qualitat.

Des del punt de vista de l'optimització combinatòria, un AG és un mètode iteratiu de recerca d'un espai de solucions que genera les solucions candidates d'una manera esbiaixada en funció d'un subconjunt de solucions ja calculades —*població*— que s'actualitza en cada iteració. D'aquesta manera, s'assumeix que els AG són més estables que no pas els mètodes que generen les noves solucions a partir d'una única solució candidata.

4.1.1 Funcionament dels AG

Els AG van ser concebuts com a un model per a simular el principi darwinià d'evolució de les espècies, d'aquí que el seu nom i la seva terminologia reflecteixin aquest origen en l'àrea de la biologia. En aquest apartat fem una breu introducció al funcionament bàsic dels AG. Per a més informació, recomanem el llibre de Goldberg [42].

En un AG, una *població* està formada per diversos *individus* de la mateixa espècie, és a dir, per solucions candidates del problema a resoldre. L'aspecte físic que presenta un individu depèn del seu *codi genètic*, el qual es representa mitjançant una *cadena* binària o *string*. En concret, cada *gen* està representat per un o més *bits* i és el responsable d'alguna característica física específica de la solució. El *descodificador*

Generar la població inicial;
Avaluar els individus; [*descodificador, funció de salut*]
repetir /* procés evolutiu */
    repetir /* generar la nova població */
        Seleccionar dos individus; [*mecanisme de selecció*]
        Crear dos descendents; [*funció d'encreuament*]
    fins que(*la nova població estigui completa*)
    Aplicar operadors genètics; [*mutació, ...*]
    Substituir la població anterior; [*estratègia de substitució*]
    Avaluar els individus;
fins que(Condicions de parada)
retornar el millor individu trobat

Algorisme 7: Esquema bàsic d'un AG

associa a cada cadena una solució que, després, és avaluada per tal de determinar-ne el cost. La *funció de salut*, (en anglès, *fitness function*), avalua la salut d'un individu en funció del seu cost, i li associa un valor que anomenarem *nivell de salut.*

En el *procés de reproducció*, parelles d'individus són seleccionades per a produir descendència a base de combinar les seves cadenes. Aquesta selecció es porta a terme de manera que els individus que tenen un nivell de salut més alt i, per tant, sembla que estan millor adaptats a l'entorn, tenen més possibilitats de ser seleccionats. Amb una probabilitat molt baixa un *gen* pot mutar, és a dir, pot prendre una configuració que no formava part del codi genètic dels seus pares. El paper de la mutació segueix sent una de les incògnites que es plantegen en biologia, tot i que sol considerar-se com un mecanisme aleatori que permet que les espècies siguin adaptades a l'entorn.

Tot el procés està basat en l'anomenat *principi de supervivència dels més sans*, considerat com la base de l'evolució de les espècies: els individus més sans tenen més opcions de sobreviure i de reproduir-se i, per això, les noves poblacions acostumen a tenir més informació genètica d'aquests individus que no pas de la resta. Per tant, els nous individus probablement heretaran aquells gens que fan que els seus antecessors estiguin millor adaptats a l'entorn. L'algorisme 7 mostra l'esquema bàsic d'un algorisme genètic.

- **Generar:** l'algorisme comença generant una població inicial. Tant el nombre d'individus com la seva longitud —*midaPob*, *cadLong*— es decideixen a priori i, generalment, es mantenen constants durant tot el procés. Cada bit de la cadena es genera aleatòriament a "cara o creu" tot i que, a vegades, s'inclouen en la població inicial algunes cadenes de qualitat contrastada.

- **Avaluar:** s'estableix una correspondència que associa a cada cadena una solució. A partir del cost d'aquesta solució s'assigna un nivell de salut a l'individu. El descodificador i la funció de salut depenen del problema a resoldre, i són el mecanisme principal a partir del qual la informació específica del problema és incorporada a l'AG.

- **Procés evolutiu:** en cada iteració una nova població és generada mitjançant la selecció i l'encreuament de parelles de cadenes de la població actual. Selecció i encreuament conformen el que anomenem *procés reproductiu.*

  - **Selecció:** en la majoria de mecanismes de selecció proposats, la probabilitat de seleccionar un individu és proporcional al seu nivell de salut, el qual es calcula en relació a la salut de la resta d'individus de la població. S'utilitzen diverses estratègies per tal de limitar el nombre de vegades que un individu amb salut privilegiada pot ser seleccionat.

  - **Encreuament:** la funció d'*encreuament* genera dues noves cadenes a base de combinar subcadenes extretes dels seus *pares.* Tal i com es pot veure en la figura 4a, si el primer fill hereta el k-èsim bit del pare[2] (i), el segon fill hereta aquest bit del pare (ii). En l'*encreuament uniforme* aquesta decisió es pren a cara o creu per a cada posició de la cadena.

- **Operadors genètics:** aquests operadors produeixen petites modificacions en les cadenes. El més àmpliament utilitzat és la mutació on, amb una probabilitat $pMuta$ molt petita i actuant de manera independent per a cada bit, canvia el valor d'alguns bits d'una cadena. (Veure la figura 5a).

[2]Tot i que utilitzarem el terme masculí "pare", tots els individus de la població pertanyen a un únic gènere i, per tant, tots poden fer de pares i de mares.

- **Estratègia de substitució**: depenent d'aquesta estratègia, la població actual és substituïda parcialment o total per la nova població.
- **Condicions de parada**: resulta molt difícil decidir quan el nombre d'iteracions és prou gran per a aconseguir una bona aproximació. Normalment, es solen utilitzar diversos paràmetres: el nombre màxim d'iteracions des que s'ha obtingut la millor aproximació *stopMaxInd*, el nombre màxim d'iteracions des de l'última millora de la qualitat mitjana de la població *stopMaxPop*, un nombre màxim constant d'iteracions *stopMaxIter*, o d'altres.

	pares	fills
(i)	$\overline{110110}$	$\underline{0}\overline{101}\underline{10}$
(ii)	$\underline{011010}$	$\overline{1}\underline{110}\overline{10}$

(a) Encreuament en un AG

pares	fills
$\{s_1, s_2, s_4, s_5\}$	$\{s_2, s_4, s_5\}$
$\{s_2, s_3, s_5\}$	$\{s_1, s_2, s_3, s_5\}$

(b) Encreuament en els AGPSG

Figura 4: Encreuament

$110\underline{1}10 \rightarrow 110\underline{0}10$

(a) Mutació en un AG

$\{s_1, s_2, s_4, s_5\} \rightarrow \{s_1, s_2, s_5\}$

(b) Mutació en els AGPSG

Figura 5: Mutació

En la propera secció es descriuen el funcionament dels AG que han estat proposats per al PSG.

4.2 Els AG per al PSG (AGPSG)

Fem servir AGPSG per a referir-nos a la família d'AG per al PSG que utilitzen la codificació per als individus proposada per Kapsalis i Rayward-Smith [63]: el KRGA,

l'EGA [30, 31], el VGGA [90], el GBGA [46] i les seves versions prèvies [44, 43]. Aquests algorismes comparteixen una filosofia similar, però es diferencien principalment per l'elecció d'alguns dels elements dependents del problema (el descodificador, alguns operadors genètics, ...) com també, per algunes de les eleccions estàndards dels AG (el mètode de selecció, la funció d'encreuament, ...) i pels valors inicials dels paràmetres (*midaPob*, *pMuta*, *stopMaxInd*, etc.). L'esquema bàsic de tots aquests algorismes és molt semblant a l'algorisme 7. Representem per I_C i I_T, respectivament, la cadena i l'arbre candidat a solució associat a un individu I.

L'objectiu dels AGPSG és trobar un subconjunt I_S de vèrtexs d'Steiner tal que $MSpT(G, S \cup I_S)$ sigui $MStT(G, S)$. En total hi ha $2^{|V|-|S|}$ subconjunts de vèrtexs candidats que, en els AGPSG, es codifiquen mitjançant cadenes binàries de longitud $|V| - |S|$. El bit i-èsim d'una cadena I_C està associat a la presència o a l'absència de l'i-èsim no-terminal en el subconjunt I_S associat, el qual serà referit com a *subconjunt de cadena.* Fent servir un mètode heurístic per al PSG com a descodificador, cada I_S es fa correspondre a un arbre de Steiner I_T amb conjunt de terminals $S \cup I_S$:

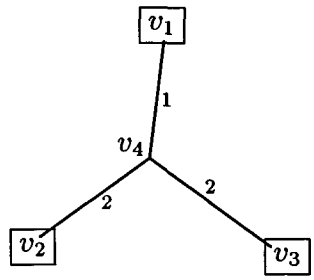
$$I_C \equiv I_S \longrightarrow I_T = StT(G, S \cup I_S).$$

El cost d'un individu $c(I)$ és el cost del seu arbre associat, després d'eliminar-ne els vèrtexs no-terminals fulla (*corbates*). Tal com en el capítol anterior (veure 3.1), la funció *trim* és l'encarregada d'eliminar aquests vèrtexs i, per tant, $c(I) = |trim(I_T)|$. Finalment, els individus amb cost menor reben associat un nivell de salut més gran.

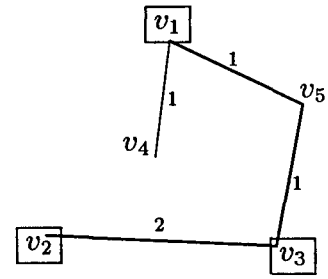
La figura 6 recopila aquestes definicions per al graf de la figura 1 fent servir l'arbre generador de cost mínim ($MSpT$) com a descodificador. La figura 4b mostra com en els AGPSG l'encreuament és una forma de redistribuir els vèrtexs de dos subconjunts de cadena per tal de construir-ne dos de nous. La figura 5 mostra com la mutació inclou o exclou vèrtexs en un subconjunt de cadena. Les principals diferències entre el KRGA i l'EGA són presentades en l'apartat que ve a continuació.

4.2.1 Els AG de Kapsalis *et al.* (KRGA) i d'Esbensen (EGA)

El descodificador del KRGA està basat en l'$MSpT$ (figura 6). Quan el subgraf induït per $S \cup I_S$ no és connex, $MSpT(G, S \cup I_S)$ no està definit. En aquest cas, s'associa



$I_C = 10 \Leftrightarrow I_S = \{v_4\}$
$I_T = MSpT(G, S \cup \{v_4\})$
$c(I) = |trim(I_T)| = |I_T| = 5$



$I_C = 11 \Leftrightarrow I_S = \{v_4, v_5\}$
$I_T = MSpT(G, S \cup \{v_4, v_5\})$
$c(I) = |trim(I_T)| = |I_T - \{v_4\}| = 4$

Figura 6: AGPSG

a cada cadena un graf *bosc*, consistent en un arbre generador de cost mínim per a cada component connex. El cost d'un individu és el cost d'aquest bosc, més una penalització constant molt gran per component addicional.

El descodificador de l'EGA, i també el que va utilitzar més endavant el VGGA, està basat en la DNH (veure 2.4.2). És conegut que la DNH retorna sempre una solució com a mínim tant bona com l'MSpT i que, a més, compleix les propietats següents:

- Com que $I_T = DNH(G, S \cup I_S)$ és un arbre d'Steiner vàlid per a tot subconjunt de cadena I_S, les penalitzacions deixen de ser necessàries.
- La DNH és un algorisme d'aproximació que garanteix que $|I_T| \leq 2|MStT(G, S \cup I_S)| \quad \forall I_S$. En la segona versió de l'EGA, Esbensen [31] treu partit a aquesta propietat i obté una FRA inferior a 2, a base d'introduir una cadena amb totes les posicions a zero en la població inicial ($I_S = \emptyset$).
- Fent servir la DNH, es pot assegurar que hi ha un I_S amb $|I_S| \leq |S| - 2$ tal que $I_T = DNH(G, S \cup I_S)$ és $MStT(G, S)$.

Aquesta darrera propietat és utilitzada a l'EGA per a introduir un nou operador genètic (*filtre*) que redueix l'espai de solucions a subconjunts de cadenes amb, com a

molt, $|S| - 2$ vèrtexs i que, a més, incrementa la velocitat del descodificador:

- *filtre*(I_C): posa a zero bits de I_C, seleccionats de manera aleatòria, fins que $|I_S| \leq |S| - 2$.

El rendiment del KRGA i de l'EGA són comparats a [30] fent servir els 18 grafs més petits, o tipus *b*, de Beasley [10], tots ells de menys de 100 vèrtexs. Mentre que el KRGA aconsegueix l'òptim absolut en el 77% dels casos, l'EGA retorna l'òptim en totes les execucions fent servir un temps espectacularment inferior.

A partir dels nostres propis experiments hem pogut constatar que, introduint petits canvis, la qualitat dels resultats del KRGA s'acosta a la dels de l'EGA per a aquests grafs. Per exemple, el percentatge de solucions òptimes del KRGA creix fins al 91% quan s'incrementa el temps d'execució[3] i arriba al 95% canviant el mètode de selecció —fent servir l'*Stochastic Remainder Selection* [42] en comptes del *Roulette Selection*. A més, la diferència entre els temps d'execució també es veu reduïda quan, com en l'EGA, s'utilitzen rutines per a reduir la mida dels grafs abans d'executar l'AG. L'efecte d'aquestes rutines en els grafs *b* és molt important, solucionant per si mateixes, és a dir, sense la intervenció de l'AG, tres de les instàncies. De tota manera, fins i tot sense les reduccions, els grafs *b* són tan febles que generant un nombre no excessiu de cadenes de manera aleatòria —*random walk*— s'obté una solució òptima en 12 dels 18 grafs, és a dir, en el 66,7% d'execucions. És en els grafs més grans de la OR-library amb $|V| \geq 500$ on les proves de veritat comencen. I les diferències tan marcades que es produeixen entre el KRGA i el EGA per a aquests grafs resulten en principi sorprenents, si es té en compte que els dos mètodes és diferencien bàsicament pel descodificador i l'operador filtre:

- La reducció de l'espai de solucions que s'aconsegueix mitjançant l'operador filtre només resulta interessant quan $|S| << |V|/2$ i, fins i tot en aquests casos, l'efecte d'aquest operador en el procés evolutiu és poc clar.

- La utilització de *penalitzacions* s'ha mostrat d'utilitat per a d'altres AG i, per tant, no té per què ser necessàriament un inconvenient en els AGPSG.

[3]L'ordinador en el qual Kapsalis i Rayward-Smith van realitzar les seves proves és, avui dia, obsolet.

- La DNH produeix resultats de més qualitat que l'MSpT [57] però, segons Esbensen [30], no és aquest el motiu principal que l'ha impulsat a seleccionar la DNH com a descodificador. El motiu principal és que aquest algorisme permet associar d'una manera relativament ràpida un arbre d'Steiner vàlid per a tot conjunt de vèrtexs donat.

- Quan G compleix la desigualtat triangular, tot $MSpT(G,S)$ podria ser obtingut[4] com a $DNH(G,S)$. Aleshores, les diferències entre el KRGA i l'EGA és minimitzen, la qual cosa fa suposar que aquestes diferències depenen de les característiques dels grafs.

En la secció següent, l'origen genètic dels AGPSG és deixa de banda i l'atenció se centra en els aspectes estratègics d'aquests algorismes per al PSG.

4.3 Línies estratègiques dels AGPSG

En els AGPSG l'espai de solucions està format pels arbres d'Steiner per al conjunt de terminals. El descodificador defineix el subespai de solucions que serà explorat. El procés d'exploració és portat a terme a base de generar de forma iterativa solucions candidates d'una forma esbiaixada, la qual depèn de la població i, en concret, de la manera en què l'encreuament genera noves cadenes una vegada els pares han estat seleccionats. La recombinació de subconjunts de cadena és la clau de la generació de solucions candidates i, per tant, ha de suportar la base estratègica d'on es deriva el rendiment de l'algorisme.

En els apartats següents, els AGPSG són presentats com a implementacions factibles d'un nou subconjunt d'estratègies per al SPG, les *estratègies de recombinació.*

4.3.1 Heurístiques per al PSG: l'estratègia de recombinació

La utilitat dels mètodes heurístics depèn de la seva capacitat per a aconseguir solucions aproximades quan no es coneixen mètodes que puguin retornar una solució

[4]El fet que l'aresta directa sigui un camí de cost mínim entre dos vèrtexs, no significa que, en cas d'igualtat, la DNH no pugui escollir altres camins.

òptima en un temps raonable. Conseqüentment, els mètodes heurístics estan basats en regles que acostumen a complir-se, o com a mínim a no fallar, en la majoria de situacions. Són regles o estratègies que moltes vegades han estat obtingudes a partir d'observacions empíriques, de sentit comú o, fins i tot, de forma intuïtiva, la qual cosa ens dóna una idea de la importància que una FRA té per a aquests algorismes. A continuació, anem a descriure breument les tres línies estratègiques que, segons [57], són utilitzades per la majoria de mètodes heurístics per al PSG:

- Heurístiques de camí: com que els terminals en un arbre d'Steiner òptim solen estar connectats entre ells mitjançant camins de cost mínim, una bona aproximació pot aconseguir-se connectant-los d'aquesta manera. L'exemple més clar el trobem en la SPH (veure 2.4.1).

- Heurístiques de vèrtexs: un cop seleccionats mitjançant algun criteri vèrtexs no-terminals que podrien formar part d'un arbre d'Steiner òptim, un arbre que connecti aquests no-terminals i els terminals podria ser una bona aproximació. L'exemple més clar el trobem en ZH (veure 2.4.3).

- Heurístiques d'arbre: a partir d'un arbre inicial que connecta tots els terminals, s'apliquen diverses estratègies per tal de millorar-ne el cost. La DNH és considerada com una heurística d'arbre (veure 2.4.2).

El rendiment dels algorismes basats en aquestes línies estratègiques està estretament lligat al grau de compliment d'aquestes estratègies per part de les instàncies de prova. És per això que moltes d'aquestes estratègies solen combinar-se en la pràctica. Per exemple, una heurística d'arbre pot ser utilitzada per tal de millorar l'arbre obtingut mitjançant una heurística de camí o de vèrtexs. O també, una heurística d'arbre o de camí pot ser aplicada a un nou conjunt de terminals en què s'inclouen "falsos" terminals, seleccionats a partir d'una heurística de vèrtexs. Anem ara a proposar un nou grup de línies estratègiques:

- Heurístiques de recombinació: com que un arbre d'Steiner òptim està format per subarbres d'Steiner òptims, una bona aproximació contindrà elements — subarbres, vèrtexs d'Steiner, arestes— que són part de solucions òptimes o

quasi-òptimes. Si fóssim capaços de localitzar aquests elements de qualitat de diferents arbres aproximats, podríem utilitzar-los per tal de construir millors aproximacions. En altres paraules, l'estratègia consisteix en construir de forma iterativa millors solucions a base de recombinar parts de qualitat de solucions obtingudes prèviament.

Tres decisions s'han de prendre per tal de dissenyar una heurística de recombinació: com generar bones aproximacions, com detectar les seves parts de qualitat i quina és la millor manera de recombinar-les.

4.3.2 AGPSG: heurístiques de recombinació de vèrtexs

El nostre objectiu és dissenyar un algorisme heurístic de recombinació basat en la idea següent: els no-terminals que acostumen a formar part d'arbres d'Steiner de qualitat tenen més possibilitats de formar part d'un arbre d'Steiner òptim que els que no acostumen a formar-ne part. Per tant, l'algorisme haurà de construir de forma iterativa millors solucions candidates a base de recombinar no-terminals que generalment formen part d'arbres d'Steiner de qualitat.

En els AGPSG, la qualitat d'un no-terminal es mesura en funció del nivell de salut del individus que contenen aquest vèrtex en el subconjunt de cadena i, per tant, en funció del costos dels arbres que el descodificador ha associat a aquests individus. En el procés reproductiu, les cadenes són seleccionades en funció del nivell relatiu de salut dels individus dins la població i, d'aquesta manera, també es seleccionen els no-terminals de subconjunts de cadena per a formar la nova població. Suposant una estratègia de substitució que afecta la totalitat de la població, el nombre de mostres d'un no-terminal en els subconjunts de cadena de la nova població depèn principalment d'aquesta selecció: l'encreuament distribueix aquests vèrtexs en els subconjunts sense modificar la seva presència en la població i la mutació actua amb una probabilitat molt petita. Per tant, un no-terminal que forma part d'un individu amb un alt nivell de salut —un arbre d'Steiner de qualitat— acostuma a ser escollit per a formar part de la següent generació. A més, suposant un mètode de selecció estàndard, és d'esperar que els no-terminals amb una qualitat per sobre de la mitjana

incrementin el nombre d'aparicions en els properes generacions. Per això, d'acord amb la idea que hem exposat al començament, el nombre esperat de proves que es porten a terme amb els no-terminals que acostumen a produir arbres d'una qualitat per sobre de la mitjana s'incrementa de generació en generació[5].

