The parameterized complexity of the Rectilinear Picture Compression Problem

Tomàs Winand

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Índex

Capítol 1

Introducció

Hi ha molta gent que veu els ordinadors com a caixes negres capaces de respondre qualsevol pregunta que els vulguem plantejar i que no han pensat mai en la possibilitat que un ordinador no fos capaç de resoldre el càlcul. Recordo el dia en que vaig proposar a un professor d'àlgebra l'ús de mètodes aproximats per trobar la inversa d'una matriu 8.000×8.000, i ell em va contestar que ho fes utilitzant Cramer. I és que hi ha infinitat d'operacions que a nivell teòric no plantegen absolutament cap dificultat però que, a la pràctica, quan les dimensions del problema o la mida dels valors augmenten es converteixen en intractables per culpa del temps de processador o la capacitat de memòria que necessiten.

L'estudi de les limitacions computacionals és una de les pedres angulars de l'evolució de la informàtica. Si els augments en la velocitat del processador o les progressives ampliacions en la capacitat de memòria dels ordinadors ens han permès un avanç tecnològic inimaginable fa 50 anys; no és menys cert que aquesta evolució s'està alentint i que els salts són cada cop més petits.

Com més potents han estat els nostres ordinadors més els hem demanat, esperant que resolguessin problemes amb dades cada cop més i més grans. Emperò, tot i els avenços tecnològics, hi ha problemes que continuen resistint-se.

La comprensió d'on és la frontera entre els problemes tractables i els que no ho són és el que em va motivar a estudiar la carrera d'informàtica, en concret sempre he sentit molta curiositat sobre els problemes que tot i tenir un senzill enunciat resulten intractables.

Normalment identifiquem el terme *tractabilitat* amb l'existència d'un algorisme determinista que el resolgui en temps polinòmic. La classe d'aquests problemes l'anomenarem P. Els problemes que pertanyen a P són aquells que admeten un algorisme que els resol en *temps raonable* [2]).

Malauradament, no tots els problemes pertanyen a P. En concret, la classe P resta inclosa dins la classe NP ($\mathsf{P} \subseteq \mathsf{NP}$). Classe que, en les mateixes condicions, ens assegura

que podem trobar un algorisme indeterminista que ens resolgui el problema en un *temps raonable* [2].

Ara bé, a l'hora de decidir si un problema pertany a P, el fet que no haguem trobat un algorisme determinista amb temps polinòmic, no vol dir pas que algú més astut no en pugui trobar un que si ho sigui. En definitiva, provar que un problema no pertany a P pot resultar francament complicat.

De fet, ni tan sols sabem si la classe $\mathsf{P} \neq \mathsf{NP}$. La majoria d'investigadors consideren que aquestes dues classes són diferents però ningú ho ha pogut demostrar. Així que, el que usualment s'intenta és demostrar que si es trobés una solució polinòmica pel problema que estem estudiant tots els altres problemes que pertanyen a NP pels que no s'ha trobat mai solució polinòmica, llavors en tindrien. D'aquests problemes en diem NP-difícils i si a més pertanyen a NP en diem NP-complets [3]. Fixem-nos que parlem de difícil en sentit que qualsevol altre problema de NP és, com a molt, tan difícil com el que estem estudiant.

Per més inri, hi ha multitud de problemes naturals que s'han demostrat NP-difícils i cal tenir en compte que demostrar que un problema és NP-difícil serveix gairebé exclusivament per desanimar-nos de trobar un algorisme tractable que el resolgui completament en temps polinòmic [3].

Però, si hi ha teories provades i contrastades que es veuen contínuament revisades, completades i retocades (em ve a la memòria el dia que vaig llegir que, contràriament el que havia cregut sempre, existeixen partícules que viatgen a velocitats més altes que la de la llum); perquè no hauríem d'estudiar els problemes NP-difícils? Doncs, hi ha molts científics que han dedicat els seus esforços a descobrir, per exemple, solucions parcials o solucions aproximades per alguns problemes NP.

Hi ha massa problemes NP-difícils interessants com per dir que simplement no tenen un algorisme eficient i despreocupar-nos-en. En aquesta línia, han anat apareixen gran quantitat de tècniques i innovadores teories que ens han permès una millor comprensió de la classe NP-difícil i treballar, a la pràctica, amb problemes que es consideren intractables.

Un dels detalls que em sembla important destacar és que tot sovint, afegint una restricció addicional al problema aconseguim resoldre'l eficientment. És a dir, a vegades, és fàcil localitzar quina és la peculiaritat que ens converteix el problema en NP-difícil i afegir alguna restricció al problema per tal d'evitar aquesta dificultat.

Tècniques d'aproximació que redueixen la duresa del problema al donar per bones solucions aproximades o algorismes aleatoritzats que esquiven la complexitat dels casos difícils són algunes de les tècniques que ens permeten superar les dificultats que ens plantegen alguns dels problemes NP-difícils. En concret, en aquest treball s'utilitzen tècniques de paràmetre fix per poder estudiar un parell de problemes NP-complets.

La idea de la complexitat parametritzada rau en analitzar detalladament l'estructura de les entrades del problema buscant franges que continguin gran part de les entrades

i que a l'hora es puguin resoldre en temps polinòmic. Un cop identificades aquestes franges s'intenta dividir qualsevol entrada en dues parts: la primera que pertanyi a alguna d'aquestes franges (anomenada part principal del problema) i la segona que, en general, no hi pertanyi i que anomenarem part parametritzada del problema. Com la part principal del problema es pot resoldre amb algun algorisme en temps polinòmic, en la pràctica, ens permetrà resoldre el problema per gran part de les entrades [4].

Exemplifiquem-ho amb el següent problema:

Vertex Cover
Instance: Un graf G amb n vèrtexs i un natural k
Parameter: k
Question: Té G un recobriment de vèrtexs de mida $\leq k$? (Un recobriment de vèrtexs d'un graf G és un subconjunt V' dels vèrtexs de G tal que qualsevol aresta e de G té almenys un dels seus extrems en V').

Considerant que l'entrada del problema és el graf de mida n i el paràmetre k, el problema resulta NP-complet [5]. Amb un algorisme $O(2^k n)$ [6]. Si, per contra, considerem que k és un paràmetre que fixem i no una part de l'entrada, l'algorisme és lineal i tractable per a casos en que la k no sigui massa gran ($k \leq 4000$) [7].

Aquesta restricció descriu el cas en que volem trobar recobriment de grafs utilitzant pocs vèrtexs en grafs arbitràriament grans. Tot i que no ens resol completament el problema, ens resol els problemes en els que ens interessin recobriments petits.

Dins de la classe NP, definirem la subclasse FPT (Fixed Parameter Tractable) com la dels problemes pels que existeix un algorisme en que la complexitat és de la forma $O(f(k)n^i)$ (on n és la mida de l'entrada, k el paràmetre i i no depèn de k ni de n) [8].

Els problemes FPT són els problemes pels que hem sigut capaços de detectar quina era la dificultat del problema i dividir-ne la complexitat en dues parts, una que creix polinòmicament amb l'entrada i una altra que depèn d'un paràmetre que podrem controlar mentre el mantinguem acotat, independentment de la mida l'entrada.

Per tal d'aprofitar-nos de les avantatges que presenta pertànyer a FPT ens aniria molt bé tenir algunes tècniques que ens permetessin detectar el paràmetre adequat i maneres d'aconseguir algorismes senzills.

Per tal d'obtenir parametritzacions que ens siguin útils no hi ha pas tècniques generals i es trien per experiència o per intuïció. Acostuma a resultar productiu provar paràmetres que apareixen amb naturalitat com poden ser: el número de vèrtexs, o el número de colors (en els problemes de grafs); o el nombre de literals, clàusules o variables (en problemes de fórmules lògiques); el número de columnes, files o elements (per problemes formulats com a matrius)...

Com veure'm més endavant, els problemes que hem estudiat estan formulats com a matrius i el paràmetre que ens ha permès resoldre'ls ha estat el número d'elements de la matriu, o dit d'altra manera, les dimensions de la matriu.

Un cop triat l'algorisme, caldria decidir-se per alguna tècnica de disseny d'algorismes parametritzats. El ventall és ample i en destacaria els següents:

- **menors de grafs** que malauradament només ens aporten solucions no constructives [9].
- **codificació de colors mitjançant funcions de dispersió** que s'han utilitzat per tractar problemes de camins o d'aparellaments [10].
- **tècniques de kernelització** que per exemple van servir per el millor algorisme per resoldre el problema de Recobriment de Vèrtexs del que parlàvem abans i, a més, és una de les metodologies més genèriques [7].

Nosaltres ens hem centrat en la tècnica de Kernelització. La Kernelització es basa en treballar sobre l'entrada, reduint-la i convertint-la en una entrada que depengui només del paràmetre. Si aconseguim que el procés de kernelització tingui un cost polinòmic en la mida de l'entrada, resolent el problema reduït mitjançant força bruta ens assegurem un algorisme FPT.

Presentades les tècniques que utilitzarem, em queda descriure el problema que hem estat estudiant. Dins de la gran quantitat de problemes NP-complets que podíem haver triat, hem decidit treballar en problemes que podríem anomenar de descripció d'imatges digitals. Aquest tipus de problemes ens permetien combinar els meus coneixements previs en compressió d'imatges al mateix temps que l'extensa experiència en complexitat parametritzada aplicada a problemes de grafs del professor Dimitrios (director del projecte).

Les imatges, en digitalitzar-se, es poden entendre com a matrius de pixels. Emmagatzemar aquestes matrius no és pas trivial ja que, per exemple, amb sistemes avui ja desfasats podríem estar parlant de matrius 800×600 on cada casella admet 256 colors diferents. La pregunta obvia és: hi ha alguna manera millor d'emmagatzemar aquestes matrius que no sigui guardar-ne tots els seus elements?

Nosaltres ens hem plantejat una versió restringida d'aquest problema en el que només admetem imatges en blanc i negre. Podem doncs, representar les imatges com a matrius en les que a cada posició hi pot haver només un element $i \in \{Blanc, Negre\}$. Utilitzant la bijecció $b : \{Blanc, Negre\} \to \{False, True\}$ tal que $b(Blanc) \mapsto False$ i $b(Negre) \mapsto True$, convertirem la imatge en blanc i negre en una matriu booleana que ens serà més còmoda d'utilitzar.

Hem estudiat dos possibles mètodes per emmagatzemar matrius booleanes:

- El primer cas estudiat consisteix en trobar un recobriment de rectangles que ens cobreixi els 1 de la matriu i només els 1. Un cop identificat aquest recobriment, guardant informació de quins són els rectangles que el formen n'hi haurà prou per

ser capaços de reconstruir la imatge. Formalment, el problema el podem enunciar com:

Rectiliniar Picture Compression (Repico)
Instance: Una matriu booleana $n \times n$ anomenada B i un enter positiu k
Parameter: k
Question: Existeixen k rectangles que cobreixin tots els uns de B?

- La segona aproximació de la que n'hem estudiat la complexitat consisteix en trobar un conjunt de corbes poligonals tancades que ens envoltin els 1 de la matriu i només els 1. L'enunciarem com:

  Poligonal Picture Hull Cover (Popico)
  Instance: Una matriu booleana $n \times n$ que anomenarem B i un enter positiu k
  Question: Podem trobar una seqüència S de vèrtexs amb k vèrtexs que sigui una envolupant de B?
  Parameter: k

L'objectiu genèric d'aquest projecte és l'estudi comparatiu d'aquests dos mètodes per emmagatzemar matrius booleanes. A continuació, voldríem descriure d'una manera més acurada quins seran els objectius perseguits al fer l'estudi:

- Demostrar que el problema de recobriment d'imatges utilitzant rectangles (Repico)pertany a la classe FPT, mitjançant tècniques de kernelització.
- Trobar una fita superior acurada per la dimensió d'aquest kernel.
- Descriure un algorisme FPT pel problema Repico.
- Trobar fites superiors i inferiors que ens relacionin les dimensions de la matriu amb el número de rectangles necessaris per recobrir-la.
- Demostrar que el problema de trobar una envolupant de la imatge utilitzant polígons (Popico) pertany a la classe FTP utilitzant tècniques de kernelització.
- Trobar una fita superior acurada per la dimensió d'aquest kernel.
- Descriure un algorisme FPT pel problema Popico.
- Trobar fites superiors i inferiors que ens relacionin les dimensions de la matriu amb el número vèrtex necessaris per descriure l'envolupant dels 1 de la matriu.

