UNA CONTRIBUCIÓ AL CÀLCUL DE VALORS I VECTORS PROPIS I A L'ANÀLISI DE L'ESCALABILITAT

UNA CONTRIBUCIÓ AL CÀLCUL DE VALORS I VECTORS PROPIS I A L'ANÀLISI DE L'ESCALABILITAT

Dolors Royo Vallés

Departament d'Arquitectura de Computadors

Universitat Politècnica de Catalunya (UPC)

Barcelona. Novembre, 1998

TESIS PRESENTADA EN COMPLIMENT
DELS REQUERIMENTS PER OBTENIR EL GRAU DE
Doctora en Informàtica

UNA CONTRIBUCIÓ AL CÀLCUL DE VALORS I VECTORS PROPIS I A L'ANÀLISI DE L'ESCALABILITAT

Dolors Royo Vallés

Tribunal de Tesi:

A la meva germana Men,
a la meva mare i al meu pare,
per tots els bons moments
que hem passat tots plegats

RESUM

El càlcul de valors i vectors propis és un nucli computacional que forma part de diverses aplicacions de tipus científic i tècnic que requereixen una potència de càlcul molt gran. Aquestes aplicacions no poden resoldre's en sistemes monoprocessadors perquè aquests sistemes no proporcionen la potència de càlcul suficient per resoldre el problema amb un temps raonable. Una solució possible a aquest problema és la utilització de sistemes paral.lels.

El contingut d'aquest treball pot dividir-se en quatre parts ben diferenciades; en les tres primeres parts es proposen un conjunt d'algorismes paral.lels per la resolució del problema del càlcul dels valors i vectors propis en sistemes multicomputadors amb diferents topologies: hipercub, malla i torus; en l'última part del treball es proposa una metodologia d'anàlisi de l'escalabilitat de sistemes paral.lels.

- En la primera part del treball es proposen un conjunt d'algorismes paral.lels per hipercubs: BR segmentat, α-optimal i $Grau$-4. Tots aquests algorismes es basen en l'algorisme *Block Recursive* proposat a [42]. Els nous algorismes proposats tenen la capacitat d'utilitzar de forma més eficient el potencial paral.lelisme de comunicacions que ofereix una arquitectura *multiple-port* amb els que s'aconsegueix una reducció del cost de la comunicació considerable respecte al cost de comunicació de l'algorisme original.
- En la segona part del treball es proposa un nou algorisme amb una topologia de comunicació en malla bidimensional ($2D$). Aquest algorisme l'hem anomenat algorisme $2D$. Es veurà que aquest nou algorisme aconsegueix reduir el cost total considerablement respecte als algorismes que han estat proposat per altres autors per malles i torus.

- En la tercera part, s'estudia l'eficiència de l'algorisme BR-segmentat (algorisme amb una topologia de comunicació en hipercub proposat en la primera part de la tesi) un cop mapejat en un multicomputador amb una topologia en malla o en torus. A l'hora de realitzar el mapeig s'ha aplicat i ampliat una metodologia desenvolupada en el grup de treball que ens permet realitzar el mapeig de forma eficient i sistemàtic d'una topologia en hipercub a una topologia en malla o torus. El cost de la comunicació del nou algorisme es compara amb el cost de l'agorisme $2D$ proposat en la segona part del treball.
- Finalment, en l'última part d'aquest treball es proposa una metodologia d'anàlisi de l'escalabilitat de sistemes paral.lels orientada a l'usuari final del sistema. S'utilitza l'algorisme $2D$ mapejat en una línia per mostrar un exemple d'aplicació de la metodologia.

AGRAïMENTS

Vull expressar el meu agraïment més sincer a totes aquelles persones que han estat al meu costat durant el temps que he estat realitzant aquesta tesi.

A Miguel, el director d'aquesta tesi, li he d'estar especialment agraïda per la paciència que ha tingut per treballar amb mi durant tot aquest temps, per tot l'esforç que ha tingut que fer per tirar endavant gran part del treball i per tot el temps que m'ha dedicat. Després de tant temps fent recerca junts, m'agradaria poder dir que he après a ser una mica més metòdica i ordenada (qualitats seves indiscutibles). L'enteniment entre el Miguel i jo ha estat difícil però s'ha consolidat en alguna cosa més que aprendre a treballar plegats, en ell he trobat un molt bon company que dessitjaria conservar per molts anys.

A l'Antonio González, li agraeixo la seva disponibilitat i entusiasme que ha demostrat quan li hem demanat un cop de mà. La seva participació ha fet possible el desenvolupament d'algunes parts representatives del treball. Al LuisMa que ha fet la seva tesi paral.lelament a la meva, un exemple clar que el paral.lelisme funciona a alguns nivells.

A la meva família; la meva mare Emilia, al meu pare Jordi, a les meves germanes: Emi, Tere, Men, Luci i Maria; que tants cops m'ha fet oblidar la dura feina de fer una tesi i que han estat una ajuda moral insubstituïble per superar tots els moments difícils d'aquest llarg període de la meva vida.

A Toni, que ha estat un company insuperable, que sempre m'ha animat en els pitjors moments i que amb la seva facilitat organitzadora i crítica m'ha fet veure més clar el camí a seguir en els moments crítics. Li estaré immensament agraïda per la confiança que a tingut sempre en mi i per tot l'amor que ha posat per demostrar-ho.

A Pedro, el germà del Toni, que m'ha fet disfrutar de xerrades interminables de les que he après que les totes les coses en aquesta vida s'han de prendre en més calma.

A la Montse, que després del Miguel és la persona que s'ha llegit més cops tot el treball i que ha fet possible que la seva edició sigui sintàcticament més correcta. Totes les possibles errades sintàctiques que podeu trobar en el document són degudes als canvis d'ùltima hora que ella no ha pogut revisar.

Al Juanjo, que va aguantar una prelectura de la tesi a la una de la matinada amb molt entusiasme, amb el que va aconseguir transmetre'm confiança en el treball que havia realitzat. A Agustin, Marta, Roger, Xavi, David, lo Josep, Fermín, Luis (l'olímpic), Enric, Josep Ramon, Cristina i Sangi per tots els moments agradables que m'han fet passar.

A Pepe, que malgrat no va poder resistir una prova per la presentació en un congrés de l'ùltima part del treball caient en mans de Morfeu, no ha dubtat mai en fer-me companyia per dinar molts dies d'estiu d'aquests darrers anys que he estat treballant en la realització de la tesi. A la Julita, Marc, Albert, Ernest, Gianluca i Toni Cortés que per supost també m'han amenitzat alguns que altres dinars estiuencs.

Enric Pastor ha estat sempre dispost a resoldre qualsevol dubte de Latex.

A la gent que va compartir amb mi els meus anys universitaris, un dels períodes mes feliços de la meva vida, i que han tingut l'humor de compartir amb mi algunes estones d'aquests anys post-universitaris amb molta paciència; Manu, Xisca i Vial, Pili, Andrés, Ernest, Maite i Rafa, Albert i Lluís.

Voldria mencionar a Betty, que sempre ha tingut el detall de preguntar pel treball de tesi els pocs cops que hem coincidit en alguna sortida cultural, i al Miodrag Stekovic, que com el seu nom deixa clar no és d'aquesta terra i que fa poc més d'un any va anar a parar just al costat del meu despatx, que havia treballat amb hipercubs casualitats de la vida, i que sempre s'ha mostrat interessat en el procés final del treball.

Per últim voldria agrair a la gent del DAC que treballa o ha treballat en tasques organitzatives, tasques administratives i tasques burocràtiques; Mateo Valero, Josep Maria Llaberia, Jesus Labarta, Eduard Aiguadé, Nacho Navarro, Jordi Cortadella, Rosa Badia, Olga Casals, Rosa Castro, Jose A. Rodríguez, Alex, Juan Escobar, Robert, Cristina, Juani, Mercé i Alícia; que amb el seu esforç m'han proporcionat l'entorn i els recursos adequats per desenvolupar la meva recerca.

CONTINGUT

1

CONTEXT DEL TREBALL

Resum

En aquest capítol es descriuen els principals conceptes que defineixen l'entorn en que es desenvolupa aquest treball. En la primera part es dóna una visió històrica dels multicomputadors i es descriuen aquelles propietats dels multicomputadors que s'han tingut en compte a l'hora de definir els algorismes que es proposen. A continuació es descriu el problema del càlcul del valors i vectors propis des del punt de vista matemàtic i es descriuen alguns algorismes que ja han estat proposats per la seva resolució en sistemes multicomputadors. Els algorismes descrits són el punt de partida d'algunes de les propostes d'aquest treball. A continuació s'introdueix el concepte d'escalabilitat de sistemes. I per últim, es descriu la metodologia d'avaluació que s'ha utilitzat a l'hora d'analitzar l'eficiència dels algorismes proposats en el treball.

1.1 SISTEMES PARAL.LELS

Al llarg de la història dels computadors, capítol 2 de [20],capítol 1 de [51], capítol 1 de [5], sempre s'han buscat computadors més ràpids per motius diversos:

- Als usuaris dels sistemes els interessa resoldre problemes cada cop més grans els quals no poden resoldre's amb els computadors disponibles en aquell moment en un temps raonable.
- Per una altra banda, sempre s'està interessat en reduir el temps d'execució dels problemes que s'estan resolent i obtenir d'aquesta manera un temps de resposta cada cop més petit.

Durant les últimes dècades s'ha aconseguit incrementar considerablement la velocitat de processament dels computadors. Gran part d'aquest increment de velocitat és degut a la utilització de components electrònics cada cop més ràpids - vàlvules, transsistors, circuits integrats (petita/mitjana/gran escala)-. Aquesta reducció, però, està limitada per la pròpia tecnologia degut a què la velocitat màxima a la que pot operar un component electrònic de certes dimensions depèn de la velocitat de la llum. Un cop arribat al límit de la tecnologia, actualment una de les opcions que ens permeten seguir augmentant la velocitat i capacitat de processat és l'aplicació del *paral.lelisme*. És aquest el motiu pel qual s'ha introduit el paral.lelisme a molts nivells dintre del sistema computador per tal d'aconseguir més velocitat de processament i en definitiva millors prestacions.

1.1.1 Visió històrica dels sistemes paral.lels

Inicialment, les millores es realitzen amb l'objectiu d'aconseguir processadors seqüencials més ràpids, capítol 1 de [51]. Algunes d'aquestes millores (seguint un ordre cronològic d'aparició) són: accés en paral.lel a una paraula de memòria (inicialment 8 bits), utilització de processadors especialitzats en la realització d'operacions d'entrada i sortida, memòria entrellaçada, memòria cache, $buffering$ d'instruccions, implementació de múltiples unitats funcionals, execució segmentada de les instruccions.

Arribats en aquest punt, és natural l'aparició d'unitats funcionals segmentades i la possibilitat d'entrellaçar vàries operacions aritmètiques. Apareixen els computadors vectorials els quals s'implementen de dues maneres diferents: els processadors en array i els computadors vectorials segmentats. Mentre que els processadors en array estan formats per un conjunt d'elements de procés amb la capacitat de realitzar de forma simultània la mateixa operació amb un conjunt de dades diferents, els processadors vectorials segmentats es caracteritzen per obtenir concurrència a partir de l'execució d'operacions amb vectors utilitzant unitats vectorials segmentades (model *SIMD -Single Instruction Multiple Data-* segons la classificació de Flynn [19][18]).

El model $SIMD$ és un dels primers models de computadors paral.lels que va aparèixer [49]. Model que van seguir els multiprocessadors *Illiac IV*, *ICL DAP*, *Goodyear MPP*, *Thinking Machines* $CM-1$ i $CM-2$ i la *Maspar* $MP-1$ i $MP-2$ (els quals es classifiquen dintre la categoris de processadors en array).

Molts supercomputadors d'aquesta època tenen processadors vectorials segmentats, com el *Cray-1* (*Cray Research*) i el *Cyber 205* (*Control Data Corporation*). Alguns multiprocessadors d'aquesta època es construeixen a partir de nodes vectorials, com per exemple el *Cray 3*. Apareixen, doncs, sistemes multiprocessadors amb varis nodes vectorials connectats en bus.

Tots els sistemes paral.lels que podem trobar en aquesta època són amb una gestió de memòria compartida.

Els dos problemes principals que tenen aquestes primeres màquines paral.leles són:

- El model de programació dels processadors en array ($SIMD$) és molt poc flexible - no tots els problemes tenen aquest tipus de paral.lelisme-. Malgrat que els computadors $SIMD$ inicialment estaven pensats com una màquina de propòsit general, aquests tipus d'arquitectura s'adapta a les necessitats de certs tipus de problemes, com el processat de la imatge o el processat del senyal.
- Són màquines amb un cost molt elevat, especialment en el cas dels *multiprocessadors* amb nodes vectorials, els quals utilitzen nodes que s'emmarquen dintre la categoria

de *supercomputadors* - nodes vectorials *custom*- amb el que s'encareix el cost del sistema de forma considerable a l'hora d'implementar màquines amb un nombre de nodes gran.

Amb l'idea de reduir el cost dels sistemes paral.lels, el següent pas vas ser implementar màquines amb molts nodes molt més senzills, nodes amb arquitectures de tipus general amb un cost molt més baix. És cap a la dècada dels 80 que els multiprocessadors comencen a prendre força, bàsicament, degut a les bones prestacions dels microprocessadors i a la possibilitat d'obtenir qualsevol component (memòria, xarxa d'interconnexió...) del que es composa el sistema computador a un cost considerablement baix.

Aquests nous computadors paral.lels segueixen un model $MIMD$, *Multiple Instruction Multiple Data*, (segons la classificació dels computadors realitzada per Flynn [18]). Un model molt més flexible que el model $SIMD$, on cada node del sistema té la possibilitat d'executar un conjunt d'instruccions diferents amb un conjunt de dades diferents.

Dintre d'aquest nou grup, inicialment, apareixen els sistemes paral.lels amb memòria distribuïda, també anomenats *multicomputadors*. En aquests sistemes paral.lels un node consisteix en un processador el qual té la seva pròpia memòria local i un hardware de comunicació que li permet la connexió directa amb alguns nodes que formen el sistema. Els processadors s'intercanvien informació mitjançant el mecanisme de pas de missatges a través de la xarxa d'interconnexió. En la figura 1.1(a) es mostra un esquema molt simple del que seria un sistema *multicomputador* [1].

Els hipercubs [29][28], són les xarxes d'interconnexió que s'implementen en els primers sistemes multicomputadors. Els primers hipercubs que podem trobar al mercat són de la família *Intel*, així podem trobar el $iIPSC-2$, el $iPSC/860$,.. o bé la família dels *nCube*. Els hipercubs, malgrat les seves bones prestacions tenen restriccions hardware per escalar el sistema a qualsevol nombre de nodes. Aquest fet dóna lloc a que es proposen altres topologies molt més escalables com les malles i torus n-dimensionals.

La programació dels *multicomputadors* amb una metodologia amb pas de missatges no és una tasca fàcil de realitzar. El programador ha de tenir cura a l'hora de distribuir

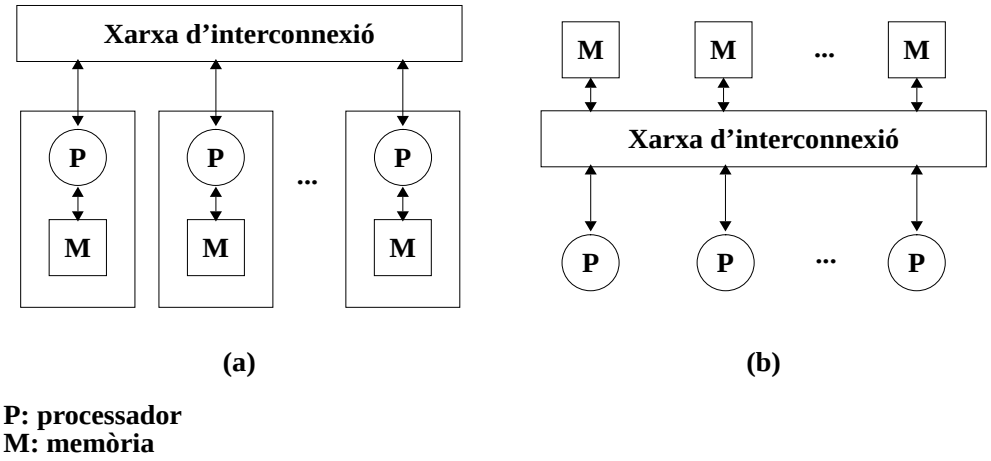


Figura 1.1 Representació esquemàtica de (a) sistemes *multicomputadors* i (b) sistemes *multiprocessadors*

les dades entre els diferents nodes del sistema per obtenir una eficiència màxima i minimitzar el pas de missatges. Per una altra banda, el software desenvolupat fins aquell moment no pot aprofitar-se en els nous sistemes amb un model de programació amb pas de missatges i s'ha de refer de nou.

Aquest problema fa aparèixer sistemes paral.lels amb gestió de memòria compartida, també anomenats *multiprocessadors* [5]. En aquests nous sistemes, la programació és molt més senzilla que en el cas dels *multicomputadors* ja que l'intercanvi de dades entre els diferents nodes del sistema és totalment transparent al programador. Les operacions de comunicació es realitzen en base a les operacions d'accés a memòria *load/store*.

En la figura 1.1(b) es mostra un esquema general d'un sistema *multiprocessador*. Com es pot observar en aquest esquema, els multiprocessadors per tal de poder compartir la memòria entre tots els nodes del sistema s'implementen situant la xarxa d'interconnexió entre els processadors i la memòria. Malgrat la senzillesa de programació que comporten els sistemes *multiprocessador*, aquests tenen una limitació hardware a l'hora d'afegir nodes al sistema, limitació que no apareix en els sistemes *multicomputadors*. Un sistema *multiprocessador* no pot incrementar el nombre de nodes de forma indefinida degut a què el temps d'accés a memòria inclou la latència d'accés a la xarxa d'interconnexió i aquest cost s'incrementa a mesura que el sistema creix.

En aquest moment tenim dos esquemes ben diferents de plantejar un sistema paral.lel amb els seus avantatges i desavantatges. El següent pas en l'evolució dels sistemes paral.lels ha estat proposar sistemes que aconsegueixen tenir els avantatges dels *multicomputadors* i els avantatges dels *multiprocessadors*, o sigui, aconseguir sistemes que siguin fàcils de programar i amb la possibilitat de tenir un nombre de nodes grans a un cost raonable (sistemes *híbrids*). Amb aquest objectiu, apareixen els sistemes amb memòria compartida físicament distribuïda, *DSM -Distributed Shared Memory-*. La topologia d'interconnexió és similar als sistemes *multicomputadors* inicials. A l'arquitectura s'afegeix el hardware necessari per la gestió d'adreces de memòria; el hardware s'encarrega de determinar si l'accés és a dades locals -memòria local al node- ó si l'accés és a dades no locals -que pertanyen a la memòria local d'un altre node-; aquesta gestió és totalment transparent a l'usuari [5].

El model de programació d'aquests sistemes *DSM* es pot realitzar mitjançant primitives de memòria compartida (*load/store*) o bé amb primitives de pas de missatges. Quan el nombre de nodes del sistema és gran és aconsellable l'utilització de primitives de pas de missatges per qüestió d'eficiència.

Actualment, podem trobar sistemes paral.lels comercials amb aquestes característiques [49], per exemple *Cray T3E* amb un nombre màxim de 2048 nodes i una topologia toroidal $3D$, la *Sequent NUMA-Q* amb 32 nodes com a màxim i una topologia en anell o la *SGI Origin2000* amb un màxim de 128 nodes i una topologia $6 - cube$.

Paral.lelament, amb el desenvolupament de xarxes d'àrea local cada cop més ràpides, ha sorgit un nou camp en la computació paral.lela que són les xarxes d'estacions de treball (*NOW* o també anomenats *clusters*). Un exemple representatiu d'aquest tipus de màquina és: el *cluster* de *100* nodes *UltraSPARC* implementat a la universitat *UC Berkeley* amb una xarxa d'interconnexió *Myrinet* [49]. En aquestes xarxes el model de programació també es realitza amb primitives de pas de missatges.

En aquest treball de tesi ens interessa qualsevol sistema paral.lel en el que sigui possible la programació de pas de missatges. Així doncs, tots les idees que es proposen en els algorismes proposats i avaluats en aquest treball són vàlids per sistemes *multicomputa-*

dors, per sistemes *multiprocessadors* del tipus DSM que suporten el pas de missatges i pel cas de les xarxes d'estacions de treball (NOW). En el treball, els algorismes s'han avaluat concretament pel cas d'un entorn *multicomputador*, és per aquest motiu que a continuació es donen més detalls sobre *multicomputadors* .

1.1.2 Organització dels multicomputadors

En aquesta secció es descriuen breument els components dels multicomputadors que s'han tingut en compte a l'hora de modelitzar els algorismes que es presenten en els capítols següents.

Xarxa d'interconnexió

La funció de la xarxa d'interconnexió en una màquina paral.lela és permetre l'intercanvi d'informació entre un node i qualsevol altre node del sistema. La classificació més habitual de les xarxes d'interconnexió en un entorn multiprocessador és en xarxes directes i xarxes indirectes.

Una xarxa directa consisteix en un conjunt de nodes, cadascun dels quals està connectat directament a un subconjunt d'altres nodes de la xarxa [5][13][38]. Un node és un sistema complet, processador, memòria i dispositius. Algunes de les propietats importants d'una xarxa d'interconnexió directa són:

- Grau d'un node: El nombre de canals que connecten un node amb els seus veïns.
- Diàmetre: La distància màxima entre dos nodes dintre la xarxa.
- Simetria: Una xarxa és simètrica si es veu igual des de qualsevol node.

Una xarxa directa es caracteritza bàsicament per dos components: topologia i gestió de l'encaminament de missatges. En el cas d'encaminament de missatges s'ha de tenir en compte els algorismes de *switching* i algorismes d'enrutament de missatges (*routing*). La topologia determina cóm estan els nodes físicament connectats uns amb els altres. La

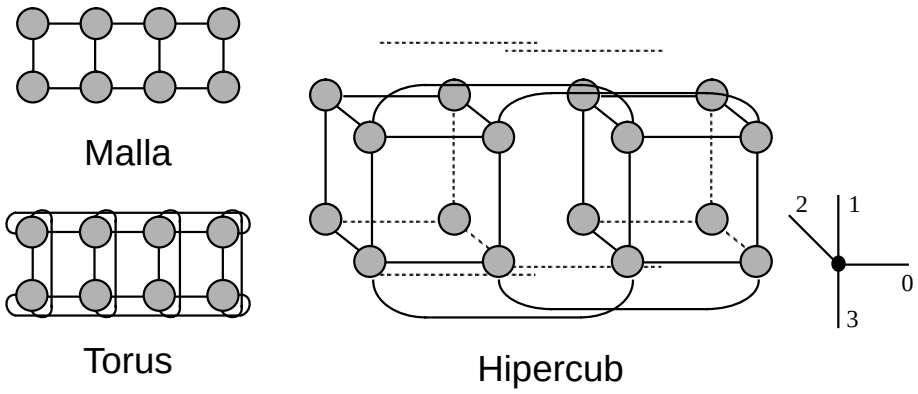


Figura 1.2 Xarxes d'interconnexió directes d'interés en aquest treball

topologia ideal és aquella en què tots els nodes estan connectats amb la resta de nodes del sistema mitjançant un enllaç directe. Quan el nombre de nodes del sistema creix, el cas ideal és impracticable. És per aquest motiu que en xarxes reals, els nodes tan sols estan connectats directament a un subconjunt de nodes dels sistema. Això implica que a l'hora d'intercanviar informació entre dos nodes qualssevol s'hagin d'utilitzar nodes intermitjos. En aquestes topologies, en les que un missatge ha de travessar nodes intermitjos per arribar al seu destí, són necessaris algorismes d'encaminament que determinen quin és el camí a seguir. Finalment, quan un missatge arriba a un node intermig, s'han de determinar quins són els recursos que s'han d'utilitzar, canals de sortida, buffers... aquestes decisions es prenen en funció de quin mecanisme de *switching* s'apliqui.