Podem concloure que el comportament bàsic de les AGPSG es correspon a la idea anterior i, per tant, que els resultats que obtinguin aquests algorismes depenen bàsicament de la validesa de l'estratègia de recombinació. És a dir, si els individus són convenientment seleccionats —escollim arbres de qualitat— i l'estratègia de recombinació és vàlida —tendeix a millorar la qualitat de les solucions—, l'algorisme que proposem anirà convergint cap a una solució propera a l'òptima.

De fet, els AGPSG inclouen també algunes de les estratègies heurístiques mencionades en l'anterior apartat. D'una banda, es basen en una heurística de vèrtexs en la qual els vèrtexs d'Steiner són seleccionats en funció de la seva presencia en la població i, per tant, en funció d'una tendència estadística a produir arbres de qualitat. De l'altra, tot el procés és suportat pel descodificador, el qual en l'EGA és una heurística d'arbre i en el GBGA serà una heurística de camí. Conseqüentment, els AGPSG són heurístiques de recombinació de vèrtexs, basades en una estratègia heurística de vèrtexs i suportats per l'heurística associada al descodificador.

En les seccions següents descriurem les decisions que s'han portat a terme al GBGA per tal de reduir el temps de convergència de l'algorisme. Totes aquestes decisions estan orientades a millorar l'efectivitat de l'estratègia de recombinació de vèrtexs: afavorint la generació de solucions de qualitat, seleccionant els vèrtexs de qualitat, i decidint una manera apropiada de recombinar-los.

4.4 GBGA: un AG per al PSG

El GBGA és un AGPSG que segueix l'esquema de l'algorisme 7. En aquesta secció es descriuen de forma detallada els elements que han estat incorporats a l'algorisme. La finalitat de cadascun d'aquests elements és analitzada en les tres seccions següents.

[5]Podem reconèixer en aquesta afirmació una versió simplificada del *teorema dels esquemes* de Holland [52].

Deixant de banda l'elecció d'alguns paràmetres que, en la nostra opinió, tenen un efecte marginal en el rendiment dels AGPSG, les principals diferències entre l'EGA i el GBGA són quatre. En primer lloc, els dos algorismes fan servir un descodificador diferent. En segon lloc, el GBGA utilitza una funció per a reordenar els vèrtexs, similar a la que utilitzava GBA (veure l'apartat 3.3.1). En tercer lloc, el GBGA no utilitza l'operador filtre, però incorpora els tres nous operadors genètics següents:

1. $aSteiner(I_T)$: afegeix a I_S els vèrtexs d'Steiner d'I_T i actualitza, en funció d'aquestes canvis, I_C.
2. $eCorbates(I_T)$: elimina d'I_S les corbates d'I_T i actualitza I_C.
3. $eGrau2(I_T)$: elimina d'I_S els no-terminals d'I_T de grau 2 i actualitza I_C.

En quart i últim lloc, la probabilitat de mutació $pMuta$ es calcula en el GBGA en funció del nombre esperat de bits mutats per cadena ($espMuta$) i es manté constant al llarg de totes les iteracions. En canvi, com en la majoria dels AG, en la primera versió de l'EGA [30] $pMuta$ es calcula en funció de la longitud de les cadenes i, en la segona [31], s'actualitza dinàmicament en funció de la *diversitat de la població*, la qual mesura les diferències entre les cadenes d'una població.

4.4.1 El descodificador

El descodificador del GBGA està basat en la SPH (veure l'apartat 2.4.1) i fa servir la mateixa distinció que en el capítol anterior entre els quatre primers passos d'aquest algorisme, corresponents a la versió originària de Takahashi i Matsuyama (TM) [88], i el càlcul d'un MSpT en el darrer pas (veure la secció 3.1). L'algorisme següent mostra l'esquema final del descodificador, essent la funció *reordenar* com la que està descrita en la pàgina 47.

Pas 1. $T_1 := TM(reordenar(G), S \cup I_S)$, $T_2 := trim(T_1)$;
Pas 2. $T_3 := MSpT(reordenar(G), vertexs(T_2))$, $I_T := trim(T_3)$.

Algorisme 8: Descodificador del GBGA

Anàlisi de la complexitat

La complexitat[6] en el pitjor cas d'una execució del descodificador és $O(|S \cup I_S|(|A| + |V| \log |V|))$, i es deu al càlcul dels camins de cost mínim necessaris. Gràcies als operadors genètics, i exceptuant la primera iteració, es pot assegurar que $|I_S| \leq |S|-2$ (veure l'apartat 4.5.2) i, per tant, la complexitat en el pitjor cas d'una única iteració esdevé $O(|S|(|A| + |V| \log |V|))$. Tot i això, tenint en compte el nombre d'individus que s'avaluen, resulta convenient calcular els camins entre cada parella de vèrtexs un únic cop. Exceptuant la primera iteració, la complexitat en el pitjor cas de l'execució de TM i de l'MSpT és, respectivament, $O(|S||V|)$ i $O(|A| + |V| \log |V|)$, quan es disposa dels camins.

Limitant el nombre d'iteracions a $c|V|/midaPob$ per a una constant c molt petita, el nombre d'execucions del descodificador esdevé $O(|V|)$. Per tant, fent servir els mateixos arguments que en el capítol anterior (veure l'apartat 3.3.5), la complexitat en el pitjor cas del GBGA és $O(|V|(|A| + |V| \log |V| + |S||V|))$, incloent un únic càlcul dels camins entre cada parella de vèrtexs. Val a dir que $|V|$ iteracions s'han mostrat suficients per a produir els resultats del GBGA que presentem més endavant.

4.5 Generació de solucions de qualitat

Amb l'excepció del descodificador del KRGA, les solucions candidates que generen els AGPSG són arbres d'Steiner per al conjunt de terminals. Per tant, podem considerar que cadascun d'aquests descodificadors defineix un subespai d'arbres d'Steiner en el qual es produeix l'exploració per a trobar una bona aproximació. Més endavant, diversos elements provoquen reduccions addicionals en aquest subespai, de manera que no tots els seus arbres poden convertir-se, de forma efectiva, en solucions candidates. Distingim tres tipus de reduccions diferents: les que es decideixen en la fase de disseny, les pròpies del mecanisme d'exploració i les que es produeixen de manera fortuïta.

[6] Els càlculs de la complexitat es realitzen considerant que s'utilitza l'estructura de dades proposada per Fredman i Tarjan [36], i sense tenir en compte la millora de la complexitat de la SPH que es proposa en el capítol 5.

Per tal d'afavorir la generació de solucions de qualitat, un descodificador amb un bon rendiment és necessari però, també, la manera en què l'espai de solucions és reduït juga un paper fonamental.

4.5.1 El rendiment del descodificador

Un descodificador amb bon rendiment hauria de produir arbres relativament bons en un temps raonable i, també, definir un subespai de solucions accessibles de qualitat. Es pot veure que, donada una cadena I_C, el millor descodificador hauria de retornar un $MStT(G, S \cup I_S)$, on I_S és el subconjunt de cadena de I_C. Queda clar, doncs, que cada cadena defineix una nova instància del PSG i, per tant, que el descodificador haurà de ser un mètode heurístic per al problema. D'aquesta manera, el cost d'un individu pot ser considerat com un valor aproximat del cost d'un MStT per a la instància que té associada.

- El descodificador del KRGA no és un algorisme d'aproximació i produeix generalment aproximacions de baixa qualitat [57]. A més, pot associar costos molt alts a individus molt propers a l'òptim. Per exemple, $MSpT(G, S \cup I_S \cup \{v\})$ pot ser una solució òptima i, en canvi, l'individu I amb conjunt de cadena I_S rebrà una forta penalització quan el subgraf de G induït per $S \cup I_S$ no és connex. Aquesta és, en la nostra opinió, la raó principal del pobre rendiment del KRGA.

Deixant de banda aquest fet, el rendiment d'un mètode heurístic depèn del grau en què la instància específica del problema a resoldre s'avé amb les seves regles heurístiques. No tenim la impressió que, ara per ara, hi hagi algun mètode heurístic que tingui un millor rendiment per a tota mena d'instàncies. Per tant, suggerim la possibilitat de canviar el descodificador dinàmicament en funció de les característiques dels grafs, sempre i quan es pugui fer una predicció fiable del rendiment dels algorismes. De moment, però, anem a justificar l'elecció del descodificador per al GBGA:

- Tant la DNH com la SPH milloren clarament el rendiment de l'MSpT [57].
- La SPH acostuma a produir millors aproximacions que la DNH [84, 93, 89, 28] (veure el capítol 5). Tots dos algorismes tenen la mateixa FRA i, quan es disposa

dels camins de cost mínim, còmplexitat similar. A més, totes les propietats de la DNH que han estat mencionades en la pàgina 71 també són també aplicables a la SPH.

- La SPH ha mostrat tenir un comportament especialment bo en grafs generats de forma aleatòria [57], com els grafs que utilitzarem per a les proves.

- La complexitat de la resta d'algorismes d'aproximació en dificulta la seva utilització en la pràctica[7].

Creiem que aquests arguments són suficients per a seleccionar la SPH com a mètode de base per al descodificador del GBGA.

4.5.2 La reducció de l'espai de solucions accessibles

El subespai de solucions que és definit pel GBGA conté tots aquells arbres que poden ser considerats $SPH(G, S \cup I_S)\ \forall I_S$, considerant la SPH de l'algorisme 8. En canvi, el subconjunt de solucions accessibles, que és actualitzat dinàmicament durant l'execució de l'algorisme conté, en la majoria de casos, només un subconjunt d'aquest subespai de solucions. És en aquest subconjunt de solucions accessibles on es produeix realment l'exploració de l'espai. Per tant, un dels objectius de l'algorisme és la progressiva reducció del subconjunt de solucions accessibles de manera que es descartin principalment arbres amb un cost relativament alt. D'aquesta manera, els subconjunts de solucions accessibles van millorant la seva qualitat i, conseqüentment, les possibilitats d'aconseguir una bona aproximació es veuen incrementades. Anem a considerar els elements del GBGA que incideixen en el subconjunt de solucions accessibles.

[7] Amb la reducció de la complexitat de l'algorisme de Zelikovsky [97] proposada per Duin i Voß [28], aquest algorisme s'ha convertit en una alternativa factible a la SPH. Tampoc es tenen en compte les variants de la SPH que proposem en el capítol següent.

Reduccions que es decideixen en la fase de disseny

És coneguda l'existència de, com a mínim, un I_S amb $|I_S| \leq |S|-2$ tal que $SPH(G, S \cup I_S)$ és un $MStT(G, S)$. Conseqüentment, considerant únicament subconjunts de cadena amb, com a molt, $|S| - 2$ no-terminals, es pot reduir l'espai de solucions accessibles sense que, en principi, la possibilitat d'aconseguir una solució òptima es vegi alterada.

En l'EGA aquesta reducció es porta a terme mitjançant l'operador filtre. El comportament d'aquest operador genètic pot ser molt abrupte, ja que no té en compte la incidència que l'eliminació de vèrtexs d'un subconjunt cadena té en el nivell de salut de l'individu i, a l'igual que la mutació, descarta vèrtexs de forma aleatòria, independentment de la seva qualitat. Per exemple, quan $|S|$ és molt petit, el filtre té una incidència tan gran que transforma l'EGA en una mena de *random walk*, restringit a subconjunts de fins a $|S| - 2$ no-terminals. De tota manera, mitjançant l'operador *filtre* la velocitat del descodificador augmenta de manera considerable i, per tant, es poden generar un nombre molt més alt de solucions candidates.

En el GBGA, una reducció semblant s'aconsegueix d'una forma més selectiva per mitjà dels operadors genètics *eCorbates*, *aSteiner* i *eGrau2*, els quals són executats sempre per aquest ordre. En l'apartat 4.6 les propietats d'aquests operadors es consideren des de l'òptica de la recombinació de vèrtexs. A continuació, anem a demostrar que aquests operadors no només garanteixen una reducció de l'espai a subconjunts de cadenes amb com a molt $|S| - 2$ vèrtexs, sinó que les cadenes que en resulten tenen arbres associats com a mínim tan bons com els de les cadenes inicials.

Teorema 1 *Siguin I_S i I_T, respectivament, un subconjunt de cadena i el seu arbre associat, I'_S el subconjunt de cadena després d'aplicar els tres operadors genètics (eCorbates, aSteiner i eGrau2) a I_S i I'_T el seu arbre associat:*

1. *I'_S té, com a molt, $|S| - 2$ vèrtexs no-terminals.*
2. *El cost d'I'_T és no superior al cost d'I_T.*

DEMOSTRACIÓ: Construïm l'arbre T a base d'eliminar els vèrtexs no-terminals de grau 2 d'I_T i reconnectar cada parella de veïns d'aquests vèrtexs mitjançant una

aresta amb cost igual a la distància entre ells. Construint T d'aquesta manera, es compleixen les dues propietats següents: el subconjunt de vèrtexs no-terminals de T es correspon a I'_S i $|T| \leq |I_T|$. Per tant:

1. Els no-terminals de T tenen com a mínim grau tres $\Longrightarrow$ I'_S té, com a molt, $|S| - 2$ vèrtexs.

2. $|I_T| \geq |T| \geq |MSpT(D, S \cup I'_S)| \geq |SPH(G, S \cup I'_S)| = |I'_T|$.

∎

Podem concloure que, degut a l'efecte d'aquests operadors genètics, el subconjunt de solucions accessibles es redueix, la salut dels individus pot millorar i, a més, la velocitat del descodificador es veu normalment incrementada[8]. Per tant, juguen un paper molt important en el rendiment final de l'algorisme.

Reduccions pròpies del mecanisme d'exploració

Al GBGA, la població inicial defineix el subconjunt de l'espai de solucions que pot ser accedit de forma efectiva per l'algorisme[9]. Aquest subconjunt inicial de solucions accessibles té tendència a esdevenir progressivament reduït, de manera que cada nou subconjunt està inclòs en el de la generació anterior. L'èxit del procés d'exploració depèn de la seva habilitat per anar descartant arbres de cost relativament alt i, d'aquesta manera, millorar la qualitat mitjana del subconjunt de solucions accessibles el qual, progressivament, hauria d'anar-se fent més petit i amb millor qualitat mitjana.

En la estratègia de recombinació de vèrtexs, els no-terminals que acostumen a formar part d'arbres de qualitat tenen tendència a ser inclosos en la majoria de subconjunts de cadena i, per contra, els que generalment no en formen part, tenen tendència a anar disminuint la seva presència en la població. D'aquesta manera, els arbres en els què hi dominen els no-terminals del segon tipus, o en què hi ha poca presència dels del primer, van sent descartats del subconjunt de solucions accessibles.

La reducció del conjunt de solucions accessibles va quedant reflectida en la *diversitat* de la població, és a dir, en les diferències que hi ha entre les cadenes. És molt

[8]La complexitat de TM esdevé $O(|S||V|)$ quan es disposa dels camins.

[9]Per tal de simplificar, no tindrem en compte l'efecte de la mutació.

important evitar, sobretot en les primeres fases de l'execució, que les cadenes de la població esdevinguin molt iguals entre elles i, per tant, que el subconjunt de solucions accessibles es vegi reduït d'una forma dràstica.

El manteniment de la diversitat en els AG [42] és un dels problemes clàssics que han estat considerats àmpliament des del punt de vista teòric i pràctic. El mètode de selecció, l'estratègia de substitució, el tipus d'encreuament, la probabilitat de mutació, etc., tots ells tenen una influència important en la diversitat de la població. En la nostra opinió, aquestes eleccions tenen un efecte marginal en el rendiment global de l'algorisme a menys que afectin en un alt grau l'estratègia de recombinació. Per exemple, un mètode de selecció pot promoure una convergència prematura si selecciona de forma repetida un individu privilegiat. O també, una probabilitat de mutació molt alta pot transformar l'algorisme en un *random walk*. És per això que la nostra selecció particular per al GBGA està basada en un subconjunt estàndard de rutines, al qual, pels motius exposats anteriorment, no hem dedicat excessiva atenció.

D'una banda, hem promogut l'exploració del màxim nombre de solucions diferents possible: la població inicial és generada de forma totalment aleatòria, sense introduir cap individu de qualitat que podria promoure la convergència prematura. Dins d'aquesta mateixa línia, la nova generació substitueix totalment l'anterior. Finalment, l'encreuament uniforme és el que permet generar un nombre més gran de cadenes diferents, i és aplicat a totes les parelles d'individus seleccionats (la probabilitat d'encreuament val 1).

D'altra banda, en el mètode de selecció del GBGA, l'*Stochastic Remainder Selection* [42], el paper de l'atzar es redueix a un grau molt petit. L'elecció del valor 2 per al paràmetre inicial fa que el millor individu de la població acostumi a ser seleccionat dues vegades.

Els operadors genètics també contribueixen al manteniment de la diversitat: els tres operadors específics del GBGA, combinats amb la funció reordenar, activen la possibilitat de transformar dues cadenes duplicades en dues de diferents. L'operador de mutació permet, de forma aleatòria, incloure o excloure arbres en el subconjunt de solucions accessibles.

El paper de la mutació

Des de la perspectiva dels AG, la mutació és considerada bàsicament com un mecanisme per a mantenir la diversitat i augmentar la capacitat d'explorar solucions. De tota manera, la forma aleatòria en què la mutació actua dificulta en molts casos justificar-ne la utilitat.

En el GBGA, i en la mateixa línia que la funció sacsejar en el GBA (veure la secció 3.3.3), la mutació és principalment un mecanisme per a donar més opcions a aquells no-terminals que no poden ser incorporats a cap arbre candidat mitjançant la SPH. Ens referim, per exemple, als vèrtexs no-terminals que poden produir arbres de qualitat tot i no ser vèrtexs intermedis de cap camí de cost mínim (veure la figura 2) o, fins i tot, de cap dels camins de cost mínim que han estat calculats (veure la figura 3). Si aquests vèrtexs són exclosos dels conjunts de cadena en les primeres iteracions de l'algorisme, deixen de ser tinguts en compte durant el procés d'exploració. Mitjançant la mutació, tots els vèrtexs tenen opcions de passar a formar part d'alguna solució.

Els canvis que la mutació introdueix en les cadenes han de ser molt petits, de manera que no afectin de forma important ni l'estructura, ni el cost, del seu arbre associat. És per això que $pMuta$ es calcula a partir del nombre de mutacions esperat $espMuta$ per cadena, nombre que, pel que acabem de comentar, ha de ser molt petit:

$$pMuta := espMuta/(|V| - |S|).$$

Reduccions que es produeixen de manera fortuïta

Donat un subconjunt de vèrtexs V_S podem trobar un nombre generalment alt d'arbres que són $TM(G, V_S)$ i, també, alguns que són $MSpT(G, V_S)$. En canvi, com que les implementacions d'aquestes funcions són deterministes, sempre que s'executin per al mateix conjunt de vèrtexs retornaran la mateixa solució. Podríem dir que en cada execució d'un d'aquests algorismes es selecciona arbitràriament[10] una solució dins del subespai de solucions definit per aquestes heurístiques. Per tant, només un dels arbres que formen part d'aquest subespai por ser accedit de forma efectiva, sense que

[10]Distingim entre una decisió arbitrària i una d'aleatòria en el sentit en què, en la mateixa situació, la primera sempre retornarà el mateix resultat, cosa que no té per què succeir amb la segona.

el criteri utilitzat per a escollir-lo formi part de les especificacions que es donen en la definició dels algorismes. La reducció de l'espai de solucions accessibles és evident i es produeix d'una forma totalment fortuïta.

El problema, però, es veu agreujat en els processos iteratius (veure l'apartat 3.3.1), ja que les arbitrarietats no solen repartir-se de manera uniforme. Per exemple, s'estableixen prioritats entre els vèrtexs les quals, en mantenir-se al llarg de tot el procés, condueixen cap a certes solucions, en detriment d'altres igualment vàlides.