- Replantejar els problemes REPICO i POPICO de tal manera que admetin un petit nombre d'errors (cobrint o envoltant algun 0 respectivament).
- Estudiar la relació entre els problemes sense errors i els que permeten algun error.
- Trobar una relació entre les solucions que presenten els problemes REPICO i POPICO per poder comparar-les. La conjectura a demostrar és que el número de rectangles necessaris per cobrir la imatge no serà mai més gran que la meitat del nombre de vèrtexs dels polígons necessaris per construir l'envolupant.

En el proper capítol, es defineixen els conceptes bàsics sobre els que es fonamenta el treball al mateix temps que es descriuen les notacions i representacions utilitzades.

El capítol número 3 tracta els objectius referents al problema REPICO i el capítol 4 els referents al problema POPICO.

El capítol número 5 pretén respondre a les inquietuds referents a replantejar els problemes admeten errors.

Finalment, al capítol 6 trobarem l'anàlisi comparatiu de les dues tècniques estudiades.

Capítol 2

Preliminars

En aquest capítol repassarem algunes de les definicions sobre les quals fonamentarem els lemes, teoremes, corol·laris i demostracions dels capítols centrals d'aquest treball.

En concret, repassarem primer les definicions bàsiques sobre complexitat clàssica per després poder parlar amb propietat de la complexitat parametritzada que és la que realment ens interessa. Finalment, intentarem fer unes definicions acurades dels objectes d'aquest estudi i les seves característiques: les matrius booleanes.

2.1 Complexitat Clàssica

Per començar voldríem centrar idees pel que fa a les definicions que ens permetran parlar del temps de càlcul dels algorismes i de les classes de complexitat.

Per no recular en excés, donarem per conegudes les definicions d'alfabets, llenguatges, llenguatges regulars, autòmats finits i màquines de Turing [12]. Iniciem aquesta secció definint què són problemes decissionals ja que inclouen tots els problemes que tractarem en el treball.

Definició 2.1.1 *Parlarem de problema decissional quan vulguem distingir el subconjunt de paraules d'un alfabet Σ que compleixen una certa propietat P.*

Normalment identificarem un problema decissional amb el llenguatge L_P d'aquestes paraules. És a dir, $L_P = \{\omega \in \Sigma^* \mid tal\ que\ P(\omega)\}$ [12].

Parlarem d'algorisme decissional quan ens referim a un algorisme que retorni YES *sobre les paraules $w \in L_P$ i* NO *per la resta de mots.*

Entendrem també, que les entrades es poden codificar en binari mitjançant una funció de codificiació. Aquesta codificació la denotarem com a $\langle w \rangle$.

Pot semblar que els problemes decissionals són només una petita porció dels problemes que ens podem plantejar però, tot i que s'ha de reconèixer que no inclouen tots

els problemes que podem resoldre utilitzant Màquines de Turing, en són una part molt amplia.

Per exemple, si N és una xarxa representada per $N = (V, E, s, t, C)$ on (V, E) és un graf amb dos nodes destacats (s el node inicial i t el final) i C és una funció $C : E \to \mathbb{N}$ que anomenarem funció de capacitat. El problema de trobar el flux màxim que pot admetre la xarxa N sense desbordar la capacitat (MAX FLOW) de cap de les seves arestes és un problema d'optimització. Però, podem entendre'l com un problema decissional preguntant per a cada possible valor del flux si aquest és el màxim que admet la xarxa [11].

Definits el tipus de problemes que tractarem, ens caldrà tenir una certa mesura de la *bondat* dels algorismes que els resolen per poder-los comparar.

Tot i les diverses possibilitats que hi ha per comparar algorismes, nosaltres usarem com a mesura el temps de càlcul que inverteixen en resoldre una entrada concreta. El temps de càlcul depèn de la mida de l'entrada del problema així que començarem amb aquesta definició:

Definició 2.1.2 *Donada una entrada $\omega \in \Sigma^*$ al nostre problema, notarem com $|\omega|$ la mida d'aquesta entrada i coincidirà amb el número de bits necessaris per codificar-la en binari.*

Definició 2.1.3 *Definirem el temps de càlcul d'un algorisme amb una entrada ω, a partir del model de Màquina de Turing que utilitza k cintes de càlcul M i ho notarem com $t_M(\omega)$ [11]. Si donada una entrada concreta w, M s'atura després de t passos, llavors $t_M(\omega) = t$. Si, per contra, M no s'atura, $t_M(\omega) = \infty$.*

En aquest projecte, considerarem tan sols els algorismes anomenats d'aturada segura que són aquells pels que $\nexists w \mid t_M(\omega) = \infty$.

En general, agruparem totes les entrades que tenen la mateixa mesura i parlarem del temps de càlcul d'un algorisme en funció no de l'entrada sinó de la mida de l'entrada.

Definició 2.1.4 *Direm que un algorisme A, té un temps de càlcul $T_M : \mathbb{N} \to \mathbb{N}$, si $T_M(n) = max_{|x|=n} t_M(x)$.*

Compararem les funcions de temps fixant-nos en els seus infinitèssims, en el seu comportament quan tendim cap a infinit, per poder classificar els problemes segons el seu cost. La notació que utilitzarem per acarar les funcions de cost serà la de $\mathcal{O}$.

Definició 2.1.5 *Siguin f i g dues funcions definides de $\mathbb{N}$ a $\mathbb{N}$. Escriurem $f(n) = \mathcal{O}(g(n))$ si existeixen dos naturals n_0 i c tal que per tota $n \geq n_0$, $f(n) \leq cg(n)$. Podríem dir que f creix assimptòticament més a poc a poc o igual que g [11].*

Continuant amb l'exemple del problema MAX FLOW, la relació entre els conjunts de màxim flux i els talls minimals ens permeten obtenir un algorisme que el resol en temps polinòmic [11].

Abans d'entrar en les definicions de les classes de complexitat, introduirem la noció de funció computable que no és altra que la de les funcions que es poden simular mitjançant una màquina de Turing.

Definició 2.1.6 *Direm que $F : \Sigma^* \to \Sigma^*$ és una funció computable si i sols si existeix una màquina de Turing M que computa la funció F. És a dir, $M(x) = F(x) \forall x \in \Sigma^*$.*

Definirem ara, les classes de complexitat de les que tractarem en la resta del treball aprofitant la tesis de Church que ens assegura que qualsevol algorisme que puguem executar en un ordinador es pot interpretar en termes d'una màquina de Túring que executa els passos de l'algorisme:

Definició 2.1.7 *Direm que un llenguatge L pertany a la classe* P *si existeix una màquina de Turing M que ens permet resoldre el problema decissional P_L en un temps de càlcul polinòmic en la mida de les entrades. És a dir, si $\forall \omega \in L$, tal que $|\omega| = n$, llavors $T_M(n) \in \mathcal{O}(p(n))$. On $p(n)$ és un polinomi en funció de n [12].*

Dels problemes, els llenguatges dels quals tinguin algorismes (màquina de Turing) polinòmics en direm, simplement, polinòmics.

De la mateixa manera, les màquines de Turing no deterministes són aquelles en les que en almenys un dels passos de càlcul es decideix quin camí seguir de forma indeterminista. Llavors, s'accepta un mot sempre i quan hi hagi un camí indeterminista fins un dels estats acceptadors.

Definició 2.1.8 *Els llenguatges L que pertanyen a la classe* NP *seran pels que existeix una màquina de Túring no determinista M que ens permet resoldre el problema decissional P_L en temps de càlcul polinòmic en la mida de les entrades [12].*

Alternativament, podem dir que $A \in$ NP si i sols si $\exists B \in$ P i un polinomi p tq: $A = \{x \in \Sigma^* \mid \exists y \in \Sigma^*, |y| \leq p(|x|) \wedge \langle x, y \rangle \in B\}$.

Anomenarem problemes NP-difícils *als problemes pels que, si en trobéssim una solució polinòmica, la resta de problemes de* NP *també es podrien resoldre en temps polinòmic.*

Si a més de ser difícil, el problema pertany a NP*, direm que és* NP-complet.

Com a exemple de problema NP, proposem el problema de trobar un conjunt independent en un graf, IDEPENDENT SET. Problema que donat un graf G i un enter $k \geq 0$ es pregunta si $\exists S \in V(G), |S| = k : \forall e \in E(G) | e \cap S| \leq 1$ [13].

2.2 Pametritzacions

En aquesta secció entrarem a definir què són els problemes parametritzats, la classe dels problemes prametritzats i el procés de kernelització. Un cop definides, podrem utilitzar la tècnica de kernelització per trobar un algorisme parametritzat per als problemes: Popico i Repico.

Per a totes les definicions d'aquesta secció suposarem fixat un alfabet Σ sobre el que construirem les entrades del nostre problema.

Definició 2.2.1 [1]

Definirem una parametrització i un problema parametritzat com:

- *Una parametrització de Σ^* és una aplicació $k : \Sigma^* \to \mathbb{N}$ que es pot calcular en temps polinòmic.*
- *Un problema parametritzat, és una parella (L, k) on $L \subseteq \Sigma^*$ és el llenguatge que volem decidir i k una parametrització de Σ^*.*

Podem, per exemple, definir una parametrització del Independent Set que sigui $k(G, k) = k$.

Finalment definirem el concepte de tractabilitat mitjançant paràmetres i la seva relació amb la kernelització:

Definició 2.2.2 *Fixada una parametrització $k : \Sigma^* \to \mathbb{N}$,*

- *Un algorisme $\mathcal{A}$ és un algorisme* FPT *(la classe dels algorismes tractables amb paràmetre fix, Fixed Parameter Tratable) respecte de k si existeix una funció computable $f : \mathbb{N} \to \mathbb{N}$ i una funció polinòmica $p : \mathbb{N} \to \mathbb{N}$ tal que per cada $x \in \Sigma^*$, l'algorisme $\mathcal{A}$ necessita menys de $f(k(x)) \cdot p(|x|)$ passos.*
- *Un problema parametritzat (L, k) és tractable amb paràmetre fix si existeix un algorisme* FPT *respecte de k que decideix L.*
  Direm que $(L, k) \in$ FPT.

Si restringim el problema de trobar un conjunt independent a grafs plans (Planar Independent Set), el problema es converteix en un problema FPT (sense aquesta restricció és NP-difícil) [14].

A continuació definim el procés de kernelització que és la tècnica que utilitzarem per resoldre els problemes que hem estat estudiant:

[1] Les definicions 2.2.1 i 2.2.2 resten extretes dels apunts de l'assignatura de Complexitat impartida en la FIB.

Definició 2.2.3 *Una funció $K : \Sigma^* \to \Sigma^*$ computable en temps polinòmic és una kernelització de (L, k) si existeix una funció computable $h : \mathbb{N} \to \mathbb{N}$ tal que:*

- $\forall x \in \Sigma^*(x \in L \Leftrightarrow K(x) \in L)$
- $|K(x)| \leq h(k(x))$

Si K és una kernelització de (L, k), llavors per cada $x \in L$, anomenarem kernel de x (respecte de K) a $K(x)$.