En una xarxa indirecta en lloc de tenir enllaços que permeten la comunicació directa entre els nodes, la comunicació entre els nodes es realitza mitjançant un mecanisme hardware (*switch*) que permet la comunicació entre dos nodes qualssevol de la xarxa amb un cost constant.

En aquest treball ens centrem en xarxes directes [5][38], més concretament es consideraran topologies en hipercubs, malles i torus (figura 1.2).

- Hipercub.

Un hipercub d-dimensional té $N = 2^d$ nodes i $d \times 2^{d-1}$ enllaços [29][28]. Cada node està identificat per un nombre binari. Dos nodes estan connectats amb un enllaç directe si i només si els corresponents identificadors difereixen en un sol bit. D'aquesta manera es compleix que un node té un nombre de veïns igual a $d = \log N$, cadascun d'aquests veïns en una dimensió diferent k de l'hipercub, $k \in [0..(d-1)]$, depenent de quina posició ocupa el bit que difereix entre els dos identificadors del nodes en questió.

La construcció d'un hipercub es pot realitzar de forma recursiva; un hipercub de dimensió d es pot construir a partir de dos hipercubs de dimensió $d-1$. En la figura 1.2 es mostra un exemple d'un hipercub de dimensió 4 el qual es construeix a partir de dos hipercubs de dimensió 3, a la vegada cadascun d'ells es construeix a partir de dos hipercubs de dimensió 2 i aquest procés ès repeteix fins que s'arriba a un hipercub de dimensió 0, un únic node.

És una topologia que degut a les bones propietats que presenta ha estat estudiada molt a fons. Les propietats més destacables són:

- Regular
- Diàmetre petit, igual a $\log N$.
- Els nodes tenen un grau igual a la dimensió d (es a dir el grau escala en funció del tamany de l'hipercub).
- És una topologia en la que fàcilment es poden mapejar altres topologies, malles, torus, arbres, etc.

En els primers multicomputadors s'implementa aquesta topologia, on podem incloure el prototipus de *CaltechResearch - Cosmic Cube* [56]-, les tres primeres generacions de l'*Intel iPSC* (*iPSC-1, iPSC-2 i iPSC/860*) i les diferents generacions de *nCUBE* (*nCUBE2* [47], *nCUBE3 -metaCube* (1995)-) [28].

- Malles i torus multidimensionals.

  La xarxa d'interconnexió més simple que podem trobar és una línia, on els nodes és numeren consecutivament $0, ..., (N-1)$ (una línia amb N nodes). La xarxa consta de $N-1$ enllacos. El diàmetre de la xarxa és $N-1$ i la distància mitja és $2/3N$. L'encaminament en xarxes d'aquest tipus és trivial ja que tenim un únic camí per connectar qualsevol parella de nodes i el cost d'implementació hardware és petit.

Un anell es construeix bàsicament afegint un altre enllaç que connecti directament els dos extrems d'una línia. El diàmetre i la distància mitja són $N/2$. L'encaminament és senzill ja que tan sols tenim dos camins possibles per comunicar dos nodes qualssevol de l'anell.

El problema principal d'aquestes topologies linials és que a mesura que el nombre de nodes augmenta les distàncies es fan grans i augmenta el nombre de conflictes de comunicació sempre i quan varis nodes de la línia/anell vulguin enviar un missatge en el mateix instant.

La generalització d'una línia i un anell és una malla i un torus respectivament. Una malla d dimensional, consisteix en $N = k_{d-1}$ x ... x k_0 nodes. Un node X dintre la xarxa està identificat per un vector de d coordenades $(x_{d-1}, ..., x_0)$, on $0 \leq i_j \leq k_j - 1$ per $0 \leq j \leq (d-1)$. Dos nodes X i Y dintre la malla estan connectats amb un enllaç directe si $x_i = y_i$ per tot $0 \leq i \leq d-1$, excepte per un j tal que $y_j = x_j + 1$ ó $y_j = x_j - 1$. En una malla un node té un nombre de veïns que varia de d fins a $2d$ segons la situació del node dintre la xarxa; els nodes interns tenen el màxim grau $2d$ i els nodes del contorn tenen graus inferiors, sent els nodes de l'extrem els que tenen el grau mínim d.

En un torus, amb els enllaços *wraparound* entre els nodes del contorn s'aconsegueix que tots els nodes tinguin un nombre de veïns igual a $2d$.

En la figura 1.2 es mostra un exemple d'una malla i d'un torus de dues dimensions.

Molts multicomputadors més actuals han adoptat aquestes topologies, malla i torus, ja que permeten escalar el sistema de forma simple i amb un cost baix respecte a altres topologies com l'*hipercub*. Podem trobar multicomputadors com *Intel Paragon XP/S* i el *Mosaic C* amb una topologia en malla *2D* [35]. El *MIT J-Machine* amb una topologia en malla *3D* i els multicomputadors de *Cray, T3E i T3D,* amb una topologia toroidal $3D$.

Gestió d'encaminament de missatges

El hardware que ens permet la comunicació entre els nodes del multicomputador és el *router* [13]. Els components principals que podem trobar en un *router* són:

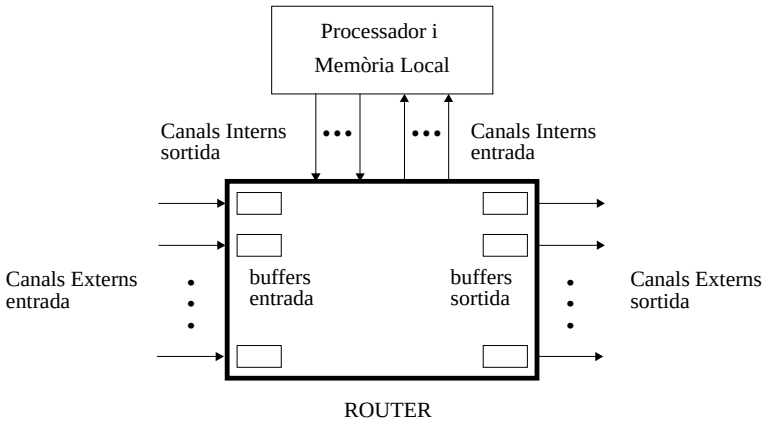


Figura 1.3 Esquema d'un *router*

- Canals externs. Els canals de comunicació utilitzats per l'intercanvi de dades entre nodes. Distingim entre canals d'entrada i canals de sortida. Els canals d'entrada, pels quals el node rep els missatges, dels quals ell és el destí, o bé rep missatges els quals s'han de reenviar ja que tenen un altre destí. Els canals de sortida, pels quals el node envia/reenvia els missatges a la resta de nodes.

- Buffers. Per emmagatzemar els missatges en trànsit. En general s'associa un buffer a cadascun dels canals d'entrada i sortida físics. El tamany del buffer ha de ser suficient per emmagatzemar almenys un missatge de tamany igual a un $flit$ (quantitat d'informació mínima que viatja per la xarxa). L'unitat de transferència es diu $phit$, i està composada per 1 o més $flits$.

- "$Switch$". Mecanisme hardware que permet la connexió dels canals d'entrada amb els canals de sortida.

- Mecanisme de $routing$. Mecanisme que implementa els algorismes d'encaminament, seleccionant quin canal de sortida ha d'utilitzar el missatge que arriba, realitzant la connexió corresponent (mecanisme de $switching$). En el cas de què varis missatges vulguin utilitzar el mateix canal de sortida el mecanisme d'encaminament ha de gestionar l'accés al mateix. En cas de què el canal de sortida estigui ocupat el missatge s'ha d'emmagatzemar en els buffers.

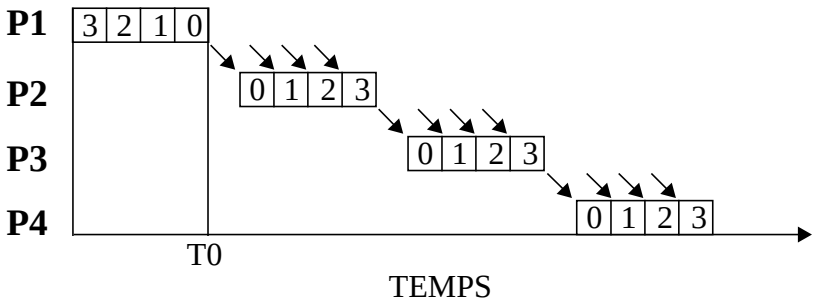


Figura 1.4 *Store* and *Forward*

- Canals interns. Permeten la comunicació entre el processador i el *router*. Aquests consisteixen en un o més canals d'entrada i sortida pels quals viatgen els missatges el destí o l'origen dels quals és el node en questió.

Podem trobar tres mecanismes de gestió d'encaminament dels missatges [5], que són els següents:

- *Store* and *Forward*.

  Amb aquest esquema un node espera a rebre tot el missatge sencer abans d'enviar-lo al següent node. Aquesta tècnica requereix que els nodes tinguin buffers molt grans.

  En la figura 1.4 es mostra de forma gràfica com es realitzaria el pas de missatges amb aquest esquema. En l'exemple es suposa que un missatge està composat per 4 paquets numerats del 0 al 3. El missatge s'envia del processador $P1$ al processador $P4$. Inicialment el processador $P1$ té emmagatzemat tot el missatge, temps $T0$. Els paquets s'envien seqüencialment un darrera l'altre al processador $P2$ i aquest no els reenvia al processador $P3$ fins que no ha rebut tot el missatge complet (els quatre paquets). Un cop rebut tot el missatge torna a començar el mateix procés però entre el processadors $P2$ i $P3$ i finalment amb els processadors $P3$ i $P4$.

- *Circuit switching*.

  Amb aquest esquema, abans d'enviar cap missatge primer s'estableix el camí, configurant els routers convenientment i indicant al node origen quin ha estat el camí

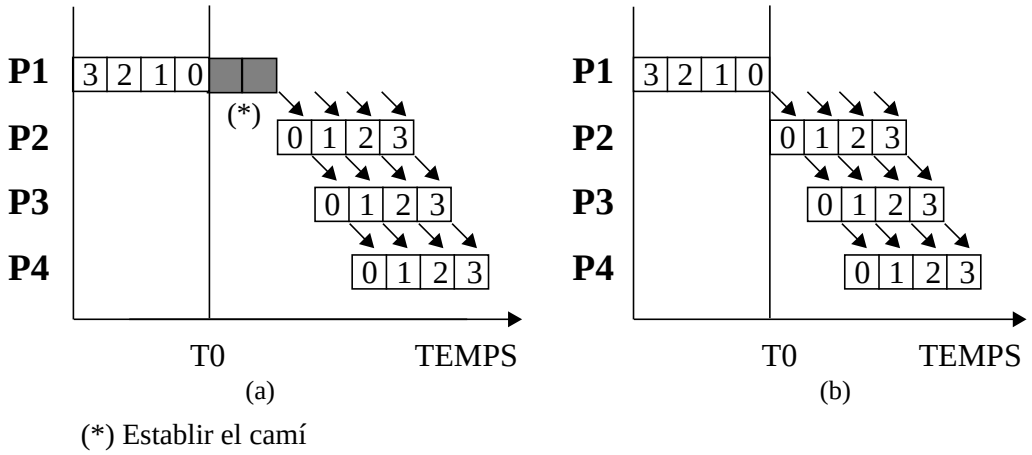


Figura 1.5 (a) *Circuit switching* i (b) *Cut through*

que s'ha establert [50]. A continuació les dades poden enviar-se a la velocitat màxima (igual a l'amplada de banda de l'enllaç). Els nodes s'alliberen un cop el missatge ha estat rebut completament pel node destí.

Aquest esquema és eficient en sistemes en els que s'han d'enviar missatges amb poca freqüència i cada transmissió involucra grans quantitats de dades. D'aquesta forma el temps d'iniciar una comunicació (establir el camí) té un pes relativament petit en el cost total de la comunicació, el qual es considera despreciable en front del cost d'enviar les dades. El problema més gran que planteja aquest esquema és el temps que es mantenen els nodes ocupats un cop establert el camí. Un node no s'allibera fins que tots els paquets que formen el missatge no han arribat al node destí. Un cop el node destí ha rebut tots els paquets envia un missatge al node origen que indica que la transferència ha finalitzat i és en aquest moment que els nodes queden lliures. En la figura 1.5(a) es mostra un esquema de com es realitzaria el pas de missatges amb aquest esquema.

- *Cut through.*

  La majoria de màquines paral.leles actuals apliquen l'esquema d'encaminament de missatges anomenat *cut through* [48] [46]. En aquest esquema el camí es crea a mesura que la comunicació progressa. Els routers decideixen quin ha de ser el camí consultant els primers bits de la capacalera del missatge, permetent d'aquesta forma

a la resta del missatge passar directament del canal d'entrada al canal de sortida del router. Daquesta forma el retard en cada node és mínim.

Aquest esquema és molt similar a l'esquema *Circuit switching* però sense la fase inicial d'establiment del camí. A més a més, en aquest esquema els nodes s'alliberen a mesura que l'últim paquet que forma part del missatge és enviat per cadascun dels nodes que formen el camí. D'aquesta forma els nodes estan ocupats el mínim temps possible. El cost de travessar els nodes intermitjos es considera despreciable. En la figura 1.5(b) es mostra un esquema de com es realitzaria el pas de missatges amb aquest esquema.

Un dels aspectes que s'han de tenir en compte en la gestió d'una xarxa és el problema de la contenció, quan dos o més missatges volen utilitzar el mateix canal de sortida. En el cas del mecanisme *cut through*, podem trobar dues solucions.

- La primera solució, anomenada *virtual cut through*, soluciona el problema emmagatzemant tot el missatge en un buffer del node on s'ha generat la contenció [34].
- En la segona solució, anomenada *wormhole*[48], cada node emmagatzema la part del missatge que té en l'instant en què es produeix la contenció, d'aquesta forma el missatge queda emmagatzemat entre tots els nodes que formen el camí. Amb aquesta segona proposta, el tamany dels buffers pot ser una mica més petit.

Interfície amb la xarxa

Un punt molt important a considerar en aquest treball és la connexió entre el processador i el *router*, que fa referència al nombre de canals interns [35][48]. Des d'aquest punt de vista podem distingir dos tipus d'arquitectures: $one - port$ i $multiple - port$.

En una arquitectura $one - port$ l'interface entre el processador i el *router* està formada per un únic enllaç d'entrada i un únic enllaç de sortida. En un moment determinat el processador tan sols pot enviar i rebre un únic missatge per/de qualsevol dels canals externs (del *router*). Una conseqüència directa d'aquest esquema és que els missatges s'han de rebre/enviar de forma seqüencial.

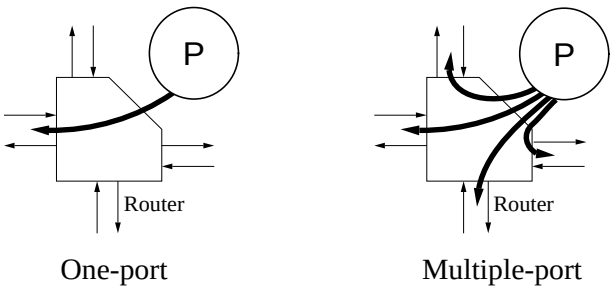


Figura 1.6 Model $one-port$ i $multiple-port$

En la figura 1.6(a) es pot veure de forma esquemàtica la capacitat d'enviar/rebre d'un node amb un model $one-port$ (un missatge com a màxim). En aquest cas es considera un *router* amb quatre canals externs d'entrada i quatre canals externs de sortida. El processador té capacitat d'enviar/rebre un únic missatge que pot sortir/ o li pot arribar per qualsevol dels quatre canals de sortida/entrada. La connexió processador-*router* es considera *full-duplex*; per tant, a la vegada que rebi un missatge es possible estar enviant un altre missatge per qualsevol dels quatre canals de sortida. En el cas que en el mateix instant es rebi més d'un missatge el processador tan sols podrà accedir a la informació de forma seqüencial (un darrera l'altre).

En una arquitectura $multiple-port$ es redueix el coll d'ampolla entre processador i *router* afegint més canals interns d'entrada i sortida. El node pot enviar i rebre varis missatges en paral.lel sempre i quan utilitzin diferents canals externs. L'arquitectura és diu $all-port$ quan el nombre de canals interns coincideix amb el nombre de canals externs. En la figura 1.6(b) es mostra de forma esquemàtica (quatre missatges a la vegada pels diferents canals externs) la capacitat d'enviar/rebre d'un node amb un model $multiple-port$ (més concretament $all-port$).

Model de programació (pas de missatges)

Els diferents processos d'una aplicació paral.lela distribuïts entre els diferents nodes d'un multicomputador molt probablement han d'intercanviar informació entre ells. En sistemes multicomputadors les dades generalment s'intercanvien entre els processos mitjançant crides de sistema de pas de missatges. Per enviar dades a un altre procés,

s'utilitzen rutines que tenen per paràmetres el node destí, el missatge i el tamany del missatge. El que rep el missatge indica de qui vol rebre el missatge i on s'ha de deixar la informació rebuda [5] [33].

En aquest treball no s'ha tingut en compte cap tipus d'implementació concreta del model de programació (PVM, MPI, active messages...), el model desenvolupat és el suficientment general per permetre avaluar qualsevol implementació sempre i quan sigui possible expressar el seu cost en funció d'un temps d'inicalització (latència) i d'un temps d'enviar les dades per un enllaç (ample de banda) [5].

1.2 CÀLCUL PARAL.LEL DE VALORS I VECTORS PROPIS

1.2.1 Valors i vectors propis

Sigui A una matriu real quadrada de dimensió m x m. Es diu que λ és un valor propi de la matriu si existeix un vector x tal que

$$Ax = \lambda x \tag{1.1}$$

El vector x és el vector propi associat al valor propi λ.

El problema que s'estudia en aquest treball és el càlcul eficient de tots els valors i vectors propis d'una matriu real i simètrica [24][59].

La necessitat de resoldre el càlcul de valors i vectors propis apareix en una gran varietat d'aplicacions, on s'inclouen l'anàlisi dinàmica d'estructures d'avions i coets, la predició del comportament d'estructures mecàniques, l'estudi de convecció solar, l'anàlisi de circuits electrònics i l'anàlisi estadística de dades. És per això que és important trobar mètodes més ràpids per resoldre el càlcul de valors i vectors propis.

El mètode més àmpliament utilitzat per a la resolució del problema dels valors i vectors propis en un computador escalar és el mètode iteratiu QR [24]. Aquest mètode es considera el més eficient per màquines escalars perque requereix menys operacions aritmètiques que qualsevol altre mètode. El problema que té, però, és que el tipus d'operacions que es realitzen presenten molt poc paral.lelisme de manera que és molt poc eficient quan s'executa en multicomputadors.

Si la matriu A és simètrica podem aplicar el mètode de Jacobi. Malgrat que requereix més operacions que el mètode de la iteració QR, les operacions permeten un grau de paral.lelisme més gran, cosa que el fa més atractiu per a computadors paral.lels.

1.2.2 Mètodes de Jacobi

El mètode de Jacobi és un mètode iteratiu. En cada iteració, un element no diagonal de la matriu, a_{pq}, i el seu simètric, a_{qp}, són anul.lats mitjançant l'aplicació d'una transformació similar, R_{pq} [24]. El procés finalitza quan la matriu original es transforma en una matriu amb estructura més simple, generalment una matriu diagonal, a partir de la qual és més fàcil obtenir els valors i vectors propis. El fet que s'utilitzin transformacions similars fa que els valors propis de la matriu resultant després d'aplicar les transformacions, coincideixin amb els valors propis de la matriu original. En concret, els valors propis de la matriu són els elements de la diagonal de la matriu resultant. Els vectors propis són les columnes d'una matriu, l'anomenarem U, que és el resultat de l'acumulació de totes les transformacions similars que s'han aplicat a la matriu original A.

Les transformacions similars que s'utilitzen en el mètode de Jacobi són rotacions en el pla. Més concretament, donada una iteració k de l'algorisme, la rotació en el pla $R_k(pq)$ s'aplica per tal d'anul.lar un element (a_{pq}) no diagonal de la matriu A i el seu simètric (a_{qp}) de la manera següent:

$$A_{k+1} = R_k^T(pq) \text{ x } A_k \text{ x } R_k(pq) \qquad \text{per k=0,1,2,...} \qquad \text{i} \qquad A_0 = A$$

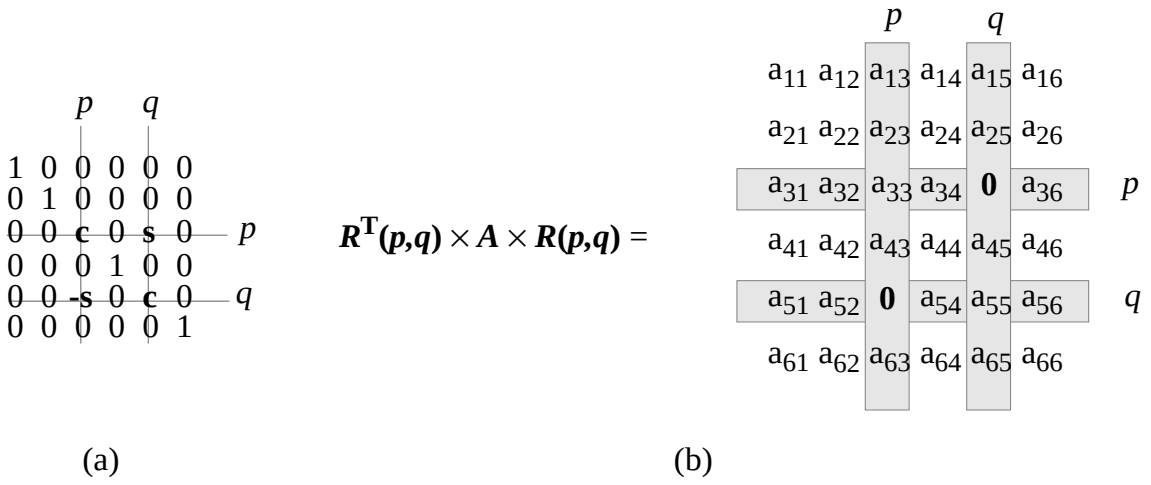


Figura 1.7 a) Matriu de transformació b) Elements modificats en l'aplicació d'una transformació

La transformació similar R_k que es requereix per anul.lar els elements $a_k(p,q)$ i $a_k(q,p)$ té una estructura igual a la que es mostra en la figura 1.7(a), on $s = sin(\alpha_{pq})$ i $c = cos(\alpha_{pq})$. L'angle de rotació, α_{pq}, s'obté a partir dels valors $a_k(p,q)$, $a_k(q,p)$ i $a_k(q,q)$ [24].

Si m és la dimensió de la matriu, són necessàries $m(m-1)/2$ transformacions diferents per anul.lar un sol cop tots els elements que no pertanyen a la diagonal. El grup de transformacions que anul.len tots els elements de fora la diagonal un sol cop s'anomena *escombrat*. El mètode és necessàriament iteratiu ja que l'aplicació d'una transformació pot modificar (donar un valor diferent de zero) algun element que havia estat anul.lat per alguna transformació prèvia. És per aquest motiu que són necessaris varis escombrats per què la matriu original quedi reduïda a una matriu diagonal.