El GBGA reordena de forma aleatòria les etiquetes dels vèrtexs abans de cada execució del TM i de l'MSpT per tal d'evitar parcialment[11] aquestes dependències arbitràries. D'aquesta manera, el descodificador esdevé no-determinista i dóna opcions a que siguin accedits un nombre molt més gran d'arbres del subespai de solucions definit pel descodificador. Gràcies a la funció *reordenar*, els resultats que obté el GBGA per a unes instàncies concretes poden ser considerats vàlids per als grafs isomorfs. A més, la tendència a produir arbres semblants a partir de cadenes similars es redueix, amb la qual cosa s'augmenta la capacitat de l'algorisme d'explorar solucions diferents. Aquest fet resulta especialment important quan, en les fases finals de l'execució de l'algorisme, les cadenes de la població acostumen a esdevenir més homogènies.

Per tal de reduir la complexitat, els camins de cost mínim entre cada parella de vèrtexs són calculats una única vegada abans de l'execució de l'algorisme. Per tant, el mateix camí entre dos vèrtexs és seleccionat de forma arbitrària quan n'hi ha més d'un del mateix cost. En el GBGA utilitzem l'estratègia de donar prioritat als camins amb vèrtexs intermedis respecte als que tenen una única aresta. Tal i com ha estat descrit en l'apartat 3.3.4, aquesta estratègia, que amb prou feines modifica la complexitat de l'algorisme, té una incidència positiva en el seu rendiment.

4.6 Selecció de vèrtexs de qualitat

La interdependència entre el nombre de mostres que s'espera que rebi un vèrtex no-terminal en la següent generació i el cost dels arbres associats als individus que contenen aquest vèrtex en el conjunt de cadena, pot produir situacions paradoxals

[11] La funció no incideix en el càlcul de camins de cost mínim.

des del punt de vista de l'estratègia de recombinació de vèrtexs. Per exemple, un no-terminal pot incrementar el nombre esperat de mostres fins i tot quan és corbata i, per tant, no contribueix de forma directa al cost de l'arbre. En canvi, un no-terminal que és vèrtex d'Steiner de l'arbre associat, no veu incrementat el nombre esperat de mostres si no forma part del subconjunt de cadena. Els operadors genètics dissenyats específicament per al GBGA permeten evitar aquestes situacions paradoxals:

- Mitjançant *eCorbates*, el nombre esperat de mostres d'un vèrtex corbata en la següent generació no es veu incrementat.
- Mitjançant *aSteiner*, els no-terminals que no formen part del subconjunt de cadena, però si de l'arbre associat, incrementen el nombre esperat de mostres.
- Mitjançant *eGrau2*, els no-terminals en el subconjunt de cadena que són prescindibles per a millorar el cost de l'arbre no incrementen el nombre esperat de mostres —els seus veïns poden reconnectar-se mitjançant una aresta amb cost igual al camí de cost mínim (veure l'apartat 3.3.2).

Podem concloure que, a més dels avantatges que han estat presentats en la secció anterior —reducció de l'espai de solucions, millora de la salut dels individus i increment de la velocitat del descodificador— els tres nous operadors genètics que utilitza el GBGA actuen d'acord amb les línies marcades per l'estratègia de recombinació de vèrtexs.

4.7 El mecanisme de recombinació

Una vegada establert el mecanisme basat en la SPH per a generar arbres de qualitat i el procediment per avaluar els vèrtexs, el nostre objectiu consisteix en dissenyar un mecanisme de recombinació adequat, per tal d'utilitzar vèrtexs de qualitat per a construir noves solucions candidates. Concretament, estem buscant una manera per a decidir quins vèrtexs han d'anar junts i quins no en els nous subconjunts de cadena que es vagin construint. Cal tenir present que les millores més substancials són generalment produïdes per la inclusió de grups de vèrtexs d'Steiner actuant conjuntament,

més que no pas per vèrtexs aïllats. Des d'aquest punt de vista, un mecanisme ideal de recombinació hauria d'intentar salvaguardar aquestes associacions de vèrtexs en la construcció dels nous subconjunts.

El mecanisme de recombinació que hem escollit per al GBGA està basat en l'encreuament uniforme i està més orientat a evitar associacions "incertes" —la probabilitat que un vèrtex formi part d'un fill o de l'altre és independent de la resta de vèrtexs— que no pas a preservar associacions "exitoses". Aquest fet es deu a la dificultat de detectar quines associacions haurien de ser preservades i quines no. De tota manera, un mecanisme de recombinació que, com l'encreuament uniforme, combina parelles de subconjunts, sembla tenir més possibilitats de preservar associacions de vèrtexs que no pas un mètode que recombini més de dos subconjunts, o que un altre que consideri cada vèrtex de forma independent d'una forma purament estadística [8]. Des d'un altre punt de vista, l'encreuament uniforme també sembla més indicat que altres tipus d'encreuament (*one-point*, *two-point*, etc., [42]) els quals creen dependències entre els bits, i per tant, entre els vèrtexs associats, en funció de les posicions que ocupen en la cadena. En la nostra opinió, aquests mètodes d'encreuament poden resultar interessants si van associats a una estratègia de distribució dels vèrtexs que permeti treure profit d'aquestes dependències.

4.8 Resultats experimentals

El GBGA ha estat provat utilitzant les 60 instàncies més grans dels grafs proposats per Beasley [10] de la OR-library [58], que denotem per c1,...,c20, d1,...,d20, e1,...,e20. Cadascun dels graf de tipus *c*, *d* i *e* té, respectivament, 500, 1000 i 2500 vèrtexs. Amb l'objectiu de comparar el rendiment del GBGA amb el de l'algorisme genètic EGA proposat per Esbensen [30, 31], les proves realitzades s'han adaptat a les que es van portar a terme per a aquest algorisme i, per tant, són les mateixes que han estat descrites en el capítol anterior per al GBA (veure la secció 3.4). Aquests dos algorismes han estat executats en la mateixa màquina i, fins i tot, comparteixen algunes parts de codi, com ara la implementació de la SPH. Per tant, les comparacions entre ells es realitzen exactament en les mateixes condicions.

Els paràmetres inicials del GBGA són $midaPob = 40$ i $espMuta = 3$. Les condicions de parada, $stopMaxInd$ i $stopMaxPop$ han estat inhibides, mentre que $stopMaxIter = nIn/midaPob$, on nIn és un nombre fix d'individus descodificats i té la mateixa funció que nIt en el GBA. En altres paraules, l'algorisme finalitza la seva execució després d'avaluar nIn individus, on el valor de nIn és generalment inicialitzat en funció del nombre de vèrtexs del graf. Per tant, quan nIn és igual a nIt, el nombre d'execucions de la SPH per part del GBGA i del GBA és el mateix. En aquest cas, tenint en compte que en tots dos algorismes el nombre de terminals per instància és limitat a $2|S|-2$, es pot considerar que el temps d'una execució de la SPH és igual per als dos mètodes. (Els temps lleugerament superiors del GBGA cal atribuir-los a la no limitació del nombre de terminals en la primera iteració i, també, als processos addicionals que ha de realitzar —selecció, reproducció, etc.)

Les taules 6, 7 i 8 mostren els resultats obtinguts pel GBGA per als grafs c, d i e, respectivament. Per a destacar les diferències entre els tres algorismes, la taula 9 mostra els resultats de cadascun d'ells per a aquells grafs en què no s'ha obtingut una solució òptima en totes les execucions. Les taules 10 i 11 conclouen els resultats d'aquestes taules i faciliten la comparació amb els resultats corresponents a la segona versió de l'EGA (EGA2) [31], i amb els del GBA. Aquesta darrera taula inclou també els resultats d'un dels mètodes de Duin i Voβ (DVA) [28]. Finalment, les taules 12, 13 i 14 permeten comparar el nombre d'execucions de la SPH per part del GBA i el GBGA fins a obtenir la solució final. En aquestes taules també s'ha inclòs informació del *branch and cut* proposat per Koch i Martin (KMBC) [67].

4.8.1 Anàlisi del rendiment

A partir de les taules 6, 7, 8 i 9 podem concloure que quan $nIn = |V|$ el GBGA troba una solució òptima en el 87,6% del total d'execucions, mentre que retorna una solució amb un error per sota de l'1% en el 99,5%. L'error només és superior a l'1% en tres de les 10 execucions per al graf $d18$, essent l'error màxim en aquest graf de l'1,3%, que és l'equivalent a 3 unitats de cost. La taula 11 mostra que la mitjana d'error per a totes les execucions és del 0,07%, i que l'error mitjà màxim és de l'1,03%, corresponent a

Taula 6: Resultats experimentals del GBGA amb els grafs c

	Mida Reduïda			Cost					Temps de CPU	
G	$\|V\|$	$\|N\|$	$\|E\|$	opt.	$nIn = 500$		$nIn = 5000$		red	$nIn = 500$
c1	145	5	263	85	—		—		5	0
c2	130	8	239	144	—		—		5	0
c3	120	35	232	754	0,1	(0-1)	0,1	(0-1)	5	1
c4	109	38	221	1079	—		—		5	1
c5	37	17	91	1579	—		—		5	0
c6	369	5	847	55	—		—		5	1
c7	382	9	869	102	—		—		5	1
c8	336	54	818	509	—		—		5	2
c9	349	78	832	707	0,3	(0-1)	—		6	4
c10	213	76	624	1093	—		—		6	2
c11	499	5	2184	32	—		—		5	1
c12	498	9	2236	46	—		—		5	1
c13	463	65	2108	258	—		—		5	4
c14	427	81	1961	323	—		—		6	3
c15	299	92	1471	556	—		—		6	3
c16	500	5	4740	11	—		—		5	1
c17	499	9	4698	18	—		—		5	1
c18	486	70	4668	113	—		—		6	3
c19	473	98	4490	146	—		—		6	4
c20	386	143	3850	267	—		—		6	5

- Mida Reduïda: característiques dels grafs un cop s'han executat les rutines de reducció.
- Cost: cost d'una solució òptima (opt.) i la diferència mitjana respecte aquest cost després de 10 execucions avaluant nIn individus per a cada graf. Un guió indica que una solució òptima s'ha obtingut en totes 10 execucions. Cas contrari, entre parèntesis i separats per un guió, s'indica la diferència entre els costos de la millor i la pitjor aproximació, respectivament, i el cost de l'òptim.
- Temps de CPU: indica els segons de CPU utilitzats per les rutines de reducció (red), i pel procés iteratiu quan $nIn = |V|$. El temps de reducció és semblant al d'una execució de l'algorisme de Floyd i, per tant, es pot fer servir de referència per a futures comparacions.
- Per exemple, per al graf $c9$, l'error absolut mitjà de 10 execucions del GBGA, cadascuna calculant 500 ($= |V|$) individus, val 0,3. El (0-1) a la dreta d'aquest valor indica que la millor aproximació obtinguda té cost 707+0, mentre que la pitjor 707+1, essent 707 el valor de l'òptim per al graf. Incrementant el nombre d'individus fins a 5.000 ($= 10|V|$) s'obté una solució òptima en totes 10 execucions.

Taula 7: Resultats experimentals del GBGA amb els grafs d

	Mida Reduïda			Cost					Temps de CPU	
G	$\|V\|$	$\|N\|$	$\|E\|$	opt.	$nIn = \|V\|$		$nIn = 10\|V\|$		red	$nIn = \|V\|$
d1	274	5	510	106	—		—		50	1
d2	285	10	523	220	—		—		50	1
d3	224	67	441	1565	—		—		51	4
d4	159	66	339	1935	—		—		51	3
d5	97	48	246	3250	—		—		52	2
d6	761	5	1741	67	—		—		50	4
d7	754	10	1735	103	—		—		50	4
d8	731	124	1708	1072	—		—		51	24
d9	654	155	1613	1448	—		—		52	28
d10	418	146	1317	2110	—		—		53	13
d11	993	5	4674	29	—		—		50	6
d12	1000	10	4671	42	—		—		50	6
d13	922	122	4433	500	—		—		51	23
d14	853	160	4173	667	—		—		52	28
d15	550	157	2925	1116	—		—		54	15
d16	1000	5	10595	13	—		—		50	5
d17	999	9	10531	23	—		—		51	5
d18	978	145	10140	223	2,3	(2-3)	1,6	(1-2)	50	24
d19	938	193	9676	310	1,6	(1-2)	1,0	(1-1)	51	33
d20	814	324	8907	537	—		—		52	59

Per exemple, per al graf $d19$, l'error absolut mitjà de 10 execucions del GBGA, cadascuna calculant 1.000 ($= |V|$) individus, val 1,6. El (1-2) a la dreta d'aquest valor indica que la millor aproximació obtinguda té cost 310+1, mentre que la pitjor 310+2, essent 310 el valor de l'òptim per al graf. Tot i incrementar el nombre d'individus fins a 10.000 ($= 10|V|$), no s'aconsegueix cap solució òptima.

Taula 8: Resultats experimentals del GBGA amb els grafs e

	Mida Reduïda			Cost					Temps de CPU	
G	$\|V\|$	$\|N\|$	$\|E\|$	opt.	$nIn = \|V\|$		$nIn = 10\|V\|$		red	$nIn = \|V\|$
e1	680	5	1286	111	—		—		773	5
e2	710	9	1328	214	—		—		773	6
e3	637	199	1233	4013	0,7	(0-2)	—		779	118
e4	435	164	964	5101	0,1	(0-1)	—		783	42
e5	222	108	649	8128	—		—		800	15
e6	1845	5	4318	73	—		—		756	23
e7	1891	10	3388	145	—		—		775	26
e8	1723	286	4193	2640	0,2	(0-1)	0,1	(0-1)	791	372
e9	1608	358	4069	3604	—		—		799	460
e10	1046	366	3388	5600	0,3	(0-1)	—		827	284
e11	2498	5	12093	34	—		—		780	39
e12	2500	10	12123	67	—		—		777	39
e13	2341	321	11760	1280	0,9	(0-1)	0,9	(0-1)	788	476
e14	2139	388	11325	1732	0,9	(0-1)	0,1	(0-1)	802	498
e15	1461	443	8514	2784	—		—		837	421
e16	2500	5	29332	15	—		—		781	34
e17	2500	10	29090	25	—		—		779	34
e18	2429	355	28437	564	4,3	(3-5)	2,7	(2-3)	781	390
e19	2351	485	27779	758	2,9	(2-3)	1,9	(1-2)	787	554
e20	1988	758	24423	1342	—		—		801	1000

Per exemple, per al graf $e18$, l'error absolut mitjà de 10 execucions del GBGA, cadascuna calculant 2.500 ($= |V|$) individus, val 4,3. El (3-5) a la dreta d'aquest valor és redueix fins a (2-3) quan el nombre d'individus s'incrementa fins a 25.000 ($= 10|V|$), la qual cosa indica que la millor aproximació obtinguda en aquestes 10 execucions té cost 564+2, mentre que la pitjor 564+3, essent 564 el valor de l'òptim per al graf.

Taula 9: Resultats per graf de l'EGA2, el GBA i el GBGA

Graf	opt.	EGA2 $nIn > 50,000$	GBA $nIn = \|V\|$	GBA $nIn = 10\|V\|$	GBGA $nIn = \|V\|$	GBGA $nIn = 10\|V\|$)
c3	754	—	0,5 (0-1)	—	0,1 (0-1)	0,1 (0-1)
c9	707	0,2	0,7 (0-1)	—	0,3 (0-1)	—
c13	258	0,3	—	—	—	—
c16	11	0,3	—	—	—	—
c18	113	1,1	—	—	—	—
c19	146	0,6	—	—	—	—
d3	1565	—	0,2 (0-2)	—	—	—
d8	1072	0,2	1,3 (0-3)	—	—	—
d9	1448	0,1	0,3 (0-2)	—	—	—
d10	2110	—	0,8 (0-2)	—	—	—
d13	500	0,5	—	—	—	—
d14	667	1,4	—	—	—	—
d18	223	3,1	1,4 (1-2)	1,1 (1-2)	2,3 (2-3)	1,6 (1-2)
d19	310	2,4	1,0 (1-1)	1,0 (1-1)	1,6 (1-2)	1,0 (1-1)
e3	4013	—	5,6 (3-8)	2,5 (2-3)	0,7 (0-2)	—
e4	5151	—	1,9 (1-3)	0,3 (0-1)	0,1 (0-1)	—
e8	2640	—	1,7 (0-4)	0,3 (0-1)	0,2 (0-1)	0,1 (0-1)
e9	3604	—	3,7 (2-6)	1,8 (0-3)	—	—
e10	5600	—	4,1 (1-5)	2,0 (1-3)	0,3 (0-1)	—
e12	67	1	—	—	—	—
e13	1280	11	1,5 (1-2)	—	0,9 (0-1)	0,9 (0-1)
e14	1732	1	1,2 (0-2)	—	0,9 (0-1)	0,1 (0-1)
e15	2784	1	—	—	—	—
e18	564	20	2,4 (2-3)	1,9 (1-2)	4,3 (3-5)	2,7 (2-3)
e19	758	5	0,1 (0-1)	—	2,9 (2-3)	1,9 (1-2)

Resultats que obtenen els algorismes de la taula 10 per a aquells grafs en què, com a mínim, algun dels algorismes no ha obtingut la solució òptima en totes 10 execucions. (Per als grafs *e*, l'EGA2 ha estat executat una única vegada). Un guió indica que l'òptim ha estat obtingut en totes les execucions. Cas contrari, entre parèntesis i separats per un guió, s'indica la diferència entre els costos de la millor i la pitjor aproximació, i l'òptim, respectivament. (Desconeixem aquests valors per a l'EGA2.)

Taula 10: Comparant els resultats amb els d'altres mètodes

Grafs	EGA2 $nIn > 50000$		GBA $nIn = \|V\|$		GBA $nIn = 10\|V\|$		GBGA $nIn = \|V\|$		GBGA $nIn = 10\|V\|$	
tipus	0%	$< 1\%$	0%	$< 1\%$	0%	$< 1\%$	0%	$< 1\%$	0%	$< 1\%$
c	90,5	94,0	94,5	100	100	100	98,0	100	99,5	100
d	83,5	95,5	80,5	100	90,0	100	90,0	98,5	90,0	100
e	70,0	90,0	61,0	100	77,5	100	75,0	100	84,5	100
Total	81,3	93,2	78,6	100	89,2	100	87,6	99,5	91,3	100

Percentatge de solucions òptimes (0%) i d'aproximacions amb error inferior a l'1% ($< 1\%$) per a cadascun dels algorismes. Els resultats de l'EGA2 han estat extrets de [31], mentre que els del GBA han estat presentats en el capítol anterior. Les altres dues columnes resumeixen els resultats del GBGA que es mostren en les taules 6, 7 i 8, quan el nombre d'individus avaluats és igual a $|V|$ i a $10|V|$, respectivament.

Taula 11: Error mitjà (*mig*) i error màxim (*max*) en % per als diferents algorismes

Graf	DVA		EGA2 $nIn > 50,000$		GB $nIn = \|V\|$		GB $nIn = 10\|V\|$		GBGA $nIn = \|V\|$		GBGA $nIn = 10\|V\|$	
tipus	*mig*	*max*	*mig*	*max*	*mig*	*max*	*mig*	*max*	*mig*	*max*	*mig*	*max*
c	0,75	3,5	0,21	2,73	0,01	0,10	0,00	0,00	0,00	0,04	0,00	0,01
d	0,54	3,4	0,13	1,39	0,06	0,63	0,04	0,49	0,08	1,03	0,05	0,72
e			0,33	3,55	0,05	0,43	0,03	0,34	0,06	0,76	0,04	0,48
Total	0,64	3,5	0,22	3,55	0,04	0,63	0,02	0,49	0,05	1,03	0,03	0,72

Els resultats del DVA es corresponen als de l'algorisme identificat com a *Svertex+reBuilt* pels seus autors, Duin i Voß [28]. Els resultats per a la resta de mètodes poden calcular-se a partir de la taula 9.

Taula 12: Nombre d'execucions de la SPH per al grafs c.