Si, a més, existeix una funció polinòmica $p : \mathbb{N} \to \mathbb{N}$ tal que $|K(x)| \leq p(k(x))$, llavors direm que K és una kernelització polinòmica.

A partir de la fórmula d'Euler per a grafs plans, sabem que existeix almenys un vertex amb com a màxim 5 arestes. Aprofitant aquest fet, podem definir una kernelització (no polinòmica) pel Planar Independent Set definit com $K(G, k) = (G - v - veins(v), k - 1)$. On v és un d'aquests vèrtexs amb almenys 5 arestes.

Per acabar aquest apartat, voldria recalcar que tot problema FPT és kernelitzable.

Teorema 2.2.4 *Un problema parametritzable és* FPT *si i sols si és kernelitzable [11].*

2.3 Matrius Booleanes

Després de formalitzar les definicions generals de la complexitat parametritzada, ara ens centrarem en conceptes més propis del problema que volem tractar.

Durant tot el treball estarem parlant de problemes les entrades dels quals són matrius booleanes. Utilitzarem les següents notacions per referir-nos a elles i als elements que contenen:

Definició 2.3.1 *Direm que una matriu $n \times m$ és una matriu booleana si els $n \cdot m$ elements que la composen són tots booleans.*

Per referir-nos a l'element que resta en la fila i-èssima i la columna j-èssima utilitzarem indistintament les següents notacions:

- *Posició (i, j) de B.*
- $B[i, j]$.

També ens cal fixar la convenció gràfica que farem servir per tal que quedi clar de què parlem. La primera convenció que segur ens ve a tots al cap és la de representar les matrius com per files i columnes de 1 i 0.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

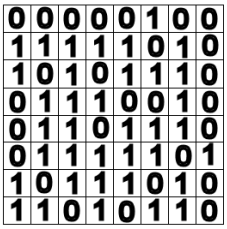

Figura 2.1: Representacions gràfiques naif

Escrites entre parèntesis o bé com a retícle, les anomenarem convencions naif per la seva simplicitat i la poca claredat que aporten. Per això mateix intentarem utilitzar-les el mínim possible.

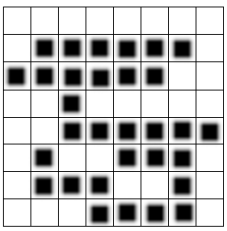

Figura 2.2: Representació gràfica marcant el recinte

En general ens interessa veure de forma clara la forma del contorn dels 1 i, per tan, buscarem alguna altre convenció que ens convingui més. Utilitzarem el conveni de representar els 1 com a caselles en negre i els 0 amb caselles en blanc. Aquesta convenció ens ajudarà moltíssim a l'hora de posar exemples amb els que il·lustrar les demostracions que acompanyen. D'aquest conveni en direm que marca el recinte.

A més, tot sovint, ignorarem que estem parlant de matrius i no en dibuixarem els seus límits, centrant-nos només en els uns que conté la matriu i el seu contorn dibuixarem

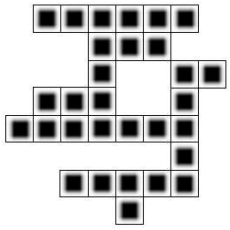

Figura 2.3: Representació gràfica mostrant només dels 1 de la matriu

tan sols els 1 i ignorarem els 0 que els acompanyen. Direm que aquest conveni només representa els 1 de la matriu.

Durant els dos propers capítols haurem de descriure rectnagles i polígons inscrits en les matrius booleanes, per fer-ho ens fa falta el concepte de coordenada dins les matrius per poder parlar dels seus vèrtexs amb propietat.

Entendrem les matrius $n \times m$, quan ens convingui, com a cel·les d'una graella amb $n+1 \times m+1$ nodes. Aquests nodes $[i, j]$, amb valors $0 \leq i \leq n$ i $0 \leq j \leq m$, seran els que usarem com a coordenades dins les matrius.

Definició 2.3.2 *Sigui B una matriu booleana de dimensions $n \times m$, definirem la seva coordenada $[i, j]$ com:*

- $[i, j]$ *serà el vèrtex inferior dret de la posició $B[i, j]$ si i sols si $i, j > 0$.*
- $[0, j]$ *serà el vèrtex superior dreta de la posició $B[1, j]$ si i sols si $j > 0$.*
- $[i, 0]$ *correspondrà al vèrtex inferior esquerra de la posició $B[i, 1]$ si i sols si $i > 0$.*
- $[0, 0]$ *serà el vèrtex superior esquerra de la posició $B[1, 1]$.*

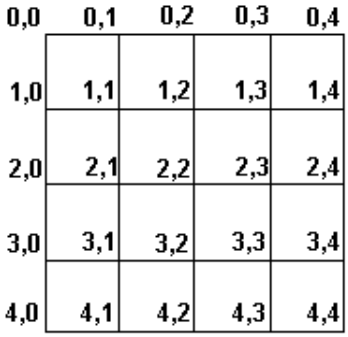

Figura 2.4: Exemple ús coordenades

Capítol 3

Solució utilitzant rectangles

En aquest capítol volem demostrar que el problema Repico pertany a FPT. El que volem és cobrir tots els 1 d'una matriu booleana amb el mínim nombre de rectangles possible. Aquest problema d'optimalitat el reduim a un problema decissional preguntant si podem cobrir-los amb un nombre concret de rectangles, tal i com hem avançat en els preliminars. Per ser precisos definim primer que entenem per un recobriment amb rectangles:

Definició 3.0.3 *Donada una matriu booleana B de dimensions $n \times m$. Representarem un rectangle en B com $r = \{[a,b],[c,d]\}$, amb $0 \leq a \leq c \leq n$ i $0 \leq b \leq d \leq n$ (on $[a,b]$ i $[c,d]$ són els vèrtexs superior esquerre i inferior dret del rectangle). És a dir, el rectangle r contindrà totes les posicions $\{B[i,j] \mid a < i \leq c \wedge b < j \leq d \wedge a \leq c \wedge b \leq d\}$.*

Direm que un conjunt de rectangles $R = \bigcup_{n=1}^{n=k} r_n$ és un recobriment dels 1 de la matriu de mida k, si i sols si es compleix que $B[i,j] = 1$ per tot element $B[i,j] \in r_n$ i $B[i,j] = 0$ per a tot $B[i,j] \notin r_n$.

La complicació d'aquest problema rau en que no podem assegurar que un rectangle concret formi part o no d'una solució òptima sense comprovar la optimalitat de la solució en el seu conjunt. Com que donat un pretendent a solució, comprovar si és o no ho és ens costa temps P, podem assegurar que el problema pertany a NP [15].

La definició formal del problema, que hem introduït en els preliminars, és la següent:

Rectiliniar Picture Compression (Repico)
Instance: Una matriu booleana $n \times n$ anomenada B i un enter positiu k
Parameter: k
Question: Existeixen k rectangles que cobreixin tots els uns de B?

En la figura 3.1 podem veure un primer exemple d'una matriu booleana coberta per el mínim nombre de rectangles (aquí recoberts per el·lipses de colors per major claredat).

Els rectangles que formen el recobriment són els següents: $R = \{\{[0,1],[2,2]\},\{[0,4],[2,5]\},\{[1,1],[2,8]\},\{[1,2],[6,3]\},\{[3,1],[4,3]\},\{[3,5],[6,6]\},\{[4,5],[6,7]\},\{[5,2],[6,7]\}\}$.

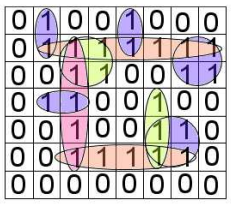

Figura 3.1: Exemple recobriment d'una matriu mitjançant rectangles

3.1 Límit superior

Les regles de reducció que hem utilitzat per aquest cas són molt senzilles: sempre que ens trobem dues files o columnes identiques dins de la matriu n'eliminem una de les dues. Utilitzant aquestes regles de reducció hem estat capaços de trobar un algorisme que redueix el problema a un kernel que només depèn de k i que podem, llavors, resoldre amb un algorisme de temps exponencial en k.

Comencem definint el concepte de matriu reduïda en el que hem basat la kernelització:

Definició 3.1.1 *Sigui B una matriu booleana $n \times n$. Direm que B és una matriu reduïda si, i sols si, no inclou dues files o columnes consecutives idèntiques. És a dir, si $i = j+1$, llavors $\exists k \mid B[i,k] \neq B[j,k]$. Equivalentment per les columnes.*

El següent lema ens permet demostrar que aquestes reduccions es poden utilitzar per kernelitzar el problema.

Lema 3.1.2 *Sigui β la matriu reduïda d'una matriu booleana B. Llavors, β es pot cobrir utilitzant k rectangles si i sols si B també.*

Demostració Donat que al reduir, mantenim els canvis del contorn l'únic que canviarà són les mides dels rectangles i no pas el nombre de rectangles que necessitem.

Els únics rectnagles que poden desapareixer són, en solucions no òptimes, els rectangles redundants que poden ser afegits en la matriu reduïda superposats a qualsevol altre rectangle.

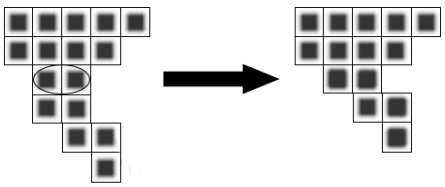

Figura 3.2: Exemple de reducció

Voldria destacar que al eliminar files (o columnes) repetides, només n'esborrem les repeticions però sempre en deixem un representant i això impedeix que hi hagi diferències en la mida dels recobriments de B i β.

□

Donada una matriu B de dimensions $l \times l$, reduir-la fins obtenir una matriu β de dimensions $n \times m$ (amb $n \leq l \wedge m \leq l$) consisteix en comparar les files de dues en dues i eliminar-ne les repetides (igualment amb les columnes). Aquesta tasca es pot resoldre comparant n files i m columnes i sempre que trobem dues files (o columnes) iguals eliminar-ne una.

El cost de comparar dues files depèn fortament del model de computació que fem servir. Com que cal fer una O exclusiva de dos valors binaris de mida $n' = max\{n, m\}$, en principi, el cost és $\mathcal{O}(n')$.

Hem de fer n' comparacions de cost $\mathcal{O}(n')$, ens queda una reducció de cost $\mathcal{O}((n')^2)$.

En la figura 3.2 podem observar un exemple de matriu booleana abans i després de la reducció. Cal destacar que després de reduir una matriu podem assegurar que en cada fila i columna, com a mínim hi ha un canvi respecte l'anterior (perquè si fossin iguals n'hauríem eliminat una de les dues).

Les matrius reduïdes ens interessen especialment perquè les seves dimensions són tan petites com ens es possible mantenint els canvis que hi pot haver en el contorn.

El primer que volem observar és que les mides de la matriu reduïda i el nombre de rectangles que formen un recobriment mínim estan relacionats. Fonamentarem aquest resultat en un lema que ens assegura que afegint tants 1 com vulguem a una matriu reduïda de manera que només calgui un rectangle addicional per cobrir-los, les dimensions de la matriu reduïda no poden augmentar en més de dos unitats:

Teorema 3.1.3 *Sigui B una matriu booleana de dimensions $n \times n$ que es pot recobrir amb k o menys rectangles, llavors la matriu reduïda de B, β, té com a màxim $2k+1$ files i $2k+1$ columnes.*

Demostració La demostració la farem per inducció sobre el nombre de rectangles k.

Cas $k = 1$: Trivialment, si B es pot cobrir amb un sol rectangle, llavors β és una matriu com a molt 3×3 (Veure figura 3.3).