Un dels factors més importants que afecten a la rapidesa amb què convergeix el mètode és l'ordre en què s'apliquen les transformacions dintre d'un escombrat; aquest es diu *ordenació* o *esquema* de Jacobi. S'han proposat diverses ordenacions per completar un escombrat. En l'esquema anomenat Jacobi clàssic, a cada iteració s'elimina l'element no diagonal més gran en valor absolut de la matriu A. Amb aquest esquema, el mètode convergeix molt ràpidament, però el fet d'haver de recórrer tots els elements de la matriu

en cada iteració per buscar quin element és el més gran en valor absolut suposa un cost en temps d'execució considerable. Una solució a aquest problema és eliminar tots els elements de fora la diagonal d'acord un ordre predeterminat (per files, columnes, etc.). Aquest nou esquema s'anomena Jacobi cíclic i malgrat que necessita més iteracions per convergir generalment és més ràpid que el Jacobi clàssic ja que cada escombrat té un cost inferior des del punt de vista de temps d'execució.

Finalment els vectors propis es corresponen a les columnes d'una matriu que s'obté a partir de l'acumulació de totes les rotacions realitzades a la matriu A, $U = I \times R_0 \times R_1 \times R_2 ....$

1.2.3 Mètode de Jacobi *unilateral* i *bilateral*

S'han proposat duess maneres diferents d'aplicar les rotacions en el mètode de Jacobi, que donen lloc al mètode de Jacobi *unilateral* ($one - sided$ [14] [15]) i al mètode de Jacobi *bilateral* ($two - sided$ [59]). Els dos mètodes requereixen el mateix nombre d'operacions sempre i quan es calculin els valors i vectors propis. Els dos mètodes es diferencien bàsicament en l'ordre en que es realitzen les operacions d'una transformació similar.

El mètode que s'ha descrit en la secció anterior correspon al mètode de Jacobi bilateral. En aquest mètode s'ha pogut veure que en cada iteració, k, l'aplicació d'una transformació similar $R_k(pq)$ suposa un premultiplicació de la matriu A_k per la matriu de transformació trasposada $R_k^T(pq)$ i una postmultiplicació de la matriu A resultant per la matriu de transformació $R_k(pq)$. Aquestes dues operacions matricials actualitzen les files p i q i les columnes p i q de la matriu A_k (figura 1.7(b)). Aquest fet fa que el mètode bilateral tingui un cost de comunicació bastant gran quan s'implementa en un sistema multicomputador, on les matrius es distribueixen entre els diferents nodes del multicomputador habitualment per files o per columnes.

Amb el mètode de Jacobi *unilateral* les operacions es realitzen en un ordre tal que a l'hora d'aplicar una transformació tan sols s'actualitzen explícitament les columnes de la matriu. En aquest cas, les operacions s'organitzen de la manera següent:

$$\bar{A}_0 = A \qquad \text{i} \qquad U_0 = I$$
$$\bar{A}_{k+1} = \bar{A}_k \times R_k \qquad \text{i} \qquad U_{k+1} = U_k^T \times R_k \qquad \text{per k=0,1,2,...}$$

Es compleix que:

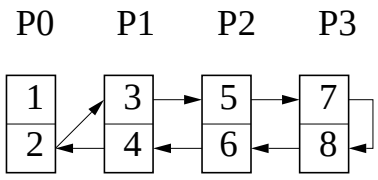
$$A_{k+1} = U_k \times \bar{A}_k$$

Per calcular la matriu de rotació $R_k(pq)$ (angle de rotació) és necessari que es calculin explícitament els elements $a_k(p,q)$, $a_k(q,p)$ i $a_k(q,q)$ de la matriu A_k (matriu que resultaria si s'apliqués el mètode bilateral). Els elements $a_k(p,q)$, $a_k(q,p)$ i $a_k(q,q)$ s'obtenen a partir de $\bar{A}_k$ i U_k tal i com s'indica a continuació:

$$\begin{aligned} a_k(p,q) &= < U_k(*,p), \bar{A}_k(*,q) > \\ a_k(p,p) &= < U_k(*,p), \bar{A}_k(*,p) > \\ a_k(q,q) &= < U_k(*,q), \bar{A}_k(*,q) > \end{aligned}$$

L'expressió $< x, y >$ fa referència al producte intern dels vectors x i y. $\bar{A}(*,p)$ fa referència a la columna p de la matriu $\bar{A}$. Es pot comprovar fàcilment que tant el càlcul de la transformació $R_k(p,q)$ com la seva aplicació requereix únicament operacions amb les columnes p i q de les matrius $\bar{A}$ i U.

El mètode de Jacobi *unilateral* generalment és més ràpid que el bilateral quan s'executa en un multicomputador ja que requereix menys comunicació. Les matrius $\bar{A}$ i U es poden distribuir per columnes i totes les operacions necessàries per calcular i aplicar una rotació poden realitzar-se en un mateix node sense necessitat de cap tipus de comunicació amb els nodes veïns. En canvi, amb el mètode *bilateral* és difícil evitar totalment la comunicació a l'hora de calcular i aplicar una rotació, ja que com s'ha vist anteriorment el mètode modifica tant files i columnes de les dues matrius $\bar{A}$ i U. És per aquest motiu que molts treballs en la literatura es centren en el mètode de Jacobi *unilateral*.



(a)

	P0	P1	P2	P3
etapa 0	(1,2)	(3,4)	(5,6)	(7,8)
etapa 1	(1,4)	(2,6)	(3,8)	(5,7)
etapa 2	(1,6)	(4,8)	(2,7)	(3,5)
etapa 3	(1,8)	(6,7)	(4,5)	(2,3)
etapa 4	(1,7)	(8,5)	(6,3)	(4,2)
etapa 5	(1,5)	(7,3)	(8,2)	(6,4)
etapa 6	(1,3)	(5,2)	(7,4)	(8,6)

(b)

Figura 1.8 Ordenació de Jacobi *round robin*. (a) Patró de comunicació (b) Emparellaments en un escombrat

1.2.4 Paral.lelisme dels mètodes de Jacobi

Una propietat molt interessant del mètode de Jacobi es que podem aplicar vàries rotacions en paral.lel, per anul.lar varis elements a la vegada, reduint d'aquesta manera el temps necessari pera completar un escombrat. Per exemple, en una matriu qualsevol podem anul.lar els elements $A(1,2)$ i $A(3,4)$ (i els seus simètrics) en paral.lel, perquè la transformació $R(1,2)$ tan sols modifica les columnes 1 i 2 de la matriu i la transformació $R(3,4)$ tan sols modifica les columnes 3 i 4 de la matriu. En aquest context, aquestes dues transformacions es diuen *transformacions independents*. En general, dues transformacions $R(i,j)$ i $R(r,s)$ són independents si es compleix que $i \neq j$, $i \neq s$, $j \neq r$ i $j \neq s$.

Aquesta propietat ha motivat la cerca d'ordenacions de Jacobi paral.leles en les que les transformacions similars per completar un escombrat s'organitzin en grups de rotacions independents. Cadascun d'aquests grups s'anomena una etapa. Els algorismes paral.lels que apliquen aquestes ordenacions de Jacobi poden explotar millor el paral.lelisme inherent d'un multicomputador ja que el treball associat a cada etapa es pot distribuir entre els diferents nodes del multicomputador i realitzar-se en paral.lel.

En la figura 1.8(b) es mostra els diferents grups de rotacions independents que s'executen en cada etapa quan s'aplica l'ordenació *round robin* [15]. En l'exemple es considera una línia de 4 processadors i una matriu de dimensió 8. Inicialment cada node de la línia rep dues columnes de la matriu A i de la matriu U. El processador $P0$ rep les columnes 1 i 2, s'indica amb la tupla $(1,2)$, el processador $P1$ les columnes $(3,4)$, el processador $P2$ les columnes $(5,6)$ i el processador $P3$ les columnes $(7,8)$. Cada processador pot calcular i aplicar una transformació. En l'exemple, quatre transformacions s'apliquen en paral.lel. Una etapa està formada per un grup de transformacions independents que s'apliquen en paral.lel.

Després de cada intercanvi i seguint el patró de comunicació que s'indica em la figura 1.8(a) cada node emmagatzema una parella nova de columnes amb les que pot calcular i aplicar una nova transformació.

Un escombrat es realitza amb el mínim nombre d'etapes, 7 en l'exemple. A més en cada etapa tots els nodes realitzen la mateixa quantitat de càlculs, els corresponents a l'aplicació d'una transformació similar. Des d'aquest punt de vista, aquesta ordenació és òptima.

S'han proposat diverses ordenacions de Jacobi per diferents escenaris (sistòlics [4][40], multiprocessadors amb memòria compartida [2], hipercubs [21][43], multicomputadors en malla [15]). En general, les característiques desitjables per qualsevol ordenació de Jacobi que s'hagi d'aplicar en un multicomputador són:

- Que un escombrat es realitzi amb el nombre mínim d'etapes amb una distribució equilibrada de la càrrega. En el cas de què la matriu a diagonalitzar és de dimensió m, s'han de realitzar un total de $m(m-1)/2$ rotacions i com a màxim podem realitzar $m/2$ rotacions en paral.lel. Per tant, per completar un escombrat necessitarem un mínim de $m-1$ etapes si m és parell i m etapes si m és imparell.
- Aconseguir que al final de cada etapa de càlcul l'intercanvi entre nodes, anomenat a partir d'ara transició, involucri la mínima quantitat de dades i es realitzi entre nodes el més propers possible (el cas ideal seria que totes les comunicacions foren locals, entre nodes veïns en el multicomputador). Si analitzem el patró de comunicació

de l'ordenació *round robin*, que es mostra en la figura 1.8(a), veiem que en una transició tan sols hi ha intercanvi d'informació entre nodes veïns però s'han de moure moltes dades ja que en cada transició un node intercanvia dades amb els seus dos veïns: s'han d'enviar dos missatges i s'han de rebre dos missatges.

És important assenyalar que en aquest cas, en cada pas de comunicació es mou tota la informació (els dos blocs de columnes s'intercanvien amb els veïns) mentre que en altres ordenacions proposades tan sols es requereix moure la meitat de la informació.

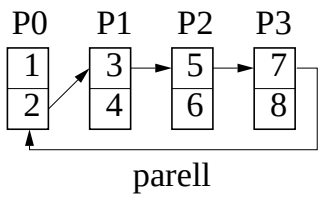
S'ha de tenir clara la diferència entre ordenació de Jacobi i algorisme de Jacobi. Una ordenació, pensada pera una determinada topologia, ens defineix bàsicament els grups de transformacions independents que s'han de realitzar en cadascuna de les etapes necessàries per completar un escombrat. Els dos elements bàsics per definir una ordenació són: pera quina topologia està pensada i quin és el patró de comunicació entre els nodes per generar els diferents grups de transformacions independents.

L'algorisme de Jacobi determina en quin ordre es realitzen les operacions de càlcul i quin tipus d'operacions de comunicació són necessàries per completar un escombrat. En el nostre treball hem proposat algorismes nous que fan servir ordenacions ja proposades i a més a més es proposen nous algorismes que apliquen noves ordenacions de Jacobi.

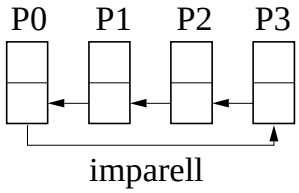
1.2.5 Ordenacions de Jacobi per Multicomputadors

Són moltes i molt diverses les propostes d'ordenacions de Jacobi que poden trobar-se en la literatura [14][15][40][4][2][21]. A continuació es descriuen dues ordenacions les quals són els punt de partida d'algunes de les propostes que es realitzen en aquest treball.

Per cadascuna de les ordenacions es descriu quina és la distribució de dades i quin és el patró de comunicació que s'aplica per generar els diferents grups de transformacions independents per completar un escombrat.



(a)



(b)

	P0	P1	P2	P3
etapa 0	(1,2)	(3,4)	(5,6)	(7,8)
etapa 1	(1,7)	(2,4)	(3,6)	(5,8)
etapa 2	(1,4)	(2,6)	(3,8)	(5,7)
etapa 3	(1,5)	(4,6)	(2,8)	(3,7)
etapa 4	(1,6)	(4,8)	(2,7)	(3,5)
etapa 5	(1,3)	(6,8)	(4,7)	(2,5)
etapa 6	(1,8)	(6,7)	(4,5)	(2,3)

(c)

	P0	P1	P2	P3
	(1,2)	(8,7)	(6,5)	(4,3)
	(1,4)	(2,7)	(8,5)	(6,3)
	(1,7)	(2,5)	(8,3)	(6,4)
	(1,6)	(7,5)	(2,3)	(8,4)
	(1,5)	(7,3)	(2,4)	(8,6)
	(1,8)	(5,3)	(7,4)	(2,6)
	(1,3)	(5,4)	(7,6)	(2,8)

(d)

Figura 1.9 Intercanvi de columnes en l'ordenació *Eberlein*:(a)Patró de comunicació per etapes parells. (b)Patró de comunicació per etapes imparells. (c)Emparellament de les columnes en etapes parells. (d)Emparellament de les columnes en etapes imparells.

Eberlein

En aquesta secció es descriu un algorisme paral.lel que utilitza l'ordenació de Jacobi proposada a [14]. Originalment, aquesta ordenació ha estat proposada per multicomputadors amb una organització unidimensional dels seus nodes en anell. A continuació es descriu l'algorisme suposant que tenim un anell amb un total de 2^d nodes i que el tamany de la matriu és $m = 2^{d+1}$. Inicialment, cada node de l'anell té assignades dues columnes de la matriu $\bar{A}_0 = A$ i les corresponents columnes de la matriu $U_0 = I$. En la figura 1.9(a) hi ha un exemple d'aquesta distribució inicial pel cas particular de $2^d = 4$, les columnes de la matriu $\bar{A}$ i de la matriu U s'han numerat de 1 fins a m.

Cada escombrat consta de $2^{d+1} - 1$ etapes, cada etapa consta de $m/2$ transformacions independents cadascuna de les quals es realitza en un node diferent de l'anell. En cada etapa, cada node calcula i aplica una transformació independent i intercanvia

una columna d'$\bar{A}$, i la corresponent columna d'U, amb un dels nodes veïns en l'anell. L'intercanvi de les columnes es realitza d'acord als patrons de comunicació que es mostren en les figures 1.9(a) (per les etapes parells) i 1.9(b) (per les etapes imparells). En la figura 1.9(c) es mostren les parelles de columnes que constitueixen cadascuna de les etapes del primer escombrat. En aquesta figura, cada parella (i, j) indica que el node té assignades les columnes i i j en l'etapa corresponent, i per tant en aquesta etapa calcula i aplica la rotació $R(i, j)$ actualitzant les matrius $\bar{A}$ i U convenientment. El mètode *unilateral* assegura que les dades necessàries per calcular i aplicar la rotació, és a dir les columnes i i j, estan assignades al mateix node per evitar d'aquesta manera qualsevol tipus de comunicació. En la figura 1.9(d) es mostren els aparellaments de les columnes que es realitzen en cadascuna de les etapes que constitueixen el segon escombrat. En general, en els escombrats imparells es realitzen sempre els aparellaments que es mostren en la figura 1.9(c) i en els escombrats parells es realitzen sempre els aparellaments que es mostren en la figura 1.9(d).

L'extensió de l'algorisme per matrius de qualsevol valor d'm és immediat. En aquest cas, les columnes de les matrius $\bar{A}$ i U s'organitzen en 2^{d+1} blocs de $m/2^{d+1}$ columnes consecutives cadascun. Ara, un índex i identifica a un bloc de columnes i l'aparellament (i, j) identifica totes les transformacions que resulten de la combinació de les columnes dels dos blocs i i j. En cadascuna de les $2^{d+1} - 1$ etapes, cada node ha de realitzar un total de $m/2^{d+1} \times m/2^{d+1}$ transformacions corresponents a l'aparellament de cada columna del bloc i amb totes les columnes del bloc j. A més a més, en la primera etapa de cada escombrat, els nodes han de realitzar $m/2^{d+1} \times (m/2^{d+1} - 1)$ transformacions addicionals que corresponen als aparellaments de cada columna d'un bloc amb la resta de columnes del mateix bloc. Finalment, en lloc de columnes són blocs de columnes el que s'han d'intercanviar al final de cada etapa.

En la figura 1.10 es mostra un exemple de tots els aparellaments que s'han de realitzar en cadascuna de les etapes pel cas concret de 4 nodes i una matriu de dimensió $m = 16$, o sigui amb blocs de dues columnes. En la figura els aparellaments en negreta (i, j) fan referència als blocs de columnes i i j, composats per les columnes $2i - 1$ i $2i$ i $2j - 1$ i $2j$ respectivament. A sota d'aquests es detalla quines transformacions es realitzen com a conseqüència de la combinació de les columnes d'un bloc amb les columnes de l'altre

	P0	**P1**	**P2**	**P3**
	(1,2)	**(3,4)**	**(5,6)**	**(7,8)**
etapa 0	(1,2) (3,4)	(5,6) (7,8)	(9,10) (11,12)	(13,14) (15,16)
	(1,3) (2,4)	(5,7) (6,8)	(9,11) (10,12)	(13,15) (14,16)
	(1,4) (2,3)	(5,8) (6,7)	(9,12) (10,11)	(13,16) (14,15)
etapa 1	**(1,7)**	**(2,4)**	**(3,6)**	**(5,8)**
	(1,13) (2,14)	(3,7) (4,8)	(5,11) (6,12)	(9,15) (10,16)
	(1,14) (2,13)	(3,8) (4,7)	(5,12) (6,11)	(9,16) (10,15)
etapa 2	**(1,4)**	**(2,6)**	**(3,8)**	**(5,7)**
	(1,7) (2,8)	(3,11) (4,12)	(5,15) (6,16)	(9,13) (10,14)
	(1,8) (2,7)	(3,12) (4,11)	(5,16) (6,15)	(9,14) (10,13)
etapa 3	**(1,5)**	**(4,6)**	**(2,8)**	**(3,7)**
	(1,9) (2,10)	(7,11) (8,12)	(3,15) (4,16)	(5,13) (6,14)
	(1,10) (2,9)	(7,12) (8,11)	(3,16) (4,15)	(5,14) (6,13)
etapa 4	**(1,6)**	**(4,8)**	**(2,7)**	**(3,5)**
	(1,11) (2,12)	(7,15) (8,16)	(3,13) (4,14)	(5,9) (6,10)
	(1,12) (2,11)	(7,16) (8,15)	(3,14) (4,13)	(5,10) (6,9)
etapa 5	**(1,3)**	**(6,8)**	**(4,7)**	**(2,5)**
	(1,5) (2,6)	(11,15) (12,16)	(7,13) (8,14)	(3,9) (4,10)
	(1,6) (2,5)	(11,16) (12,15)	(7,14) (8,13)	(3,10) (4,9)
etapa 6	**(1,8)**	**(6,7)**	**(4,5)**	**(2,3)**
	(1,15) (2,16)	(11,13) (12,14)	(7,9) (8,10)	(3,5) (4,6)
	(1,16) (2,15)	(11,14) (12,13)	(7,10) (8,9)	(3,6) (4,5)

Figura 1.10 Sequència detallada dels aparellaments que es realitzen en cada etapa quan s'aplica l'ordenació *eberlein* per 4 nodes i una matriu de dimensió $m = 16$

bloc. En la primera etapa s'apliquen, a més a més, les transformacions que resulten de la combinació de les columnes d'un bloc entre elles mateixes.

Aquest algorisme té dues característiques destacables que afavoreixen la seva execució paral.lela. Primer es pot observar que amb aquest algorisme s'aconsegueix realitzar un escombrat amb el nombre mínim d'etapes. Per una altra banda, a l'hora de realitzar l'intercanvi de columnes al final de cada etapa, es requereix tan sols que cada node intercanviï un únic bloc de columnes de cadascuna de les dues matrius A i U amb un node veí en l'anell, a diferència de l'ordenació *round robin* en la que és necessari intercanviar els dos blocs de columnes que cada node té assignat.

Aquest és l'algorisme base del qual parteix el nou algorisme proposat per una malla multidimensional que es descriu en el capítol 3, l'algorisme $2D$.

Block Recursive

A continuació es descriu l'algorisme que utilitza una ordenació pensada per un multiprocessador amb una topologia en hipercub proposat per Gao i Thomas [21] i revisat posteriorment a [42].

Descriurem l'algorisme mitjançant un exemple concret en el que la dimensió de l'hipercub és $d = 3$ (vuit processadors) i la dimensió de la matriu és $m = 16$. Les columnes de la matriu s'han numerat de l'1 fins al 16. Inicialment, s'assignen a cada node un parell de columnes de la matriu $\bar{A}_0 = A$, i el mateix parell de columnes de la matriu U. En cada parella, la columna amb l'índex més gran s'identificarà a partir d'ara com a columna inferior i la columna amb l'índex més petit s'identificarà a partir d'ara com a columna superior. L'exemple de la figura 1.11 concretament mostra els aparellaments corresponents al primer escombrat complet i algunes etapes del segon escombrat. La distribució inicial es pot veure en el primer rectangle de la mateixa.

Els aparellaments de columnes en la distribució inicial corresponen a la primera etapa de l'ordenació BR. Els següents 14 rectangles de la figura corresponen a la resta d'etapes del primer escombrat. Si una parella de columnes (i, j) es troben en un determinat node en una determinada etapa, llavors aquest node és responsable de calcular i aplicar la transformació corresponent, $R(i, j)$. La transformació es pot aplicar sense necessitat d'intercanviar cap tipus d'informació ja que totes les dades necessàries es troben al node en questió.

En cada escombrat podem distingir tres tipus diferents de fases que s'anomenen *fase d'intercanvi, fase de divisió* i *última transició*. Una fase d'intercanvi consisteix en una sèrie d'etapes (tal i com s'ha definit anteriorment, una etapa és un conjunt de rotacions independents), cadascuna d'elles seguida d'una transició. Una transició implica l'intercanvi de les columnes, corresponents als blocs inferiors, entre nodes veïns en una

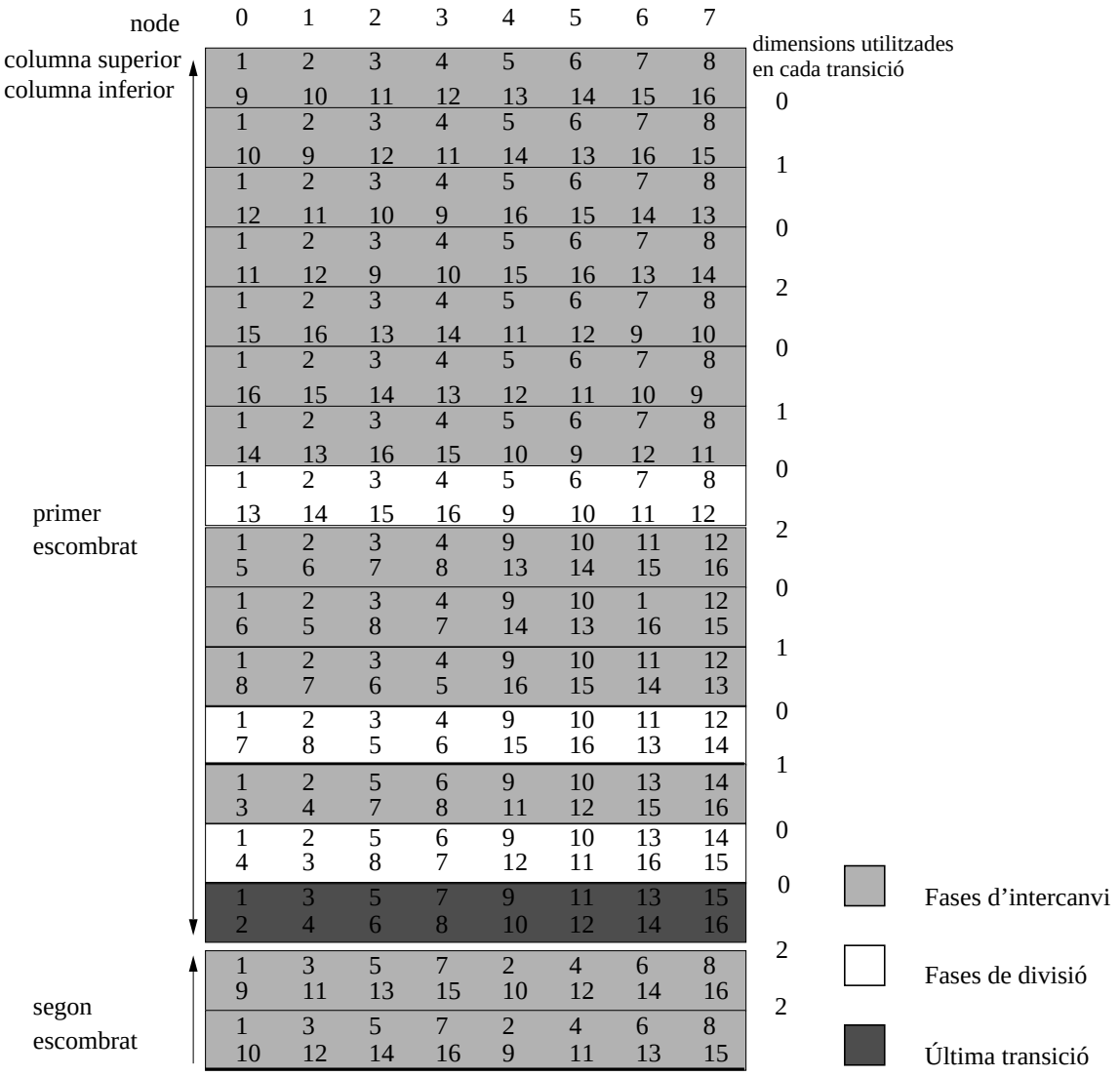


Figura 1.11 Sequència d'etapes de l'algorisme BR per $d = 3$

determinada dimensió de l'hipercub. Tots els intercanvis d'una transició es realitzen al mateix temps i tots els nodes utilitzen la mateixa dimensió de l'hipercub.