	KMBC	GBA (500)		GBGA (500)	
graf	total	mig	màxim	mig	màxim
c1	253	1	1	40	40
c2	330	1	1	60	80
c3	88	(152)	(483)	(92)	(160)
c4	88	143	446	68	120
c5	143	1	1	40	40
c6	528	6	46	52	80
c7	385	6	46	48	80
c8	253	26	228	84	120
c9	187	(142)	(426)	(340)	(520)
c10	77	92	222	80	200
c11	1298	15	75	68	120
c12	825	6	13	44	80
c13	275	72	301	212	400
c14	165	63	175	120	200
c15	99	13	143	168	240
c16	242	5	28	44	80
c17	825	17	47	52	80
c18	440	36	70	304	480
c19	264	55	152	308	400
c20	77	12	27	72	120
mitjà	242	43	147	115	178

La columna KMBC indica el nombre total d'execucions de la SPH que realitza el mètode *branch and cut* proposat per Koch i Martin [67] fins que es para. Els altres dos mètodes, el GBA i el GBGA, executen 500 vegades aquest algorisme, degut a la condició de parada. La columna (mig) mostra el nombre mitjà d'execucions de la SPH fins a trobar la millor solució, òptima o quasi-òptima, en les 10 proves que s'han realitzat per graf. El màxim nombre d'execucions necessari en aquestes 10 proves s'indica en la columna (màxim). Els valors entre parèntesis denoten que en alguna d'aquestes proves no s'ha obtingut la solució òptima.

Taula 13: Nombre d'execucions de la SPH per al grafs *d*.

	KMBC	GBA (1000)		GBGA (1000)	
graf	total	mig	màxim	mig	màxim
d1	429	8	16	40	40
d2	308	2	11	40	40
d3	77	(198)	(519)	336	600
d4	143	3	9	56	80
d5	88	84	321	100	160
d6	1419	16	45	40	40
d7	1100	1	1	68	80
d8	242	(257)	(620)	398	600
d9	132	(274)	(556)	304	360
d10	121	(303)	(889)	276	400
d11	1727	128	270	64	120
d12	1177	1	1	40	40
d13	220	108	344	188	280
d14	231	71	195	140	200
d15	121	89	385	112	120
d16	770	11	53	40	40
d17	1265	3	13	48	80
d18	385	(406)	(967)	(632)	(840)
d19	517	(182)	(468)	(604)	(920)
d20	132	26	78	156	240
mitjà	530	109	280	184	264

Nombre (total) d'execucions de la SPH que realitza el KMBC [67] fins que es para, i nombre mitjà (mig) i màxim (màxim) d'execucions de la SPH fins que el GBA i el GBGA troben la millor solució en les 10 proves que s'han realitzat per graf. Els valors entre parèntesis denoten que en alguna d'aquestes no s'ha obtingut la solució òptima.

Taula 14: Nombre d'execucions de la SPH per al grafs *e*.

	KMBC	GBA (2500)		GBGA (2500)	
graf	total	mig	màxim	mig	màxim
e1	473	2	6	40	40
e2	781	13	37	48	120
e3	143	(1370)	(2255)	(492)	(2280)
e4	99	(960)	(1914)	(1272)	(2120)
e5	143	164	393	236	160
e6	869	7	28	48	80
e7	1496	23	128	72	120
e8	275	(852)	(1841)	(772)	(1280)
e9	253	(1112)	(2364)	612	1160
e10	143	(379)	(1638)	(1760)	(2160)
e11	1430	7	27	40	40
e12	1452	107	641	48	80
e13	451	(1403)	(2183)	(496)	(680)
e14	220	(627)	(1791)	(1272)	(2040)
e15	264	(201)	(504)	236	320
e16	2937	19	80	40	40
e17	1936	11	35	68	80
e18	3366	(1201)	(2366)	(2036)	(2520)
e19	330	(889)	(2116)	(1700)	(2360)
e20	176	37	51	300	480
mitjà	862	469	1020	579	908

Nombre (total) d'execucions de la SPH que realitza el KMBC [67] fins que es para, i nombre mitjà (mig) i màxim (màxim) d'execucions de la SPH fins que el GBA i el GBGA troben la millor solució en les 10 proves que s'han realitzat per graf. Els valors entre parèntesis denoten que en alguna d'aquestes no s'ha obtingut la solució òptima.

la mitjana de les 10 execucions per al graf $d18$. Incrementant nIn fins a $10|V|$, el nombre de solucions òptimes arriba fins al 91,3% del total d'execucions —passa del 75% al 84,5% per als grafs e—, mentre que l'error màxim es manté sempre per sota de l'1%.

Els temps d'execució de la part iterativa de l'algorisme es mantenen generalment per sota dels de calcular els camins de cost mínim entre cada parella de vèrtexs utilitzant l'algorisme $O(|V|^3)$ de Floyd [33], temps que és similar al de reducció de la mida dels grafs (red). Aquest temps, així com els resultats, es veuen clarament influenciats per les condicions de parada. Les taules 12, 13 i 14 mostren el nombre mitjà i màxim d'individus avaluats fins a trobar la solució final que retorna l'algorisme quan $nIn = |V|$. Com a mínim s'avalua la població inicial i, per tant, el nombre mínim d'individus és 40. En aquestes tres taules es pot apreciar la feblesa de molts dels grafs, sobretot aquells acabats en números congruents amb 1 o 2 mòdul 5, que tenen molt pocs vèrtexs terminals. Per a tots aquests grafs s'aconsegueix una solució òptima amb menys de quatre iteracions i, per tant, contribueixen a augmentar de manera artificial els percentatges de solucions òptimes que aconsegueixen els algorismes.

Comparació amb l'EGA

Les diferències entre l'EGA i el GBGA són encara més marcades que les que han estat descrites en el capítol anterior entre l'EGA i el GBA (veure la 3.4.1). Els resultats del GBGA poden comparar-se amb els de la segona versió de l'algorisme d'Esbensen [31] (EGA2) a partir de les taules 9, 10 i 11. La millora del GBGA respecte l'EGA2 és molt important, la qual cosa resulta significativa tenint en compte que l'EGA2 obté una solució òptima en el 81,3% d'execucions, i que està per sota de l'1% d'error en el 93%. Més remarcable encara, però, resulta la comparació relativa al nombre d'arbres que avalua cada algorisme per tal d'obtenir els resultats que presentem en aquestes taules, tenint en compte que la complexitat d'avaluar un arbre és similar. En funció de les condicions de parada dels algorismes, podem assegurar que quan $nIn = |V|$, el GBGA avalua com a mínim 100, 50 i 20 vegades menys arbres que l'EGA2 per als grafs c, d i e, respectivament.

Comparació amb el GBA

Tenint en compte l'autoria de tots dos algorismes, les comparacions entre el GBGA i el GBA no haurien de resultar difícils. Tot i això, la qualitat de les solucions que obtenen tots dos mètodes dificulta establir conclusions definitives.

D'una banda, a partir de la taula 10, resulta clara la superioritat del GBGA en el percentatge de solucions òptimes que aconsegueix, quan es compara en igualtat de condicions amb el GBA. Per exemple, quan $nIn = |V|$, el nombre de solucions òptimes obtingudes pel GBGA és sempre superior al del GBA, arribant a aconseguir una millora de quasi un 23% per als grafs e. Aquesta relació, però, pot resultar enganyosa, tal i com es pot observar en la taula 11. La mitjana d'error relatiu del GBA és lleugerament inferior a la del GBGA, fins i tot per els grafs e. De tota manera, i tornant a la dificultat d'establir conclusions definitives, les diferències són mínimes i venen produïdes, tal com es pot veure en la taula 9, pels grafs $d18$, $d19$, $e18$ i $e19$. En aquests quatre grafs, els únics pels quals el GBGA no obté cap solució òptima, el GBA millora els resultats del GBGA, resultant especialment sorprenents en el cas d'$e19$. D'altra banda, el GBGA supera els resultats del GBA en 13 dels grafs, marcant diferències menors, per a les instàncies de prova menys denses. A més, els intervals en què varien les solucions del GBA són generalment més grans, especialment quan $nIt = |V|$.

Finalment, les taules 12, 13 i 14 permeten comparar el nombre d'arbres avaluats pels dos algorismes fins a trobar la solució que retornen. Podem observar que, si bé el nombre mitjà d'iteracions fins a obtenir la solució final és sempre inferior en el GBA, el nombre d'iteracions del GBGA es va mostrant més estable, a mesura que augmenta el nombre de vèrtexs.

Comparació amb el DVA

La comparació del GBGA amb els mètodes de Duin i Voβ [28], els quals han estat descrits en l'apartat 2.5.3, resulta complicada. En un principi, la complexitat dels dos algorismes és similar, tenint en compte que el factor dominant és degut més al càlcul dels camins que al procés iteratiu. L'excepció la trobem quan els algorismes

treballen amb camins aproximats —calculats mitjançant un mètode de complexitat quadràtica que proposen els mateixos autors—, sense que aquest fet repercuteixi de forma excessiva en la qualitat de les solucions.

La taula 11 presenta els resultats en el mateix format en què Duin i Voβ exposen els seus. El mètode DVA de la taula es correspon a l'algorisme *Svertex+reBuilt* [28], el qual millora en qualitat a la resta d'algorismes proposats. Tot i que el DVA no ha executat rutines per a reduir la mida dels grafs, la taula 11 mostra la clara superioritat del GBGA per als grafs *c* i *d*, els únics que disposem per a fer les comparacions.

Comparació amb el KMBC

El *branch and cut* proposat per Koch i Martin [67] (KMBC) (veure l'apartat 2.5.5), és el primer mètode exacte capaç d'obtenir una solució òptima per a tots els grafs de Beasley, essent el primer algorisme que troba l'òptim per al graf *e18*[12]. Tenint en compte la complexitat del PSG, el principal interès del KMBC es troba, des del nostre punt de vista, en la possibilitat d'obtenir una fita dinàmica de l'error comès, sempre que aquesta fita sigui suficientment ajustada. És a dir, no considerem tan important el fet que sigui un mètode exacte, com la possibilitat que ofereix al donar una garantia de la qualitat de l'aproximació que millori la FRA. Els mètodes basats en la programació lineal tindran interès en funció de la seva capacitat per a produir bones aproximacions en un temps raonable i afinar el seu marge d'error.

En la taules 12, 13 i 14 es mostra el total d'execucions de la SPH per graf en el KMBC. Aquests valors tenen un interès purament orientatiu, ja que no poden ser comparats directament amb els dels altres mètodes. D'una banda, les execucions del KMBC estan orientades a trobar les solucions òptimes, sense que es disposi d'informació intermèdia. Per tant, els valors que es mostren en les taules no fan referència a l'execució en que s'obté per primera vegada una solució òptima, sinó al nombre d'execucions necessàries fins que es pot assegurar, a partir de la fita obtinguda via programació lineal, que la solució és òptima. De l'altra, el KMBC modifica els costos

[12]Prèviament, una execució exhaustiva de rutines de reducció va deixar aquest graf amb $|V| = 2224$, $|A| = 5996$ i $|S| = 394$. Tot i això, van ser necessàries més de 19 hores de CPU d'una *Sun SPARC 20 Model 71*. Sembla, però, que els temps d'execució del KMBC han millorat de forma sensible [66].

associats a les arestes en funció del resultat de les relaxacions del problema. Aquest fet obliga a recalcular els camins[13] de cost mínim en cada execució de la SPH[14] i, per tant, la complexitat d'una execució d'aquest algorisme és molt superior en el cas del KMBC. Cal afegir que el temps que aquest algorisme dedica a l'execució de la SPH és, de mitjana, inferior a la quarta part del temps global d'execució.

4.9 Conclusions

El contingut d'aquest capítol està motivat per la dificultat de justificar, tot i les seves similituds, la impressionant diferència de rendiment mostrada entre el KRGA i l'EGA, i la convicció que els resultats obtinguts mitjançant un AG no són considerats tan valuosos com els que s'obtenen utilitzant mètodes heurístics no evolutius. Veiem totes dues motivacions com una conseqüència de les dues maneres en què tradicionalment el funcionament dels AG és justificat: assumint l'existència d'algunes propietats inherents als AG que són independents del problema, o bé a partir dels resultats experimentals.

Proposem un nou algorisme, el GBGA, basat en un nou conjunt d'estratègies heurístiques per al PSG, les heurístiques de recombinació. El GBGA és també un AG i pot ser considerat com un successor dels dos algorismes mencionats a dalt, amb una diferència important: mentre que el GBGA es basa en una estratègia per al PSG que pot ser implementada mitjançant un AG, el KRGA i l'EGA es basen en les hipòtesis generals que donen suport als AG. En altres paraules, un AG és un mecanisme indicat per a implementar el conjunt de línies estratègiques que hem escollit per al PSG. En canvi, les línies estratègiques que regeixen el KRGA i l'EGA s'han d'extreure del concepte general d'evolució que permet als AG obtenir bons resultats per a alguns problemes determinats.

L'heurística de recombinació de vèrtexs proporciona les línies bàsiques que donen suport a les decisions en el disseny del GBGA, i que permeten interpretar el funcionament dels AG per al PSG. A partir d'aquestes línies, les diferències importants de

[13]No es té en compte la implementació alternativa de la SPH que es proposa en el capítol següent.
[14]En realitat, els camins s'han de recalcular cada 11 execucions de la SPH.

rendiment entre el KRGA i l'EGA es justifiquen, l'objectiu de les principals funcions que incorpora el GBGA s'analitza, i es poden prendre decisions per tal de millorar el rendiment en relació al de l'EGA. El GBGA utilitza un descodificador diferent, incorpora un subconjunt d'operadors genètics més selectiu, evita parcialment la dependència en l'ordre dels vèrtexs i considera un mètode alternatiu per a calcular la probabilitat de mutació. Com a resultat, el GBGA aconsegueix resultats sensiblement superiors als de l'EGA en un temps clarament inferior.

L'algorisme ha estat provat fent servir els grafs més grans de Beasley [10] de la OR-library, aconseguint una solució amb error inferior a l'1% en el 99,5% d'execucions. La solució òptima s'obté en el 87,6% de casos quan el nombre d'arbres avaluats és igual a $|V|$, i arriba fins al 91,3% quan aquest nombre s'incrementa fins a $10|V|$. En el primer cas, els temps d'execució de la part iterativa són normalment inferiors als de calcular els camins de cost mínim entre cada parella de vèrtexs mitjançant l'algorisme de Floyd, que té complexitat $O(|V|^3)$. Quan el nombre d'arbres avaluats és limita a $O(|V|)$, la complexitat en el pitjor cas del GBGA és $O(|V|(|A|+|V|\log|V|+|S||V|))$, incloent un càlcul previ dels camins de cost mínim entre cada parella de vèrtexs. A més a més, amb la utilització de la funció *reordenar*, els resultats del GBGA poden ser considerats quasi independents de l'ordre dels vèrtexs.

Per acabar, el GBGA pot ser vist des de dues perspectives ben diferents, que considerem interessants. D'una banda, és un dels pocs AG que poden competir, tant en temps com en qualitat, amb els millors mètodes heurístics dissenyats específicament per al problema. De l'altra, l'alt grau de dependència de les característiques específiques del problema per part de l'algorisme, sembla apuntar cap a les limitacions dels AG com a mètode general de resolució de problemes.

Capítol 5

La SPH i la DNH

En els capítols anteriors hem pogut constatar la importància decisiva que té l'algorisme d'un pas que s'utilitza com a base en el rendiment dels *mètodes competitius*, aquells algorismes orientats a obtenir bons resultats en la pràctica. La majoria d'aquests mètodes són d'execució múltiple i, per tant, generen un nombre relativament alt de solucions candidates, retornant al final la millor aproximació obtinguda. Cada solució candidata és el resultat d'un nou problema d'Steiner de característiques similars a les del problema inicial, per al qual s'obté una aproximació mitjançant l'algorisme d'un pas. Ara per ara, l'*heurística del camí més curt* [88] (SPH) (veure l'apartat 2.4.1) i l'*heurística del graf distància* (DNH) (veure l'apartat 2.4.2) són, amb diferència, els dos algorismes més àmpliament seleccionats com a algorisme de base, especialment en aquells mètodes competitius que estan donant millors resultats: la DNH és utilitzada a [30, 31, 90, 2], mentre que la SPH a [93, 72, 45, 67, 46]. Hi ha tres factors principals que, afegits a la seva simplicitat, afavoreixen l'elecció d'aquests algorismes i ajuden a decidir quin d'ells és més indicat: la complexitat computacional, la qualitat de les solucions que produeixen i l'estratègia en què es basen.

Complexitat

En general, sol considerar-se que $O(|V|^3)$ marca el llindar entre els algorismes que poden ser utilitzats en la pràctica, i els que només poden tenir un ús limitat per a instàncies petites. La complexitat, per tant, permet descartar la major part dels

algorismes d'aproximació[1] proposats aquests darrers anys, especialment aquells amb millor FRA [12, 82, 99, 54] (veure l'apartat 2.3.1). D'altres algorismes amb complexitat $O(|V|^3)$, superen aquest límit quan són executats de manera iterativa. Aquest és el cas, per exemple, de l'*heurística de distància mitjana* (ADH), proposada per Rayward-Smith i Clare [83] l'any 1983 i, també, de l'*heurística de contracció*, proposada per Plesník [78, 79] l'any 1981. Per tant, deixant apart algunes variants, la complexitat limita l'elecció de l'algorisme de base a la DNH i la SPH, a les quals cal afegir-hi l'algorisme de Zelikovsky [97, 98] (ZH) (veure l'apartat 2.4.3).

La DNH pot ser implementada en $O(|A| + |V| \log |V|)$ mitjançant el mètode proposat per Mehlhorn [73], essent l'algorisme d'aproximació per al PSG que té menor complexitat. D'aquesta manera, permet la construcció de mètodes d'execució múltiple amb complexitat $O(|V|(|A| + |V| \log |V|))$, quan el nombre d'execucions es limita a $O(|V|)$. Tal i com es pot veure, aquesta complexitat és la mateixa que la de calcular els camins de cost mínim entre cada parella de vèrtexs fent servir l'estructura de dades proposada per Fredman i Tarjan [36].

La SPH té complexitat en el pitjor cas $O(|S|(|A|+|V| \log |V|))$, però esdevé $O(|A|+|V| \log |V| + |S||V|)$ quan es disposa dels camins de cost mínim[2] (veure la pàgina 45). Tal i com hem vist en els dos capítols anteriors, aquest algorisme pot ser utilitzat per a construir mètodes d'execució múltiple $O(|V|(|A| + |V| \log |V| + |V|^2))$, complexitat que s'assoleix considerant que el nombre d'execucions és $O(|V|)$ i calculant un únic cop els camins entre cada parella de vèrtexs. A més a més, el $|V|^2$ final pot transformar-se en $|V||S|$ limitant el nombre de "falsos" terminals a $|S| - 2$ mitjançant una funció del tipus *eGrau2* (veure l'apartat 2.5.2).

El ZH s'ha convertit en una bona alternativa a la SPH des que Duin i Voß [28] n'han proposat una implementació amb complexitat $O(|S|(|A| + |V| \log |V|))$. A més, un cop calculats els camins de cost mínim i de manera semblant a la SPH, l'execució del ZH esdevé $O(|V| \log |V| + |S||V|)$. Per tant, considerant un nombre d'execucions $O(|V|)$ i fent servir una funció per a limitar el nombre de vèrtexs, es poden construir

[1] Tenint en compte les prestacions que ofereixen alguns dels algorismes d'aproximació, descartem d'entrada tots aquells algorismes que no ho siguin.

[2] El factor $|A| + |V| \log |V|$ pot suprimir-se si no es calcula l'arbre generador de cost mínim al final.

algorismes basats en el ZH de complexitat $O(|V|(|A|+|V|\log|V|+|V||S|))$, és a dir, la mateixa[3] que tenen els mètodes GBA i GBGA proposats en els capítols anteriors. De moment, però, l'únic mètode d'execució múltiple del qual tenim constància que està basat en aquest algorisme va ser proposat per Alexander i Robins [2] (veure l'apartat 2.5.2), i és anterior a aquesta millora de la complexitat.

Qualitat de les solucions

No resulta gens senzill escollir entre dos algorismes basant-se en la qualitat de les solucions que generen. Plesník [79] va comparar l'any 1992 fins a 7 algorismes diferents per al PSG, entre els quals hi havia la SPH i la DNH. Amb algunes excepcions, es proposaven per a cada parella d'algorismes exemples en què un obtenia una millor solució que l'altre, així com exemples en què succeïa el contrari. Entendrem, per tant, la qualitat d'un algorisme d'una forma relativa, en funció del percentatge esperat d'error en la solució. Tot i que aquest percentatge no sembla pas que pugui ser calculat en la pràctica, s'acostumen a utilitzar indicadors que, com ara la FRA i els resultats experimentals, permeten fer-ne una estimació.