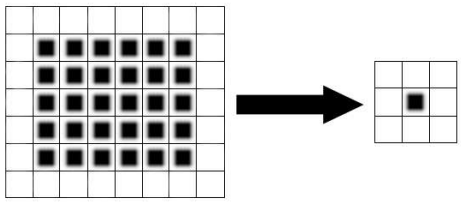

Figura 3.3: Cas $k = 1$

Cas $k + 1$: Sigui B una matriu booleana que es pot recobrir $k+1$ rectangles, sigui β la seva matriu reduïda. Sigui R un recobriment de mida $k + 1$ de B. Triem $\nu \in R$, un rectangle qualsevol i considerem B' la matriu en la que hem transformat en 0 els 1 que sols pertanyien a ν:

- $B'[i, j] = B[i, j]\ \forall (i, j) \notin R$
- $B'[i, j] = B[i, j]\ \forall (i, j) \in R \setminus \nu$
- $B'[i, j] = 0\ \ \forall (i, j) \in \nu \wedge (i, j) \notin \delta, \quad \forall \delta \neq \nu \in R$

Per construcció, B' es pot recobrir utilitzant k rectangles aprofitant com a recobriment $R' = R \backslash \nu$.

Per hipòtesi d'inducció, β', la matriu reduïda de B' té com a màxim unes dimensions de $2k + 1 \times 2k + 1$.

Estudiant en paral·lel el procés de reducció per les matrius B i B' completarem la demostració. Per simetria ens fixarem només en les reduccions de les files podent-se reproduir el mateix raonament per les columnes.

Totes les files que no conten ν són exactament iguals en B que en B' i per tan el procés de reducció de totes aquestes files serà exactament el mateix. Al ser idèntica la reducció en aquestes files, si β ha d'acabar tenint més files que β ha de ser fruit de les files que conten ν que no s'han de poder reduir tan.

Dividirem ν en tres parts que estudiarem per separat:

- La primera fila en la que apareix ν: en aquesta fila hi ha un diferència substancial entre B i B', estem afegint un rectangle que no hi era i podem estar provocant un canvi en B que faci que aquesta fila no es pugui reduir (no sabem si en B' es podia reduir o no però en el pitjor dels casos ara tenim una fila que abans es reduïa i ara no).

  Aquesta diferència entre B i B' ens permet assegurar que β pot necessitar una fila més que β' però només una d'addicional. O millor dit només una de provocada per la fila en la que comença ν.

- Les files que corresponen a l'interior de ν: és un cas molt curiós perquè en la porció d'aquestes files que no inclouen ν és clar que no hi ha més canvis que els que hi havia abans d'afegir ν. Però, a més, en les columnes de la fila en la que tenim 1 de ν, precisament per ser interior a ν, ens trobem que almenys en les columnes anterior i posterior també hi ha 1. És a dir, en les files interiors no hi ha més canvis en B dels que hi havia en B' perquè ν no n'hi afegeix cap. Per tan es podran reduir d'igual manera.

- La fila en que acaba ν: de la mateixa manera que la primera fila de ν podia afegir un canvi addicional, ja que pot afegir alguna diferència més (és clavat al primer cas intercanviant el paper que fan els 1 i els 0). Per tan, per culpa d'aquesta fila, β podria ser una fila més ampla que β'.

Resumint β pot tenir dues files més que β': les dues úniques files en les que pot afegir canvis (en la que comença i en la que acaba). Com β' té, com a molt, $2k+1$ files, β en té com a molt $2k+1+2 = 2(k+1)+1$ que precisament ens demostra el pas d'inducció.

□

La possibilitat de reduir la matriu fins aconseguir una matriu que tan sols ens representi els canvis que hi ha entre les files i columnes és una de les dues idees que ens ajuda a kernelitzar el problema. Ja que, combinant la reducció de matrius amb el resultat del teorema anterior podem assegurar un kernel de mida $2k+1$. Vegem-ne l'algorisme resultant:

Input: B una matriu $n \times n$ i k un número enter
Output: Una matriu β de com a molt $2k+1 \times 2k+1$ o bé la resposta definitiva que amb k rectangles no en fem prou per cobrir els 1 de B

1. Obtenim β la matriu reduïda de B
2. **If** (β té més de $2k+1$ files o columnes)$\rightarrow$ llavors **return** NO
3. **Else If** $\rightarrow$ **return** β

Després d'aplicar l'algorisme obtenim, o bé la certesa que no podem cobrir-ho amb k rectangles, o bé un problema, les dimensions del qual depenen exclusivament de k. Per resoldre el problema que només depèn de k podem usar un algorisme molt senzill però exponencial en la mida de la matriu:

Input: B una matriu $n \times m$
Output: R recobriment dels 1 de B
1. **While:** $(Acabat \wedge Trobat)$ **Do:**
2. (R,Acabat):=Generar nou recobriment()
3. Trobat=Comprovar(T)
4. **If** $\rightarrow$ **return** T

On *Generar nou recobriment* ens ha d'anar generant seqüencialment cada una de les possibles combinacions que tenim de recobriments. Cada recobriment està format per k rectangles i cada rectangle necessita de 4 valors entre 0 i $n' = max\{n, m\}$ per definir-se (recordem que representàvem els rectangles com a $\{[a, b], [c, d]\}$)donant-nos un total de $(4k)^{n'}$ combinacions possibles de les quals podem eliminar-ne la meitat que corresponen a els rectangles malformats (en els que $a > c$ o bé $b > d$) quedant-nos amb $(2k)^{n'}$.

Comprovar és una funció que donat un recobriment R comprova que tots els 1 de la matriu estiguin coberts per algun rectangle de R i que cap d'aquests rectangles cobreixi un 0. El cost d'aquest algorisme és polinòmic en les mides de la matriu perquè sols cal comprovar cada posició de la matriu (que són $n \cdot m$). Per tan, en el pitjor dels casos, haurem de fer tantes comprovacions com recobriments puguin existir i en conseqüència obtenim un algorisme $\mathcal{O}((2k)^{n'})$.

Com que, en el nostre cas, generarem recobriments de matrius quadrades de com a molt $2k + 1 \times 2k + 1$ obtenim un algorisme de cost $\mathcal{O}((2k)^{2k+1})$ que és exponencial en k i per tan ens demostra que Repico$\in$ FPT.

3.2 Límit inferior

Continuarem ara amb un resultat complementari al mostrat per demostrar usant tècniques combinatòries que qualsevol matriu que necessiti d'almenys k rectangles per a ser coberta, té com a mínim unes dimensions de $\sqrt{2k} \times \sqrt{2k}$.

Teorema 3.2.1 *El màxim nombre de rectangles que podem necessitar per cobrir els 1 d'una matriu B de dimensions $n \times n$, és $\lceil \frac{n^2}{2} \rceil$ i s'assoleix quan els 1 de la matriu segueixen la distribució de les caselles negres d'un taulell d'escacs.*

Demostració La farem per inducció sobre n la mida de la matriu B

Figura 3.4: Cas $n = 1$

Cas $n = 1$**:** Trivialment, si B és una matriu 1×1, com a molt pot contenir un únic 1 que pot ser cobert amb un sol rectangle.

Cas $n + 1$**:** Dividirem la matriu en dues parts, una formada per les primeres n files i columnes (que anomenarem B') i l'altre per la fila $n + 1$ i la columna $n + 1$.

Per hipòtesi d'inducció, B', pot contenir com a molt $\lceil \frac{n^2}{2} \rceil$ rectangles distribuïts talment com un taulell d'escacs. Veiem ara, quants rectangles addicionals podem afegir a la matriu B en la última fila o columna.

Qualsevol rectangle addicional, κ que vulguem afegir als de B' tindrà com a mínim un 1 en la fila $n + 1$ o en la columna $n + 1$ (perquè sinó trencaríem l'esquema de taulell d'escacs en B' i contradiríem la hipòtesi d'inducció). Si κ té almenys un 1 en la franja $B \backslash B'$ no pot contenir al mateix temps un altre $1 \in B'$ perquè aquest nou rectangle no estaria augmentant el nombre total de rectangles donat que podríem ampliar el rectangle λ que cobria l'$1 \in B$ fins que $\lambda = \kappa$.

Per motius de densitat, tampoc podem canviar un $0 \in B'$ per un 1 i considerar que el rectangle que estem afegint, κ, pot contenir un dels antics $0 \in B'$ i un $1 \in B \backslash B'$ perquè podríem ampliar qualsevol dels rectangles que cobreixen algun dels uns veïns d'aquest zero fins que cobris dos uns del taulell B' utilitzant el zero com a pont. D'aquesta manera tampoc ampliaríem el nombre total de rectangles.

Aquestes dues indicacions ens permeten ampliar el número de rectangles només afegint 1 en la franja $B \backslash B'$. És clar, que els rectangles que afegim en aquesta franja han d'estar formats per un únic 1 perquè si els fem més grossos, al no poder superposar-los, ens allunyarem del màxim.

Si ajuntem aquestes condicions ens trobem que tan sols podem ampliar el nombre de rectangles de B' ampliant l'esquema de taulell d'escacs de B a B'.

El número d'uns que caven a la franja $B' \backslash B$ és $n - 1$ si n és parell i n si és senar. Però en qualsevol cas és menor o igual que n.

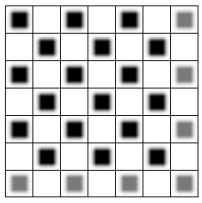

Figura 3.5: Cas $n+1$

Donat que en B' caven $\lceil \frac{n^2}{2} \rceil$ i que en $B \backslash B'$ en podem afegir com a molt n. En total obtenim que en B queven:$\lceil \frac{n^2}{2} \rceil + n \leq \lceil \frac{n^2+2n}{2} \rceil \leq \lceil \frac{(n+1)^2}{2} \rceil$

Que és exactament el que volíem demostrar.

□

Vista la demostració pel cas en el que la matriu és quadrada queda palès que el fet que sigui quadrada no hi té cap més repercussió que simplificar la demostració. Així doncs, el següent corol·lari és immediat a partir del teorema.

Corol·lari 3.2.2 *El màxim nombre de rectangles que podem necessitar per cobrir els uns d'una matriu B de dimensions $n \times m$, és $\lceil \frac{n \cdot m}{2} \rceil$ i s'assoleix si extenem la configuració d'un taulell d'escacs a matrius no quadrades.*

Com a conseqüència directa dels teoremes 3.1.3 i 3.2.1 obtenim el següent corol·lari:

Corol·lari 3.2.3 *Si una matriu booleana B es pot recobrir utilitzant k rectangles, llavors β, la seva matriu reduïda, és una matriu $n \times n$ on $\sqrt{2k} \leq n \leq 2k+1$.*

A més, tal i com hem demostrat en els respectius teoremes, les dues fites s'assoleixen i per tan no es poden millorar.

Capítol 4

Solució utilitzant arestes

Després de classificar el problema de cobrir els 1 d'una matriu booleana utilitzant rectangles, ens vam adonar que, probablement podríem mantenir la informació referent a on eren els 1 de la matriu d'una altra manera.

Si volem emmagatzemar les posicions dels 1 de la matriu i utilitzem rectangles per cobrir-los, guardant informació sobre la posició de dos vèrtexs diametralment oposats de cada rectangle ja en tenim prou per localitzar tots els rectangles i per tan per localitzar tots els 1 de la matriu.

Però, i si en comptes de cobrir els 1 amb rectangles utilitzem les arestes que envolten els conjunts d'1 de la matriu? Ens permet resoldre el problema de descriure les matrius? És millor o pitjor que la solució utilitzant rectangles?