En cadascuna de les etapes de la fase d'intercanvi, una columna inferior visita un node diferent. En la primera fase d'intercanvi, cada columna superior s'aparella amb totes les columnes inferiors. Per exemple, en la figura 1.11 es pot veure que en la primera fase d'intercanvi en el node 3 la columna 4 s'aparella amb totes les columnes inferiors, les

columnes de la 9 fins la 16. En l'exemple que es mostra en la figura 1.11, en la primera transició de la primera fase d'intercanvi, cada node intercanvia la seva columna inferior amb el node veí en la dimensió 0 de l'hipercub. La resta de transicions necessàries per completar la primera fase d'intercanvi es realitzen a través de les dimensions $1, 0, 2, 0, 1$ i 0, tal i com es pot veure en la figura 1.11.

Després de cada fase d'intercanvi es realitza una fase de divisió. Una fase de divisió consisteix en una única etapa seguida per una transició. En la transició de la primera fase de divisió els nodes amb un índex $i < 2^{d-1}$ intercanvien la seva columna inferior amb la columna superior que pertany als nodes veïns en la dimensió $d-1$ (2 en l'exemple). Després d'aquest intercanvi, cada node reordena les columnes de manera que la columna amb l'índex més petit passa a ser la columna superior i la columna amb l'índex més gran passa a ser la columna inferior. En la figura 1.11 el node 4 abans de la fase de divisió té assignades les columnes 5 i 9, sent la columna 9 la columna inferior ja que $9 > 5$; després de la fase de divisió el node 4 té les columnes 9 i 13, com que ara $9 < 13$ la columna 9 passa a ser una columna superior i la columna 13 passa a ser una columna inferior. És important adonar-se que després d'aquest intercanvi, totes les columnes que eren columnes superiors en la etapa d'intercanvi anterior, estan assignades als nodes d'un mateix subcub (el subcub format pels nodes 0, 1, 2 i 3 en l'exemple), i el mateix passa amb totes les columnes inferiors, les quals queden repartides entre tots els nodes d'un mateix subhipercub, en l'exemple els nodes 4, 5, 6 i 7.

En la segona fase d'intercanvi es repeteix el mateix esquema en cadascun dels dos subcubs de dimensió $d-1$ que s'han generat. En la segona fase d'intercanvi s'aparellen totes les columnes superiors amb totes les columnes inferiors que estan en el mateix subcub de dimensió 2. La transició de la segona fase de divisió es realitza a través de la dimensió $d-2$ (1 en l'exemple). Igual que en la primera fase de divisió, les columnes inferiors d'un subcub s'intercanvien amb les columnes superiors de l'altre subcub. En el nostre exemple, aquesta segona fase d'intercanvi consisteix en tres etapes i tres transicions a través de les dimensions 0, 1 i 0. De nou, cada transició implica l'intercanvi de les columnes inferiors.

En general, per un hipercub de dimensió d, un escombrat consisteix en d fases d'intercanvi, cadascuna d'elles seguida per una fase de divisió. Després de l'última fase de divisió es realitza l'última transició. Aquesta última transició consisteix en una etapa seguida d'una transició a través de la dimensió $d-1$ (2 en l'exemple). En aquesta transició s'intercanvien les columnes superiors i les columnes inferiors seguint el mateix esquema de comunicació que s'aplica en la primera fase de divisió. Després d'aquesta última transició j'ha s'ha complert l'escombrat. A partir d'aquest punt es generen de nou els aparellaments inicials però en nodes diferents. Per tal d'obtenir els mateixos aparellaments que els que es realitzen en el primer escombrat, les transicions que s'havien realitzat a través de les dimensions 0, 1 i 2 en el primer escombrat, en el segon escombrat es realitzen per les dimensions 2, 0 i 1 respectivament. En el tercer escombrat les transicions s'han de realitzar a través de les dimensions 1, 2 i 0. Després de l'última transició del $d-1$-èssim escombrat es torna a generar la distribució original i els intercanvis es realitzen tal i com es van realitzar en el primer escombrat. En la figura 1.11 es mostren algunes de les etapes del segon escombrat.

En general, si tenim un hipercub de dimensió d, cada escombrat té d fases d'intercanvi, d fases de divisió i una última transició. Farem servir l'índex e ($e \in [d,1]$) per identificar cadascuna de les d fases d'intercanvi.

La fase d'intercanvi e consta de 2^e-1 intercanvis. Anomenem D_e^{BR} a la tupla que identifica l'ordre en què s'utilitzen les dimensions durant la fase d'intercanvi e del primer escombrat. D_e^{BR} es pot obtenir de forma sistemàtica:

$$\begin{aligned} D_1^{BR} &= <0> \\ D_i^{BR} &= <D_{i-1}^{BR}, i-1, D_{i-1}^{BR}> \qquad 1 < i \leq e \end{aligned}$$

Per exemple, si $e=4$, la seqüència que s'obté és $D_4^{BR} = \{010201030102010\}$.

Per obtenir les tuples de D_e^{BR} corresponents a les etapes d'intercanvi del j-éssim escombrat ($j \geq 0$) , cadascun dels valors de k ($k \in [0..d-1]$) de les tuples D_e^{BR} del primer es-

combrat es substitueix pel valor $(k-j)$ mod d. Per exemple, la tupla D_4^{BR}corresponent a l'escombrat 2 és $D_4^{BR} = \{232023212320232\}$.

L'extensió de l'algorisme per matrius de qualsevol dimensió m és immediat. En aquest cas les columnes de les matrius $\bar{A}$ i U s'organitzen de nou en 2^{d+1} blocs de $m/2^{d+1}$ columnes consecutives cadascun. De nou, un índex i identifica un bloc de columnes i la parella (i, j) identifica totes les transformacions que resulten de la combinació de les columnes dels dos blocs i i j entre ells. En cada etapa un node ha de realitzar $m/2^{d+1} \times m/2^{d+1}$ transformacions corresponents a l'aparellament de les columnes del bloc i amb les columnes del bloc j. A més a més en la primera etapa s'han de realitzar $m/2^{d+1} \times (m/2^{d+1} - 1)$ transformacions que resulten de l'aparellament d'una columna amb la resta de columnes del mateix bloc al que pertany. Finalment, al final de cada etapa es realitza l'intercanvi d'un bloc de columnes sencer. És important assenyalar que el concepte de columna superior i columna inferior es segueix aplicant però a nivell de bloc superior i bloc inferior.

En [42], a més a més de completar l'ordenació de Jacobi de *Gao i Thomas* definint l'última transició, també es demostra la correctesa del mètode. L'ordre resultant s'anomena *Block Recursive (BR)*.

Resumint, aquesta ordenació que s'acaba de descriure té les següents propietats:

- És una ordenació pensada per a una topologia en hipercub.
- En cada operació de comunicació es requereix un mínim moviment de dades (un node envia i rep un únic missatge) entre els nodes veïns en una determinada dimensió (comunicacions locals).
- Un escombrat es realitza amb un nombre mínim d'etapes.

Aquest algorisme serà el punt de partida de noves propostes que es realitzen en aquest treball en el capítol 2.

1.3 ESCALABILITAT

L'escalabilitat és una propietat desitjable dels sistemes paral.lels. Tothom té una noció intuitiva del concepte d'escalabilitat; un sistema es considera escalable si és capaç d'utilitzar de forma eficient un nombre creixent de recursos del sistema [5][30][60][36].

La definició intuitiva ha de concretar-se en una definició més rigurosa per tal que els resultats d'una anàlisi d'escalabilitat puguin ser interpretats de forma adequada. En la literatura s'han proposat diverses definicions d'escalabilitat de forma més rigurosa [36]. Definicions que per ser diferents poden donar resultats diferents e inclús contradictoris quan s'apliquen a un cas concret.

Per exemple, a [26] es proposa una metodología d'anàlisi de l'escalabilitat, anomenada *isoeficiència*. Aquesta metodologia s'aplica a sistemes, es a dir, al conjunt format per una arquitectura determinada i un algorisme determinat. Amb aquesta metodologia, un sistema és escalable si quan s'incrementa el nombre de nodes és possible resoldre un problema de tamany raonable (amb l'algorisme que s'està avaluant) que permeti mantenir l'eficiència paral.lela del sistema a un valor predeterminat. Podem observar que aquest anàlisi pot ser molt valuós per un arquitecte de sistemes mentre que per un usuari final la informació que li proporciona és menys valuosa, ja que molt probablement un usuari final estarà interessat en altres paràmetres com per exemple el temps d'execució o bé increment del tamany del problema.

Davant la diversitat de metodologies i resultats és important tenir clar per una part que és el què interessa a l'usuari que realitza l'anàlisi i per una altra part quin tipus d'anàlisi realitza el mètode escollit; d'aquesta forma sabrem cóm interpretar de forma correcta els resultats obtinguts.

1.4 RESUM DE LES CONTRIBUCIONS D'AQUEST TREBALL

Un cop establert el contexte del treball, és possible descriure amb més detall les principals contribucions del mateix:

- En el capítol 2, es proposen un conjunt d'algorismes paral.lels per hipercubs multiport (BR segmentat, α-optimum, $Grau$-4) tots ells basats en l'algorisme $Block\ Recursive$ proposat a [21] i revisat a [42]. Amb els nous algorismes s'aconsegueix reduir considerablement el cost de la comunicació respecte al cost de la comunicació de l'algorisme original BR. Aquesta contribució ha donat lloc a dues publicacions [53][52].
- En el capítol 3, es proposa un nou algorisme anomenat algorisme $2D$ amb una topologia de comunicació bidimensional en malla. Aquest algorisme està basat en l'algorisme proposat en [14] per a un anell de processadors. Segons siguin les característiques del sistema paral.lel on s'executi, amb el nou algorisme s'aconsegueix reduir considerablement el cost respecte al cost de l'algorisme original. Aquesta contribució ha donat lloc a una publicació a [54].
- En el capítol 4, es presenta un nou algorisme paral.lel per la resolució dels valors i vectors propis en malles i torus bidimensionals que resulta del mapeig de l'algorisme $BR\ segmentat$ inicialment pensat per un hipercub en una topologia en malla o torus bidimensional. Amb aquest algorisme s'aconsegueix reduir el cost de la comunicació respecte al cost de la comunicació de l'algorisme $2D$, especialment en sistemes amb una arquitectura *multiple-port* amb un nombre de nodes moderat (fins a *256* nodes).
- En el últim capítol, capítol 5, es proposa una metodologia d'anàlisi de l'escalabilitat de sistemes paral.lels orientada a l'usuari final del sistema. Més concretament es proposen tres mètodes diferents de realitzar l'anàlisi segons les necessitats de l'usuari. Des d'aquesta prespectiva l'anàlisi d'escalabilitat que es proposa en aquest treball proporciona suficient informació a l'usuari final de com evoluciona el sistema quan aquest creix (augmenta el nombre de nodes) de forma que l'usuari és capaç per decidir si és rentable o no afegir més nodes al sistema. Aquesta contribució ha donat lloc a una publicació a [55].

1.5 METODOLOGIA D'AVALUACIÓ

Per avaluar l'eficiència dels algorismes que es proposen en aquesta tesi s'utilitzen models analítics. Els models analítics tenen els següents avantatges:

- Per una part permeten realitzar avaluacions ràpides (creació del model) si ho comparem amb altres mètodes d'avaluació com la simulació i l'execució en màquines reals.
- Per una altra banda, els resultats que s'obtenen mitjançant aquest mètode són reproduïbles, és a dir, es fàcil generar-los de nou i per tant permeten realitzar fàcilment comparacions amb nous algorismes de forma ràpida i fàcil.

Podríem considerar com un desavantatge dels models analítics el grau de precisió de l'anàlisi que es realitza, és a dir, és molt difícil construir models analítics molt precisos i que tinguin en compte tots els factors que influeixen en un sistema. D'aquí que amb els models analítics tan sols és possible realitzar avaluacions qualitatives. Un model no ens permet saber quin serà el comportament real de l'algorisme quan s'executi en un sistema concret, sinó que el model ens dóna una idea del comportament de l'algorisme en un entorn que tingui unes característiques similars a les que s'han tingut en compte a l'hora de crear-lo.

En aquest treball els models analítics ens permeten mostrar que els algorismes proposats són prometedors i, que per tant, pot ser interessant la seva implementació en màquines reals.

A l'hora de definir els models s'han tingut en compte paràmetres que depenen de l'arquitectura, paràmetres que depenen de l'algorisme i paràmetres que depenen del problema.

Els paràmetres que depenen de l'arquitectura que considerem en els nostres models són:

- T_c : temps de realitzar una operació en coma flotant.

- T_e : temps de transmetre un element en coma flotant doble precisió.
- T_s : temps d'iniciar una operació de comunicació ($start - up$).
- d : fa referència a la dimensió de l'hipercub i ens diu quin nombre de nodes té els sistema (2^d).

El paràmetre que depèn del problema és:

- m : tamany del problema, dimensió de la matriu.

Respecte als paràmetres que depenen de l'algorisme s'especificaran més endavant en la desripció de cadascun dels algorismes proposats.

El cost d'enviar un missatge de tamany L entre dos nodes veïns a distància 1 es modelitza com [5]:

$$T_s + L \times T_e \tag{1.2}$$

Quan els algorismes tan sols realitzen comunicacions amb els nodes veïns (locals a distància 1), l'expressió 1.2 és vàlida per qualsevol model de comunicació que s'utilitzi, *store and forward*, *circuit switching* o *cut trough* (*wormhole* o *virtual cut through*).

En el cas de què l'intercanvi d'informació sigui entre nodes no veïns, tenint en compte que en aquest treball tan sols es consideren multicomputadors amb el model de comunicació *wormhole*, el cost d'enviar un missatge es modelitza de la manera següent:

$$T_s + n \times T_i + L \times T_e \tag{1.3}$$

on n fa referència al nombre de nodes (exceptuant els nodes origen i destí) intermitjos que s'han de travessar i t_i el cost de travessar cadascun d'aquests nodes. El cost de travessar un node (t_i) és el suficientment petit per a considerar-se despreciable (sempre

i quan no hagin conflictes), és per això que el cost d'enviar un missatge entre nodes no veïns coincidirà amb el cost d'enviar un missatge entre nodes veïns, el qual es pot expressar com s'indica en 1.2.

Considerem el cas d'enviar C missatges per diferents enllaços. Si l'arquitectura és $one-port$ s'han d'enviar seqüencialment un darrera l'altre, un missatge no pot enviar-se fins que el missatge anterior no ha estat rebut. En aquest cas, el cost es modelitza com:

$$\sum_{i=1}^{C} T_s + L_i \times T_e \tag{1.4}$$

sent L_i el missatge que s'envia per l'enllaç i.

Si l'arquitectura és $all-port$, els missatges es poden enviar en paral.lel. Considerem que el node es queda bloquejat fins que l'últim missatge arribi al seu destí. La inicialització ($start-up$) de cadascun d'aquests missatges que s'envien en paral.lel no es pot solapar ja que és una tasca que l'ha de realitzar el node per cadascun del missatges que s'envien. Per tant, el temps d'enviar C missatges per C enllaços diferents de l'hipercub es pot modelitzar de la manera següent:

$$C \times T_s + L_{max} \times T_e \tag{1.5}$$

on L_{max} és la longitud del missatge més llarg. L'expressió 1.5 és una fita superior del temps de comunicació ja que es correspon al cas en què el missatge més llarg és l'últim que s'envia. Aquesta presuposició simplifica els models que es desenvoluparan en capítols posteriors.

L'expressió 1.5 considera que no hi ha contenció en el bus intern ni en el bus de memòria. Aquesta contenció pot ser deguda a l'organització de la memòria i pot no ser nul.la en el cas concret d'alguna màquina real. No obstant, si l'arquitectura permet enviar més d'un missatge en paral.lel aquesta contenció ha de tenir molt poc pes; en

	T_s / T_c	T_e / T_c
iPSC/860	6400	160
nCUBE 2	300	7.5
PARAGON	1500	2.28
Cray T3D	1974	27.85

Figura 1.12 Temps de $star - up$ i temps d'enviar un real amb doble precisió

cas contrari, l'eficiència que pot obtenir-se serà com si estéssim enviant els missatges seqüencialment.

Per validar els models els hem avaluat per un rang de valors molt ampli de forma que es pugui veure el comportament dels algorismes per entorns amb qualsevol valor dels paràmetres T_e, T_s i T_c. A més a més, els models també s'han avaluat tenint en compte la relació de paràmetres T_c, T_e i T_s d'algunes de les màquines reals representatives de les topologies que s'estudien en aquest treball [12][49]. Les màquines considerades i la relació de paràmetres es mostren en la taula de la figura 1.12. Concretament, les dades de la taula 1.12 s'han obtingut de [5].

2

CÀLCUL DE VALORS I VECTORS PROPIS EN HIPERCUBS *MULTIPLE – PORT*

Resum

En la literatura s'han proposat diversos algorismes per resoldre el problema del càlcul de valors i vectors propis en multicomputadors amb una topologia en hipercub [3] [21] [42]. En tots aquests algorismes (un d'ells l'algorisme Block recursive descrit en el capítol 1) en cada operació de comunicació els processos tan sols intercanvien dades amb un únic veí en una determinada dimensió, per tant en una operació de comunicació un node de l'hipercub tan sols utilitza un únic enllaç dels d possibles (sent d la dimensió de l'hipercub). Aquest comportament no treu profit del potencial paral.lelisme de comunicacions que ofereixen les arquitectures multiple − port.

En aquest capítol es proposen nous algorismes, els quals en una operació de comunicació poden utilitzar més d'un enllaç de l'hipercub de forma simultània i per tant exploten el paral.lelisme de comunicacions que ofereix el sistema multiple − port.

El contingut del capítol és el següent: inicialment es recorda el funcionament de l'algorisme Block Recursive descrit en el capítol 1 *el qual és el punt de partida de totes les propostes que es realitzen posteriorment. Per augmentar el paral.lelisme de les comunicacions s'aplica la tècnica de la segmentació de les comunicacions, el funcionament de la mateixa es descriu en la segona part del capítol. Aquesta tècnica s'aplica a l'algorisme Block Recursive amb el que s'aconsegueix una reducció del cost de la comunicació a la meitat. Finalment es presenten els nous algorimes amb els que s'aconsegueix reduir molt més el cost de la comunicació. En la primera proposta el nou algorisme permet utilitzar tots els enllaços de l'hipercub (cada procés d enllaços) de forma simultània amb una reducció del cost de la comunicació agirebé òptima. Aquesta primera proposta tan sols és vàlida per problemes molt grans. En la segona proposta, el nou algorisme permet utilitzar quatre enllaços simultàniament amb una reducció de la comunicació a una quarta part del cost original. Aquesta segona proposta és vàlida per qualsevol tamany del problema.*

2.1 INTRODUCCIÓ

En el cas de màquines paral.leles amb memòria distribuïda (multicomputadors, DSM, NOW's) el cost de la comunicació juga un paper molt important en l'eficiència d'un determinat algorisme [1]. En tots els algorismes de Jacobi que s'han proposat en la literatura per multicomputadors amb una topologia en hipercub [3] [21][42] una operació de comunicació tan sols utilitza un únic enllaç dels d possibles. En aquest capítol es proposen nous algorismes de Jacobi per multicomputadors amb una topologia en hipercub en els que la majoria d'operacions de comunicació permeten explotar el paral.lelisme de comunicació que ofereix un hipercub amb una arquitectura $multiple - port$ [39][35].

El punt de partida de totes les propostes és l'algorisme $Block\ Recursive$ ([42]) descrit en detall en el capítol 1. A continuació es mostrarà com amb els nous algorismes és possible reduir el cost de la comunicació considerablement respecte al cost de comunicació de l'algorisme BR.

2.1.1 Recordatori de l'algorisme BR

En aquesta secció recordarem els punts més importants de l'algorisme BR proposat a [21] i revisat a [42], el qual ha estat descrit en detall en el Capítol 1. En general, considerarem que tenim un hipercub de dimensió d i una matriu de dimensió m.

L'algorisme BR organitza les dades de la matriu $\bar{A}$ i de la matriu U en blocs de columnes consecutives. Un bloc consta de $m/2^{d+1}$ columnes. Inicialment a cada node de l'hipercub se li assignen 2 blocs de columnes de la matriu A i els mateixos dos blocs de la matriu U. En l'exemple de la figura 2.1 cada parella de blocs de columnes assignades a cada node està representat per una tupla (i, j), on i és el bloc de columnes superior i j el bloc de columnes inferior.