D'una banda, la FRA proporciona una fita superior de l'error màxim comès, encara que aquesta fita sol estar molt allunyada dels resultats que produeix l'algorisme. Tot i això, es reconeix l'existència d'una certa correlació entre el valor d'aquestes fites i la qualitat dels resultats que obtenen els diferents algorismes. La FRA de la DNH era considerada, fins fa ben poc, lleugerament millor a la de la SPH [84, 73, 93, 57], la qual cosa pot haver influït en alguns casos en l'elecció d'aquest algorisme en detriment de la SPH [2]. Tot i no fer-ho constar de forma explícita, Duin i Voβ [28] acabaren amb aquest malentès l'any 1997, en demostrar que tots dos algorismes tenen la mateixa FRA.

D'altra banda, els resultats experimentals són també habituals a l'hora de comparar algorismes, i permeten afinar les previsions a base d'adaptar els grafs de prova a les característiques de les instàncies a resoldre. Val a dir que, en tots els treballs que coneixem en què la SPH i la DNH han estat comparats experimentalment, la SPH ha obtingut resultats clarament superiors als de la DNH [84, 93, 89, 28].

[3]No tenim en compte la millora de la complexitat de la SPH que proposem en la secció 5.3.

L'estratègia

La comparació d'algorismes heurístics en funció de l'estratègia només és possible quan els algorismes són molt semblants i estan basats en les mateixes estratègies. En aquest cas, es pot afirmar que un algorisme és millor que un altre respecte a una estratègia donada en el sentit que segueix més fidelment aquesta estratègia i, per tant, si l'estratègia és encertada, és d'esperar que assoleixi resultats millors.

En el llibre de Hwang, Richards i Winter [57] dedicat al problema d'Steiner, es classifiquen els mètodes heurístics per al PSG en funció de la seva estratègia, destacant tres grups que permeten encabir la majoria de mètodes: les heurístiques de camí, les heurístiques d'arbre i les heurístiques de vèrtexs. Mentre que la SPH és classificada com a una heurística de camí, la DNH és considerada una heurística d'arbre.

L'objectiu principal d'aquest capítol és analitzar la SPH i la DNH per tal de poder decidir quin algorisme és més indicat per a la construcció de mètodes competitius i, a partir d'aquesta anàlisi, intentar introduir millores en tots dos algorismes. Tot i que la SPH acostuma a produir millors resultats, i que la DNH pot ser implementada amb una complexitat sensiblement inferior, les similituds entre tots dos algorismes són evidents [93], i han de servir com a punt de partida per als nostres propòsits.

El capítol està organitzat de la manera següent: en la primera secció es comparà el funcionament de la SPH i la DNH des de tres perspectives diferents: com a heurístiques de camí, de vèrtexs i d'arbre. Tal i com veurem, tots dos algorismes poden ser classificats en cadascuna d'aquestes tres categories en funció de la versió que es consideri. D'aquesta manera, s'aprofundeix en el coneixement del algorismes i es posen en relleu algunes de les diferències que permetran introduir-hi millores.

En la secció 5.2 es demostra que la FRA de la SPH és igual a la de la DNH. Aquesta demostració va ser el punt de partida que ens va permetre considerar la SPH com a heurística d'arbre tal i com la presentem en l'apartat 5.1.2, origen de les variants dels algorismes que proposem en la secció 5.4. Tot i això, el resultat que es demostra no és nou, tenint en compte la demostració alternativa que proposaren Duin i Voβ [28] l'any 1997.

En la secció 5.3 es proposa una nova implementació de la SPH que en redueix

sensiblement la complexitat computacional [47]. La idea principal en què es basa aquesta implementació sorgeix de combinar la implementació proposada per Mehlhorn [73] per a reduir la complexitat de la DNH, amb la versió d'aquest algorisme com a heurística de camí [84, 93]. Traslladant aquesta idea a la SPH, la seva complexitat esdevé similar a la de la DNH per a la majoria d'instàncies. Destaquem d'aquesta secció la demostració que l'algorisme és una implementació vàlida de la SPH, en l'apartat 5.3.4, així com l'algorisme a mig camí entre la DNH i la SPH que es proposa en l'apartat 5.3.5.

En la secció 5.4 es proposen noves variants de la SPH que creiem poden resultar molt interessants [48], les quals es basen en la versió de la SPH com a heurística d'arbre. Hem anomenat a la variant que proposem en l'apartat 5.4.3 *Improved-SPH* (ISPH), ja que millora el rendiment de la SPH. De tota manera, la variant que considerem més interessant des d'un punt de vista d'estratègia és l'anomenada *Generalized-SPH* (GSPH), que proposem en l'apartat 5.4.4. Tots dos algorismes tenen una complexitat semblant a la de la SPH, i admeten una implementació similar a la que ha estat proposada per a aquest algorisme en la secció anterior.

Finalment, en la secció 5.5 es conclou el capítol, remarcant les implicacions que les millores proposades en aquest capítol tenen, des del nostre punt de vista, en la construcció de mètodes competitius per al PSG.

5.1 Les estratègies de la SPH i de la DNH

En el capítol dedicat als algorismes heurístics per al PSG del llibre de Hwang, Richards i Winter [57] sobre el problema d'Steiner, es descriuen i classifiquen fins a 25 algorismes diferents. En aquesta classificació, 22 dels algorismes pertanyen a algun d'aquests tres grups[4]: les heurístiques de camí, les heurístiques d'arbre i les heurístiques de vèrtexs. Els altres tres algorismes són difícils de classificar.

- Les heurístiques de camí construeixen un arbre d'Steiner a base d'anar connectant, mitjançant camins que solen ser de cost mínim, terminals que no formen

[4]Els tres grups han estat comentats breument, des d'un punt de vista una mica diferent, en l'apartat 4.3.1.

part de l'arbre amb vèrtexs que ja en formen part. La SPH és un clar exponent d'aquest grup, tal i com es pot veure en l'algorisme de l'apartat 2.4.1.

- Les heurístiques d'arbre construeixen un arbre inicial que connecta tots els terminals, generalment utilitzant alguna variant de l'arbre generador de cost mínim i, després, apliquen estratègies diverses per tal de millorar-ne el cost. La DNH és un dels exemples típics d'aquest tipus d'estratègia, tal i com queda reflectit en l'algorisme de l'apartat 2.4.2.

- Les heurístiques de vèrtexs intenten seleccionar no-terminals que, en funció d'algun criteri, puguin ser considerats bons candidats a formar part d'un arbre d'Steiner òptim. Un cop seleccionats, es construeix un arbre que connecti tant els terminals com aquests vèrtexs. El ZH [97] és considerat una heurística de vèrtexs.

En els tres apartats que venen a continuació, es presenten versions alternatives tant de la SPH com de la DNH que ressalten les diferències entre tots dos algorismes. El darrer apartat de la secció, compara tots dos mètodes en funció de les diferents estratègies que utilitzen.

5.1.1 DNH: heurística de camí

Tal i com van descriure Rayward-Smith i Clare [84], per a obtenir una versió de la DNH com a heurística de camí només cal introduir una petita modificació a l'algorisme de la SPH descrit en l'apartat 2.4.1: substituir V_1 per $V_1 \cap S$ en el *Pas 2*. Aquesta modificació està subratllada en l'algorisme de la DNH com a heurística de camí que presentem en aquest apartat.

En la versió clàssica de la DNH (veure l'apartat 2.4.2), després de construir un arbre $MSpT(D, S)$ inicial es substitueixen les seves arestes pels camins de cost mínim corresponents de G. En canvi, en la versió heurística de camí la substitució es realitza en el moment en què l'aresta passa a formar part de l'arbre i, per tant, no s'ajorna fins que aquest ha estat construït del tot. El fet que els vèrtexs no-terminals ja en l'arbre no siguin tinguts en compte fins al *Pas 5* confirma l'equivalència entre totes

dues versions. Aquesta és, a més a més, l'única diferència entre aquest algorisme i la SPH, en la qual els no-terminals que ja formen part de l'arbre són tractats com si fossin terminals.

Pas 1. Començar amb $G_1 = (V_1 = \{s_0\}, A_1 = \emptyset, c)$, $s_0 \in S$;
Pas 2. Seleccionar $s, t \in V$ i P_{st} de manera que
$d(s,t) = min\{d(v,v')\}\, \forall v \in S - V_1, \forall v' \in \underline{V_1 \cap S}$, i
P_{st} és un camí de distància $d(s,t)$ de s a t;
Pas 3. Afegir P_{st} a G_1;
Pas 4. Si V_1 no conté tots els terminals
Tornar al *Pas 2*; ($|S| - 1$ vegades)
Pas 5. Retornar $T := trim(MSpT(G, V_1))$;

Algorisme 9: La DNH com a heurística de camí

5.1.2 SPH: heurística d'arbre

Si bé en la versió clàssica de la DNH queda clara la construcció d'un arbre inicial que es va transformant, a base de substitucions d'arestes per camins, per tal de reduir-ne el cost, aquest arbre no està present de forma explícita en la SPH. Ara presentem la versió d'aquest algorisme com a heurística d'arbre, on es construeix inicialment un $MSpT(D, S)$ i es va reduint el seu cost tot combinant el mecanisme clàssic de substitució d'arestes de la DNH amb un procés de reestructuració de l'arbre que s'executa cada cop que s'hi incorpora un nou vèrtex d'Steiner.

Durant tot el procés iteratiu es poden distingir en l'arbre T_0 dues parts diferenciades: una part inamovible, corresponent a l'arbre parcial G_1 que es va construint en la versió clàssica de la SPH, i una altra susceptible de ser reestructurada en el *Pas 3*, que és la que connecta en D els terminals que no formen part de G_1. La funció *reestructurar*, que detallem a continuació, substitueix algunes de les arestes de D per tal que es tinguin en compte els camins que connecten terminals que no formen part de G_1 amb no-terminals que en formen part:

Reestructurar T_0: Sigui A_N el subconjunt d'arestes de D que connecten els vèrtexs intermedis de P_{st} amb els terminals que no formen part de V_1, i sigui G' l'arbre T_0 un cop se l'hi han afegit aquestes arestes. Reestructurar T_0 consisteix en transformar aquest arbre en un $MSpT(G', V_0)$.

Pas 1. Construir $T_0 = (V_0, A_0)$ tal que $T_0 := MSpT(D, S)$,
$V_1 := \{s_0\}, s_0 \in S$;
Pas 2. Seleccionar $(s,t) \in A_0$ de manera que
$c(s,t) = min\{c(v,v')\}\, \forall v \in V_0 - V_1, \forall v' \in V_1$, i
P_{st} és un camí de s a t amb cost $c(s,t)$ a G;
Pas 3. Substituir (s,t) per P_{st} a T_0,
$V_1 := V_1 \cup vertexs(P_{st})$,
Si P_{st} té vèrtexs intermedis **reestructurar** T_0;
Pas 4. Si V_1 no conté tots els terminals
Tornar al *Pas 2*; ($|S| - 1$ vegades)
Pas 5. Retornar $T := trim(MSpT(G, V_1))$;

Algorisme 10: La SPH com a heurística d'arbre

Per a obtenir un algorisme equivalent a la DNH a partir de la modificació de l'algorisme anterior, només cal eliminar la línia referent a la funció reestructurar, en el *Pas 3.* En aquest cas, es pot veure que els passos 2, 3 i 4 equivalen a una simple substitució d'arestes per camins, sense que l'ordre en què es porta a terme tingui cap incidència en el resultat final. De tota manera, considerarem una altra versió de la DNH com a heurística d'arbre igual a la de l'algorisme 10, en què s'utilitza el procés de reestructuració de l'arbre següent:

Reestructurar T_0: Substituir les arestes de D que comparteixen un tram del camí associat amb G_1 per l'aresta corresponent al subcamí que no forma part de G_1.

L'únic efecte que té aquest procés és el d'anticipar la reducció del cost de l'arbre que es produiria més endavant i, d'aquesta manera, acostar-se més al funcionament de la mateixa versió de la SPH.

Tot i que la funció reestructurar dóna lloc a un mecanisme alternatiu per a seleccionar el camí de cost mínim en la SPH, no sembla tenir massa interès el fet de mantenir l'arbre T_0 de manera explícita en l'execució de l'algorisme. En qualsevol cas, aquest arbre podria ser utilitzat per tal d'avaluar el guany —diferència de cost respecte l'arbre anterior— que produeix la substitució d'una aresta per un camí, de manera semblant a la que s'utilitza en el ZH [97], o a la proposada per Duin i Voß [28].

5.1.3 SPH i DNH: heurístiques de vèrtexs

Afirmar que la SPH i la DNH poden ser considerats com a heurístiques de vèrtexs és una obvietat si tenim en compte que en el darrer pas d'aquests dos algorismes només s'utilitza el subconjunt de vèrtexs del graf que s'ha anat construint. Per aquest motiu, la versió de la SPH com a heurística de vèrtexs només ha de prescindir de la part en què es construeix l'arbre, i dedicar-se a seleccionar els vèrtexs que van sent incorporats a través dels camins. L'algorisme següent mostra la versió de la SPH com a heurística de vèrtexs. Per a obtenir una versió per a la DNH només cal substituir $v' \in V_1$ per $v' \in V_1 \cap S$ en el *Pas 3*, tal i com s'ha fet en l'apartat 5.1.1.

Pas 1. $V_1 := \{s_0\}$, $s_0 \in S$;
Pas 2. Seleccionar $s, t \in V$ i P_{st} de manera que
$d(s,t) = min\{d(v,v')\}\, \forall v \in S - V_1, \forall v' \in V_1$, i
P_{st} és un camí de distància $d(s,t)$ de s a t;
Pas 3. $V_1 := V_1 \cup vertexs(P_{st})$;
Pas 4. Si V_1 no conté tots els terminals
Tornar al *Pas 2*; $(|S| - 1$ vegades)
Pas 5. Retornar $T := trim(MSpT(G, V_1))$;

Algorisme 11: SPH: heurística de vèrtexs

Creiem interessant remarcar la manera indirecta en què la SPH i la DNH seleccionen els no-terminals: exclusivament com a vèrtexs intermedis de camins de cost mínim i, per tant, sense tenir en compte aquests vèrtexs en cap moment. Aquest fet posa en relleu la importància de la construcció i la selecció dels camins, tal i com ha

estat destacat en els capítols anteriors (veure l'apartat 3.3.4). Tenint en compte que la no incorporació de vèrtexs intermedis equival a la no millora respecte el cost de l'arbre inicial[5], creiem que seria interessant estudiar i definir criteris heurístics per a seleccionar els camins que acostumessin a donar bons resultats. Per exemple, en cas d'igualtat es podria seleccionar el camí amb més vèrtexs intermedis, el que produeix un major guany, etc.

5.1.4 Comparant la SPH amb la DNH

A partir de les versions de la SPH i la DNH que acabem de presentar, podem concloure que tots dos algorismes combinen, d'una manera implícita o explícita, tres estratègies diferents: heurística d'arbre, de camí i de vèrtexs. En el pitjor dels casos, l'arbre aproximat que serà retornat per aquests algorismes tindrà cost igual al d'un $MSpT(D,S)$, que denotem per T_D, la qual cosa queda garantida pel fet que el cost del camí no serà mai superior al de l'aresta de D que substitueix (veure la secció següent). Per tant, l'objectiu final de tots dos algorismes és aconseguir allunyar-se tant com sigui possible del cost d'aquest arbre. L'elecció dels camins, la inclusió de nous vèrtexs i la reestructuració de l'arbre, són les eines de què disposen aquests mètodes per tal d'aconseguir aquest objectiu.

El primer aspecte que resulta rellevant és l'aparent independència del criteri base que guia l'algorisme respecte al cost que, en darrer terme, tindrà la solució final. Per exemple, per molt curt que sigui el camí que s'incorpora a l'arbre, només si té vertexs intermedis pot contribuir directament a millorar el cost en relació al de T_D. Per tant, la qualitat del resultat final depèn, en darrer terme, de la presència de vèrtexs d'Steiner de grau superior a dos i del nivell en què aquests vèrtexs contribueixen a reduir el cost de l'arbre en relació al de T_D.

Com a heurístiques de camí, l'única diferència entre la DNH i la SPH és que aquesta última permet seleccionar camins que connecten un terminal amb un no-terminal que formi part de l'arbre en construcció. Considerant una execució per a una mateixa instància, el comportament de tots dos algorismes és idèntic en aquelles

[5] Podem considerar els algorismes en la seva versió d'heurística d'arbre.

iteracions en què la SPH selecciona un camí que uneix dos terminals. En canvi, la SPH es comporta de forma diferent en els casos en què selecciona un camí que connecta un terminal amb un no-terminal. Aquest no-terminal passa a tenir temporalment grau més gran que dos, i té moltes opcions de conservar aquest grau en l'arbre que retorna l'algorisme. Considerant que les expectatives d'incloure vèrtexs intermedis en un camí són independents dels vèrtexs finals que connecta, sembla clar que la SPH tendeix a promoure, en un grau més alt que la DNH, que els no-terminals que incorpora passin a ser vèrtexs d'Steiner de grau superior a dos i, d'aquesta manera, contribueixin a reduir el cost de l'arbre resultant.

Com a heurístiques d'arbre, els costos dels arbres que s'obtenen en cada iteració en tots dos algorismes són no creixents i, per tant, actuen com a fita superior del cost de la solució final. Aquesta fita superior, però, pot resultar poc ajustada, tenint en compte les dues parts diferenciades que formen aquests arbres. D'una banda, la part corresponent a l'arbre G_1 en construcció, que és un subgraf de G, no té per què connectar els seus vèrtexs de manera òptima i, per això, cal el *pas 5* (veure l'apartat 5.4.2). De l'altra, els camins associats a les arestes de la part corresponent al subgraf de D poden tenir trams compartits que, per tant, són comptabilitzats més d'una vegada a l'hora d'avaluar el cost de l'arbre. Aquests trams són una de les claus per entendre les diferències que es produeixen entre el cost de la solució final i el de T_D.

Tenint en compte la versió de la DNH que ha estat proposada en l'apartat 5.1.2, l'única diferència entre tots dos algorismes es troba en la utilització d'un procés de reestructuració de l'arbre diferent, procés que consisteix en la substitució d'algunes de les arestes de la part en D per unes altres. Les noves arestes que passen a formar part de l'arbre, a més de disminuir-ne el cost, connecten sempre un terminal amb un no-terminal de G_1 i, per tant, contribueixen a augmentar el grau d'aquests vèrtexs. Comparant els dos processos de reestructuració, el de la DNH es basa exclusivament en la supressió de trams que resulten coincidents en els camins associats a les arestes, coincidència que es dona per atzar, sense que hi intervingui cap element estratègic[6].

[6]Aquest fet sembla apuntar cap a la conveniència d'escollir sempre el mateix camí entre dos vèrtexs.

En canvi, en el de la SPH, a més de contemplar la supressió d'aquests trams, es produeixen altres intercanvis d'arestes propiciats per l'estratègia de selecció de camins. Considerant que en aquests intercanvis el camí associat a la nova aresta no té menys possibilitats de tenir un tram compartit que el camí associat a l'aresta que substitueix, podem afirmar que la SPH integra un mecanisme addicional que pot revertir en un major nombre de vèrtexs d'Steiner de grau superior a dos en la solució.

Com a heurístiques de vèrtexs, només veiem dues possibles explicacions que podrien justificar un comportament diferenciat entre la SPH i la DNH: que un dels algorismes tendeixi a incorporar un nombre superior de no-terminals, o que el nombre de no-terminals que produeixen guany que inclou un dels algorismes tendeixi a ser més gran. Tenint en compte que el cost de la solució mai superarà el cost de T_D, un major nombre final de no-terminals pot associar-se a un major nombre de vèrtexs que produeixin guany respecte a aquest arbre i, per tant, a una solució que estigui generalment a més distància de T_D.