Per tal de respondre aquestes preguntes, ens cal definir formalment que vol dir envoltar els 1 amb arestes, plantejar el nou problema i estudiar-ne la seva complexitat. Definirem primer que és un polígon rectilini, per després definir com seran les seves envolupants:

Definició 4.0.4 *Un polígon rectilini és un polígon amb els seus costats paral·lels als eixos de coordenades. Pot tenir forats o no, però, cas que els tingui, aquests seran també rectilinis.*

A partir d'aquest punt entendrem que tots els polígons dels que parlem són polígons rectilinis simples (simples en el sentit que dues arestes no poden tenir cap punt en comú fora dels seus vèrtexs) [16].

Definició 4.0.5 *Donat el polígon p, definirem l'envolupant de p com la seqüència dels seus vèrtexs de tal manera que els vèrtexs del polígon apareixen en sentit horari i els dels seus forats en sentit antihorari,* $S_p = [[a_1, b_1], [a_n, b_n]]$ *[16].*

Si tenim una col·lecció de polígons $P = [p_1, ..., p_m]$*, l'envolupant de la col·lecció serà* $S_P = [S_{p_1}, ..., S_{p_m}]$.

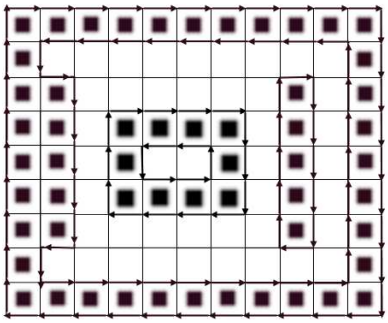

Figura 4.1: Exemple envolupant

Remarcar només que els 1 d'una matriu booleana formaran un conjunt de polígons i que parlarem de l'envolupant dels 1 de la matriu o de l'envolupant de la matriu com l'envolupant dels polígons formats pels 1 de la matriu.

En la figura 4.1, podem veure una matriu booleana i quin aspecte tindria la seva envolupant. En concret, l'envolupant, serà: S=[[[3,3],[3,7],[6,7],[6,3],[4,4],[5,4],[5,6],[4,6]], [[0,0],[0,11],[9,11],[9,0],[1,1],[2,1],[2,2],[7,2],[7,1],[8,1],[8,10],[1,10]], [[2,8],[2,9],[7,9],[7,8]]]

El problema de l'envolupant l'enunciarem com apareix a continuació:

Poligonal Picture Hull Cover (Popico)
Instance: Una matriu booleana $n \times n$ que anomenarem B i un enter positiu k
Question: Podem trobar una seqüència S de vèrtexs amb k vèrtexs que sigui una envolupant de B?
Parameter: k

Aquest problema és NP complet [16] i, tal i com veurem a continuació, pertany al conjunt de problemes FPT. Per demostrar-ho començarem estudiant les fites de les dimensions de la matriu segons el nombre de vèrtexs necessaris per construir una envolupant dels 1 de la matriu.

4.1 Límit superior

Tractarem primer el límit superior, és a dir, estudiarem quines són les dimensions màximes de les matrius que es poden envoltar usant sols k vèrtexs. Aquest anàlisi ens portarà fins una reducció que ens permetrà kernelitzar el problema d'una manera semblant al Repico.

La clau d'aquesta reducció rau en entendre quants vèrtexs són necessaris per poder resseguir els canvis que es donen en la matriu. Comencem definint que volem dir quan ens referim als canvis que hi ha en una matriu:

Definició 4.1.1 *Definim un* canvi horitzontal *de la matriu booleana B com qualsevol posició de B, (i, j) tal que: o bé $i = 0$, o bé $B[i, j] \neq B[i - 1, j]$.*

Definició 4.1.2 *Definim un* canvi vertical *de la matriu booleana B com qualsevol posició de B, (i, j) tal que: o bé $j = 0$, o bé $B[i, j] \neq B[i, j - 1]$.*

Fixem-nos que definim com a canvis tota la primera fila i tota la primera columna de qualsevol matriu. Ho definim així perquè per la resta de casos fixem el criteri d'igualtat segons la comparació amb la fila (o columna) anterior. Com la primera fila (o columna) no té anterior cal definir-la com a cas base i ens convindrà considerar-la com a *diferent* per tal que no sigui reduïda posteriorment.

Utilitzarem aquesta definició per mantenir un cert control sobre la dificultat d'envoltar els 1 amb polígons ja que, com més canvis hi hagi, més vèrtexs seran necessaris. Aquesta idea intuïtiva la formalitzem mitjançant el següent lema:

Lema 4.1.3 *Sigui B una matriu booleana $n \times m$. Per cada columna $j \neq 1$, que contingui com a mínim un canvi horitzontal, es necessiten almenys dos vèrtexs per descriure el canvi produït.*

Demostració La demostració la farem amb l'ajut de les següents representacions gràfiques donat que així és molt més aclaridora. Podem trobar-nos amb 3 casos: que apareguin 1 on hi ha havia 0, que apareguin 0 on hi havia 1 o bé que es donin els dos casos anteriors a l'hora.

Els dos primers casos tenen en comú que només generen un canvi, en la figura 4.2 hem representat aquests dos casos mitjançant dos exemples en els que apareixen o desapareixen 1 respectivament.

Mentre que en la figura 4.3, es representen les tres possibilitats que en una columna hi hagi dos canvis:

- Que apareguin o desapareguin dos 1 (representades en el primer exemple).

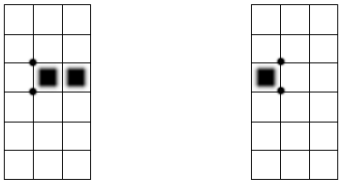

Figura 4.2: Amb un sol canvi

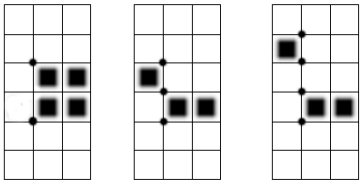

Figura 4.3: Amb dos canvis

- Que aparegui un 1 i en desaparegui un altre i que ambdós canvis es produeixin en files contigües (el segon exemple).

- Que aparegui un 1 i en desaparegui un altre en files no contigües (en el tercer exemple mostrat de la figura 4.3).

No podem tenir en compte la primera columna de la matriu perquè hem definit de manera arbitraria que tingui canvis. Com que ho estem forçant artificialment, es pot donar el cas en el que ni tan sols hi hagi cap 1 per envoltar en aquesta columna i, per tan, és obvi que no caldria cap vèrtex.

□

Presentem com a corol·lari el lema simètric per les files de la matriu:

Corol·lari 4.1.4 *Sigui B una matriu booleana $n \times m$. Per cada fila $i \neq 1$ que contingui com a mínim un canvi horitzontal, es necessiten almenys dos vèrtexs per descriure el canvi produït.*

Altra vegada, el procés que seguirem serà primer reduirem la matriu eliminant les files i columnes en les que no hi hagi canvis. A continuació, següent lema ens assegurarà que al fer la reducció no podem ni augmentar ni reduir la quantitat de vèrtexs que formen l'envolupant:

Lema 4.1.5 *Una matriu booleana B té una envolupant S amb k vèrtexs si i sols si la seva matriu reduïda β té una envolupant ζ amb k vèrtexs.*

Demostració En el procés de reducció sols eliminem files i columnes en les que no hi ha canvis. Per definició, si no hi ha canvis no poden contenir vèrtexs del polígon (la reducció no afecta el contorn del polígon, només pot arribar a escurçar la longitud d'algunes de les seves arestes). Per tan, al reduir no poden aparèixer ni desaparèixer vèrtexs.

□

Aquest lema ens permetrà kernelitzar el problema ja que és una reducció que no fa augmentar el paràmetre utilitzat i, gràcies al lema 4.1.3 aconseguirem que la part exponencial depengui només del paràmetre:

Lema 4.1.6 *Tota matriu booleana β de dimensions $n \times m$ reduïda, necessita un envolupant S amb almenys $2 \cdot max\{n, m\} - 2$ vèrtexs per ser descrita.*

Demostració La demostració és senzilla i es basa en la definició de matriu reduïda. Per ser β una matriu reduïda, en cada fila (o columna) hi ha com a mínim un canvi. Aquest fet ens assegura que tenim n files (i m columnes) en les que hi ha canvis i pel lema 4.1.3 necessitem almenys 2 vèrtexs per cada canvi. Així doncs necessitem $2 \cdot max\{n, m\} - 2$ vèrtexs com a mínim.

Fixem-nos que el -2 prové dels dos canvis que estem afegint de forma artificial en la primera fila i columna. Així doncs, en el cas pitjor, aquests canvis no només seran artificials sinó que resultaran falsos i no els podem contar al cercar una fita inferior.

□

Teorema 4.1.7 *A tota matriu booleana B, l'envolupant de la qual tingui k vèrtexs, li correspon una matriu reduïda β amb unes dimensions de com a molt $\frac{k}{2} + 1 \times \frac{k}{2} + 1$.*

Demostració Aquest teorema resulta una conseqüència immediata del lema 4.1.6

□

Un cop més, aprofitarem el fet de poder reduir la matriu fins aconseguir una matriu que tan sols ens presenti els canvis que hi ha entre les files i columnes de la matriu original. Com que la matriu reduïda necessitarà almenys el doble de vèrtexs que el seu número de files (o columnes), si la matriu reduïda resultant és massa gran podrem assegurar que no es pot envoltar. Obtenint així un kernel de mida $\frac{k}{2}+1$. Vegem-ne l'algorisme:

Input: B una matriu $n \times n$ i k un número enter
Output: Una matriu β de dimensions, com a molt grans de $\frac{k}{2}+1 \times \frac{k}{2}+1$ o bé la resposta definitiva que amb k vèrtexs no en fem prou per envoltar els uns de B
1. Obtenim β la matriu reduïda de B
2. **If** (β té més de $\frac{k}{2}+1$ files o columnes)$\rightarrow$ llavors **return** NO
3. **Else If** $\rightarrow$ **return** β

Fixem-nos que en haver aplicat l'algorisme obtenim o bé una resposta negativa (si la matriu reduïda és massa gran no hi haurà manera d'envoltar-la utilitzant sols amb k vèrtexs perquè hi haurà massa canvis); o bé obtindrem una matriu reduïda β de dimensions $\frac{k}{2}+1 \times \frac{k}{2}+1$ que tan sols depenen del paràmetre k.

Al problema reduït, li podem aplicar un algorisme de cerca exhaustiva molt semblant al descrit pel problema Repico (lema 3.1.3).

Ara bé, *Generar nou recobriment* ens haurà de donar com a resultat k vèrtexs que estan formats per dos números entre 0 i $\frac{k}{2}+1$ (les seves coordenades) per un total de $(2k)^{\frac{k}{2}+1}$ combinacions possibles.

Comprovar serà un xic més complexa perquè per convèncer-nos que tots els 1 de la matriu resten envoltats al mateix temps que els 0 queden fora d'una envolupant que pot quedar molt embolicada. A grans trets, l'algorisme passa per marcar tots els 1 que queden a la dreta de les arestes formades per vèrtexs consecutius de l'envolupant. Aquest algorisme es pot resoldre en temps polinòmic [16] deixant-nos en total un algorisme amb cost $\mathcal{O}((2k)^{\frac{k}{2}+1})$.

Com que la reducció del kernel es pot computar en temps polinòmic, ajuntant els dos algorismes obtenim una solució FPT del problema Popico.

4.2 Límit inferior

De la mateixa manera que ho vam fer amb el cas del recobriment de rectangles, volem ara demostrar que també tenim un límit inferior en les dimensions de les matrius booleanes si fixem el nombre de vèrtexs que podem utilitzar per envoltar els 1 de la matriu.