Un escombrat consta de $2^{d+1} - 1$ etapes. En cada etapa, un node realitza un conjunt de transformacions com a resultat de l'aparellament de cada columna d'un bloc amb totes les columnes de l'altre bloc. En la primera etapa, a més a més, també s'apliquen

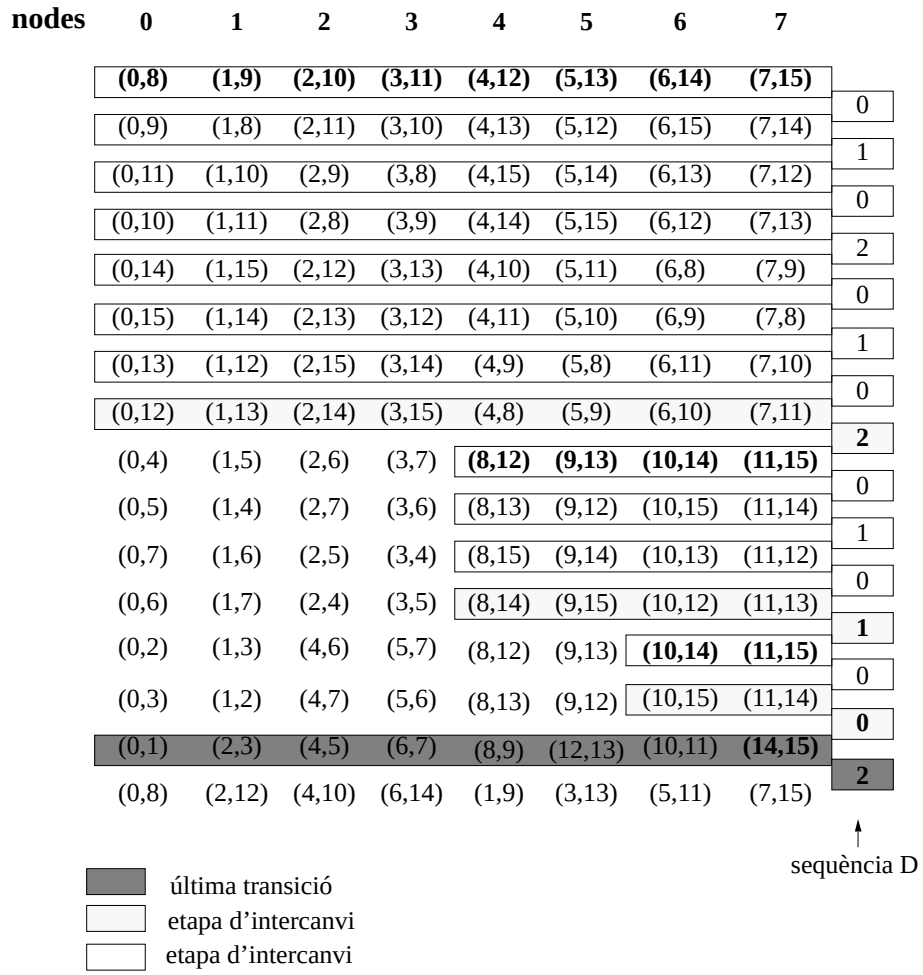


Figura 2.1 Aparellaments d'un escombrat quan s'aplica l'algorisme BR

les transformacions que resulten de l'aparellament de cada columna amb la resta de columnes del mateix bloc. A l'hora de calcular i aplicar cada transformació no és necessari cap tipus de comunicació entre nodes.

Les $2^{d+1}-1$ etapes per completar un escombrat s'agrupen en tres tipus de fases diferents: intercanvi, divisió i última transició. Una fase es composa d'una o més etapes, cadascuna d'elles seguida per una transició. Un escombrat consisteix en d fases d'intercanvi, cadascuna d'elles seguida per una fase de divisió i per últim l'última transició. Les fases d'intercanvi es numeren de d fins a 1. Una fase d'intercanvi e ($e \in [d,1]$)

consisteix en $2^e - 1$ etapes seguides cadascuna d'una transició. Per exemple, la primera fase d'intercanvi (per $e = 3$) de l'exemple que es mostra en la figura 2.1 consta de $2^3 - 1 = 7$ etapes, en cadascuna d'aquestes etapes un node calcula les transicions corresponents a la combinació adequada de les columnes dels dos blocs que té assignats, i un cop totes les transformacions han estat aplicades es realitza un intercanvi (transició) del bloc inferior amb algun node veí en l'hipercub. En la primera fase de l'exemple les transicions es realitzen per les dimensions $0, 1, 0, 2, 0, 1, 0$ respectivament.

En general, la seqüència d'enllacos (dimensions de l'hipercub) que defineixen cadascuna de les transicions de la fase d'intercanvi es representa per la seqüència D_e^{BR}. Aquesta seqüència pot generar-se de forma sistemàtica tal i com es mostra a continuació:

$$\begin{aligned} D_1^{BR} &= < 0 > \\ D_i^{BR} &= < D_{i-1}^{BR}, i-1, D_{i-1}^{BR} > \qquad 1 < i \leq e \end{aligned}$$

Per exemple, ja hem vist que en la primera fase d'intercanvi la seqüència per $e = 3$ és $D_3^{BR} =< 0102010 >$, en la segona fase d'intercanvi la seqüència per $e = 2$ és $D_2^{BR} =< 010 >$ i finalment per la tercera fase d'intercanvi la seqüència per $e = 1$ és $D_1^{BR} =< 0 >$.

Després de cada fase d'intercanvi e es realitza una fase de divisió que consisteix en una única etapa seguida d'una transició que utilitza l'enllaç e; en l'exemple cadascuna de les fases de divisió es realitza per la dimensió 2, 1 i 0 respectivament. Per finalitzar, es realitza l'última transició que consisteix en una etapa seguida d'una transició per l'enllaç $d - 1$, en l'exemple per la dimensió 2.

El següents escombrats utilitzen el mateix algorisme però es permuta l'ordre en què s'utilitzen els enllaços. En particular, la permutació que s'ha d'aplicar a l'escombrat s, suposant que el primer escombrat és per $s = 0$, es defineix de la forma següent:

$$\begin{aligned} \alpha_0(i) &= i \\ \alpha_s(i) &= (\alpha_{s-1}(i) - 1) \bmod d \qquad \text{for i=0...}d\text{-1} \end{aligned}$$

Les permutacions que s'aplicarien en el segon escombrat en el cas d'un hipercub de dimensió igual al de l'exemple de la figura 2.1 ($d = 3$) serien $0 \to 2$, $1 \to 0$ i $2 \to 1$. Les seqüències de dimensions de les diferents fases d'intercanvi en aquest segon escombrat quedarien doncs com $D_3^{BR} =< 2021202 >$, $D_2^{BR} =< 202 >$ i $D_1^{BR} =< 2 >$, mentre que les fases de divisió es realitzaran per les dimensions 1, 0 i 2 i l'última transició per la dimensió 1. I en el tercer escombrat s'aplicarien les permutacions $0 \to 1$, $1 \to 2$ i $2 \to 0$, per tant quedarien les seqüències de dimensions per les diferents fases d'intercanvi com $D_3^{BR} =< 1210121 >$, $D_2^{BR} =< 121 >$ i $D_1^{BR} =< 1 >$; les fases de divisió es realitzen per les dimensions 0, 2 i 1, i finalment l'última transició es realitza per la dimensió 0. Seguint aquest mateix esquema, al quart escombrat es tornen a utilitzar les dimensions en el mateix ordre en què es van utilitzar en el primer escombrat.

2.1.2 Model Analític del cost de la comunicació de l'algorisme BR

El cost de la comunicació d'un escombrat complet quan s'aplica l'algorisme BR ve donat per l'expressió següent:

$$T_{BR} = (2^{d+1} - 1)(T_s + \frac{m^2}{2^d} \times T_e) \tag{2.1}$$

El model es correspon al cost de la comunicació de les $2^{d+1}-1$ transicions necessàries per completar un escombrat. En cada transició s'intercanvia un missatge que conté $m/2^{d+1}$ columnes de la matriu $\bar{A}$ i les columnes corresponents de la matriu U ($m/2^{d+1}$). Cada columna té m elements.

2.1.3 Anàlisi de l'algorisme BR per arquitectures $multiple - port$

L'algorisme BR, tal i com s'acaba de descriure, és eficient per multicomputadors amb una topologia en hipercub on els nodes tinguin una arquitectura $one-port$. En aquestes condicions, l'algorisme és capaç d'utilitzar tots els enllaços disponibles (un únic enllaç en aquest cas) en cada etapa de comunicació (transició).

Si executem l'algorisme BR en un sistema amb una arquitectura $multiple - port$, el comportament serà el mateix que pel cas de tenir un únic port, ja que l'ordre en què es realitzen les operacions de càlcul i comunicació en l'algorisme BR tan sols permet utilitzar un únic port dels d ports possibles que ens permet utilitzar el sistema en cada etapa de comunicació. Aquest comportament fa que l'algorisme BR sigui molt poc eficient per arquitectures $multiple - port$.

Si analitzem quin seria el cost de la comunicació, cost hipotètic, de l'algorisme BR en el cas que en cada etapa de comunicació en lloc d'utilitzar un únic enllaç de l'hipercub utilitzés els d enllaços, el cost de la comunicació podria expressar-se com:

$$T_{LB} = (2^{d+1} - 1)(d \times T_s + \frac{m^2}{d \times 2^d} \times T_e) \tag{2.2}$$

Expressió que s'ha calculat a partir del cost de comunicació de l'algorisme BR suposant que es pot utilitzar tot l'ample de banda que ofereix el sistema, amb el que s'ha dividit el terme multiplicat per T_e pel nombre d'enllaços d i s'ha multiplicant el terme T_s pel nombre de missatges que hipotèticament poden enviar-se en paral.lel, d. Aquesta expressió serà considerada com una fita inferior del cost de la comunicació.

En la gràfica de la figura 2.2 es mostra la relació entre el cost de comunicació de l'algorisme BR i la fita inferior, s'han considerat els valors de $T_e = 1$ i $T_s = 10000$. Es veu clarament que si fóssim capaços d'utilitzar més eficientment l'ample de banda

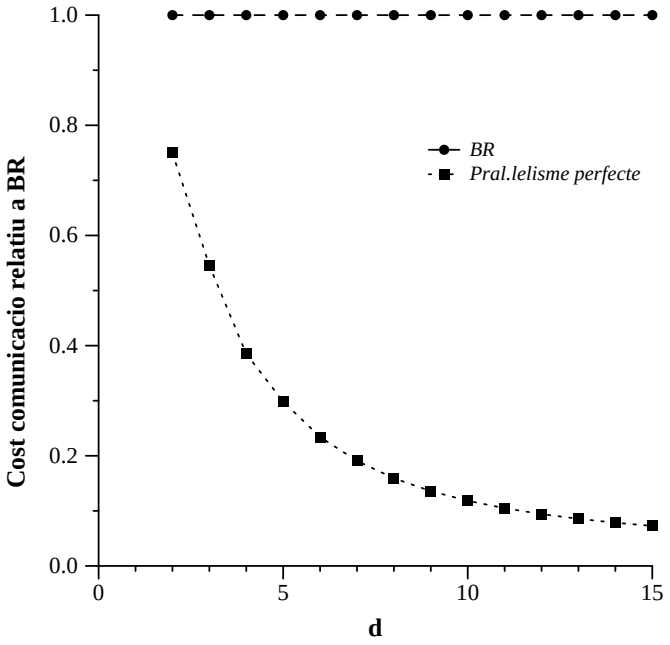


Figura 2.2 Cost de la comunicació de la fita inferior relatiu al BR

que ens ofereix el sistema $multi-port$, el cost de la comunicació podria reduir-se considerablement respecte al cost de la comunicació de l'algorisme BR original.

En les següents seccions es proposen nous algorismes amb els que s'aconsegueixen costos de comunicació inferiors al de l'algorisme BR per hipercubs $multiple-port$. Com ja s'ha dit anteriorment, el punt de partida de totes les noves propostes és l'algorisme BR. Per tal de generar els nous algorismes es realitzen les següents transformacions a l'algorisme BR original:

- En primer lloc es reorganitza l'ordre en què es realitzen les operacions de càlcul i de comunicació de l'algorisme original BR. Aquesta reorganització es realitza aplicant la tècnica de la segmentació de les comunicacions a l'algorisme BR descrita breument a continuació i en més detall a [8]. En aquest primer pas s'aconsegueix reduir a la meitat el cost de la comunicació.
- S'observa que la reducció del cost de la comunicació que s'aconsegueix segmentant les comunicacions de l'algorisme BR segueix sent bastant lluny de la fita inferior que

s'ha definit anteriorment. La causa d'aquesta reducció tant poc significativa es deu a l'ordre en què l'algorisme BR utilitza les dimensions de l'hipercub en les diferents transicions. És per això que a continuació es proposen noves ordenacions de Jacobi en les que l'ordre d'utilització de les dimensions de l'hipercub afavoreix l'increment del paral.lelisme de les comunicacions. Amb les noves ordenacions s'aconsegueixen reduccions pròximes a la fita inferior.

2.2 TÈCNICA DE LA SEGMENTACIÓ DE LES COMUNICACIONS

A [8] es proposa un transformació sistemàtica que s'aplica a un tipus determinat d'algorismes amb una topologia de comunicació en hipercub. El tipus d'algorismes s'anomena CC-cub. La transformació sistemàtica es diu segmentació de les comunicacions. La tècnica de la segmentació de les comunicacions ha estat proposada per tal d'aconseguir incrementar el grau de paral.lelisme de les comunicacions i permetre el solapament del càlcul i la comunicació. En aquest treball tan sols s'aplica la tècnica per aconseguir augmentar el paral.lelisme de les comunicacions.

Els algorismes CC-cube estan formats per 2^d processos els quals treballen en paral.lel. L'esquema de les comunicacions entre processos defineix una topologia en hipercub. En cada etapa de comunicació tots els processos utilizen les mateixes dimensions de l'hipercub.

Cada procés executa un codi que segueix la següent estructura:

do i=1, K
    Calcular $x_i[1..N]$ i altres dades locals
    intercanviar x_i amb el procés veí en la dimensió d_i
enddo

L'algorisme consisteix en K iteracions. En una iteració i cada procés realitza certs càlculs i intercanvia algunes dades amb algun procés veí en una determinada dimensió de l'hipercub. Les dades que es calculen en cada iteració estan representades pel vector x_i, aquestes dades són les que s'intercanvien al final de cada iteració. d_i fa referència a una dimensió de l'hipercub ($d_i \in [0, d-1]$). És important assenyalar que una dimensió de l'hipercub pot ser utilitzada en més d'una iteració i que alguna dimensió de l'hipercub pot no ser utilitzada en cap iteració.

Per poder aplicar la tècnica de la segmentació a l'algorisme CC-cub el càlcul del vector x_i ha de complir la següent condició:

do j=1, N
$x_i[j] = \text{f}(x_a[b], \text{dades locals})$ amb $a \leq i, a+b \leq j+i-1$
enddo

és a dir, el càlcul de $x_i[j]$ tan sols és funció de part d'altres vectors que han estat rebuts en iteracions prèvies i possiblement de dades locals, veure [8] per més detalls.

La idea de la segmentació de les comunicacions es basa en el fet de dividir els N elements que s'han d'enviar en una iteració en Q paquets de mida igual a N/Q; suposarem que N és divisible per Q per simplificar les expressions analítiques. La tècnica reorganitza les operacions de càlcul i comunicació de les diferents iteracions amb la intenció d'incrementar el paral.lelisme de les comunicacions. El nou algorisme segmentat que resulta després de l'aplicació de la segmentació a l'algorisme original segueix el següent esquema d'execució. Primer, tots els processos calculen el primer paquet (d'N/Q elements) corresponent a la primera iteració i l'envien al node veí en la dimensió d_1. A continuació cada node té les dades necessàries per calcular el segon paquet corresponent a la primera iteració i el primer paquet de la segona iteració. Aquests dos nous paquets s'envien als nodes veïns en les dimensions d_1 i d_2. Si d_1 és diferent de d_2, els dos paquets es poden enviar en paral.lel. Si $d_1 = d_2$, els dos paquets poden combinar-se en

un únic paquet i enviar-se al seu destí. Seguint aquest mateix procediment, el nombre d'enllaços que poden ser utilitzats en paral.lel per cada node pot incrementar-se fins a un màxim de d (dimensió de l'hipercub). En aquest cas, estaríem utilitzant tots els enllaços de l'hipercub en paral.lel.

El valor òptim de Q depèn dels paràmetres del sistema (temps d'iniciar una operació de comunicació, temps d'enviar les dades per la xarxa) i de paràmetres del problema (tamany de les dades). El valor de Q que s'escull per un problema particular té un impacte decisiu en l'eficiència de l'algorisme resultant. Q pot prendre qualsevol valor entre 1 i N. Quan $Q = 1$ no s'està aplicant la segmentació de les comunicacions. Quan $Q = N$ significa que el tamany d'un paquet és exactament igual a un element. El valor de Q indica el grau de segmentació de les comunicacions, és a dir, el grau de paral.lelisme que potencialment un node pot explotar. A mesura que Q s'incrementa alhora s'incrementa el grau de paral.lelisme (potencialment podrem utilitzar més enllaços de l'hipercub en paral.lel) però, per una altra banda, el nombre de missatges s'està incrementant (més missatges de tamany més petit) i amb el nombre de missatges s'incrementa el cost degut a la inicialització d'una operació de comunicació ($start-up$). A [8] es mostra que el valor òptim de Q pot obtenir-se de forma sistemàtica per qualsevol algorisme CC-cub, per qualsevol dimensió de l'hipercub i per qualsevol valor dels paràmetres del sistema i del problema.

La relació entre Q i K (K és el nombre d'etapes de l'algorisme CC-cub original) determina dos esquemes de segmentació diferents. Quan $Q < K$ l'esquema de la segmentació s'anomena *shallow pipelining*, mentre que si s'escull un tamany de Q més gran o igual que K la segmentació s'anomena *deep pipelining*. Veiem amb un exemple en què consisteixen cadascun d'aquests esquemes.

En la figura 2.3(a) es mostra de forma esquemàtica l'execució d'un CC-cub en un hipercub de dimensió 3. L'algorisme consisteix en quatre iteracions, $K = 4$. Els càlculs estan representats per rectangles i la comunicació representada per fletxes. Les operacions de comunicació en cada iteració es representen amb una fletxa amb un to de gris diferent. L'ordre en què les dimensions de l'hipercub són utilitzades en cadascuna de les quatre etapes de comunicació són $< 0, 1, 2, 1 >$.

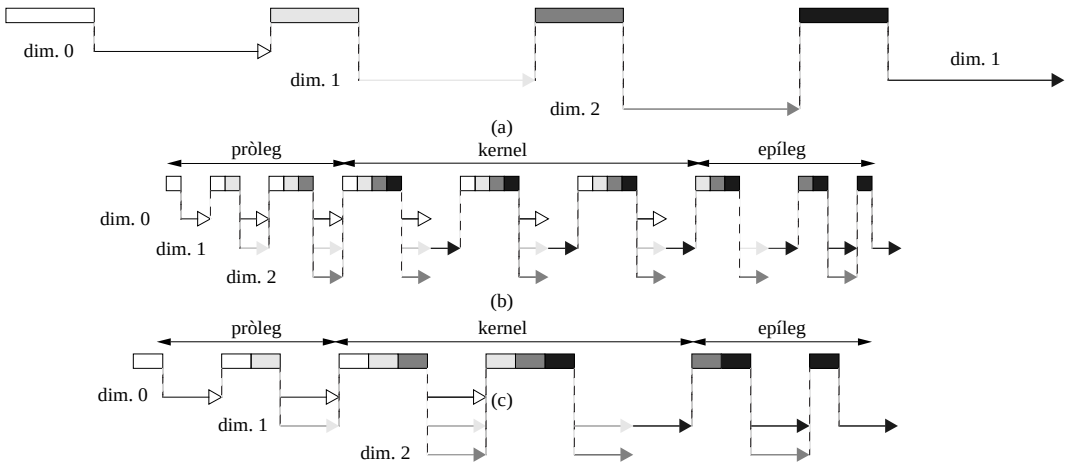


Figura 2.3 (a)Algorisme CC-cub sense segmentar les comunicacions. (b)Aplicació de Deep pipelining (Q=6). (c)Aplicació de Shallow pipelining.

Deep pipelining

La figura 2.3(b) mostra l'execució de l'algorisme quan s'aplica *deep pipelining*. Els càlculs que es realitzaven en cadascuna de les etapes de l'algorisme original, es divideixen en Q paquets de manera que $Q > K$ (6 paquets en l'exemple). Cada paquet consisteix en el càlcul d'N/Q elements. En el nou algorisme generat després d'aplicar la segmentació de les comunicacions, es poden distingir tres fases, *pròleg, kernel i epíleg* .

La primera fase, el *pròleg*, consisteix en $K-1$ iteracions. En la iteració i del *pròleg*, cada node calcula i paquets (que pertanyen a diferents etapes de l'algorisme original) i envia i paquets als nodes veïns en les dimensions $d_1...d_i$. Els paquets que utilitzen diferentes dimensions s'envien en paral.lel. Aquells paquets que s'han d'enviar per la mateixa dimensió es combinen en un únic missatge i són enviats a la seva destinació.

Les següents $Q-K+1$ iteracions formen la fase del *kernel*. En cada iteració de la fase del *kernel* cada node calcula K paquets i els envia utilitzant tots els enllaços de comunicació a la vegada, els enllaços que apareixen en la seqüència D utilitzada.

Finalment, el nombre de paquets que ha de calcular i d'enviar cada node decreix en les $K-1$ últimes etapes, que constitueixen la fase d'*epíleg*.

Shallow pipelining

El mètode *shallow pipelining* s'il.lustra en la figura 2.3(c). De nou, cada iteració de l'algorisme original es descomposa en Q paquets, en aquest cas el valor de Q és més petit que K (3 paquets en l'exemple). Ara la fase de *pròleg* consisteix en $Q-1$ iteracions. Les següents $K-Q+1$ iteracions formen la fase de *kernel* i les últimes $Q-1$ iteracions constitueixen la fase d'*epíleg*. En cada iteració del kernel cada node envia Q paquets utilitzant com a màxim Q dimensions contínues de la seqüència D en paral.lel.

La idea de la tècnica de la segmentació de les comunicacions s'inspira en el *software pipelining* [37] i va ser utilitzada per primer cop de forma molt restrictiva per [32] per optimitzar el càlcul de la FFT en la CM2. En [8] s'ha generalitzat la tècnica amb la possibilitat d'una anàlisi de tots els possibles valors que pot prendre Q (grau de segmentació).

2.3 BR SEGMENTAT

La tècnica de la segmentació de les comunicacions pot aplicar-se a qualsevol algorisme del tipus CC-cub que compleixi els requeriments especificats a [8]. En particular, en el cas de l'algorisme BR la tècnica no pot aplicar-se a tot l'algorisme, però sí que és possible aplicar la tècnica de la segmentació de les comunicacions a les fases d'intercanvi, que a més a més podem considerar que són les fases amb més cost de tot l'algorisme.

Recordem quin era el procediment de l'algorisme BR: en cadascuna de les iteracions que formen una fase d'intercanvi es podien distingir una etapa computacional (càlcul i aplicació de rotacions) seguida d'una transició (intercanvi d'informació). En cada iteració s'intercanvia un bloc d'$m/2^{d+1}$ columnes de la matriu $\bar{A}$ i les corresponents

columnes de la matriu U (cada parella de veïns intercanvia els respectius blocs inferiors). En cada iteració un procés ha d'aparellar una columna que pertany al bloc inferior amb totes les columnes que pertanyen al bloc superior. Quan una columna del bloc inferior s'ha aparellat amb totes les columnes del bloc superior, aquesta columna ja no es modificarà més durant la iteració en qüestió i per tant és possible enviar-la al node que correspongui (segons la dimensió utilitzada en la transició següent) abans de finalitzar totes les transformacions que s'han de realitzar durant aquesta iteració. Això fa que sigui possible aplicar la segmentació de les comunicacions a totes les fases d'intercanvi.

La segmentació de les comunicacions no es pot aplicar a les etapes de divisió. Per una part, no és possible aplicar-la individualment a la fase de divisió ja que en aquesta fase tan sols s'utilitza un enllaç de l'hipercub i per tant és impossible explotar cap tipus de paral.lelisme des d'aquest punt de vista. Per una altra part, podríem considerar la fase de divisió conjuntament amb les fases d'intercanvi i aplicar la segmentació de les comunicacions a tot plegat. Però això tampoc no és possible ja que en les fases de divisió s'intercanvien alguns blocs superiors. Per poder realitzar aquest intercanvi és absolutament necessari que s'hagin rebut totes les columnes del bloc inferior que s'hagin generat en la iteració immediatament anterior a la fase de divisió, iteració que es correspon a l'última transició de la fase d'intercanvi prèvia a la fase de divisió. Aquesta situació fa que una de les condicions necessàries per aplicar la segmentació de les comunicacions no es compleixi i per tant aquesta no pugui aplicar-se.