D'una banda, tenint en compte que tots dos algorismes seleccionen $|S|-1$ camins diferents, la probabilitat de que un d'ells tendeixi a incorporar més no-terminals sembla estar relacionada amb el fet que l'arbre resultant de la connexió de tots aquests camins, un cop eliminats els trams compartits, sol tenir un cost major[7]. En aquest cas, caldria veure si un nombre superior de vèrtexs és capaç de compensar el cost més gran d'aquest arbre que, tal i com hem vist, representa una fita superior del cost de l'aproximació final.

De l'altra, la probabilitat de que un dels dos algorismes tendeixi a introduir més vèrtexs de guany, vèrtexs d'Steiner de grau superior a dos, també depèn dels vèrtexs intermedis que formen part dels $|S|-1$ camins que són seleccionats. En la DNH aquests vèrtexs intermedis ho són sempre de camins, o trams de camins, entre dos terminals. En canvi, en la SPH també poden ser incorporats els vèrtexs intermedis d'alguns camins que van des d'un vèrtex terminal a un no-terminal, essent força excepcionals els casos de vèrtexs que poden formar part d'una solució de la DNH i

[7]Aquesta afirmació només té sentit si hi ha una correlació entre els costos de les arestes i la distribució de vèrtexs intermedis, cosa que pot succeir, per exemple, si els costos corresponen a distàncies entre vèrtexs.

que no poden formar part de cap solució vàlida de la SPH. Per tant, el subconjunt de vèrtexs que poden formar part d'una solució de la SPH sol ser més gran que el de la DNH, fet que suposaria una millora si aquests vèrtexs addicionals tinguessin una major tendència a produir guany. En qualsevol cas, cada cop que s'incorpora un d'aquests vèrtexs, lligat a un camí entre un terminal i un no-terminal, s'està incrementant el grau d'aquest últim. Tot indica, doncs, que els vèrtexs addicionals que poden ser accedits per la SPH contribueixen, de manera indirecta, a incrementar el grau dels vèrtexs d'Steiner.

Considerant les tres estratègies esmentades, podem concloure que en totes elles s'han trobat arguments a favor de la superioritat de la SPH enfront de la DNH. A més, hem pogut observar alguns dels aspectes que moltes vegades no es tenen en compte i que tenen una influència molt gran en la qualitat final dels resultats. Per exemple, donada una instància $G(V, A \cup A_T)$, on A_T és el subconjunt d'arestes de T_D, ni la SPH ni la DNH aconseguiran cap millora respecte el cost de T_D si en el procés de selecció dels camins es dóna prioritat al camí que té menys arestes. Aquests són alguns dels criteris que, en la nostra opinió, podrien contribuir a millorar el resultats:

- Intentar construir els camins de manera que tinguin vèrtexs intermedis.
- Intentar que la solució final tingui el màxim nombre possible de vèrtexs, per exemple, donant prioritat en la selecció als camins que tenen vèrtexs intermedis respecte als formats per una sola aresta.
- Afavorir que els vèrtexs d'Steiner passin a tenir grau superior a dos, per exemple, promovent el càlcul de camins amb trams en comú, i donant prioritat en la selecció als camins que connecten un terminal amb un no-terminal, en el cas de la SPH.

Finalment, la versió de tots dos algorismes com a heurístiques d'arbre de l'apartat 5.1.2 possibilita avaluar localment el guany que produeix un vèrtex o un camí respecte la solució actual, la qual cosa pot ser utilitzada per a dissenyar algorismes basats en una estratègia *greedy*.

5.2 La FRA de la SPH

L'any 1980, Takahashi i Matsuyama [88] van proposar la versió originària de la SPH, i van demostrar que la FRA d'aquest algorisme no estava per sobre de $2(1 - 1/|S|)$. Era la primera ocasió en què es demostrava una FRA per a un algorisme per al PSG i, per tant, es confirmava l'existència d'algorismes d'aproximació per a aquest problema. Un any més tard, Kou, Markovsky i Berman [68] van demostrar que la DNH tenia una FRA no superior a $2(1 - 1/l)$, on l és el nombre mínim de fulles en un arbre d'Steiner òptim. Com que $l \leq |S|$, es va considerar que la FRA de la DNH era lleugerament millor que la de la SPH [84, 73, 93, 57].

En aquesta secció proposem una demostració de que la FRA de la SPH és no superior a $2(1 - 1/l)$ i, per tant, igual a la de la DNH. En realitat, l'únic que cal demostrar és la proposició 2 i, a continuació, fer servir la proposició 3, utilitzada en la major part dels algorismes que tenen aquesta FRA [57]. El valor de la demostració rau en què representa un punt de partida per a arribar a la versió de la SPH com a heurística d'arbre de l'apartat 5.1.2, tal i com es pot veure a partir de la seqüència d'arbres que es genera en la primera demostració. La proposició 2 ja va ser demostrada de forma diferent per Duin i Voβ [28] l'any 1997.

Siguin, T_{SPH} l'arbre resultant de l'execució de la SPH per al graf G i el conjunt S de terminals, T_D un arbre generador de cost mínim per a S en el graf distància D de G i T_{MStT} un arbre de Steiner òptim per a S en G. Tots els passos de la SPH que s'esmenten en la resta d'aquest capítol fan referència a l'algorisme de l'apartat 2.4.1.

Proposició 2 $|T_{SPH}| \leq |T_D|$.

Demostració:

Denotem per T_I^i l'arbre inicial de la i-èsima execució del *Pas 2* de la SPH, $1 \leq i \leq |S| - 1$, i que està representat per G_1 en l'algorisme. P_i fa referència al camí P_{st} de cost mínim que és afegit a T_I^i en el *Pas 3*. Anem a construir T_{SPH} substituint cadascuna de les arestes de T_D per un P_i amb cost no superior, $1 \leq i \leq |S| - 1$.

Sigui C_1 el circuit que es forma en afegir P_1 a T_D. Construïm l'arbre T_D^1 substituint P_1 per l'aresta a_1 de C_1 que uneix un vèrtex de T_I^1 amb un vèrtex que no forma part

d'aquest arbre. Si l'aresta a_1 no existís, els terminals de T_I^1 no tindrien cap mena de connexió a T_D amb els terminals que no estan en aquest arbre. A més a més, el cost de P_1 no pot ser superior al de cap aresta de D que connecti un terminal de l'arbre T_I^1 amb un terminal que no està en aquest arbre. Conseqüentment, $|T_D| \geq |T_D^1|$.

Sigui C_2 el circuit que es forma en afegir P_2 a T_D^1. Construïm l'arbre T_D^2 substituint P_2 per l'aresta a_2 de C_2 que uneix un vèrtex de T_I^2 amb un vèrtex que no forma part d'aquest arbre. Com a dalt, a_2 existeix i $|T_D^1| \geq |T_D^2|$. A més a més, com que T_I^2 és un subgraf de T_D^1, a_2 és l'única aresta de C_2 que connecta un vèrtex de T_I^2 amb un vèrtex que no forma part d'aquest arbre.

El procés de construcció es repeteix fins arribar a $T_D^{|S|-1} = T_{SPH}$. La relació següent mostra una nova versió de la SPH com a heurística d'arbre:

$$|T_D| \geq |T_D^1| \geq \cdots |T_D^{|S|-2}| \geq |T_{SPH}|$$

∎

La demostració de la proposició següent és anàloga a la que s'utilitza a [57] per demostrar la FRA de la DNH.

Proposició 3 $|T_D| \leq 2(1-1/l)|T_{MStT}|$, *on l és el nombre de fulles de T_{MStT}.*

Demostració: Sigui W el circuit que es forma quan es ressegueix en el sentit de les agulles del rellotge la part externa de les arestes de l'arbre T_{MStT}, fins a tornar al punt inicial. Es pot veure W, el qual passarà exactament dues vegades per cadascuna de les arestes de T_{MStT}, com si estigués format per l camins disjunts d'arestes, cadascun d'ells unint dues fulles de l'arbre. Sigui P el camí que té cost més gran de tots els que uneixen dues fulles. En aquest cas, $|P| \geq 2|T_{MSpT}|/l$, donant-se la igualtat si tots aquests camins tinguessin el mateix cost. Construïm W' eliminant de W les arestes de P i, per tant, $|W'| = |W| - |P| \leq 2(1-1/l)|T_{MStT}|$. Es pot veure W' com si fos un camí que connecta tots els terminals, així, com que cada camí entre dos terminals pot ser substituït per una aresta de D, $|W'| \geq |T_D|$. ∎

Teorema 4 $|T_{SPH}| \leq 2(1-1/l)|T_{MStT}|$, *on l és el nombre mínim de fulles que hi ha en un MStT per N en G.*

Demostració: Es dedueix directament de les dues proposicions anteriors. ∎

5.3 Reduint la complexitat de la SPH

La complexitat reconeguda de la SPH és $O(|S|(|A| + |V|\log|V|))$ i està dominada pel càlcul dels camins de cost mínim entre els terminals i la resta de vèrtexs del graf. Aquest càlcul pot portar-se a terme executant $|S|$ vegades una implementació $O(|A|+|V|\log|V|)$ de l'algorisme de Dijkstra [23], que requereix l'estructura de dades proposada per Fredman i Tarjan [36].

Tot i la seva similitud amb la SPH, Mehlhorn [73] va proposar l'any 1988 una implementació de la DNH amb complexitat $O(|A| + |V|\log|V|)$, amb la qual cosa esdevenia l'algorisme d'aproximació per al PSG amb menor complexitat. En la implementació proposada per Mehlhorn s'evita calcular de forma exacta molts dels camins entre parelles de terminals que, en principi, eren considerats necessaris en les implementacions anteriors.

En aquesta secció proposem una implementació de la SPH que en redueix la complexitat fins a $O(min\{|S|, \frac{|S'|}{|S|}\}(|A|+|V|\log|V|))$, on $|S'| \leq |V|-|S|$ és el nombre de vèrtexs d'Steiner en l'arbre que retorna l'algorisme. Tal i com es pot veure, la complexitat de l'algorisme esdevé igual a la de la DNH per a la majoria d'instàncies, com ara aquelles en què $2k|S| \geq |V|$, o $k|S| \geq |S'|$, per a una constant $k \geq 1$ molt petita. El cas més desfavorable es dóna quan $|S| \approx \sqrt{|V|}$ i $|S'| \approx |V|-|S|$, en el qual la complexitat esdevé $O(\sqrt{|V|}(|A|+|V|\log|V|))$ i, per tant, en el pitjor dels casos la complexitat ha passat de ser $O(|V|^3)$ a ser $O(|V|^{2+1/2})$.

La idea principal en què es basa la implementació que proposem combina la manera en què Mehlhorn redueix la complexitat de calcular els camins necessaris, amb la perspectiva de la DNH com a heurística de camí. D'aquesta manera, els camins es poden anar obtenint sobre la marxa, la qual cosa permet afegir les modificacions pertinents a l'algorisme de Mehlhorn per tal d'introduir-hi el tractament diferenciat que dóna la SPH als nous vèrtexs que s'incorporen a l'arbre, sense deixar d'evitar que es calculin una gran quantitat de camins innecessaris.

El contingut de la secció ha estat presentat a [47]. Un cop introduïdes les definicions de l'apartat que ve a continuació, en l'apartat 5.3.2 es presenta l'algorisme corresponent a la nova implementació. En l'apartat 5.3.3 es descriuen els detalls d'aquest algorisme i s'analitza la seva complexitat. En l'apartat 5.3.4 es demostra que aquest algorisme és una implementació vàlida de la SPH. Finalment, en l'apartat 5.3.5 es proposa un nou algorisme, que hem anomenat *quasi-SPH* (qSPH) que té exactament la mateixa complexitat de la DNH i un funcionament molt proper al de la SPH.

5.3.1 Definicions prèvies

Les dues definicions que donem a continuació, la de la partició $\mathcal{P}$ de V en $|S|$ subconjunts, i la de la distància d' han estat extretes de [73]:

$$\mathcal{P} = \{N(s)|s \in S\} \text{ tal que } N(s) = \{v \in V|d(s,v) \leq d(s',v) \forall s' \in S\}.$$

És a dir, $N(s)$ conté els vèrtexs que estan més propers a $s \in S$ que a qualsevol altre vèrtex de S. D'una forma gràfica, la partició $\mathcal{P}$ pot ser representada com un conjunt de boles o esferes, cadascuna d'elles centrada en un vèrtex terminal, de manera que cada vèrtex forma part d'una única bola. En aquest sentit, ens referirem a $N(s)$ com a la *bola de s*.

$$d'(s,t) = min\{d(s,u) + d(u,v) + d(v,t); (u,v) \in A, u \in N(s), v \in N(t)\}.$$

Per tant, $d'(s,t)$ és el cost del camí de cost mínim que va de s a t utilitzant únicament vèrtexs de les boles de s i de t.

5.3.2 L'algorisme

En la implementació de Mehlhorn [73] de la DNH, el $MSpT(D_S, S)$ que es construeix inicialment en versió clàssica d'aquest algorisme (veure l'apartat 2.4.2) és substituït per un $MSpT(D'_S, S)$, on D_S és el subgraf de D induït per S, i D'_S és l'equivalent d'aquest subgraf quan s'utilitza la distància d' en comptes de la distància d. Tenint en compte que aquesta és l'única diferència entre tots dos algorismes, es pot observar

que per a traslladar la implementació de l'algorisme de Mehlhorn a les altres versions de la DNH només cal substituir la distància d per la distància d'.

La implementació de la SPH que presentem a continuació utilitza la distància d' en comptes de la distància d. A més, com que d' només està definida entre terminals, actualitza la partició $\mathcal{P}$ cada cop que un no-terminal passa a formar part de l'arbre en construcció, —els únics casos en què la distància entre un no-terminal i un terminal és necessària en l'algorisme. Aquesta actualització reflecteix que en la SPH els no-terminals són tractats com si fossin terminals tant bon punt passen a formar part de l'arbre, única diferència que hi ha entre aquest algorisme i la DNH.

Pas 0. **Construir** la partició $\mathcal{P}$; (*)
Pas 1. Començar amb $G_1 = (V_1 = \{s_0\}, A_1 = \emptyset, c)$, $s_0 \in S$;
Pas 2. **Seleccionar** $s, t \in V$ i P_{st} de manera que
$d'(s,t) = min\{d'(v,v')\}\, \forall v \in S - V_1, \forall v' \in V_1$, i
P_{st} és un camí de distància $d'(s,t)$ de s a t;
Pas 3. Afegir P_{st} a G_1;
Pas 4. Si V_1 no conté tots els terminals:
Si P_{st} té vèrtexs intermedis: **Actualitzar** $\mathcal{P}$. (**)
Tornar al *Pas 2*; ($|S| - 1$ vegades)
Pas 5. Retornar $T := trim(MSpT(G, V_1))$;

Algorisme 12: Nova implementació de la SPH

Novament, l'algorisme és molt semblant al de Prim [81] per a trobar l'arbre generador de cost mínim en un graf. De fet, es va construint un subgraf arbre G_1 de G al qual s'hi afegeix en cada iteració un camí de cost mínim en distància d' que uneix un terminal s que no forma part de G_1 amb un vèrtex t que ja en forma part.

Les marques ((*) i (**)) que hem posat a l'algorisme serveixen per identificar algunes de les versions, tant de la SPH com de la DNH, que hem anat veient al llarg d'aquesta memòria. Per exemple, la versió clàssica de la SPH s'obté fent servir la distància d en comptes de la distància d' en el *Pas 2*, i suprimint les dues sentències (*) i (**) que fan referència a la partició $\mathcal{P}$. Si, a més, substituïm $v' \in V_1$, que ha estat subratllat en el *Pas 2*, per $v' \in V_1 \cap S$, l'algorisme esdevé igual a la versió de la DNH com a heurística de camí de l'apartat 5.1.1. Finalment, una versió heurística de camí equivalent a la

que va proposar Mehlhorn [73] es pot obtenir eliminant la sentència marcada amb $^{(**)}$, i substituint $v' \in V_1$ per $v' \in V_1 \cap S$ com abans. Aquesta versió s'aparta de l'estàndard seguint la proposta de Rayward-Smith i Clare [84], també utilitzada posteriorment per Winter i Smith [93].

5.3.3 Implementació i complexitat

A l'hora d'implementar l'algorisme anterior, hi ha tres processos claus a tenir en compte: la manera en què la partició $\mathcal{P}$ és construeix en el *Pas 0*, la forma en què aquesta partició s'actualitza cada cop que nous vèrtexs no-terminals són incorporats a G_1 en el *Pas 4* i, finalment, la manera en què el camí P_{st} és seleccionat durant el procés iteratiu en el *Pas 2*.

Construir $\mathcal{P}$

El mecanisme per a construir la partició $\mathcal{P}$ en el *Pas 0* va ser proposat per Mehlhorn [73] i només necessita una execució de l'algorisme de Dijkstra [23] per a trobar el camí de cost mínim entre dos vèrtexs en un graf:

- Construir el graf $G' = (V \cup \{v_0\}, A \cup A_0, c')$, en el qual $A_0 = \{(v_0, v) | v \in S\}$ i $c'(a) = c(a)$ quan $a \in A$ i $c'(a) = 0$ en cas contrari. És a dir, G' s'obté afegint un nou vèrtex v_0 a G i connectant aquest vèrtex a tots els terminals amb una aresta de cost 0.

- Executar l'algorisme de Dijkstra començant per v_0 per tal d'obtenir un *arbre de camins mínims* ($T_{\mathcal{P}}$), el qual és un graf dirigit que conté un camí de cost mínim des de v_0 fins a cadascun dels altres vèrtexs.

- El subconjunt $N(s)$ de $\mathcal{P}$ es correspon al subconjunt de vèrtexs del subarbre de $T_{\mathcal{P}}$ que té arrel s, $\forall s \in S$. Notem que, per la manera en què $T_{\mathcal{P}}$ ha estat construït, si s'elimina v_0 de $T_{\mathcal{P}}$ s'obtenen $|S|$ subarbres cadascun dels quals conté un únic terminal s, que és l'arrel, i un camí de cost mínim que va des de s a la resta de no-terminals en el subarbre.

La complexitat d'aquest procés és $O(|A| + |V| \log |V|)$ si es fa servir la implementació de l'algorisme de Dijkstra que utilitza l'estructura de dades de Fredman i Tarjan [36]. L'arbre $T_{\mathcal{P}}$ és utilitzat més endavant en el procés d'actualització de $\mathcal{P}$.

Actualitzar $\mathcal{P}$

El mecanisme d'actualització de la partició $\mathcal{P}$ modifica aquest subconjunt i l'arbre $T_{\mathcal{P}}$, i també està basat en una execució de l'algorisme de Dijkstra que, en aquest cas, està restringida a un subconjunt de vèrtexs.

Sigui I_S el subconjunt de vèrtexs intermedis del camí més curt P_{st} que és seleccionat en la i-èsima iteració del *Pas 2*. El nostre objectiu és construir $N(v), \forall v \in I_S$ de manera que cadascun d'aquests subconjunts pot incloure vèrtexs que formaven part d'altres boles de $\mathcal{P}$:

- Modificar $T_{\mathcal{P}}$ substituint l'aresta (v, v') per una nova aresta (v_0, v') amb cost 0, $\forall v' \in I_S$.

- Executar l'algorisme de Dijkstra per a reestructurar $T_{\mathcal{P}}$ de manera que només es tinguin en consideració aquells vèrtexs que poden ser accedits en l'arbre $T_{\mathcal{P}}$ des dels vèrtexs que formen part de I_S. Per tant, només aquests vèrtexs podran ser explorats.

En cada iteració de l'algorisme de Dijkstra es selecciona per a ser explorat un dels vèrtexs no explorats que estan a menor distància de v_0. Durant tot el procés es manté un conjunt que conté els camins de cost mínim que van des de v_0 a cadascun dels vèrtexs pendents d'explorar, passant únicament per vèrtexs ja explorats. Explorar un vèrtex v consisteix en actualitzar els camins d'aquest conjunt substituint, per a tot vèrtex v' pendent d'explorar, el camí de v_0 a v' pel camí alternatiu de v_0 a v' que passa per v, sempre i quan aquest tingui cost menor. Per tant, explorar un vèrtex requereix una comparació per veí. L'arbre de camins de cost mínim conté els camins que van de v_0 al vèrtex que és explorat en cada iteració.