Teorema 4.2.1 *El màxim nombre de vèrtexs que podem necessitar per envoltar els 1 d'una matriu B de dimensions $n \times n$, és $4\lceil \frac{n^2}{2} \rceil$ i s'assoleix quan els 1 de la matriu segueixen la distribució de les caselles negres d'un taulell d'escacs.*

Demostració La demostració és un corol·lari de la demostració feta per rectangles si tenim present que cada rectangle pot necessitar com a molt 4 vèrtexs per envoltar aquest rectangle (pròpiament els quatre vèrtex del rectangle).

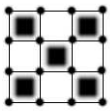

Figura 4.4: Necessitem 20 vèrtexs per descriure aquesta envolupant

Com que el nombre màxim de rectangles que podem inscriure en una matriu $n \times n$ és $\lceil \frac{n^2}{2} \rceil$, si demostrem que no ens podem estalviar cap dels vèrtexs d'aquests rectangles haurem demostrat que és el màxim número de vèrtexs que podem necessitar (qualsevol altra configuració tindria o bé menys rectangles o bé una ràtio de vèrtexs per rectangle menor).

L'únic lloc on podríem estalviar-nos algun vèrtexs és en els casos en que trobem dos rectangles en diagonal que comparteixen un vèrtex però no un costat (no tenim rectangles que comparteixin costats). Per la definició que tenim de polígon, ambdós rectangles són polígons i per la definició que tenim d'envolupant aquest tipus de vèrtexs els haurem de contar dues vegades perquè són vèrtexs als que arriben quatre arestes i que per tan s'hauran de recórrer dues vegades (una en resseguir el quadrat superior i una en resseguir el quadrat inferior).

Així doncs, cada rectangle necessitarà que els seus quatre vèrtexs formin part de l'envolupant encara que aquests siguin compartits amb els d'algun altre rectangle donant-nos com a resultat el límit proposat al lema.

□

Com a conseqüència directa del lema obtenim el següent corol·lari:

Corol·lari 4.2.2 *Si tenim una matriu booleana β reduida $n \times n$ l'envolupant de la qual necessita k vèrtexs , llavors $n \geq \sqrt{\frac{k}{2}}$.*

Ajuntant els límits demostrats en els teoremes 4.1.7 i 4.2.1, obtenim les dues cotes que ens permeten enunciar el següent corol·lari:

Corol·lari 4.2.3 *Una matriu booleana β reduïda $n \times n$, l'envolupant de la qual està formada per k vèrtexs, llavors* $\sqrt{\frac{k}{2}} \leq n \leq \frac{k}{2} + 1$.

Capítol 5

El cas on permetem errors

Havent estudiat els dos problemes dels capítols 3 i 4 vam pensar en què passaria si no fóssim tan estrictes amb el concepte de solució i acceptéssim solucions aproximades. En concret, ens permetrem el luxe de recobrir o envoltar algun 0 barrejat entre els 1 que formen la solució.

El capítol resta dividit en dues seccions que tracten el problema de recobriment amb rectangles i el de l'envolupant amb poligonals tancades respectivament.

5.1 Utilitzant Rectangles

En aquesta secció ens ocupem del problema de recobriment amb rectangles, en aquest cas acceptant solucions que cometin algun error. Entenent com errors només els 0 classificats com a 1, però en cap cas ens conformarem amb solucions en les que algun 1 hagi estat classificat com a 0.

Després d'aquestes consideracions podem definir formalment el problema:

Rectiliniar Picture Compression with errors
Instance: Una matriu booleana B, $n \times n$ i un enter positiu. k
Parameters: k i s
Question: Existeixen k rectangles que cobreixin tots els 1 i com a molt s 0 de B?

Admetem doncs, s errors, s 0 inclosos dins d'algun dels rectangles que formen la solució.

Per tal d'estudiar la complexitat d'aquest problema, ens hem preguntat si, havent resolt el problema sense errors, podíem convertir un 0 de la matriu en un 1 de manera prou intel·ligent com per necessitar menys rectangles. El resultat d'aquest plantejament l'enunciem en forma de lema:

Lema 5.1.1 *Sigui B una matriu booleana que es pot recobrir amb k rectangles i no amb menys. Si convertim s 0 de B en 1, necessitarem almenys $k' = max\{k - 3s, 0\}$ rectangles per cobrir tots els 1 de B.*

Demostració

Cal que el recobriment sigui òptim perquè quan eliminem un rectangle, si aquest rectangle simplement estigués repetit, podríem eliminar el mateix rectangle varies vegades i això ens impediria donar una fita assolible pels rectangles que ens estalviem.

Observem que canviar un 0 per un 1 no tindrà cap influencia sobre els rectangles que no estaven en contacte amb el 0 commutat. Aquest fet és trivial perquè el nou 1 no pot afectar en absolut la forma dels rectangles que no estan en contacte amb ell.

Voldria destacar també que l'aparició d'un 1 no pot fer desaparèixer cap rectangle, en tot cas, pot eliminar la diferència entre dos rectangles o més que passin a convertir-se en un de sol.

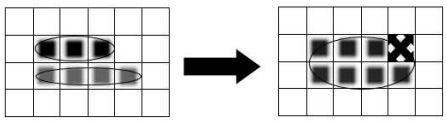

Figura 5.1: Exemple de fusió de dos rectangles en un

De tots els rectangles que aconseguim ajuntar en un de sol commutant un 0, no n'hi pot haver dos en el mateix costat. Fixem-nos que, si tenim dos rectangles que, per exemple, estan per sobre del 0 que volem commutar, han de diferir com a mínim en una posició per raons d'optimalitat. Aquesta diferencia que els distancia no pot quedar solventada de cap manera commutant un 0 que ambdós rectangles tenen per sota seu (tal i com podem observar en l'exemple 5.2 en el que la casella marcada amb una X no ens permet fusionar els dos rectangles, ens caldria commutar el valor de la casella marcada amb una estrella que té els rectangles en costats diferents).

El conjunt de les tres observacions anteriors ens diu que com a molt podem aconseguir ajuntar quatre rectangles en un de sol commutant un 0 (un per cada costat del 0). En la figura 5.3 podem observar que aquest màxim és realitzable mitjançant l'exemple mostrat.

Per cada 0 commutat, si tenim sort i convertim quatre rectangles en un de sol, ens faran falta 3 rectangles menys per cobrir els 1. Si ho aconseguim amb cada 0 que commutem, en tindrem prou amb $k - 3s$ rectangles per cobrir tots els 1 de B.

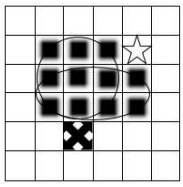

Figura 5.2: Exemple impossibilitat de fusionar dos rectangles al mateix costat del 0

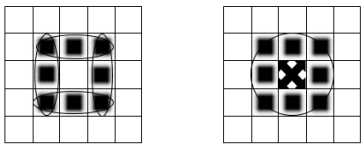

Figura 5.3: Exemple òptim

Els casos degenerats ens obliguen a afegir el màxim en la fórmula. Per exemple una matriu de 0 que no necessita cap rectangle per ser recoberta no pot reduir més el número de rectangles del seu recobriment.

□

Utilitzant aquest lema podem construir un kernel pel problema seguint el mateix esquema del capítol anterior:

Teorema 5.1.2 *El problema de recobrir matrius booleanes amb k rectangles acceptant s errors admet un kernel de mida $2(k + 3s) + 1$.*

Demostració Si podem cobrir una matriu amb k rectangles acceptant s errors, pel lema anterior (5.1.1) sabem que si no acceptéssim aquests errors necessitaríem com a molt $k + 3s$ rectangles per cobrir la matriu.

Dit això, convertim l'entrada pel problema en el que acceptem errors $[B, (k, s)]$ (on B és la matriu booleana, s el nombre d'errors que acceptem i k el número de rectangles que podem utilitzar), en una entrada pel que no n'acceptem $[B', k']$ fent la següent transformació:

- $B' = B$
- $k' = k + 3s$

Pel teorema 3.1.3, el problema $[B', k']$ accepta un kernel de mida $2k' + 1$ i per tan, el problema $[B, (k, s)]$ accepta un kernel de mida $2(k + 3s) + 1$.

□

5.2 Utilitzant Arestes

Analitzarem ara el problema d'envoltar els 1 de la matriu amb poligonals tancades. D'igual manera que en el cas en el que usàvem rectangles acceptarem alguns errors, errors que seran 0 classificats com a 1 però no al revés. Formalment descriurem el problema com:

Poligonal Picture Hull Cover with errors
Instance: Una matriu booleana $n \times n$ que anomenarem B i un enter positiu k
Parameters: k i s
Question: Podem trobar una seqüència S amb k vèrtexs que sigui una envolupant de B acceptant s errors?

La solució d'aquest problema és francament molt similar a la proposada utilitzant rectangles. Primer resolem el problema sense admetre errors i llavors estudiem si podem millorar la solució acceptant algun error. D'igual manera que amb els rectangles, ens preguntarem quants vèrtexs menys necessitem al convertir un 0 de la matriu en un 1.

Qualsevol 0 que hi hagi a la matriu, com a molt estarà en contacte amb quatre vèrtexs diferents de l'envolupant i per tant, si la solució sense errors necessita de k vèrtexs en l'envolupant, la solució amb un error en necessitarà almenys $k - 4$. Generalitzant, la solució admetent s errors necessitarà almenys $k - 4s$ vèrtexs.

Per tal de no deixar lloc a dubtes, voldria comentar un cas que tot i que no enterboleix aquest raonament podria semblar que necessita d'un estudi més detallat. Recordem que els vèrtexs que formen part de la solució poden ser dobles perquè poden ser recorreguts dues vegades. Aquests vèrtexs s'anomenen vèrtexs degenerats i, tal com mostra la primera matriu de la figura 5.5, l'envolupant podria necessitar passar dues vegades pel mateix vèrtex per tal d'envoltar aquesta forma (però no més de dues vegades).

La pregunta que ens ve al cap al fixar-nos en aquesta situació és si podria ser que canviant un únic 0 reduíssim el nombre de vèrtexs necessaris en 8 (els quatre vèrtexs del 0 que acceptem com error contats dues vegades perquè són vèrtexs degenerats)?

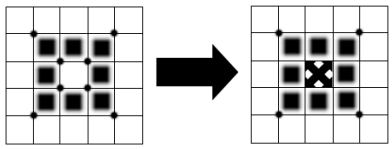

Figura 5.4: Exemple òptim

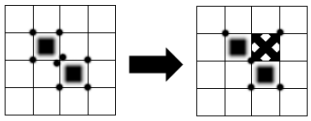

Figura 5.5: Exemple de conversió un vèrtex degenerat en un vèrtex simple

La resposta és que no, els vèrtexs degenerats, en el millor dels casos es poden arribar a convertir en vèrtexs simples perquè tot i que el grau d'incidència d'un vèrtex degenerat és quatre, tan sols dues d'aquestes arestes poden envoltar el 0 que convertirem en un 1. Així doncs, canviant el 0, com a molt eliminarem dos de les arestes incidents en el vèrtex degenerat convertint-lo en un vèrtex simple.

A continuació formalitzem aquest raonament a través d'un lema:

Lema 5.2.1 *Sigui B una matriu booleana que té una envolupant S de mida k i aquesta k és la mínima possible. Si convertim s 0 de B en 1, necessitarem almenys $max\{k-4s, 0\}$ vèrtexs per cobrir tots els 1 de B.*

A partir d'aquest lema podem construir un kernel pel problema seguint exactament el mateix esquema de la secció anterior:

Teorema 5.2.2 *El problema d'envoltar els 1 d'una matriu booleana utilitzant k vèrtexs acceptant s errors admet un kernel de mida $\frac{k+4s}{2} + 1$*

Demostració Si podem envoltar els 1 d'una matriu utilitzant k vertexs accepteant s errors, pel lemma (5.2.1) sabem que si no acceptéssim aquests errors necessitaríem com a molt $k + 4s$ vèrtexs en l'envolupant la matriu.