L'algorisme que resulta un cop aplicada la tècnica de la segmentació de les comunicacions a les fases d'intercanvi de l'algorisme BR l'anomenarem BR segmentat.

2.3.1 Modelització del cost de la comunicació de l'algorisme BR segmentat

A continuació s'avalua l'eficiència de l'algorisme BR segmentat. Aquest es compara amb l'algorisme BR per veure quant es redueix el cost quan s'aplica la tècnica de la segmentació de les comunicacions. També es compara amb la fita inferior definida anteriorment; aquesta comparació ens permet determinar quant bo és l'algorisme segmentat i si val la pena buscar alternatives millors.

Si analitzem la seqüència D_e^{BR}, que especifica l'ordre en què s'utilitzen les dimensions durant la fase d'intercanvi e, s'observa que la dimensió 0 apareix exactament 2^{e-1} cops. Per exemple, la seqüència $D_4^{BR} = 010201030102010$ consta de 15 elements i la dimensió 0 apareix un total de 8 cops, una mica més de la meitat. Des del punt de vista de la comunicació això suposa que la meitat de la informació que s'ha de transmetre en cada fase d'intercanvi s'envia a través d'aquesta dimensió. Aquesta observació suggereix que el màxim benefici que es podrà obtenir quan s'apliqui la tècnica de la segmentació de les comunicacions, serà reduir el cost de la comunicació de cada fase d'intercanvi a la meitat. Per una altra banda, també s'observa que qualsevol subseqüència de dos elements consecutius de D_e^{BR} consta de dues dimensions diferents. Per tant, despreciant l'etapa de pròleg i d'epíleg, es pot aconseguir reduir el cost de la comunicació a la meitat amb un grau de segmentació igual a 2, és a dir amb $Q = 2$. A més a més, amb aquest grau de segmentació tan petit s'aconsegueix minimitzar el cost de la comunicació degut a l'inicialització dels missatges (temps de *startup*), el qual com ja s'ha vist anteriorment és proporcional al grau de segmentació.

És possible demostrar fàcilment que el grau de paral.lelisme òptim de l'algorisme BR segmentat amb el que s'aconsegueix una reducció màxima del cost de la comunicació és per $Q = 2$. L'expressió del cost de la comunicació de l'algorisme BR segmentat per completar un escombrat, considerant el grau de segmentació per $Q = 2$ és:

$$T_{BRs} = (2^{d+1} - 1)(2 \times T_s + \frac{m^2}{2^{d+1}} \times T_e)$$

on $2^{d+1} - 1$ són el nombre d'etapes per completar un escombrat. En cada operació de comunicació s'intercanvien dos missatges de tamany igual a $m^2/2^{d+1}$. El cost d'inicialitzar les dues operacions de comunicació no és possible solapar-les i tenen un cost igual a $2 \times T_s$. Degut a què els dos missatges s'envien sempre a dos nodes veïns en diferents dimensions de l'hipercub el cost d'enviar els dos missatges es pot solapar i per tant el cost és igual al cost d'enviar un únic missatge, $(m^2/2^{d+1})T_e$. Tenint en compte que el valor que es considera és $Q = 2$, estem aconseguint el màxim paral.lelisme possible (sempre utilitzem dos enllaços diferents de l'hipercub).

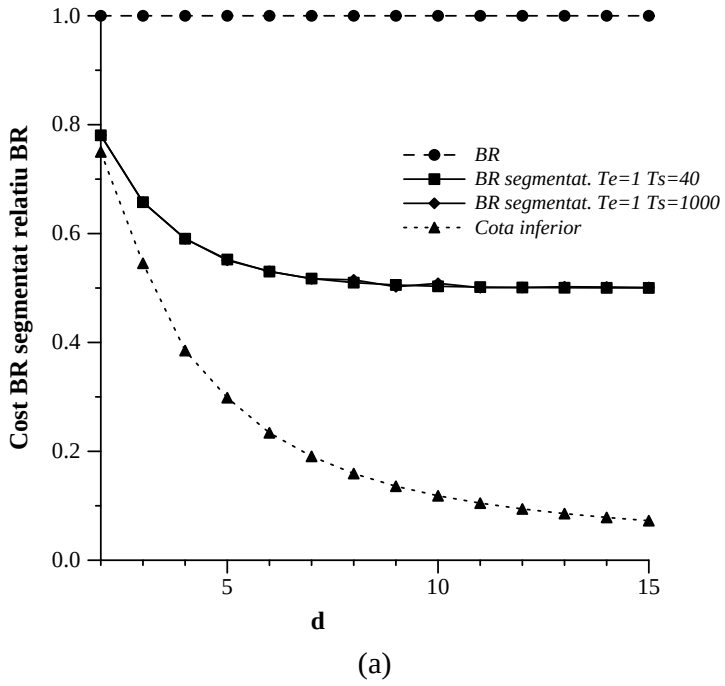


Figura 2.4 Anàlisi de l'algorisme BR segmentat

El terme que multiplica a T_e es divideix en dos factors, el primer factor amb un valor de 1 indica el nombre de vegades que apareix en cada operació de comunicació la dimensió més frequent (α), en aquest cas podem observar que totes les subseqüències de dues dimensions contenen diferents dimensions de l'hipercub i per tant el valor és 1. El segon factor amb un valor igual a $\frac{m^2}{2^{d+1}}$ indica el tamany del paquet que es transmet en cada operació de comunicació. Es pot veure que aquest és igual a la meitat del paquet que es transmetia en l'algorisme BR sense segmentar degut a què $Q = 2$.

En la figura 2.4 es compara el cost de la comunicació que requereix la resolució d'un problema de dimensió 2×2^{15} quan s'apliquen cadascun dels tres algorismes considerats en aquesta secció: l'algorisme BR, l'algorisme BR segmentat amb grau de segmentació igual a 2 i la fita inferior. En la gràfica es representa una corba en la que s'han pres els valors de T_e i de T_s igual a 1 i 40 respectivament, aquests valors serien els valors equivalents a la relació de paràmetres del multicomputador $nCUBE$ 2 que s'ha descrit en el capítol 1. Tenint en compte les tendències descrites en el capítol 1 de l'evolució

dels paràmetres en els multicomputadors, en la gràfica es representa una segona corba en la que s'han pres els valors de T_e i de T_s igual a 1 i 1000 respectivament.

Es pot observar dels resultats que s'han obtingut que el cost de la comunicació és poc sensible a la variació del valor dels paràmetres T_e i T_s i les dues corbes gairebé es superposen. Això és degut a què en aquest cas té molt més pes el fet que puguem solapar el cost d'enviar dos missatges que el fet d'augmentar el nombre de missatges al doble respecte l'algorisme BR sense segmentar.

En la gràfica s'observa que amb l'algorisme BR segmentat el cost de la comunicació tendeix a reduir-se a la meitat tal i com era d'esperar. Malgrat aquest guany, aquesta reducció està bastant lluny de l'òptim. Això és degut a la pròpia estructura de la seqüència D_e^{BR} i al desequilibri en la utilització de les dimensions. Aquesta observació és la que ens ha motivat la cerca de noves ordenacions més adequades per treure profit de la tècnica de la segmentació de les comunicacions.

2.4 NOVES ORDENACIONS

Si analitzem la seqüència de dimensions d'una etapa d'intercanvi de l'algorisme BR podem observar que aquesta defineix un camí hamiltonià d'un hipercub. És a dir, després de tots els intercanvis segons les dimensions de la seqüència un paquet de dades visita tots els nodes d'un hipercub un sol cop. En la figura 2.5 es mostra un exemple en el que es veu quin és el camí que recorre un paquet quan s'apliquen els intercanvis que determinen la seqüència $D_3^{BR} =< 0102010 >$, els quals permeten visitar tots els nodes d'un hipercub de dimensió 3 un sol cop.

Si substituïm aquesta seqüència en l'algorisme BR per qualsevol altra seqüència de dimensions que defineix també un camí hamiltonià, el nou algorisme generat tindrà un comportament totalment equivalent a l'algorisme BR original, és a dir es generen tots els aparellaments requerits a la fase d'intercanvi corresponent.

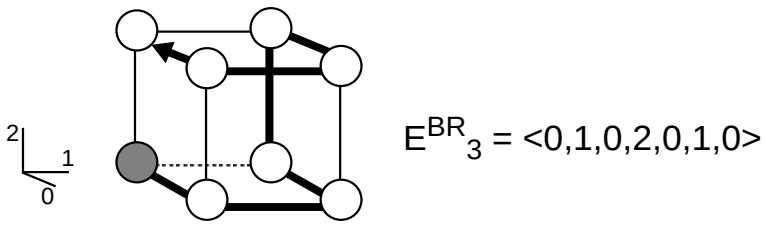


Figura 2.5 Camí hamiltonià que recorre un hipercub de dimensió 3

La seqüència de dimensions d'una etapa d'intercanvi de l'algorisme BR defineix un camí hamiltonià, amb l'objectiu que tots els bloc inferiors visitin tos els nodes de l'hipercub considerat. Si modifiquem la seqüència de dimensions i aquesta nova seqüència defineix un altre camí hamiltonià en el mateix hipercub que la seqüència original, s'assoleix el mateix objectiu de l'algorisme original, per tant podem considerar que totes dues seqüències de dimensions són equivalents i vàlides.

Això implica que tots els possibles algorimes que es generen amb aquest procediment, proposant camins hamiltonians alternatius, completaran un escombrat amb el nombre mínim d'etapes amb l'única diferència entre ells de l'ordre en què s'aplicaran les transformacions.

Des del punt de vista de la segmentació de les comunicacions la seqüència que defineix el camí hamiltonià seria interessant que complís les següents propietats:

- Equilibri

  Que totes les dimensions de l'hipercub apareguin el mateix nombre de vegades dintre la seqüència. Aquesta propietat afavoreix *deep pipelining*.

- Distribució

  Que les dimensions dintre la seqüència estiguin ben distribuïdes (*0,1,2...0,1,2...*). Aquesta propietat afavoreix *shallow pipelining*.

No ha estat possible trobar cap seqüència que defineixi un camí hamiltonià i que compleixi aquestes dues propietats simultàniament, seqüència òptima. A continuació però,

es proposen dues noves seqüències, que malgrat que no són òptimes presenten unes propietats millors que la seqüència de l'algorisme BR.

La primera, anomenada *BR-permutada*, forma part de l'ordenació de Jacobi α-*optimal* que es descriu a continuació. Amb aquesta seqüència s'aconsegueix que totes les dimensions de l'hipercub s'utilitzin bastant equilibradament (es compliria la primera propietat) però no aconsegueix una bona distribució dintre la seqüència. És per això que aquesta primera proposta funciona bé amb *deep pipelining* i per tant requereix valors de Q grans. Això implica que per poder aplicar aquesta ordenació de forma eficient es requereixen problemes de tamany gran. En el millor dels casos el cost de la comunicació de l'algorisme original es pot dividir per d.

La segona proposta, anomenada *Grau*-4, aconsegueix una distribució tal que per un valor de $Q = 4$ aconsegueix utilitzar gairebé sempre grups de quatre dimensions diferents amb el que el cost de la comunicació queda dividit per 4. Aquesta es pot utilitzar per qualsevol tamany de problema.

2.4.1 Ordenació α-*optimal*

L'ordenació que es proposa en aquesta secció, que hem anomenat ordenació α-*optimal*, és el resultat de la combinació de dues noves ordenacions, l'ordenació α-mínima i l'ordenació *BR-permutada*. Ambdues pretenen complir la propietat d'equilibri definida anteriorment. A continuació es descriuen en detall cadascuna d'aquestes dues noves ordenacions.

Equilbri òptim

Donada una seqüència de dimensions, definim α com el nombre de vegades que apareix la dimensió que més cops apareix dintre la seqüència. El grau d'equilibri d'una ordenació es caracteritza pel valor d'α. Interessarà, doncs, tenir valors petits d'α ja que la seqüència serà més equilibrada amb el que potencialment tindrem un grau de solapament més gran i per tant el cost de la comunicació serà més petit.

En una fase d'intercanvi e, el nombre d'elements per qualsevol seqüència D_e és $2^e - 1$ i el nombre de dimensions diferents que podem trobar és e. El valor d'α d'una sequència totalment equilibrada es pot expressar com:

$$\lfloor \frac{2^e - 1}{e} \rfloor \tag{2.3}$$

Aquest valor, que determina la fita inferior d'α, ens permetrà tenir un punt de referència a l'hora d'analitzar l'eficiència de les noves ordenacions.

Ordenació α-mínima

Donat un hipercub d'una determinada dimensió existeix més d'un camí hamiltonià i en conseqüència hi ha moltes seqüències alternatives que poden utilitzar-se a l'hora d'implementar les fases d'intercanvi. Si volem ser exhaustius, per tal d'obtenir la seqüència amb l'α mínima hauríem de generar tots els camins hamiltonians possibles. Hem realitzat un programa per obtenir la seqüència de dimensions que sigui camí hamiltonià i tingui el valor d'α mínima. Degut a la complexitat del mateix tan sols s'han pogut obtenir seqüències per valors d'$e \leq 7$. Les seqüències que s'han obtingut són les següents:

- $D_2^{min} = <010>$
- $D_3^{min} = <0102010>$
- $D_4^{min} = <010203212303121>$
- $D_5^{min} = <0120103010214123212303234143 23>$
- $D_6^{min} = <010201030102010401021312152131243231323435054245354241434525 4345>$

El valor d'α corresponent a cadascuna d'aquestes seqüències és 2,3,4,7 i 11 respectivament, que com es pot comprovar coincideix amb la fita inferior d'α que s'ha definit en l'apartat anterior.

Ordenació *BR-permutada*

A continuació es presenta una nova ordenació de Jacobi. Aquesta s'obté aplicant una sèrie de permutacions a la seqüència de dimensions d'una etapa d'intercanvi de l'ordenació BR. La seqüència resultant l'hem anomenat *BR-permutada*, abreujat D_e^{p-BR}. El valor d'α d'aquesta nova ordenació és bastant pròxim al valor de l'α mínima que s'ha definit anteriorment. En particular, en l'annex 1 es demostra que per valors grans d'e el valor d'α és $1,25$ vegades el valor de l'α òptim. La metodologia que es descriu a continuació s'aplicarà per tal d'obtenir seqüències quan $e > 6$. Per valors de $e \leq 6$ s'aplicaran les seqüències α-mínima definides en la secció anterior.

La metodologia que s'aplica a l'hora de generar la nova ordenació, es basa en la següent propietat dels hipercubs.

Propietat 1. Sigui D_e una seqüència d'enllaços que defineixen un camí hamiltonià en un subcub de dimensió e. Sigui ϕ qualsevol permutació dels identificadors dels enllaços dintre de la seqüència. Si apliquem la permutació ϕ a qualsevol subseqüència de D_e que és un camí hamiltonià d'un subcub, llavors la seqüència resultant segueix sent un camí hamiltonià del subcub de dimensió e.

Demostració. Donat un cub de dimensió n, si apliquem una permutació a tots els enllaços de la seqüència i renombrem els nodes d'acord amb la permutació, el graf resultant és isomòrficament equivalent al graf original. És per aquest motiu que si una permutació d'aquest tipus s'aplica a una seqüència d'enllaços que defineixen un camí hamiltonià en un cub de dimensió n, la seqüència resultant també és un camí hamiltonià del mateix subcub.

Vegem alguns exemples de com s'aplicaria aquesta propietat.

- $< 010 >$ és un camí hamiltonià d'un cub de dimensió 2. Si els enllaços 0 i 1 s'intercanvien en tota la seqüència, la seqüència resultant és $< 101 >$ que com pot comprovar-se també és un camí hamiltonià.

- $< 0102010 >$ és un camí hamiltonià d'un cub de dimensió 3. Si apliquem la permutació de l'exemple anterior a la subseqüència composada pels tres últims elements, subseqüència que és un camí hamiltonià d'un subcub de dimensió 2, s'obté la seqüència $< 0102101 >$, la qual segueix sent un camí hamiltonià d'un cub de dimensió 3.

Descripció de la metodologia per generar seqüències D_e^{p-BR}

La metodologia per obtenir la seqüència de les dimensions corresponents a una etapa d'intercanvi de l'ordenació *BR-permutada* es mostrarà amb un exemple concret per $e = 5$. S'ha escollit aquest valor per posar un exemple curt, s'ha de tenir en compte, però, que per aquest valor es coneix una ordenació α–mínima.

Es parteix de la seqüència generada amb l'ordenació BR quan $e = 5$:

$$D_5^{BR} =< 010201030102010401020103010201 0 >$$

La seqüència corresponent a l'ordenació *BR-permutada*, D_5^{p-BR}, s'obté aplicant una sèrie de permutacions a la seqüència inicial D_5^{BR}. Les subseqüències a les que s'apliquen les permutacions, s'escullen de tal manera que sempre es corresponen al recorregut d'un subcub complet de l'hipercub inicial.

Es pot observar que la seqüència D_e^{BR} està formada per 2^{e-1} elements que realitzen un recorregut d'un subcub de dimensió $(e-1)$, seguits per un element central i a continuació una altra seqüència de 2^{e-1} elements que realitzen un recorregut d'un altre subcub de dimensió $(e-1)$. Cadascuna d'aquestes dues subseqüències de dimensió $(e-1)$ estan formades per dues subseqüències de dimensió $(e-2)$ separades per un element central. Aquesta propietat pot aplicar-se recursivament a cadascuna de les quatre subseqüències fins que el resultat són subseqüències composades per un únic element.

S'han de realitzar un total de $\lfloor \log_2(e-1) \rfloor$ permutacions per tal d'obtenir la seqüència definitiva, aquestes s'apliquen en $\lfloor \log_2(e-1) \rfloor$ etapes. En la primera etapa, la primera

permutació s'aplica a la segona subseqüència de dimensió $(e-1)$. En la segona etapa, la segona permutació s'aplica a la segona i quarta subseqüència de dimensió $(e-2)$. En general, en la permutació k-èssima (per $k = 0...\lfloor \log_2(e-1) \rfloor - 1$) s'aplica a les subseqüències de dimensió $(e-k-1)$, començant per la segona subseqüència. Degut a la Propietat 1, definida anteriorment, sempre es compleix que la nova seqüència generada segueix sent vàlida.

Les permutacions que s'apliquen a les diferents subseqüències en una mateixa etapa són diferents. Per cada etapa, a partir de la permutació que s'aplica a la primera subseqüència es poden obtenir les permutacions que s'apliquen a la resta de subseqüències realitzant una composició de totes les permutacions que s'han aplicat anteriorment a cada subseqüència en particular.

En general, en l'etapa k (per $k = 0...\lfloor \log_2(e-1) \rfloor - 1$) la permutació aplicada a la primera subseqüència de dimensió $(e-k-1)$ durant aquesta etapa es defineix com la transposició de les següents parelles de dimensions:

$$i \rightarrow \lfloor (e-1)/2^k \rfloor - 1 - i \qquad k \in [0, \lfloor \log_2(e-1) \rfloor - 1] \qquad i \in [0, \lfloor (e-1)/2^k \rfloor - 1]$$

La permutació que s'aplica a la resta de subseqüències de dimensió $(e-k-1)$, S_1, durant l'etapa k s'obté a partir de la composició de la permutació aplicada a la primera subseqüència de dimensió $(e-k-1)$ amb totes les permutacions que s'han aplicat anteriorment a qualsevol subseqüència S_2 que inclou a S_1 ($S_1 \in S_2$).

En el nostre exemple, per obtenir la seqüència *BR-permutada* són necessàries 2 etapes de permutacions ($\lfloor \log_2 4 \rfloor$). En la primera etapa, la permutació s'aplica a la segona subseqüència de dimensió 3 i consisteix en intercanviar cada dimensió i , $i \in [0,3]$, amb la dimensió $3-i$ respectivament. La seqüència resultant és:

$< 0102010301020104\mathbf{323132303231323} >$

on apareixen en negreta les dimensions afectades per la permutació aplicada en la primera etapa.

En la segona etapa, s'aplica una permutació a la segona i quarta subseqüència de dimensió 2. En la primera subseqüència s'intercanvien les dimensions i, $i \in [0,1]$, amb les dimensions $1-i$ respectivament. Amb aquest intercanvi les dimensions 0 i 1 s'intercanvien en el segon quart. A l'hora de definir la permutació per la subseqüència del quart quart, s'han de tenir en compte les permutacions que s'han aplicat en l'etapa anterior. Com que la dimensió 0 es va intercanviar amb la dimensió 3 i la dimensió 1 es va intercanviar amb la dimensió 2, la permutació que resulta de la composició de la permutació de la segona etapa amb la permutació de l'etapa anterior consisteix en intercanviar la dimensió 3 amb la dimensió 2. Així doncs, aplicant aquestes permutacions la seqüència final queda com:

$$< 01020103\mathbf{1012101}43231323\mathbf{02321232} >$$

on apareixen de nou en negreta les dimensions afectades per la permutació aplicada en la segona etapa.

En la taula 2.6 es mostren els valors del paràmetre α per vàries seqüències de l'ordenació *BR-permutada* per diferents valors d'e.

Algorisme α-*optimal*

Les dues ordenacions que s'acaben de descriure, α-mínima i *BR-permutada*, intenten minimitzar el valor d'α. Aquestes dues ordenacions es combinen per formar l'algorisme α-*optimal*. Aquest algorisme consisteix en l'aplicació del mètode Jacobi *unilateral* conjuntament amb segmentació de les comunicacions amb la nova ordenació. Aquesta ordenació utilitza l'ordenació α–mínima per les etapes d'intercanvi que s'apliquen en subcubs de dimensió més petits de 7, ja que aquesta ordenació tan sols s'ha definit per subcubs d'aquestes dimensions. En el cas de les etapes d'intercanvi que s'apliquen amb subcubs de dimensions més grans o iguals a 7, s'aplica l'ordenació *BR-permutada*,

e	α	cota minima	α /cota minima
7	23	19	1.21
8	43	32	1.34
9	67	58	1.16
10	131	103	1.28
11	289	187	1.55
12	577	342	1.69
13	776	631	1.23
14	1543	1171	1.32

Figura 2.6 Valor d'α corresponent a l'ordenació *BR-permutada*

que com s'acaba de veure aconsegueix valors pròxims al paràmetre α mínim (definit al principi de la secció).

Avaluació de l'eficiència de l'algorisme α-*optimal*

En aquesta secció s'avalua l'eficiència de l'algorisme α-*optimal* i es compara amb l'algorisme BR, l'algorisme BR segmentat i la fita inferior. És difícil determinar de forma analítica el valor òptim de Q per la seqüència α-*optimal*, així que per cada problema concret s'ha realitzat per programa un escombrat de tots els casos per determinar el valor òptim de Q.