La complexitat conjunta de tots els processos d'actualitzar $\mathcal{P}$ es pot avaluar en funció de dues variables diferents, el nombre total de camins que contenen vèrtexs

intermedis n_{cam}, $0 \leq n_{cam} \leq |S| - 1$ i, també, del nombre total de vèrtexs explorats n_{exp}. En el primer cas, n_{cam} es correspon al nombre d'execucions de l'algorisme de Dijkstra en la versió que restringeix el nombre de vèrtexs. Sense tenir en compte aquesta restricció en el nombre de vèrtexs, la complexitat en el pitjor dels casos és $O(n_{cam}(|N| + |V| \log |V|))$, i dóna una fita superior de la complexitat, generalment més ajustada, que avaluem a continuació fent servir n_{exp}.

Per a què un vèrtex sigui explorat és necessari que pugui ser accedit a T_P des del conjunt de vèrtexs intermedis i, per tant, només els vèrtexs que canvien de bola en la partició poden ser-ho. Conseqüentment, el nombre de vèrtexs que serà explorat en una iteració és igual al nombre de vèrtexs que passaran a formar part de les noves boles que es creen en aquella iteració i, per tant, que estan centrades pels vèrtexs intermedis del camí seleccionat. Tot i que el nombre de vèrtexs que conté cada bola pot ser molt variable, considerarem que les boles de nova creació tenen de mitjana un nombre d'elements $\overline{n_{ele}}$ similar al de la mitjana de totes les boles en la partició inicial, és a dir, $\overline{n_{ele}} \approx |V|/|S|$. Aquesta restricció no és gens forta tenint en compte que el nombre de vèrtexs esperat d'aquestes boles es troba sempre per sota d'aquest valor, i va decreixent a mesura que augmenta el nombre de boles. D'aquesta manera, si S' és el conjunt de vèrtexs no-terminals a l'inici del *Pas 5*, $n_{exp} \approx |S'|\overline{n_{ele}} \approx |S'||V|/|S|$.

Explorar un vèrtex necessita una comparació per a cadascun dels seus veïns. Considerarem que la mitjana de veïns $\overline{n_{vei}}$ dels vèrtexs que són explorats és similar a la mitjana de veïns per vèrtex, és a dir, $\overline{n_{vei}} \approx 2|A|/|V|$. Tal i com es pot veure, aquesta restricció només fa referència a la mitjana de veïns i, per tant, no representa cap mena de limitació en el grau dels vèrtexs.

A la complexitat d'explorar un vèrtex cal afegir-hi la que necessita l'algorisme de Dijkstra per a seleccionar aquest vèrtex per a què sigui explorat: si n_i és el nombre de vèrtexs que són seleccionats en la i-èsima iteració, la complexitat de seleccionar aquests vèrtexs és $O(n_i \log n_i)$ i, per tant, no superior a $O(n_i \log |V|)$. Així, considerant que $n_i \leq |V|$ obtenim una complexitat $O(n_{cam}|V| \log |V|)$. En canvi, considerant que $n_{exp} = \sum_i n_i$, la complexitat total de seleccionar els vèrtexs és $O(n_{exp} \log |V|) \approx O(\frac{|S'|}{|S|}|V| \log |V|)$.

Podem concloure, per tant, que la complexitat final d'executar els processos d'actualització de $\mathcal{P}$ és

$$O(min\{n_{cam}, \frac{|S'|}{|S|}\}(|A| + |V| \log |V|)).$$

Seleccionar P_{st}

El mecanisme de selecció de P_{st} està basat en el que utilitza l'algorisme de Prim [81] per a seleccionar l'aresta de cost mínim mentre construeix l'arbre generador de cost mínim d'un graf. El mecanisme que proposem, però, processa boles de la partició $\mathcal{P}$ en comptes de vèrtexs i selecciona camins de cost mínim en distància d' en comptes d'arestes. Al llarg de tot el procés iteratiu, l'algorisme manté un subconjunt de camins candidats C_P que té $|S - V_1|$ camins de cost mínim, cadascun d'ells connectant un dels terminals que no formen part de V_1 amb un dels vèrtexs de V_1 que es troben a menor distància d' d'aquest terminal. Per a mantenir C_P s'utilitza el mecanisme que descrivim tot seguit:

Després de seleccionar el camí de cost mínim $P_{st} = (s = v_1, ..., v_k, t)$ que forma part de C_P, s'actualitza la partició $\mathcal{P}$ i s'elimina P_{st} de C_P. A continuació, el procés següent és executat per a cada v_i, $1 \leq i \leq k$:

$$\forall P_{s't'} \in C_P \text{ if } d'(s', t') > d'(s', v_i) \text{ aleshores } C_P := C_P - \{P_{s't'}\} \cup \{P_{s'v_i}\}.$$

És a dir, si de la incorporació dels nous no-terminals a l'arbre en construcció se'n deriven alguns camins més curts en d' que els que hi ha a C_P —camins que connecten un terminal que no forma part de l'arbre amb un vèrtex de l'arbre—, aquests camins substitueixen els anteriors.

Per a trobar les distàncies $d'(s', v_i)$ es requereix considerar els veïns terminals que no formen part de l'arbre de cadascun dels vèrtexs de $N(v_i)$, $1 \leq i \leq k$, la qual cosa és una operació similar a la d'explorar els vèrtexs d'aquesta bola de manera semblant a la que es porta a terme en el procés d'actualització de $\mathcal{P}$.

El nombre total d'exploracions per a mantenir C_P és $|V| + n'_{exp}$, on $|V|$ considera una exploració de cada vèrtex, i n'_{exp} el nombre de reexploracions. Un vèrtex es torna a explorar quan, havent estat explorat, és canviat de bola en el procés d'actualització de la partició $\mathcal{P}$, cosa que garanteix que $n'_{exp} \leq n_{exp}$. Per tant, la primera exploració

de tots els vèrtexs és $O(|A|)$, i les reexploracions són $O(n'_{exp}\overline{n_{vei}})$, on $\overline{n_{vei}}$ està definit com en el subapartat anterior. Afegint $O(|S|\log|S|)$ operacions necessàries per a seleccionar el camí de cost mínim en el conjunt C_P, la complexitat final de seleccionar P_{st} és

$$O(|A| + |S|\log|S| + n'_{exp}\overline{n_{vei}}).$$

Anàlisi de la complexitat

La complexitat de l'algorisme ve dominada per dos factors: l'execució de l'algorisme de Dijkstra en el *Pas 0*, i el procés d'actualització de $\mathcal{P}$ en el *Pas 4*. Tal i com es pot veure, la complexitat de seleccionar P_{st} en el *Pas 2* és inferior a la de combinar els altres dos processos esmentats, mentre que la del *Pas 5* no pot superar la del *Pas 0*. Podem concloure, per tant, que la complexitat final de l'algorisme és

$$O((1 + min\{n_{cam}, \frac{|S'|}{|S|}\})(|A| + |V|\log|V|)),\ 0 \leq n_{cam} < |S|,\ |S'| \leq |V| - |S|.$$

Tenint en compte que, fins ara, la complexitat de la SPH era $O(|S|(|A|+|V|\log|V|))$:

- En aquells casos en què $|S|$ és superior a un valor proper a la meitat dels vèrtexs, la complexitat es redueix fins a $O(|A| + |V|\log|V|)$, passant de cúbica a quadràtica en aquells casos en què el graf és dens.

- El cas més desfavorable de l'algorisme correspon a $|S| \approx \sqrt{|V|}$ i l'arbre resultant del *Pas 4* ha incorporat $|S'| \approx |S|^2$ vèrtexs d'Steiner. En aquest cas, $min\{n_{cam}, \frac{|S'|}{|S|}\} \approx \sqrt{|V|}$ i, per tant, la complexitat de l'algorisme esdevé $O(\sqrt{|V|}(|A| + |V|\log|V|))$. Tot i que la situació sembla poc usual, sobretot si el graf és dens, aquest fet significa que la complexitat de l'algorisme en aquesta situació extrema és $O(|V|^{2+1/2})$ i, per tant, deixa de ser $O(|V|^3)$.

- Contràriament al que passava amb les implementacions anteriors, la complexitat de l'algorisme decreix a mesura que el nombre de terminals va augmentant a partir d'un cert llindar. Aquest comportament resulta lògic si es té en compte que a mesura que el nombre de terminals augmenta el nombre d'elements per bola en la partició decreix i, per tant, hi ha una disminució del nombre de canvis de bola.

- En la pràctica, la complexitat de la SPH s'acosta molt a la de la implementació de la DNH proposada per Mehlhorn [73] per a la majoria d'instàncies.

5.3.4 L'algorisme és una implementació de la SPH

En aquest apartat anem a demostrar que l'arbre que retorna l'algorisme que hem proposat pot ser també obtingut mitjançant la SPH. Com a element afegit, aquesta demostració és una alternativa a la que Mehlhorn [73] proposa per a la seva implementació de la DNH.

Utilitzarem $N^{-1}(v)$ per indicar el vèrtex v' tal que $v \in N(v')$. És a dir, $N^{-1}(v)$ és el vèrtex central de la bola que conté v en la partició. També, per a facilitar la comprensió, representen per s, s' o s'' els terminals que no formen part de G_1, i per t, t' o t'' els que formen part d'aquest arbre.

Lemma 5 *Tot camí seleccionat en el Pas 2 és un camí de cost mínim a G.*

Demostració: Sigui P_{st} un dels camins que són seleccionats en el *Pas 2* i suposem que P_{st} no és un camí de cost mínim en distància d, és a dir, $d(s,t) < d'(s,t)$. Sigui P'_{st} un camí de cost mínim en distància d que connecta s amb t, i sigui V'_{st} el seu conjunt de vèrtexs.

Seleccionem $v \in V'_{st}$, $s' = N^{-1}(v)$, $v' \in V'_{st}$ i $t' = N^{-1}(v')$ de manera que: v és el vèrtex més proper a t en P'_{st} tal que $s' \notin V_1$, i v' és el veí de v en el subcamí de P'_{st} que va de v a t. (D'aquesta manera $s' \notin V_1$ i $t' \in V_1$). En aquest cas,

$$d(s,t) = d(s,v) + d(v,v') + d(v',t) \geq d(s',v) + d(v,v') + d(v',t') \geq d'(s',t') \geq d'(s,t)$$

La primera desigualat es compleix perquè $v \in N(s')$ i $v' \in N(t')$, la segona es conseqüència de la forma en què hem seleccionat v i v', i la tercera del mecanisme de selecció de s i t en l'algorisme. ∎

Lemma 6 *Tot camí seleccionat en el Pas 2 és un dels camins que tenen cost menor entre tots els que connecten un terminal que no forma part de G_1 i un vèrtex que en forma part.*

Demostració: Sigui P_{st} un dels camins que són seleccionats en el *Pas 2* i suposem que no és un dels camins de menor cost entre els que connecten un terminal que no forma part de G_1 i un vèrtex que en forma part. Sigui $P_{s't'}$ un dels camins de cost mínim entre els que connecten $s' \in S - V_1$ amb $t' \in V_1$, i sigui $V_{s't'}$ el seu conjunt de vèrtexs. D'aquesta manera, $d(s',t') < d(s,t) = d'(s,t) \leq d'(s',t')$.

Seleccionem $v \in V_{s't'}$, $s'' = N^{-1}(v)$, $v' \in V_{s't'}$ i $t'' = N^{-1}(v')$ de manera que: v és el vèrtex més proper a t' en $P_{s't'}$ tal que $s'' \notin V_1$, i v' és el veí de v en el subcamí de $P_{s't'}$ que va de v a t'. (D'aquesta manera $s'' \notin V_1$ i $t'' \in V_1$). En aquest cas,

$$d(s',t') = d(s',v)+d(v,v')+d(v',t') \geq d(s'',v)+d(v,v')+d(v',t'') \geq d'(s'',t'') \geq d(s,t)$$

La primera desigualat es compleix perquè $v \in N(s'')$ i $v' \in N(t'')$, la segona es conseqüència de la forma en què hem seleccionat v i v', i la tercera del mecanisme de selecció de s i t en l'algorisme. ∎

Tant la SPH com la DNH seleccionen un camí de forma arbitrària quan n'hi ha més d'un de disponible amb el mateix cost. Com a conseqüència, i tal i com hem pogut veure en els capítols anteriors, fins i tot dues implementacions del mateix algorisme poden produir resultats diferents. Direm que un arbre d'Steiner és una *solució vàlida* per a un algorisme si pot ser produïda per aquest algorisme en alguna de les possibles seleccions arbitràries de camins de cost mínim.

Teorema 7 *L'algorisme que hem proposat retorna una solució vàlida per a la SPH.*

Demostració: A partir dels lemes anteriors es pot concloure que tot camí P_{st} que és seleccionat per l'algorisme que proposem és un dels camins de cost mínim dels que connecten un terminal $s' \in S - V_1$ amb un altre vèrtex $v \in V_1$. Per tant, la SPH podria seleccionar el mateix camí en cada iteració i, d'aquesta manera, retornar el mateix arbre. ∎

5.3.5 Quasi-SPH (qSPH)

Tot i que la SPH ha produït millors resultats que la DNH en tots aquells casos en què tots dos mètodes han estat comparats [84, 93, 89, 28], i que la implementació

que acabem de proposar de la SPH té, per a la majoria d'instàncies, la mateixa complexitat que la DNH, hi ha algunes situacions en què la complexitat de la SPH continua essent més gran. Es tracta dels casos ja esmentats en la pàgina 125 en què, quan $|S| \approx |V|$ i el nombre de vèrtexs d'Steiner és quadràtic respecte a $|S|$, la complexitat de la nova implementació és $O(\sqrt{|V|}(|A| + |V| \log |V|))$. La qSPH és una senzilla modificació de la SPH que, representada també per l'algorisme 12, introdueix algunes modificacions en el procés d'actualitzar la partició $\mathcal{P}$.

En la SPH, un vèrtex $v \in N(t)$ és reexplorat quan, formant part t de V_1, és ubicat en una altra bola, amb la qual cosa s'augmenta la complexitat de l'algorisme. En la qSPH els vèrtexs que no formen part de V_1 es descarten després de ser explorats i, d'aquesta manera, el total d'exploracions es manté sempre per sota del nombre de vèrtexs. La complexitat d'actualitzar $\mathcal{P}$ passa a ser $O(|A| + |V| \log |V|)$, la mateixa que la complexitat de la DNH.

Durant les primeres fases d'una execució la SPH, quan només uns quants terminals formen part de V_1, resulta molt poc freqüent que un vèrtex sigui reexplorat. Per tant, en aquestes fases de l'execució el comportament de la qSPH és molt semblant al de la SPH. En canvi, en la part final de l'execució, quan la majoria de terminals ja formen part de l'arbre en construcció, els canvis de bola dels vèrtexs solen anar acompanyats de reexploracions. Com que la qSPH no realitzarà aquestes noves exploracions, actuarà de la mateixa manera en què ho faria la DNH. Arribem a la conclusió, per tant, que estratègicament la qSPH es troba en algun punt a mig camí entre la SPH i la DNH, la qual cosa fa preveure uns resultats situats en algun punt intermedi entre els de la SPH i els de la DNH.

La qSPH té la mateixa complexitat que la DNH, millora el temps d'execució de la SPH i és més senzilla d'implementar. Tenint en compte que el nombre de vèrtexs intermedis que són incorporats a l'algorisme acostuma a ser molt més alt en les primeres fases de l'execució, conjecturem que els resultats que produeix estan molt més propers als de la SPH que als de la DNH, d'aquí que l'hàgim anomenat quasi-SPH a l'algorisme. En qualsevol cas, la seva utilització sembla més indicada que la de la DNH i, podria substituir la SPH en els casos en què, en funció del nombre de terminals, el temps d'execució d'aquest algorisme pogués resultar crític.

5.4 Noves variants de la SPH

Des d'una perspectiva d'heurística d'arbre (veure l'apartat 5.1.2), la SPH realitza dos processos de reestructuració diferents en l'arbre[8], susceptibles de millorar-ne el cost. El primer es porta a terme en el *Pas 3* cada cop que s'incorpora un vèrtex no-terminal, i afecta la part corresponent al subarbre del graf distància D. En realitat, aquest procés es realitza de manera implícita en les altres versions de l'algorisme i, per tant, resulta innecessari. El segon procés de reestructuració de l'arbre es porta a terme un únic cop en el *Pas 5*, i consisteix en calcular un arbre generador de cost mínim (MSpT) per tal d'assegurar que els vèrtexs de l'arbre resultant es connecten de manera òptima. Aquest procés és comú a totes les versions de la SPH des que l'any 1986 Rayward-Smith i Clare [84] van suggerir d'afegir-lo a la versió originària proposada per Takahashi i Matsuyama [88]. En aquesta secció es proposen diverses variants de la SPH que incideixen, per tal d'introduir millores, en aquest segon procés de reestructuració.

En la SPH, un no-terminal és considerat com si fos un terminal tant bon punt passa a formar part de l'arbre G_1 en construcció. Per tant, es pot dir que aquest vèrtex no és considerat com si fos terminal fins el moment en què passa a formar part de l'arbre. Aquest és el motiu pel qual l'arbre resultant pot no ser un MSpT i, per tant, que calgui el *Pas 5* —és possible que els vèrtexs que ja formen part de G_1 haguessin pogut ser connectats de manera millor utilitzant les arestes que els uneixen amb aquest no-terminal.

Les tres noves variants de la SPH que proposem en aquesta secció han estat presentades a [48], i es diferencien per la manera en què l'arbre G_1 és reestructurat en cada iteració quan s'incorporen no-terminals:

- En l'*Alternative*-SPH (ASPH) l'arbre G_1 és reestructurat fent servir l'MSpT, és a dir, $G_1 := MSpT(G, V_1)$. Si V_1 és el conjunt de vèrtexs de l'arbre resultant del *Pas 4* de la SPH, es pot veure que l'ASPH retorna un $MSpT(G, V_1)$ i, per tant, eliminant els vèrtexs corbata resulta equivalent a la SPH. El *Pas 5* de l'algorisme pot ser suprimit.

[8] Connectar els vèrtexs de l'arbre de manera diferent.

- En l'*Improved*-SPH (ISPH) l'arbre G_1 és reestructurat en el graf distància D, és a dir, $G_1 := MSpT(D, V_1)$. Abans del *Pas 5* es realitza una substitució de les arestes de D pels camins en G, tal i com es porta a terme en la DNH. Es pot veure que el resultat de l'ISPH és un $DNH(G, V_1)$, on V_1 és el conjunt de vèrtexs de l'arbre resultant del *Pas 4* i, per tant, té un cost no inferior[9] al de l'arbre que retorna la SPH, $MSpT(G, V_1)$.

- En la *Generalized*-SPH (GSPH) l'arbre G_1 es reestructurat mitjançant la SPH, la qual cosa equival a tractar els nous no-terminals com si haguessin estat terminals des del principi. L'arbre resultant serà un arbre generador de cost mínim tant en el graf D com en el G. La diferència entre l'ISPH i la GSPH és molt similar a la que hi ha entre la DNH i la SPH, ja que considera els nous no-terminals que s'incorporen per reestructuració com si fossin terminals.

Tot i que no es demostra en aquesta secció, creiem important destacar que tant l'ISPH com la GSPH admeten una implementació calculant els camins de cost mínim tal i com es proposa en la secció 5.3, la qual cosa fa que la complexitat d'aquests algorismes sigui molt similar a la d'aquesta implementació de la SPH. Sembla clara, per tant, la utilitat que poden tenir tots dos algorismes per tal de ser utilitzats en comptes de la SPH com a heurística de base per a mètodes competitius. D'altra banda, resulta evident la possibilitat d'aplicar les mateixes modificacions a la DNH per tal d'obtenir les variants ADNH, IDNH i GDNH d'aquest algorisme.

En l'apartat següent es presenten alguns dels processos que seran utilitzats al llarg d'aquesta secció. En els apartats 5.4.2, 5.4.3 i 5.4.4 es presenten l'ASPH, l'ISPH i la GSPH, respectivament. Finalment, en l'apartat 5.4.5 es comparen aquests algorismes amb el que van proposar Duin i Voβ [28], i es suggereixen possibles millores.

[9]No tenim en compte l'eliminació dels vèrtexs corbata.

5.4.1 Mètodes preliminars

Incorporar un nou vèrtex a un MSpT

Diversos algorismes han estat proposats per tal de reduir la complexitat del problema següent:

- Donat $G(V,A)$ i $T = MSpT(G,V_T)$ trobar $T' = MSpT(G,V_T \cup \{v\})$, essent $V_T \subset V$ i $v \in V - V_T$.