Dit això, convertim l'entrada pel problema en el que accpetem errors $[B, (k, s)]$ (on B és la matriu booleana, s el nombre d'errors que acceptem i k el número de rectangles que podem utilitzar), en una pel que no n'acceptem $[B', k']$ de la següent manera:

- $B' = B$
- $k' = k + 4s$

Pel teorema 4.1.7, el problema $[B', k']$ accepta un kernel de mida $\frac{k'}{2} + 1$ i per tan, el problema $[B, (k, s)]$ accepta un kernel de mida $\frac{k+4s}{2} + 1$.

□

Capítol 6

Relació entre el problema de recobriment i el d'envolupant

L'últim problema que abordarem és el d'analitzar comparativament les dues aproximacions proposades en els capítols 3 i 4 per descriure matrius booleanes. El criteri pel que ens regirem serà el d'espai, de l'espai en memòria que ocupa la descripció de la matriu.

Comencem amb una relació que apareix creuant les fites demostrades pels dos plantejaments proposats. Ja que, l'espai en memòria depèn directament del número de rectangles en el problema de REPICO i del número de vèrtexs en el POPICO

Lema 6.0.3 *Sigui B una matriu booleana de dimensions $n \times n$. Siguin α, el nombre de vèrtexs que formen l'envolupant dels uns de B i γ el nombre de rectangles que formen el recobriment de dels uns de B. Llavors es compleixen les següents relacions:*

a.) $\alpha \leq 4\lceil \frac{(2\gamma+1)^2}{2} \rceil$

b.) $\gamma \leq \lceil \frac{(\frac{\alpha}{2}+1)^2}{2} \rceil$

Demostració La demostració és conseqüència dels límits superiors i inferiors dels dos problemes.

a.) Comencem suposant que la matriu necessita de γ rectangles per ser recoberta. El teorema 3.1.3 ens assegura que la matriu reduïda de B, β, té unes dimensions no superiors a $2\gamma + 1 \times 2\gamma + 1$. El teorema 4.2.1 ens permet dir que el nombre de vèrtexs que podem necessitar com a màxim per envoltar una matriu amb aquestes dimensions és $4\lceil \frac{(2\gamma+1)^2}{2} \rceil$.

b.) L'altra desigualtat es demostra de manera similar. Si necessitem α vèrtexs en l'envolupant de B, el teorema 4.1.7 ens diu que la matriu reduïda de B, β, té com a molt $\frac{\alpha}{2} + 1 \times \frac{\alpha}{2} + 1$ elements. El teorema 3.2.1 ens permet afirmar que com a molt, en

β caven $\lceil \frac{(\frac{\alpha}{2}+1)^2}{2} \rceil$ rectangles. Per tant, el nombre de rectangles γ satisfà la desigualtat $\gamma \leq \lceil \frac{(\frac{\alpha}{2}+1)^2}{2} \rceil$ que ens demostra la segona desigualtat.

Ajuntant els dos resultats obtenim la demostració que estàvem buscant.

□

Ara bé, aquest resultat, tot i que natural a partir dels apartats anteriors, dista molt de ser útil. És una fita massa generosa per treure'n alguna conclusió. Nosaltres voldríem alguna fita que ens permetés assegurar quina de les dues descripcions de matrius booleanes ocuparà menys espai en memòria i les fites anteriors no ens ho permeten.

Arribats a aquest punt vam plantejar la següent conjectura $\gamma \leq \frac{\alpha}{2}$. Una fita fantàstica perquè decantaria la discussió a favor del recobriment de rectangles. Recordem que per cada rectangle necessitem guardar en memòria la posició de dos dels vèrtexs oposats del rectangle, així que si per guardar l'envolupant necessitéssim, com a mínim, dos punts per rectangle, ens sortiria sempre més barat utilitzar recobriments per rectangles.

Aquesta conjectura prové de l'estudi de matrius que tenen els 1 col·locats en forma d'escala. En la figura 6.1, podem comprovar que en una escala necessitem tants rectangles com esglaons té l'escala. Si considerem l'envolupant de la figura, necessitem 2 vèrtexs en cada esglaó més els dos vèrtexs de la base.

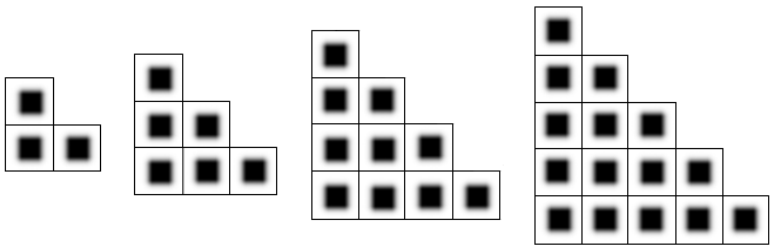

Figura 6.1: Matrius amb els 1 en escala

En un primer estadi vam estar considerant la següent desigualtat $\gamma \leq \frac{\alpha-1}{2}$ que prové del cas de les escales. Finalment vam desestimar el -1 quan vam trobar un exemple que satisfeia la igualtat $\gamma = \frac{\alpha}{2}$. Aquest exemple és el mateix que hem fet servir per demostrar els kernels quan acceptem errors, la figura 5.3. És un exemple en el que tenim 4 rectangles i 8 vèrtexs.

Fixada la conjectura, els primers intents per resoldre-la van ser infructuosos: Primer vam intentar atacs estructurals al problema basats en la hipòtesi que els 1 de la matriu

havien de tenir a la força un vèrtex convex (un vèrtex convex és aquell que només té 1 en una sola direcció, en la part dreta de la figura 6.2 la direcció sud-oest). En les primeres demostracions la idea era que al eliminar el rectangle al que pertanyia aquest vèrtex estàvem eliminant un vèrtex, si aconseguíem demostrar que qualsevol dels altres vèrtex del rectangle també desapareixia aconseguíem la ràtio de 2 a 1 que ens interessava.

Tal i com podem veure en la part esquerra de la figura 6.2, tot i que la hipòtesi sembla versemblant, el vèrtex en qüestió no té perquè desaparèixer a l'eliminar el rectangle perquè pot pertànyer a dos rectangles diferents.

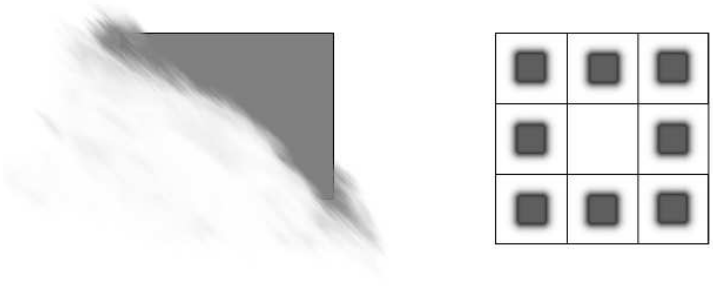

Figura 6.2: Exemples de vèrtexs convexos

El següent intent va ser una demostració per inducció en la que estudiàvem que passava quan eliminàvem un rectangle. Altra vegada, l'exemple dels kernels en que acceptem errors ens mostrava una de les dificultats més importants. Nosaltres volíem demostrar que a l'eliminar un rectangle desapareixien almenys dos vèrtexs, però en la figura 5.3, qualsevol rectangle que trèiem ens fa aparèixer dos vèrtexs!.

Tot i aquest impediment, encara vam provar de fer una inducció en la que el salt fos de dos en dos ja que donava la sensació que si eliminant un rectangle apareixien nous vèrtexs, això volia dir que, en un següent pas podriem eliminar més de dos vèrtexs. Amb aquesta idea en ment vam generar totes les possibles configuracions de matrius booleanes 2×2 (llevat de simetries) per demostrar el cas base (figura 6.3). Després vam desestimar la demostració pel ventall de possibilitats que trobàvem a l'eliminar dos rectangles arbitraris de la matriu on, a més, volíem demostrar que reduïen el nombre de vèrtexs necessaris.

També vam generar totes les configuracions possibles de matrius 3×3 i 4×4 per veure si això ens inspirava. No reprodueixo aquí aquestes configuracions perquè hi ha una gran quantitat de casos i finalment ens van aportar molt poc. A més, tal i com vam descobrir per casualitat, qualsevol exemple que resolgués la conjectura necessitava d'una matriu 5×5 per representar-se.

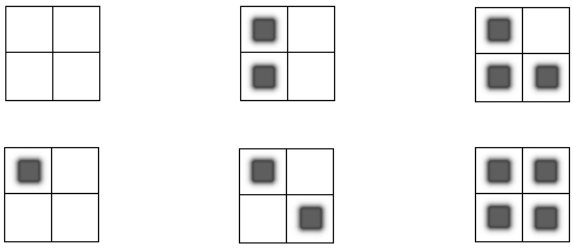

Figura 6.3: Cas base inducció

Malauradament, després de 15 dies intentant demostrar la conjectura, no només no la vam demostrar sinó que a més la vam refutar. En la figura 6.4 podem observar un contraexemple que necessita un recobriment de com a mínim 7 rectangles i tan sols 12 punts en la seva enolupant, clar exemple que $\gamma \leq \frac{\alpha}{2}$ no té perquè ser cert.

Amb aquest resultat negatiu acabem el capítol, esperàvem demostrar que un dels dos plantejaments proposats era millor que l'altre i hem acabat trobant exemples en que utilitzar rectangles ens proporciona solucions molt millors de la mateixa manera que hem trobat exemples en els que les solucions utilitzant envolupants ocupen molt menys lloc. Probablement, completar aquest capítol amb resultats que ens permetessin classificar en quines situacions és millor una aproximació o l'altre necessitaria de tot un altre treball de final de carrera.

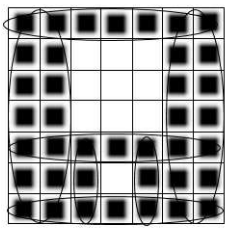

Figura 6.4: Contraexemple

Capítol 7

Balanç Econòmic

A l'hora d'escriure el balanç econòmic, com que ens trobem davant d'un treball teòric, no hem necessitat: ni maquetes, ni hores de càlcul de cap ordinador, ni materials. Les úniques despeses han estat: una vintena de bitllets de metro per acostar-me a la UPC, la gens despreciable matrícula del projecte i el cost d'impressió.

És a dir, l'únic a facturar són les hores de feina dedicades al treball. Sense cap ànim de menys valorar la feina feta per la Maria José Serna i en Dimitrios Thilikos, crec que el pes de la seva aportació sobre el total d'hores dedicades és prou petit per no tenir-lo en compte. És per això que em permetré el luxe de parlar en primera persona a partir d'aquí.

Han estat al voltant de 700 hores dedicades al projecte. D'aquestes: al voltant de 250 dedicades a documentar-me sobre el tema, unes 50 utilitzades en entrevistes amb els tutors del treball, 300 realitzant les demostracions a llapis i 100 invertides en escriure el projecte.

Com que ningú em pagarà aquestes hores de feina, presentaré 3 totals diferents basats en 3 preus per hora que descriuré a continuació:

- El primer preu per hora correspon al sou de becari de doctorat de la UPF. Trio aquest primera minuta perquè he estat becat a la Pompeu durant part del projecte i m'agradaria reflectir d'alguna manera en el projecte el, al meu entendre, mal tracte que reben els becaris allà i a tot arreu. El import d'aquesta beca és de 5.5 euros/hora. Obtenim doncs un total de 3 850 euros per realitzar aquest projecte.

- El segon preu, el calcularé a partir de la factura que els *Lampistes Gomis* van deixar a casa meva després d'arreglar una avaria. Em van cobrar 22.3 euros/hora. En total, 15 610 euros.