En les gràfiques de la figura 2.7 es compara el cost de la comunicació de l'algorisme BR segmentat i de l'algorisme α-*optimal* per diferents dimensions de l'hipercub, de dimensió 2 fins a dimensió 15. Les gràfiques també mostren l'eficiència de l'algorisme BR i de la fita inferior. Cada gràfica avalua el sistema per un tamany de problema

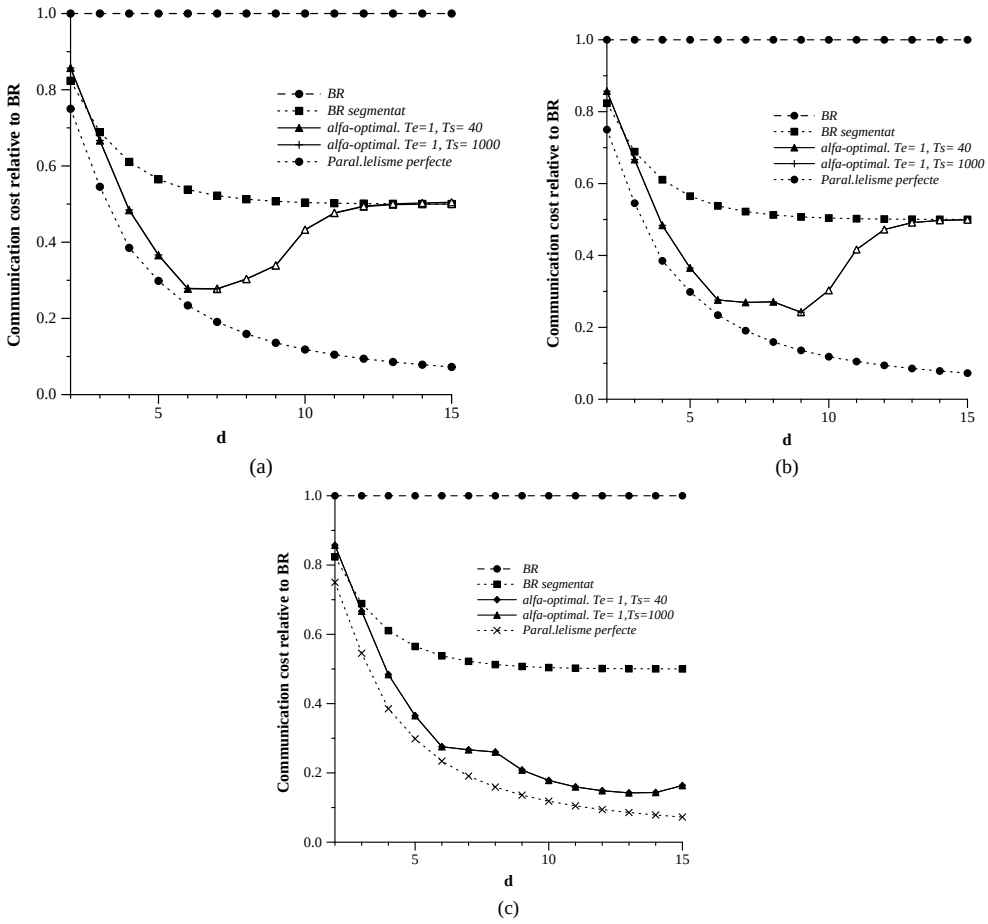


Figura 2.7 Avaluació de l'eficiència dels algorismes BR, BR segmentat i α-*optimal*. En (a) s'ha considerat un tamany de problema 2^{18}; en (b) s'ha considerat un tamany de problema 2^{20} i en (c) s'ha considerat un tamany de problema 2^{32}.

diferent; en la gràfica 2.7(a) s'ha considerat un tamany de problema igual a 2^{18}, en la gràfica 2.7(b) s'ha considerat un tamany de problema igual a 2^{20} i en la gràfica 2.7(b) s'ha considerat un tamany de problema igual a 2^{32}.

En cadascuna de les tres gràfiques es representen dues corbes en traç contigu. En la primera s'han considerat els valors dels paràmetres T_e i T_s igual a 1 i 40 respectivament, valors equivalents a la relació de paràmetres del multicomputador $nCUBE$ 2 especificats en el capítol 1. En la segona s'han considerat els valors dels paràmetres T_e

i T_s igual a 1 i 1000, valors que s'han escollit per avaluar sistemes amb una relació dels paràmetres T_e/T_s més gran.

Es pot observar dels resultats que s'han obtingut que el cost de la comunicació és poc sensible a la variació del valor dels paràmetres T_e i T_s i les dues gràfiques gairebé es podrien considerar idèntiques. Això es degut a que en aquest cas té molt més pes el fet que puguem solapar el cost d'enviar d missatges que el fet d'augmentar el nombre de missatges a un nombre igual a d respecte l'algorisme BR sense segmentar.

En la gràfica 2.7(a) s'ha considerat una matriu de tamany igual a 2^{18} columnes; en aquest cas es pot observar que amb l'algorisme *α-optimal* es pot aconseguir una eficiència pròxima a l'òptim pel cas d'hipercubs petits. Quan la dimensió de l'hipercub creix, l'eficiència tendeix a aproximar-se a l'eficiència obtinguda amb l'algorisme BR segmentat. La raó d'aquest comportament és que en les primeres fases de cada escombrat (les fases que consumeixen més temps) no és possible utilitzar *deep pipelining*. Tan sols pot aplicar-se *deep pipelining* quan el grau de paral.lelisme és més gran o igual que el nombre d'etapes de la fase d'intercanvi. L'algorisme *α-optimal* ha estat pensat per obtenir eficiències pròximes a l'òptim sempre i quan sigui possible aplicar *deep pipelining*; en cas que no sigui possible el seu comportament és equivalent al de l'algorisme BR segmentat.

En la gràfica 2.7(b) es considera una matriu amb 2^{20} columnes. Com que ara el tamany del bloc és més gran s'aconsegueixen eficiències pròximes a l'òptim per hipercubs de dimensions més grans, en aquest cas concret fins a hipercubs de dimensió igual a 10.

Finalment, en la gràfica 2.7(c) es considera una matriu amb 2^{32}columnes. En aquest cas, és possible aplicar el model *deep pipelining* en totes les fases d'intercanvi de qualsevol hipercub considerat, $2 \leq d \leq 15$, amb el que s'aconsegueixen eficiències pròximes a l'òptim per tots els casos considerats a la gràfica.

És important assenyalar que l'ordenació *BR-permutada* és eficient sempre que sigui possible aplicar *deep pipelining* a les primeres fases d'intercanvi, fases que tenen la major part del pes de la comunicació. Això tan sols és possible si el nombre de columnes

d'un bloc és més gran o igual que el nombre d'etapes que formen una fase d'intercanvi que recorre l'hipercub de dimensió més gran, dimensió igual a d, i per tant s'ha de complir que $N \geq 2^e - 1$. Aquesta restricció imposa que la dimensió de la matriu sigui igual a $(2^d - 1) \times 2 \times (2^d - 1)$. A mesura que d creix la dimensió de la matriu es fa molt gran.

Si les dimensions de la matriu són més petites, per restriccions del mètode no és possible aplicar *deep pipelining* i forçosament hem d'aplicar *shallow pipelining*. Quan s'analitzen amb més detall subseqüències més petites de l'ordenació *BR-permutada* es pot observar que el comportament és equivalent a l'ordenació BR. És per aquest motiu que la reducció de la comunicació tendeix a dividir-se per 2 tal i com ja s'ha vist en el cas del BR segmentat.

2.4.2 Ordenació *Grau*-4

S'ha vist en la secció anterior que amb l'ordenació α-*optimal* s'aconsegueix reduir el cost de la comunicació considerablement sempre i quan s'apliqui el model de segmentació *deep pipelining* i el cost de la comunicació estigui dominat per l'etapa del nucli. Dit d'una altra manera, quan el valor de Q és molt gran, per tant només és efectiu quan el problema és gran. En cas contrari s'obtenen resultats bastant iguals als que s'obtenien amb l'algorisme BR segmentat.

Per una etapa d'intercanvi e, si s'aplica *shallow pipelining*, interessa que l'ordenació utilitzada, D_e, tingui subseqüències de tamany Q amb el nombre mínim de dimensions repetides. El cas òptim serà aquell en què totes les dimensions dintre la subseqüència siguin diferents; això tan sols es pot complir si $Q \leq e$, o bé que totes les dimensions es repeteixin el mateix nombre de vegades, quan $Q > e$ amb una diferència màxima d'1. En les seqüències de les ordenacions $D_e^{min-\alpha}$ i D^{p-BR} es pot veure que si es considera tota la seqüència completa, totes les dimensions es repeteixen el mateix nombre de cops, però aquest comportament no es compleix per subseqüències més petites, on es pot observar que gairebé la meitat dels elements són iguals.

A continuació es descriu una nova ordenació de Jacobi que degut a què utilitza les dimensions de forma més equilibrada aconsegueix més bons resultats quan s'aplica *shallow pipelining*.

Definició de l'ordenació *Grau*-4

A continuació es defineix una nova ordenació de Jacobi que té la propietat que gairebé totes les subseqüències de tamany igual a 4 de la seqüència D_e que representa les transicions d'una fase d'intercanvi estan composades per diferents elements (dimensions).

Direm que una seqüència D_e és de grau n si qualsevol subseqüència d'n elements consecutius està formada per n elements diferents i a més a més la majoria de subseqüències d'$n+1$ elements tenen menys d'$n+1$ elements diferents. Per exemple, D_e^{BR} és de grau 2 per qualsevol valor d'e.

Una ordenació que consisteix en subseqüències de grau n ens permetrà reduir el cost de la comunicació per un factor n si s'aplica la segmentació de les comunicacions amb un grau de segmentació igual a n. La millor solució seria trobar una ordenació en la que totes les seqüències D_e tinguessin grau e per qualsevol e. Aquest és un problema al que no li hem pogut donar solució. Malgrat això, hem pogut trobar una metodologia que ens permet trobar seqüències que són de grau "quasi" 4. Diem que és de grau "quasi" 4, ja que gairebé totes les subseqüències de 4 elements estan composades per elements diferents. L'ordenació resultant l'anomenarem ordenació *Grau*-4 i les corresponents seqüències s'indicaran per D_e^{D4}. Aquesta ordenació es defineix de la manera següent:

$$\begin{aligned} E_3 &= < 0121012 > \\ E_i &= < E_{i-1}, i, E_{i-1} > \qquad 4 \leq i < e \\ D_e^{D4} &= < E_{e-1}, 1, E_{e-1} > \qquad e \geq 4 \end{aligned}$$

Per exemple, $D_5^{D4} =<$ 012301240123012101230124012301 2 $>$. Es pot observar que les seqüències generades són de grau "quasi" 3, ja que tan sols dues subseqüències

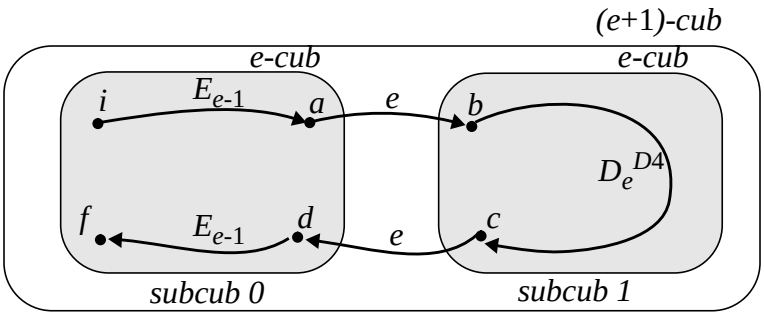


Figura 2.8 Representació gràfica de D_{e+1}^{D4}

de tamany 3, situades en el centre, no tenen tots els elements diferents ($< 121 >$ i $< 101 >$ en l'exemple). D_e^{D4} també es pot considerar de grau "quasi" 4 ja que de totes les subseqüències de 4 elements tan sols quatre no tenen tots els seus elements diferents ($< 0121 >$,$< 1201 >$,$< 2101 >$,$< 1012 >$ en l'exemple). Això es compleix per qualsevol $e > 3$. De totes maneres, quan e és gran, aquestes subseqüències que tenen elements repetits tenen un pes despreciable en l'eficiència total de l'algorisme.

Demostració que l'ordenació *Grau*-4 és un camí hamiltonià

En aquesta secció es demostra que l'ordenació *Grau*-4 proposada en aquest treball és un cami hamiltonià. La demostració es basa en el següent lema:

Lema 1. Sigui i un node arbitrari d'un hipercub de dimensió e. Sigui f el node al que s'arriba des del node i després de seguir el camí definit per la seqüència D_e^{D4}. Podem assegurar que els nodes i i f són veïns en la dimensió 1.

Demostració. La demostració del lema es realitza per inducció. És fàcil veure que el lema es compleix per D_4^{D4}. Veurem que si es compleix per D_e^{D4} també es compleix per D_{e+1}^{D4}. Primer reescriurem l'expressió de D_{e+1}^{D4} de la manera següent:

$$D_{e+1}^{D4} =< E_e, 1, E_e >=< E_{e-1}, e, E_{e-1}, 1, E_{e-1}, e, E_{e-1} >=< E_{e-1}, D_e^{D4}, E_{e-1} >$$

Qualsevol hipercub de dimensió $e+1$ es pot descomposar en dos hipercubs de dimensió e connectats entre si per la dimensió e. Tal i com es mostra a la figura 2.8, el camí descrit per D_{e+1}^{D4} consisteix en el camí E_{e-1} en el subcub 0, fins l'enllaç e que no pertany a E_{e-1}, un salt al subcub1 (per la dimensió e), el camí descrit per $D_e 6 D4$ pel subcub 1, de nou un altre salt al subcub 0 (per la dimensió e) i el camí E_{e-1} en el subcub 0.

Per hipòtesi d'inducció, els nodes b i c del subcub 1 són veïns en la dimensió 1. Com que els nodes a i b són veïns en la dimensió e, i els nodes d i c també, els nodes a i d han de ser veïns en la dimensió 1. És important adonar-se que el camí del node i fins al node a, a continuació del node a al node d per la dimensió 1, seguit del camí del node d fins al node f utilitza els mateixos enllaços que D_e^{D4}. De nou, per inducció, els nodes i i f han de ser veïns en la dimensió 1.

Teorema 1. D_e^{D4} és un camí hamiltonià.

Demostració. El teorema 1 també es demostrarà per inducció. El teorema es compleix per D_4^{D4}. Suposem que D_e^{D4} és un camí hamiltonià. De nou fem referència a la figura 2.8, en la que es mostra l'esquema del camí seguit per la seqüència D_{e+1}^{D4}. Tal i com s'ha vist en la demostració del lema anterior, el camí del node i fins al node a, seguit del node d a través de la dimensió 1, a continuació el camí del node d fins al node f, utilitza tots els enllaços de D_e^{D4}. És per això que D_{e+1}^{D4} és un camí hamiltonià en el subcub 0 que s'interromp a la meitat per passar al subcub 1, el qual es travessa per una seqüència que és hamiltoniana i després retorna al subcub 0. Com a resultat podem assegurar que D_{e+1}^{D4} és un camí hamiltonià.

Avaluació de l'eficiència de l'ordenació *Grau*-4

A l'hora d'avaluar l'eficiència de l'ordenació *Grau*-4 hem considerat un grau de segmentació $Q = 4$ ja que l'ordenació ha estat pensada per aquest cas concret. Recordem que en aquesta ordenació gairebé totes les subseqüències de tamany 4 utilitzen quatre dimensions diferents.

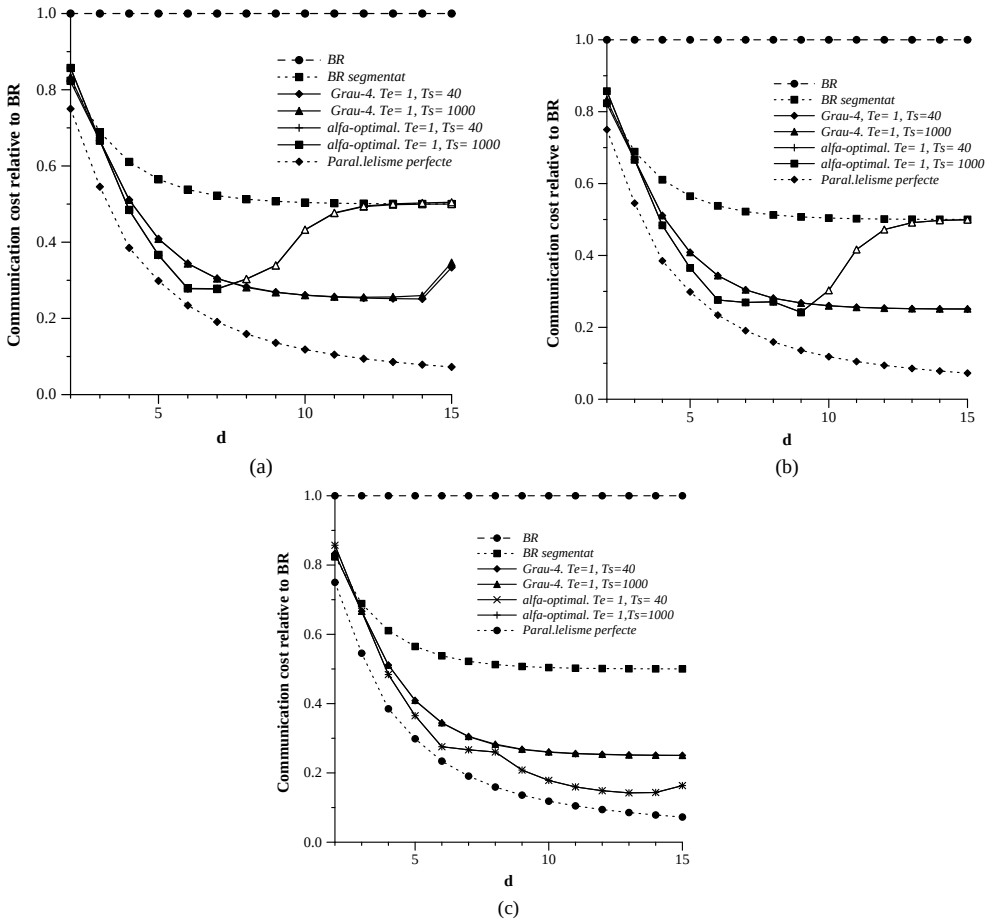


Figura 2.9 Avaluació de l'eficiència dels algorismes BR, BR segmentat, α-*optimal* and *Grau*-4. En (a) s'ha considerat un tamany de problema 2^{18}; en (b) s'ha considerat un tamany de problema 2^{20} i en (c) s'ha considerat un tamany de problema 2^{32}

En les gràfiques de la figura 2.9 es mostra l'eficiència de l'algorisme que aplica l'ordenació de Jacobi *Grau*-4. Les gràfiques també mostren l'eficiència dels algorismes BR, BR segmentat, α-*optimal* i la fita inferior. L'eficiència dels algorismes s'avalua per tres tamanys de problema diferents i per hipercubs de dimensions dintre del rang $2 \leq d \leq 15$. Les corbes amb els valors dels paràmetres Te i Ts igual a 1 i 40 respectivament es corresponen a la relació de paràmetres de l'$nCUBE$ 2 descrits en el capítol 1. Les corbes amb els valors dels paràmetres Te i Ts igual a 1 i 1000 respectivament s'han escollit per veure que el cost de comunicació de l'algorisme segueix sent poc sensible a la variació en la relació dels paràmetres T_s i T_e.

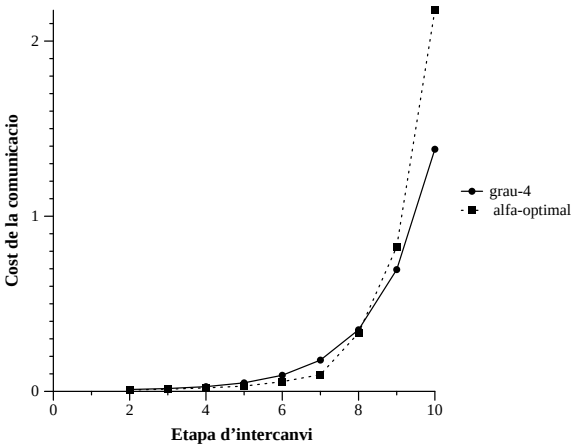


Figura 2.10 Cost de la comunicacio de cada etapa d'intercanvi per un hipercub de dimensio 10

En les gràfiques de la figura 2.9 s'observa que amb l'algorisme de *Grau*-4 el cost de la comunicació tendeix a reduir-se a un quart del cost de comunicació de l'algorisme de partida (BR sense segmentar) i per tant és dos cops més ràpid que l'algorisme BR segmentat. Respecte a l'algorisme α-*optimal*, l'algorisme de *Grau*-4 és millor en el cas d'hipercubs que per restriccions de tamany de la matriu no és possible aplicar el model de segmentació *deep pipelining*. Per exemple, en la gràfica 2.9(a) per hipercubs de dimensió més gran que 8 l'algorisme de *Grau*-4 és millor que l'algorisme α-*optimal*.

En les gràfiques de la figura 2.9 s'ha considerat que per cada etapa d'intercanvi necessària per completar un escombrat s'ha aplicat la mateixa ordenació. En la figura 2.10 es compara el cost de la comunicació dels algorismes α-*optimal* i de l'algorisme de *Grau*-4 per les diferents fases d'intercanvi de l'algorisme que s'executa en un hipercub de dimensió 10. Es pot veure que per aquest cas concret, l'algorisme α-*optimal* és millor per les fases entre 2 i 8, ambdues incloses, mentre que l'algorisme de *Grau*-4 és millor per la resta de fases, la 9 i 10. En un principi es podria intentar millorar els algorismes que s'han descrit considerant un nou algorisme en el que en cada fase d'intercanvi apliqués l'ordenació que és millor. S'han realitzat algunes proves d'aquest nou algorisme però els resultats finals no són molt millors que els que s'obtenen quan s'aplica la mateixa ordenació a totes les fases de divisió, ja que el cost total està determinat gairebé per les primeres fases d'intercanvi (amb els índexos més grans) i generalment

m	P	*BR*	α-mínima	*grau-4*
8	4	3.76	3.76	3.76
	2	3.23	3.23	3.23
16	8	4.50	4.60	4.50
	4	4.26	4.26	4.26
	2	4.03	4.03	4.03
32	16	5.03	5.16	5.03
	8	5.03	5.06	5.03
	4	5.00	5.00	5.00
	2	4.56	4.56	4.56
64	32	6.03	6.03	6.03
	16	6.00	6.00	6.00
	8	5.96	6.00	5.96
	4	5.73	5.73	5.73
	2	5.00	5.00	5.00

Figura 2.11 Anàlisi de la convergència

el cost final tendirà a l'òptim si en les primeres fases d'intercanvi és possible aplicar l'algorisme α-*optimal*, o bé el cost de la comunicació tendirà a un quart del cost de la comunicació de l'algorisme BR si en les primeres fases d'intercanvi s'ha d'aplicar l'algorisme $Grau$-4.

2.5 SOBRE LA CONVERGÈNCIA DE LES NOVES ORDENACIONS

Una anàlisi detallada de la convergència de les noves ordenacions que s'han proposat en aquest capítol no està dintre dels objectius del treball desenvolupat en aquesta tesi. En general, demostrar la convergència d'una ordenació qualsevol de Jacobi és un treball

difícil [57][41][44][61]. De fet, de moltes ordenacions proposades per altres autors no se n'ha demostrat formalment la seva convergència degut a la seva dificultat.

En molts treballs presentats anteriorment podem trobar anàlisis no formals de la convergència d'algunes ordenacions. Normalment s'utilitzen matrius amb certes característiques específiques (matrius de test) i s'estudia el nombre d'escombrats necessaris per convergir amb les diferents ordenacions de Jacobi. A continuació es mostren els resultats que s'han obtingut dels tests de convergència que s'han realitzat. En la taula 2.11 es mostra el nombre d'escombrats necessaris per convergir amb els algorismes BR, BR segmentat, α-mínima i l'ordenació $Grau$-4, per matrius de diferents tamanys (m) i per diferents nombre de nodes (P). Les matrius de test utilitzades s'han generat amb números aleatoris amb una distribució uniforme amb valors dintre l'interval $[-1, 1]$, aquest mateix mètode d'anàlisi de la convergència es pot trobar a [15]. Per cada parella de valors d'm i P s'han testejat 30 matrius diferents; els resultats presentats en la taula 2.11 són els valors mitjos dels 30 tests que s'han realitzat.

Els resultats mostren que la convergència de les ordenacions proposades en aquest treball és pràcticament igual a la de l'ordenació BR.