La idea que hi ha darrera d'aquests algorismes és que T' es pot obtenir com a $MSpT$ del subgraf de G que conté, únicament, les arestes de T i les arestes que connecten v amb els vèrtexs de V_T. Per exemple, Spira i Pan [87] van proposar l'any 1975 un algorisme que té complexitat, en el pitjor cas, $O(|V_T|)$. També, l'algorisme proposat per Minoux [74] l'any 1990 té la mateixa complexitat en la mitjana de tots els casos.

En els apartats següents haurem de resoldre un problema de característiques similars al que acabem de plantejar. En la formulació del problema que donem tot seguit, utilitzem $G \cup (v_i,v_j)$ per indicar el graf resultant d'afegir l'aresta (v_i,v_j) a G:

- Donat $G(V,A)$ i $T = MSpT(G \cup (v_i,v_j),V_T)$ trobar $T' = MSpT(G,V_T \cup \{v_k\})$, essent (v_i,v_j) una aresta de D, $V_T \subset V$, $v_k \in V - V_T$, i (v_i,v_k,v_j) un camí de cost mínim.

T' és un MSpT de G que ha de contenir el vèrtex v_k però no pot incloure l'aresta (v_i,v_j). Com en el problema anterior, un $MSpT(G \cup (v_i,v_j),V_T \cup \{v_k\})$ pot ser calculat com a un MSpT del graf que conté, únicament, les arestes de T i les que uneixen v amb vèrtexs de V_T. No resulta difícil adonar-se que un arbre calculat d'aquesta manera no pot contenir l'aresta (v_i,v_j). Conseqüentment, l'arbre T pot ser calculat fent servir qualsevol dels dos algorismes que hem esmentat per al problema anterior.

Transformar un subgraf arbre de D en un de G amb cost no superior

Sigui $T_D = (V_T,A_T,c')$ un subgraf arbre del graf distància D de G. El mètode que proposem a continuació permet transformar T_D en un arbre de G que connecta V_T, o $St(G,V_T)$, amb cost no superior al de T_D.

- $V_G := V_T$, $A_G := A_T$ i $T_G = (V_G, D_G, c')$.
- Mentre $\exists (s,t) \in A_G$ tal que no pertany (o té cost més gran) a A:

  Sigui v un vèrtex intermedi en el camí de cost mínim de s a t:

  $T_G := MSpT(D, V_G \cup \{v\})$.

L'arbre resultant $T_G = (V_G, A_G, c')$ és un MSpT per a V_G en G. A més a més, T_G és també un MSpT per a V_G en D i, per tant, cap aresta $(v_i, v_j) \in A_G$ pot ser substituïda per un camí entre dos vèrtexs v_k i v_l de l'arbre de manera que en resulti un arbre de cost menor.

Cadascun dels càlculs de l'MSpT que es realitzen està associat a la incorporació d'un nou vèrtex v en l'arbre, i pot portar-se a terme mitjançant un dels dos algorismes $O(|V_G|)$ que hem esmentat abans[10]. Per tant, el nombre d'iteracions necessàries en aquest mètode és igual al nombre de vèrtexs addicionals que són inclosos en l'arbre.

Finalment, es pot veure que l'arbre resultant pot dependre de l'ordre en què les arestes de D són seleccionades.

5.4.2 Alternative-SPH (ASPH)

En el *Pas 3* de la SPH (veure l'apartat 2.4.1), consistent en afegir el camí P_{st} a G_1, els no-terminals són incorporats a G_1 com a vèrtexs intermedis del camí de cost mínim seleccionat. A diferència dels terminals, els no-terminals no són tinguts en compte per l'algorisme fins que passen a formar part de l'arbre. Per tant, la inclusió d'un no-terminal activa la possibilitat de reduir el cost de G_1 a base de connectar millor els vèrtexs que ja formen part d'aquest arbre. En altres paraules, no es pot assegurar que l'arbre en construcció sigui un MSpT i, per això, a vegades pot ser escurçat mitjançant el *Pas 5*.

Al contrari que en la SPH, en l'ASPH es garanteix que l'arbre en construcció és en tot moment un MSpT. Aquest fet obliga a utilitzar un mecanisme més acurat que el del *Pas 3* per a incorporar un nou vèrtex en l'arbre, però també evita el càlcul de

[10]Aquesta complexitat no té en compte el càlcul dels camins de cost mínim que van de v als vèrtexs de V_G.

l'MSpT en el *Pas 5*. L'ASPH s'obté a base de substituir els passos 3 i 5, pels passos 3′ i 5′ següents:

- *Pas* 3′. Sigui v el vèrtex més proper a t en el camí P_{st}:

  si $v \neq t$ aleshores $G_1 := MSpT(G, V_1 \cup \{v\})$;

  si no afegir (s,t) a G_1;

- *Pas* 5′. Retornar $T = trim(G_1)$.

Tal i com hem vist en la secció 5.4.1, el càlcul de l'MSpT es pot fer mitjançant un algorisme amb complexitat $O(|V_1|)$. Per tant, la complexitat addicional que afegeix aquest procés a la SPH és $O(|S'|(|S|+|S'|))$ però, a canvi, evita el càlcul de l'MSpT en el *Pas 5* que és $O(|A| + (|S| + |S'|)\log(|S| + |S'|))$. Per exemple, quan no s'incorpora cap vèrtex intermedi, el comportament de l'ASPH és igual al de la SPH quan no s'executa el *Pas 5* que, en aquest cas, resulta innecessari. Per tant, la complexitat de l'ASPH és molt semblant a la de la SPH, incloent la possibilitat d'utilitzar una implementació amb complexitat similar a la que ha estat proposada per a la SPH en la secció anterior.

L'arbre que és retornat tant per la SPH com per l'ASPH és un $MSpT(G, V_1)$ al qual se li han eliminat els vèrtexs corbata. La distribució de les corbates depèn de l'ordre en què els vèrtexs i les arestes són seleccionats en el càlcul de l'MSpT final i, per tant, els arbres que retornen tots dos algorismes no tenen per què ser iguals i, fins i tot, poden tenir costos diferents. De tota manera, l'arbre que retorna l'ASPH és un arbre vàlid per a la SPH, ja que pot ser generat en com a mínim una de les seleccions arbitràries de l'MSpT en el *Pas 5*. Per tant, sense descartar la possibilitat de que la SPH pugui retornar algun arbre que no pot ser generat per l'ASPH, podem considerar que l'ASPH és una implementació vàlida de la SPH (veure la pàgina 127).

5.4.3 Improved-SPH (ISPH)

En l'ASPH es poden distingir dos processos diferents que són portats a terme en cada iteració: mentre que un procés se n'encarrega de seleccionar els vèrtexs que passaran

a formar part de l'arbre en construcció, l'altre intenta reduir el cost d'aquest arbre connectant els seus vèrtexs d'una manera diferent. A l'ISPH, sense que tingui cap mena d'incidència en el primer procés, s'augmenta la capacitat de reduir el cost d'aquest arbre en el segon a base d'admetre la utilització d'arestes del graf distància D.

L'ISPH es diferencia de l'ASPH en dos aspectes. D'una banda, el *Pas* 3′ és substituït pel *Pas* 3″ següent, que utilitza el graf D en comptes del graf G:

- *Pas* 3″. Sigui v el vèrtex més proper a t en el camí P_{st}:

  si $v \neq t$ aleshores $G_1 := MSpT(D, V_1 \cup \{v\})$;

  si no afegir (s,t) a G_1;

De l'altra, el graf G_1 resultant del *Pas* 5′, que és un graf vàlid en D, és transformat en un subgraf arbre de G amb cost no superior, la qual cosa es pot realitzar de dues maneres diferents. En la primera, es substitueixen totes les arestes de D pels camins associats i, després, es calcula un $MSpT$ per al conjunt de vèrtexs resultant. El resultat que s'obté equival a aplicar la DNH fent servir el conjunt de vèrtexs de l'arbre resultant de la SPH. En la segona, s'utilitza el mètode que ha estat descrit en l'apartat 5.4.1 i que té l'avantatge que l'arbre resultant és un $MSpT$ en el graf D i, a més a més, permet introduir una eliminació de corbates en cada iteració que podria millorar el resultat final. Sigui quin sigui el procediment final utilitzat, si T_{SPH} és l'arbre resultant de l'ASPH, T_D és un $MSpT(D, vertexs(T_{SPH}))$ i T_{ISPH} és l'arbre resultant de l'ISPH, es complirà la relació següent:

$$|T_{SPH}| \geq |T_D| \geq |T_{ISPH}|.$$

A diferència de la SPH, en l'ISPH és necessari calcular els camins de cost mínim entre els vèrtexs no-terminals que passen a formar part de l'arbre i els vèrtexs que ja en formen part. Per tant, respecte la implementació clàssica de la SPH, l'ISPH haurà d'executar $|S'|$ vegades més l'algorisme de Dijkstra [23] per a calcular els camins. Aquests camins, però, poden ser obtinguts a partir de la partició $\mathcal{P}$ tal i com es porta a terme en la implementació alternativa de la SPH que

hem presentat en la secció anterior. En aquest cas, la complexitat de l'ISPH és $O((1 + min\{n_{cam}, \frac{|S'|}{|S|}\})(|A| + |V| \log |V|) + |S'||V|)$, on $n_{cam} \leq |S'|$ és el nombre de camins que tenen vèrtexs intermedis que, en aquest cas, també inclou els que sorgeixen en el procés final de transformació de l'arbre[11]. L'últim terme és el mateix que hem inclòs en la complexitat de l'ASPH. Podem concloure, per tant, que la complexitat de l'ISPH és molt semblant a la de la SPH.

5.4.4 Generalized-SPH (GSPH)

Durant el procés iteratiu de l'ISPH, l'arbre G_1 en construcció és un arbre vàlid en D, però no té per què ser-ho en el graf G. Els vèrtexs intermedis de les arestes en D no són tinguts en compte per l'algorisme i, per tant, de manera semblant al que succeïa en la DNH (veure l'apartat 5.1.1), tampoc són tinguts en compte a l'hora de seleccionar el camí en el *Pas 2.* En el GSPH els no-terminals són utilitzats en aquesta selecció i, a més a més, l'arbre G_1 és actualitzat en cada iteració de manera que sigui sempre un $MSpT$ en els dos grafs, G i D. La GSPH s'obté a partir de substituir el *Pas* 3′ de l'ASPH pels dos passos següents:

- *Pas 3.1.* Sigui v el vèrtex més proper a t en el camí P_{st}:

  si $v \neq t$ aleshores $G_1 := MSpT(D, V_1 \cup \{v\})$;

  si no afegir (s, t) a G_1;

- *Pas 3.2.* Transformar G_1 en un subgraf arbre de G com en l'apartat 5.4.1.

El *Pas 3.1* és idèntic al *Pas* 3″ de l'ISPH. La complexitat del *Pas 3.2* és $\mathcal{O}(k|V_T|)$, on k és el nombre de vèrtexs intermedis addicionals que són incorporats a G_1 en aquest pas. Per tant, la complexitat de la GSPH és semblant a la de l'ISPH i, conseqüentment, a la de la SPH.

Tot i que el cost de l'arbre resultant de l'ASPH pot ocasionalment ser millor que el de la GSPH, aquest algorisme incorpora dos mecanismes addicionals per tal de disminuir el cost de l'arbre en construcció: mitjançant la utilització del graf D

[11]Notem que $n_{cam} < |S|$ ha estat substituït per $n_{cam} \leq |S'|$.

en comptes del graf G en el *Pas 3.1*, i a base de transformar G_1 en un arbre de cost no superior en el *Pas 3.2*. Tots dos mecanismes fan augmentar les possibilitats d'incorporar no-terminals a l'arbre i, per tant, de reduir la distància entre els terminals que no formen part de G_1 i els vèrtexs que en formen part, la qual cosa representa una millora que serà reflectida més endavant.

Tal i com hem vist, la inclusió d'un nou no-terminal en l'arbre G_1 no pot tenir un efecte negatiu en el cost d'aquest arbre. Tot i això, la presència d'aquest vèrtex pot en alguns casos evitar que siguin incorporats altres no-terminals que podrien produir beneficis més grans. Aquesta situació resulta difícil de preveure i és comuna tant en la SPH com en l'ASPH. Per tant, sembla que no té massa sentit descartar la millora que la inclusió d'un no-terminal produeix a menys que tinguem la certesa que, actuant d'aquesta manera, es podrà aconseguir una millora posterior més gran.

Finalment, tot i que l'ISPH té com valor afegit el fet que permet garantir una millora respecte la SPH, la relació entre l'ISPH i la GSPH és la mateixa que la que hi ha entra la DNH i la SPH. Per tant, fent servir els mateixos arguments que en l'apartat 5.1.4, és de preveure que la GSPH tendeixi a produir millors resultats que l'ISPH.

5.4.5 Intercanvis de camins per arestes

Tot i tenir uns objectius diferents, l'algorisme que Duin i Voß (VGA) [28] van proposar l'any 1997 té força aspectes en comú amb les dues variants de la SPH que acabem de proposar. A grans trets, el VGA està orientat a millorar el rendiment de la SPH a base de permetre la substitució no només d'arestes, sinó també de camins que contenen no-terminals de grau dos. Si no es té en compte aquest fet, el funcionament del DVA és molt semblant al del mètode de la pàgina 131 per a passar d'un subgraf arbre de D a un de G, i es pot sintetitzar, de manera aproximada[12], de la forma següent:

- En una primera fase, les arestes de D en l'arbre inicial van sent explorades una a una per tal d'introduir els seus vèrtexs intermedis en l'arbre mitjançant el

[12]Resulta difícil de determinar si la descripció que donem del DVA cobreix tots els controls que es realitzen en aquest algorisme.

càlcul d'un MSpT en D. Després de cada exploració, els camins que contenen no-terminals de grau dos són substituïts per arestes de cost mínim en D.

- En la segona fase, es va executant de forma recurrent un procés similar que contempla la possibilitat de connectar un no-terminal amb un vèrtex intermedi del camí associat a una aresta de D. El procés finalitza quan l'arbre deixa de millorar. Un cop substituïdes les arestes de D, l'arbre resultant és un $MSpT$ en els grafs D i G.

Un dels problemes que té el DVA és distingir, a priori, els casos en què la supressió d'un no-terminal de grau dos possibilitarà un guany per substitució de l'aresta en D, dels casos en els quals aquesta supressió evita un guany a l'impedir connectar un nou vèrtex amb aquest no-terminal. Aquesta és la causa principal que porta a utilitzar la segona fase, que és la que hem classificat dins d'una estratègia iterativa recurrent (veure l'apartat 2.5.3). Tenint en compte que, en general, la supressió d'aquests vèrtexs sembla ser una bona aposta per a millorar el resultat final, podria resultar interessant introduir un mecanisme similar en les variants de la SPH que proposem en aquesta secció. Tot i que no hem analitzat aquesta possibilitat en detall, ens fa l'efecte que les modificacions necessàries poden ser encabides d'una forma natural en les noves versions de la SPH.

El DVA parteix d'un arbre inicial en D, que no té per què ser MSpT, i no contempla cap criteri especial en l'ordre de selecció de les arestes que són explorades. Quan l'ISPH i la GSPH utilitzen els vèrtexs d'aquest arbre com a subconjunt de vèrtexs d'entrada, i el DVA selecciona les arestes en el mateix ordre en què ho fan aquests algorismes, es poden establir alguns paral·lelismes entre el comportament del DVA i el de l'ISPH o l'GSPH. Des d'una perspectiva d'heurística d'arbre (veure l'apartat 5.1.2), aquests dos algorismes reestructuren en cada iteració, d'una forma implícita, la part en D de l'arbre que tenen associat. Aquest fet sembla apuntar cap a la possibilitat de proposar una implementació per al DVA de menor complexitat.

5.5 Conclusions

La SPH i la DNH són, amb diferència, els dos algorismes d'aproximació més àmpliament utilitzats com a algorismes de base per a la construcció de mètodes competitius per al PSG. Amb el propòsit de decidir quin d'aquests algorismes pot contribuir en un nivell més alt en el rendiment d'aquests mètodes, la SPH i la DNH han estat comparades des de tres perspectives diferents: els resultats que produeixen, l'estratègia en què es basen i la complexitat computacional.

Des del punt de vista dels resultats que produeixen, la SPH ha mostrat un comportament millor que el de la DNH en tots aquells estudis dels que tenim coneixement que han comparat els algorismes [84, 93, 89, 28]. A més, contràriament al que es suposava [84, 73, 93, 57], la FRA de la SPH és igual a la de la DNH, tal i com van demostrar Duin i Voß [28] l'any 1997 i ha estat demostrat de manera alternativa en aquest capítol.

Des d'un punt de vista estratègic hem pogut constatar les profundes similituds entre tots dos algorismes, que poden ser classificats indistintament com a heurístiques de vèrtexs, de camí, o d'arbre [57]. L'algorisme que acostumarà a produir millors resultats és aquell que tingui tendència a allunyar-se en un grau més alt del cost d'un $MSpT(D,S)$, fita superior del cost del resultat que retornen tots dos algorismes, i aspecte clau en la demostració de la seva FRA. Per a totes les estratègies es poden trobar arguments en favor de la superioritat de la SPH respecte la DNH. Resulta sorprenent, però, que cap dels dos algorismes tingui en compte en cap moment alguns factors que són decisius en el rendiment dels algorismes, com ara la manera en què es calculen els camins, la presència de vèrtexs intermedis o la manera en què aquests vèrtexs contribueixen a millorar la solució.

Des del punt de vista de la complexitat, s'ha proposat un nou mètode per a implementar la SPH que en redueix la complexitat d'$O(|S|(|A| + |V|\log|V|))$ fins a $O((1 + min\{n_{cam}, \frac{|S'|}{|S|}\})(|A| + |V|\log|V|))$, on $n_{cam} < |S|$ és el nombre de camins amb vèrtexs intermedis que són seleccionats per l'algorisme i $|S'| \leq |V| - |S|$ és el nombre de vèrtexs no-terminals en la solució. Exceptuant aquells casos en què $|S|$ és extremadament petit, no resulta massa habitual que el nombre de vèrtexs d'Steiner

en la solució arribi a ser diverses vegades superior al nombre de terminals i, per tant, la complexitat de la SPH esdevé, per a la majoria d'instàncies, igual a la de la DNH, que és l'algorisme d'aproximació per al PSG que té menor complexitat.

Podem concloure, per tant, que la SPH resulta una millor elecció que la DNH com a mètode de base per a construir mètodes competitius i, per tant, podria ser usada per a millorar el rendiment dels mètodes que es basen en la DNH [2, 30, 31, 90]. D'altra banda, la millora en la complexitat de la SPH permet millorar els temps d'execució dels mètodes competitius que utilitzen aquest algorisme [72, 67, 93, 44, 45]. La millora és especialment important en aquells mètodes que no poden calcular els camins un únic cop a priori [72, 67] (veure l'apartat 2.5.5) i, tot i que no representa una millora significativa en els altres, possibilita evitar la dependència de l'ordre dels vèrtexs en el càlcul dels camins (veure l'apartat 3.3.1).

Un cop determinada la preferència per la SPH, hem proposat dues noves variants d'aquest algorisme, l'ISPH i la GSPH, amb el propòsit de millorar-ne el rendiment. Tot i que els resultats que obté l'ISPH no poden ser inferiors als que produeix l'ASPH —una implementació vàlida de la SPH—, tot sembla indicar que el mètode que produirà millors resultats és la GSPH: l'arbre resultant d'aquest algorisme és simultàniament subgraf de G i de D i, a més, la diferència entre l'ISPH i la GSPH és la mateixa que la que hi ha entre la DNH i la SPH. Tots dos algorismes tenen una complexitat similar a la de la SPH i, per tant, podrien substituir aquest algorisme per tal de millorar el rendiment dels mètodes competitius que hem esmentat abans. D'altra banda, el rendiment d'aquests algorismes podria millorar amb la inclusió d'algunes de les propostes que fan Duin i Voβ [28].

L'únic algorisme d'aproximació que podria rivalitzar amb la SPH com a algorisme de base d'alguns mètodes competitius és el ZH [98], especialment després que Duin i Voβ en reduïssin la complexitat. Amb la nova implementació de la SPH, les diferències de complexitat entre els dos algorismes tornen a ser molt importants.