- Finalment, el tercer preu per hora l'extrec de la meva nòmina a l'institut en el que treballo atualment, 16.2 euros/hora per un total de 11 340 euros.

No vull fer valoracions més enllà de les que es puguin extreure dels exemples triats que crec que són, a totes llums, aclaridors de la meva opinió.

Capítol 8

Conclusions

Començo dient que els objectius s'han acomplert, almenys els llistat en la Introducció. Suposo que això no sorprèn ningú perquè els objectius, quan els escrivia, era molt conscient de quin grau de realització tindrien. És per això que no parlaré d'ells en aquestes conclusions, la resta del treball està enfocat a demostrar que s'han assolit. Prefereixo parlar de com s'ha arribat a proposar aquests objectius.

Jo vaig estudiar Matemàtiques i, mentre feia totes les optatives de la branca d'Informàtica Teòrica vaig pensar que la meva formació tenia una mancança: com podia ser que algú que es volia especialitzar en complexitat i algorísmica no sabés programar ni una finestra?

Vist amb perspectiva, segurament no hagués estat necessari, però en aquell moment em va semblar indispensable començar la carrera d'Informàtica.

Moltes han estat les assignatures que no m'han interessat el més mínim perquè les meves pulsions anaven cap a camps molt concrets de la Informàtica i, pel meu gust, força oblidades en pla d'estudis. Ara bé, els coneixements realment no ocupen lloc i si en aquella època m'haguessin dit que gràcies a haver estudiat Informàtica em passaria 6 anys ensenyant estructures de computadors no m'ho hagués cregut.

Després de diversos intents i, en el quadrimestre que acabava la carrera, van obrir l'assignatura de Complexitat per tal de fer callar el grup de *freaks* que contínuament la demanàvem. Allà conec, entre d'altres la Carme i en Dimitrios.

Després d'un parell de converses amb en Dimitrios li demano que em dirigeixi el projecte i ell accepta sense pensar-s'ho. És el moment de començar a posar-se objectius. En Dimitrios m'imprimeix un llistat d'uns 250 problemes NP-difícils [13] i em demana que me'ls llegeixi i li digui en quins em veig capaç de treballar.

De la majoria de problemes a dures penes n'entenc l'enunciat, en marco al voltant de 40. D'aquests 40, en Dimitrios n'elimina els que sap que són massa difícils i destria els que creu que ell m'hi pot ajudar més. Després de la discusió ens quedem amb els

problemes Repico i Roman Domination.[1]

Comencem atacant els dos problemes en paral·lel, estudiant-ne la complexitat. En dues setmanes se m'acut la reducció mitjançant la que obtenim el kernel del Repico (teorema 3.1.3). És fruit d'aquest kernel que comencem a marcar-nos objectius. Repico queda resolt amb aquest resultat, podem revestir-lo amb alguns resultats complementaris? podem trobar variants interessants al problema? podem aprofitar els resultats en el problema de Roman Domination?

La desbordant imaginació de en Dimitrios s'inventa tres línies per continuar el treball:

- El problema Popico, que més endavant vam descobrir que era un problema conegut i molt relacionat [16].
- Estudiar els límits inferiors per tal de tenir cotes sobre les mides de les matrius.
- Generalitzar Repico i Popico a matrius amb més dimensions.

Suposo que a aquestes alçades, és clar que l'objectiu d'estudiar Popico resta assolit. El teorema 4.1.7 en demostra la seva pertinença a FPT. De la mateixa manera, els límits inferiors de pels problemes Repico i Popico queden demostrats en els teoremes 3.2.1 i 4.2.1 respectivament.

Les generalitzacions a 3 o més dimensions dels problemes no aporten res d'interessant. Per Repico, sols cal canviar els rectangles per cubs o hipercubs i tots els resultats s'hi poden reproduir. En canvi, Popico és força més complex perquè el concepte de corba poligonal és difícilment generalitzable a polígons, políedres i hiper-políedres. És per tot això que desestimem afegir aquest apartat al treball.

A més, paral·lelament a la decisió de no fer les generalitzacions apareix l'objectiu més ambiciós del treball: la comparació entre les dues descripcions a nivell d'espai utilitzat (tot i que en aquell moment utilitzàvem una demostració errònia que ens feia pensar que era una trivialitat).

Pot semblar que acabar el projecte amb aquest resultat negatiu deixa un mal regust de boca, jo crec que no. Realment l'esforç per arribar a algun tipus de resultat (positiu o negatiu) ha estat molt gran i sempre és gratificant superar un repte com aquest.

Ens queden moltes idees en el tinter que com que sabíem que no arribaríem a completar per ser massa ambicioses ja ni tan sols les hem escrit en la Introducció:

- No hem estudiat el problema de Roman Domination i molt menys la seva relació amb els dos problemes estudiats. Tot i que després de fer el treball semblen més

[1] Roman Domination és un problema que té el seu origen en l'imperi Romà i en la seva necessitat de defensar un territori molt extens. Aquest problema es pregunta, donada una configuració de ciutats i carreteres entre elles, quantes guarnicions hem de disposar per tal que cada ciutat tingui una guarnició o bé tingui una ciutat veïna amb dues guarnicions [13].

llunyans del que pensàvem ja que tots els estudis fets fins ara es fonamenten en la teoria de grafs i sembla agosarat pensar que podríem tractar-ho mitjançant matrius d'adjacències [16].

- El contraexemple amb el que acabem el capítol 6, lluny de ser una espina clavada es converteix en una porta oberta per futurs treballs. Com que cap de les dos descripcions és sempre millor que l'altra podríem intentar dividir les matrius entre les que es descriuen millor usant rectangles i les que es descriuen millor usant arestes. Estudiar en definitiva, les característiques de les matrius que ens permetrien, a priori, seleccionar el millor sistema.

A nivell personal he assolit objectius també molt importants, objectius que no es poden considerar objectius del treball sinó objectius pels que les enginyeries tenen projecte de final de carrera.

He pogut treballar amb en Dimitrios, pot semblar una tonteria però treballar amb algú que té interessos semblants als teus però que et passa la mà per la cara en coneixements és molt motivador i hi he après moltíssim d'ell.

No puc dir que hagi integrat tot el que he après en la carrera en el projecte: no he programat, no he usat bases de dades, no he pensat en sistemes operatius, no he dissenyat sorftware i no m'he preocupat de l'arquitectura dels computadors. Com ja he explicat, vaig venir a la FIB buscant uns coneixements molt concrets i el treball es centra en l'especialitat que m'agrada. M'he dedicat a aprofundir en aquesta àrea.

He après LaTeX, ha estat indispensable per escriure el treball, m'ha robat moltíssimes hores i molts cops he estat a punt de llençar la tovallola per culpa seva: aconseguir, per exemple, que posi una imatge on vols pot ser tota una odissea. Ara bé, el resultat és simplement espectacular. Quan veig el meu treball imprès, la qualitat final em fa sentir orgullós, em fa sentir professional.

El repte d'escriure un treball tan llarg, que ha de ser rigorós i ha de seguir un seguit de normes ha estat molt educatiu. En aquesta part la pobre Maria ha patit la meva inexperiència escrivint textos científics o tècnics. Cada vegada que li he presentat un nou apartat me l'ha tornat ple d'anotacions i correccions que em feien caure la cara de vergonya. No seré tan petulant com per dir que ara ja en sé, simplement en sé més que abans de començar i això ja és molt.

Acabo dient que estic molt satisfet d'aquest projecte, crec que tot el suc que n'he tret a nivell personal és enorme. Al final, em serà molt més profitós tot el que he après pel camí que no pas les demostracions sense importància que aquí presento.

Em permeto la llicència d'aprofitar-me de qui, sens dubte, escriu millor que no pas jo: *Caminante, no hay camino: se hace camino al andar* d'Antonio Machado.

Capítol 9

Agraïments

No voldria acabar aquest treball sense agrair a tota la gent que m'ha ajudat a realitzar-lo (ja sigui directa o indirectament) :

- Gràcies a en Dimitrios Thilikos per motivar el treball, per l'energia que transmets, per la teva joia de treballar. Gràcies Dimitrios.
- Gràcies a na Maria José Serna per l'esforç d'agafar un projecte a mig fer, per ajudar-me a refinar el projecte fins arribar al que és avui, per quedar a hores intempestives, per no engegar-me a fer punyetes. Gràcies Maria.
- Gràcies a la Carme Álvarez i en Josep M. Llaberia per perdre el vostre temps llegint les tonteries que un alumne pot arribar a escriure. Gràcies Carme, Josep M.
- Gràcies a l'escola SUNION per no posar cap trava a la realització d'aquest treball i per facilitar-me un horari prou flexible per acabar-lo. Gràcies SUNION.
- Gràcies a José Luis Balcázar, per ensenyar-me el camí, et reconec com el meu Obi Wan. Gràcies J.L.
- Gràcies a Mercè Molné i Douglas Winand per permetre'm arribar fins aquí: meu és el mèrit i l'esforç, vostre el sacrifici. Gràcies pares.
- Gràcies a Simó Winand per entendre els meus alts i baixos, per compartir el meu sentir, per la convivència necessària per poder treballar amb tranquil·litat. Gràcies germà.
- Gràcies a tothom que cregui que en algun moment em pot haver ajudat, ja sigui en termes tècnics o bé oferint-me certa estabilitat emocional. Simplement, gràcies.

Índex de figures

Bibliografia

[1] L. Lamport. *LaTeX A Document Preparation System* Addison-Wesley, California (1986).

[2] J.L. Balcázar, J.Diaz, J.Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Computer Science*. Springer-Verlag, Berlin (1988).

[3] M. Sipser. *Introduction to the theory of computation*. Boston, MA, PWS Publishing Company (1996).

[4] Dimitrios Thilikos. *Introcucción del Proyecto Investigador*, document no publicat, Barcelona (2002).

[5] R.M. Karp. *Reducibility among combinational problems*. Plenum, New York (1972).

[6] R.Balasubramanian, M.R. Fellows, V. Raman. *An improved fixed algorithm for vertex cover*, Inform. Process. Lett. 65, pp 163-168 (1998).

[7] J. Chen, I. Kanj i W. Jia. *Vertex cover: Further observations and further improvements*, 27th International Workshop on Graph - Theoretic Concepts in Computer Science, WG 2001, Vol. 1665, pp. 313-324 (2001).

[8] R.G.Downey, M.R. Fellows, A. Vardy i G.Whittle. *The parametrized complexity of some fundamenal problems in coding theory*, J. SIAM Comput. 29, pp 545-570 (1999).

[9] M.R. Fellows i M.A. Langston. *Nonconstructive tools for proving polynomial-time decidability*, J. Assoc. Comput. MAch. 35, pp 727-739 (1988).

[10] R. Yuster, U.Zwick. *Finding even cycles even faster*, SIAM J. Discrete Math. 10, pp 209-222 (1997).

[11] C.H.Papadimitrou. *Computational Complexity*, Pearson Education (1994).

[12] J.E. Hopcroft, R. Motwani, J.D. Ullman. *Introducion to Automata Theory, Languages, and Computation*, Pearson Education (1979).

[13] M.Cesati, *Compendium of PArameterized Problems*, CET, citeseer.ist.psu.edu/cesati01compendium.html.

[14] H.Fernau, *Parameterized algorithms: A graph-theoretic approach*, Apr.2005 Habilitationsschrift(Universität Tübingen,Tübingen,Germany).

[15] W.J. Masek, *Some NP-complete set covering problems*. MIT, Cambridge, MA, manuscrit no publicat (1978).

[16] P.Berman, B. DasGupta *Complexities of Efficient Solutions of Rectiliniar Polygon Cover Problems*. Canadian Conference on Computational Geometry, pp 331-356 (1992).