2.6 RESUM DE CONTRIBUCIONS

En la literatura s'han proposat diversos algorismes per resoldre el problema del valors i vectors propis en una topologia en hipercub. Tots aquests algorismes en cada operació de comunicació tan sols utilitzen un únic enllaç de comunicació de l'hipercub (dels d possibles en un hipercub amb 2^d nodes). En el cas d'arquitectures $multiple - port$, aquests algorismes són poc eficients ja que no exploten el potencial paral.lelisme que ofereix l'arquitectura.

En aquest capítol s'han proposat tres nous algorismes, els quals intenten treure profit de les prestacions que ofereix l'arquitectura utilitzant més d'un enllaç de comunicació en cada operació de comunicació. Els tres algorismes proposats són:

- *BR* segmentat.

  Aquest s'obté aplicant la tècnica de la segmentació de les comunicacions a l'algorisme *BR* proposat en [8]. S'aconsegueix reduir el cost de la comunicació a la meitat respecte a l'algorisme original *BR* sense segmentar.

- α-*optimal*.

  Composat per dues ordenacions: α-mínima *BR-permutada*. L'ordenació α-mínima tan sols es defineix per subcubs de dimensions més petites o iguals a 6; en aquests casos l'ordenació és òptima. Per qüestions de temps d'execució de programa aquesta no es pot definir per subcubs de dimensió més gran que 6, en aquest cas s'aplica l'ordenació *BR-permutada*, que com ja s'ha vist és quasi òptima.

  Amb dimensions de problema gran amb l'ordenació α-*optimal*, el cost de la comunicació tendeix a dividir-se per d (màxima reducció que pot obtenir-se en un sistema de dimensió d).

- *Grau*-4.

  Amb aquesta nova ordenació per gairebé qualsevol tamany de problema s'aconsegueix reduir el cost de la comunicació a una quarta part del cost original.

Relacionats amb aquest treball s'han realitzat les següents publicacions:

- "Jacobi Orderings for Multi-port Hypercubes"; D.Royo, A. Gonzàlez i M. Valero-García. *12th. International Parallel Symposium and 9th. Symposium on Parallel and Distributed Processing.* 30 de març- 3 d'abril, 1998. Orlando, Florida (USA).

- "Low Communication overhead Jacobi Algorithms for Eigenvalue Computation on Hypercubes"; D. Royo, A. Gonzàlez i M. Valero-García. Acceptat pendent de publicació en el *Journal on Supercomputing.*

3

ALGORISMES DE JACOBI BIDIMENSIONALS

Resum

En un entorn multicomputador, els pocs algorismes que han estat proposats per malles i torus multidimensionals apliquen el mètode de Jacobi bilateral, mètode que com ja s'ha vist en el primer capítol és menys eficient per entorns multicomputadors que el mètode unilateral, per les seves necessitats de comunicació. Els algorismes que hem trobat en la literatura que resolen el problema aplicant el mètode unilateral són propostes pensades per topologies unidimensionals, línies o anells.

En aquest capítol es proposa un nou algorisme per al càlcul dels valors i vectors propis d'una matriu real i simètrica que aplica el mètode de Jacobi unilateral i que fan servir una topologia de comunicació en malla bidimensional, amb 2^r files i 2^c columnes -per qualsevol valor d'r i c-. Amb el nou algorisme es pot reduir considerablement el cost

de la comunicació respecte a les propostes per topologies unidimensionals, especialment per un nombre de nodes gran.

El capítol s'organitza de la forma següent: primer es realitza una descripció de l'arquitectura considerada, a continuació es modelitza l'algorisme unidimensional (1D) en el qual es basen la resta de propostes d'aquest capítol. En les seccions següents es descriu l'algorisme bidimensional (2D) que es proposa en aquest treball. S'avalua el comportament de l'algorisme bidimensional proposat en dos entorns diferents. Primer s'avalua l'eficiència de l'algorisme en un entorn sense cap tipus de restricció hardware, (que hem anomenat configurable) en el que l'organització dels nodes (nombre de files i columnes) del multicomputador pot decidir-se en funció de l'organització dels processos en l'algorisme. En l'entorn configurable tan sols es consideren malles de dues dimensions. Després s'avalua l'eficiència de l'algorisme en un entorn no tan flexible (que hem anomenat fix), on el nombre de files i columnes del multicomputador no es poden modificar. Aquesta anàlisi en un entorn fix es realitza per malles de dues i tres dimensions regulars amb un nombre de nodes igual a una potència de 2.

3.1 INTRODUCCIÓ

En la literatura s'han proposat algorismes per resoldre el problema de valors i vectors propis en una topologia en malla multidimensional. Per una part podem trobar algorismes per malles els quals apliquen el mètode de Jacobi *bilateral*, que com ja s'ha vist en el primer capítol no és el mètode més òptim en un entorn multicomputador degut als requeriments de comunicació del mateix mètode [59]. Per una altra banda també s'han proposat algorismes específics per sistòlics [4][40], que a part d'aplicar el mètode de resolució de Jacobi *bilateral*, són algorismes que consideren les comunicacions gairebé despreciables, aspecte que en un entorn multicomputador, DSM i xarxes d'estacions de treball no és aplicable. Altres algorismes han estat proposats per malles de processadors els quals utilitzen el mètode bilateral, com [22], [23].

La majoria d'algorismes proposats en la literatura per resoldre el problema dels valors i vectors propis en multicomputadors amb una topologia en malla o torus multidimensional i que apliquin el mètode *unilateral* [1][14], utilitzen una organització unidimensional dels nodes (en línia o anell respectivament). Per aquesta raó els hem anomenat algorismes unidimensionals, $1D$. A [15][4][61] i [45] podem trobar algunes propostes d'algorismes $1D$.

En aquest capítol es proposa un nou algorisme que utilitza una organització bidimensional dels nodes. Aquest nou algorisme l'hem anomenat algorisme bidimensional, $2D$. L'algorisme utilitza el mètode *unilateral* i l'ordenació de Jacobi que aplica és l'ordenació proposada a [15] proposada per a un anell de nodes (ordenació que s'ha descrit en detall en el capítol 1). L'algorisme $2D$ utilitza una organització arbitrària de la malla (un nombre arbitrari de files i columnes). Mitjançant models analítics demostrarem que l'algorisme $2D$ pot reduir significativament el cost de la comunicació en relació al cost de la comunicació dels algorismes $1D$.

3.2 DESCRIPCIÓ DE L'ARQUITECTURA

En aquest capítol es proposa un nou algorisme bidimensional i s'avalua la seva eficiència en dos entorns diferents:

Primer es considera el cas d'una arquitectura sense cap tipus de restriccions, que hem anomenat *configurable*. En aquest cas, es parteix d'un conjunt de nodes, 2^d, els quals poden organitzar-se físicament en malles de dues dimensions, on cadascuna de les dimensions pot tenir un nombre de nodes qualsevol sempre i quan es compleixi que $2^r \times 2^c = 2^d$, amb un total de 2^r files i 2^c columnes.

Aquesta arquitectura ens permet determinar quina és la configuració òptima de l'algorisme $2D$, nombre de files i columnes que minimitzen el cost de la comunicació. Aquesta configuració es considerarà una fita mínima del cost de la comunicació de l'algorisme $2D$ quan aquest s'executi en sistemes amb més restriccions.

A més a més, aquesta anàlisi és interessant pel cas de tenir arquitectures *configurables* (transputers) i pot ser especialment útil en el cas de sistemes en els que hi han múltiples usuaris treballant, on a cada usuari se li assigna un subconjunt de nodes del sistema. En aquest cas, seria possible escollir el grup de nodes més òptim per resoldre el problema de forma que s'aconseguiria minimitzar el cost de la comunicació; per exemple, si l'algorisme $2D$ és òptim en una malla de 2 files i 8 columnes, podria buscar-se un subconjunt de nodes amb aquesta configuració concreta.

No sempre és possible tenir arquitectures tan flexibles com les arquitectures *configurables*. Així, doncs, en la segona part del capítol es consideren arquitectures amb una configuració estàtica, que hem anomenat *fixes*. Per simplificar l'anàlisi tan sols es consideren malles de dues i tres dimensions, en les que totes les dimensions tenen el mateix nombre de nodes igual a una potència de 2. L'estudi realitzat fàcilment pot extendre's a malles en les que les dimensions no tinguin un nombre de nodes potència de 2.

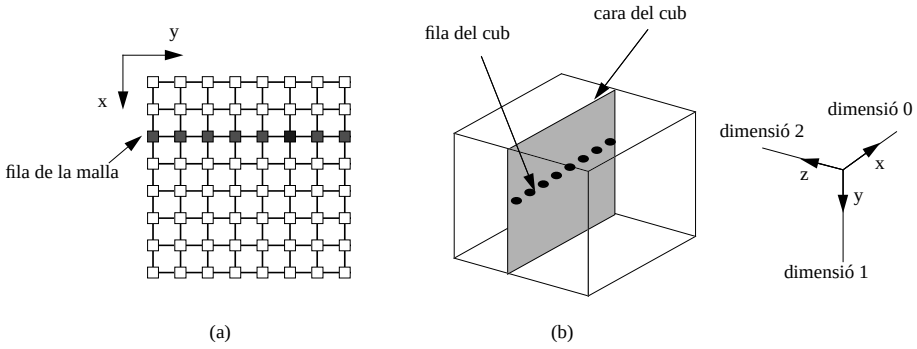


Figura 3.1 Descripció de l'arquitectura d'una malla bidimensional (a) i d'una malla de tres dimensions (b)

Pel cas de malles $fixes$ $2D$, figura 3.1(a), els nodes s'organitzen en malles regulars de $2^{d/2} \times 2^{d/2}$, amb d parell.

En qualsevol cas, malles $configurables$ i malles $fixes$, en la resta del capítol farem referència a aquesta organització simplement amb el terme *malla*. El terme *fila de la malla* s'utilitzarà per designar a tots els nodes que formen part d'una mateixa fila en la malla. Cada node de la malla s'identifica per una parella de números (x, y), els quals indiquen la fila (x) i la columna (y) de la malla als quals el node pertany, figura 3.1(a).

Pel cas de malles regulars $3D$, figura 3.1(b), els nodes s'organitzen en malles regulars de $2^{d/3} \times 2^{d/3} \times 2^{d/3}$, amb d múltiple de 3. Tan sols considerarem malles de tres dimensions en un entorn fix. En la resta del capítol farem referència a aquesta organització simplement amb el terme *cub*. Els valors 0, 1 i 2 designen les dimensions del cub tal i com es mostra en la figura 3.1(c). Cada node del cub s'identifica a partir de la tupla (x, y, z), la qual identifica les coordenades dels nodes en la dimensió 0, 1 i 2 respectivament. Es defineix *línia del cub* com el conjunt de nodes del cub que tenen les coordenades en les dimensions 1 i 2 iguals. Es defineix *cara del cub* com el conjunt de nodes que tenen la coordenada en la dimensió 1 igual.

Degut a què el patró de comunicació dels algorismes analitzats tan sols requereix que un node hagi d'enviar i rebre com a màxim un únic missatge, que la comunicació és $full\ duplex$ i que l'algorisme d'encaminament aplicat és *wormhole* [48], els models

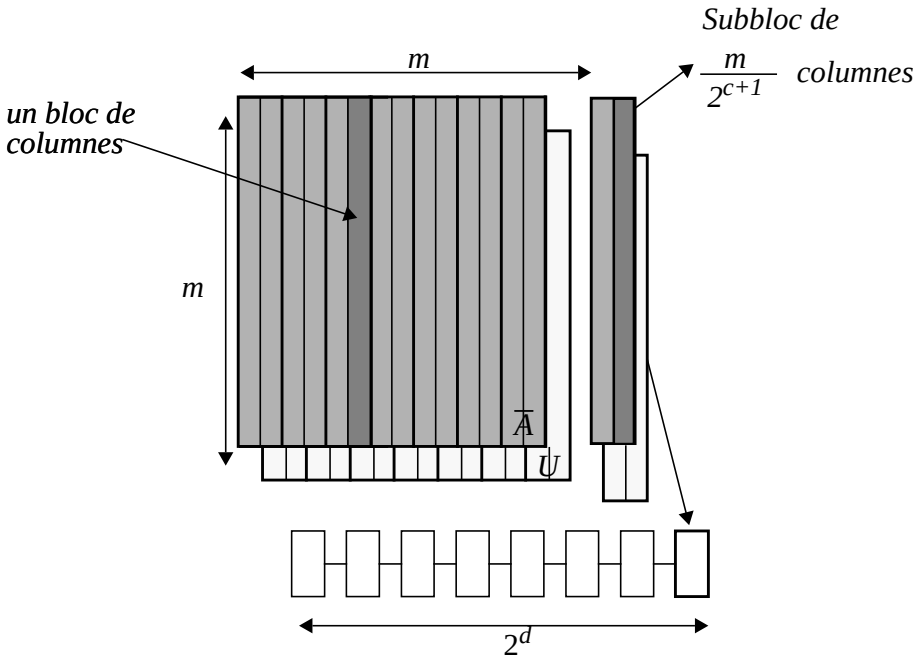


Figura 3.2 Distribució de les matrius A i U en una línia de processadors

desenvolupats en aquest capítol són vàlids tant per sistemes $one-port$ com per sistemes $all-port$ [35].

3.3 ANÀLISI DE L'ALGORISME UNIDIMENSIONAL ($1D$)

En aquesta secció es modelitza el temps d'execució de l'algorisme unidimensional, a partir d'ara algorisme $1D$, quan s'executa en una topologia en malla de dues, $2D$, i tres, $3D$, dimensions. De tots els algorismes unidimensionals que hem trobat en la literatura hem escollit l'algorisme d'Eberlein [15](descrit en el capítol 1) degut als requeriments de comunicació òptims tant per arquitectures $one-port$ com per arquitectures $all-port$.

3.3.1 Descripció de l'algorisme $1D$

Distribució de les dades

Considerem un anell amb 2^d nodes i una matriu de dimensió m. Tal i com ja s'ha vist de forma detallada en el capítol 1, en l'algorisme $1D$ les columnes de les matrius $\bar{A}$ i U es divideixen en blocs de columnes contínues de tamany igual a $m/2^{d+1}$ columnes. A cada processador de l'anell se li assignen inicialment dos blocs de la matriu $\bar{A}$ i els mateixos blocs de la matriu U. En la figura 3.2 es mostra un esquema de com quedarien les dades de les matrius $\bar{A}$ i U distribuïdes en una línia de processadors.

Organització del càlcul i la comunicació

Podem veure que per completar un escombrat són necessàries $2^{d+1}-1$ etapes de càlcul. En cada etapa de càlcul, un node calcula i aplica totes les transformacions que resulten de combinar cada columna d'un dels blocs que té assignats amb totes les columnes de l'altre bloc. En la primera etapa, a més a més, s'apliquen les transformacions que resulten de combinar una columna amb la resta de columnes del mateix bloc al que pertany. A l'hora d'aplicar una transformació, $R(p,q)$, no és necessari cap tipus de comunicació entre els nodes ja que un node té totes les dades necessàries (columnes p i q de la matriu $\bar{A}$ i de la matriu U) per al càlcul i aplicació de la transformació.

Cadascuna de les etapes de càlcul necessàries per completar un escombrat va seguida per una etapa de comunicació. En una etapa de comunicació (i), els nodes de l'anell s'intercanvien blocs de columnes per tal de generar les transformacions de l'etapa de càlcul següent ($i+1$). L'esquema de comunicació que es segueix és el que va proposar Eberlein a [15] per un anell, el qual s'ha descrit en el capítol 1. Amb aquest esquema, s'apliquen uns patrons de comunicació en els que tan sols s'intercanvia informació entre els nodes veïns en l'anell. A més a més en cada etapa de comunicació un node tan sols envia un únic bloc de columnes a un dels nodes veïns (o dreta o esquerra) i tan sols rep un únic bloc de columnes d'un dels nodes veïns (o esquerra o dreta). A diferència d'altres esquemes, com el cas de l'esquema *round robin* pensat per una línia de nodes [4], el qual en cada etapa de comunicació un node (no extrem) ha d'enviar dos blocs de

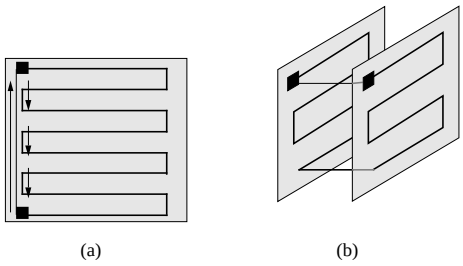


Figura 3.3 Exemple d'aplicació del mapeig "ondulat" en malles i torus multidimensionals

dades als nodes veïns en la línia (dreta i esquerra) i a la vegada ha de rebre dos blocs de dades dels nodes veïns.

3.3.2 Mapeig d'un anell en malles i torus $2D$ i $3D$

Es fàcil comprovar que en multicomputadors que presenten una organització dels nodes en malla/torus, sempre és possible mapejar un anell (amb tots els nodes del multicomputador) de manera que els nodes que eren veïns en l'anell segueixin sent veïns en la malla/torus en la que es mapegen.

Aquest objectiu s'aconsegueix utilitzant l'algorisme de mapeig que es mostra en la figura 3.3, on es pot veure un exemple concret de com es realitzaria el mapeig d'un anell en una malla de dues dimensions (3.3(a)) i com es realitzaria el mapeig d'un anell en una malla de tres dimensions (3.3(b)). Aquest mapeig l'anomenarem mapeig "ondulat" per files ja que els nodes de l'anell es mapegen en files consecutives de la malla o el torus. És possible realitzar un mapeig equivalent a l'anterior però amb columnes consecutives, al que ens referirem com a mapeig "ondulat" per columnes.

Els nodes extrems de l'anell (representats per quadrats en la figura 3.3) que tenen un enllaç directe que els uneix en l'anell, es mapegen en la malla a una certa distància. Recordem que amb l'algorisme d'encaminament considerat (*wormhole* [48]) una operació de comunicació té un cost independent de la distància. Per una altra banda no existeix cap tipus de conflicte entre les diferents operacions de comunicació que han de

realitzar els diferents nodes de l'anell. Per tant, un cop mapejat en la malla, l'execució de l'algorisme $1D$ no suposa cap tipus d'increment del cost de comunicació respecte a l'execució en l'anell.

En el cas que la topologia en la que es mapegi l'anell sigui un torus, com que es consideren topologies regulars amb un nombre de nodes potència de 2, els nodes de l'extrem en l'anell (els quadrats en el diguix 3.3) es comunicaran en el torus utilitzant l'enllaç directe *wraparound*.

D'aquesta manera tant si l'anell es mapeja en una malla com en un torus el cost de comunicació es manté igual que quan s'executava en l'anell.

3.3.3 Model analític

A l'hora de calcular els paràmetres de la transformació són necessaris tres productes interns, amb un cost en nombre d'operacions aritmètiques (multiplicacions i sumes) igual a $6 \times m$. Aplicar la transformació suposa l'actualització de quatre columnes (dues de la matriu $\bar{A}$ i dues de la matriu U), amb un total de $12 \times m$ operacions aritmètiques. En un escombrat es realitzen un total de $m(m-1)/2$ transformacions i la càrrega està totalment equilibrada entre els nodes de l'anell. Si disposem d'un total de 2^d nodes, el cost en nombre d'operacions aritmètiques que tarda cada node en aplicar el grup de transformacions que li són assignades és igual a:

$$\frac{9m^2(m-1)}{2^d}T_c$$

Al final de cada etapa, hi ha un intercanvi de blocs entre els nodes veïns en l'anell per tal de generar les transformacions de l'etapa següent. S'intercanvien dos blocs de columnes, un bloc amb $m/2^{d+1}$ columnes senceres (m elements) de la matriu $\bar{A}$ i un altre bloc de $m/2^{d+1}$ columnes senceres (m elements) de la matriu U. El mapeig *ondulat* per files permet realitzar tots aquests intercanvis sense conflictes amb un únic pas de comunicació. El cost d'enviar aquests dos blocs és igual a:

$$T_s + \frac{m^2}{2^d} \times T_e$$

Per completar un escombrat són necessàries un total de $2^{d+1} - 1$ etapes. El temps total, incloent càlcul i comunicació, necessari per completar un escombrat es modelitza com:

$$t_{1D} = \frac{9m^3 - 9m^2}{2^d} T_c + (2^{d+1} - 1)(T_s + \frac{m^2}{2^d} \times T_e) \tag{3.1}$$

L'expressió 3.1 posa de manifest que el cost de la comunicació de l'algorisme $1D$ és proporcional al nombre de nodes del multicomputador $(2^{d+1} - 1)$, per tant, per valors grans de d el cost de la comunicació creix considerablement i fa que l'algorisme $1D$ sigui poc eficient.

El model que s'ha desenvolupat en 3.1 és vàlid per anells mapejats en malles o en torus de dues o tres dimensions. Aquest model ens servirà de punt de comparació per veure com s'aconsegueix reduir el cost de la comunicació amb l'algorisme $2D$.

3.4 DESCRIPCIÓ DE L'ALGORISME BIDIMENSIONAL ($2D$)

En aquesta secció es descriu l'algorisme bidimensional que es proposa en aquest capítol, a partir d'ara algorisme $2D$, el qual es presenta com una alternativa a l'algorisme $1D$ amb la que s'aconsegueix reduir el cost de la comunicació considerablement, tal i com es mostrarà a continuació. L'algorisme $2D$ utilitza una malla de $2^r \times 2^c$ processos (amb $r + c = d$). Els càlculs del nou algorisme s'organitzen de manera que les transformacions que en l'algorisme $1D$ es calculaven i aplicaven en un únic procés (node), en l'algorisme $2D$ es calculen i s'apliquen amb col.laboració del conjunt de procesos (nodes) que formen part d'una mateixa columna de la malla o el torus $2D$. En el nou algorisme,

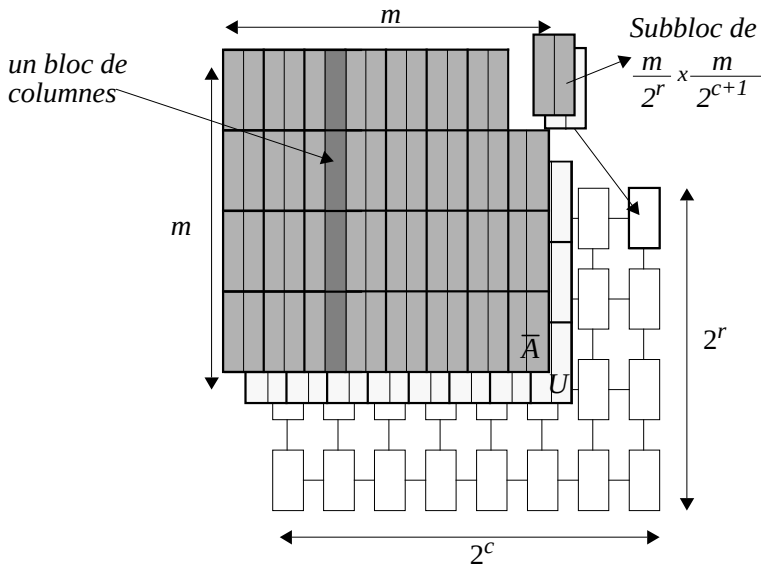


Figura 3.4 Distribució de les dades en l'algorisme $2D$ en una malla de processadors

apareix la necessitat de comunicació entre els processos (nodes) d'una mateixa columna (comunicació *vertical*) per poder calcular els paràmetres de la transformació.

En l'algorisme $2D$ parlem de comunicació *horitzontal* quan ens referim a l'intercanvi que es realitza al final de cada etapa per tal de generar els nous aparellaments dels blocs de columnes que determinaran les transformacions que s'aplicaran en l'etapa següent. Aquesta comunicació segueix el mateix patró que s'aplica en l'algorisme $1D$.

La comunicació *horitzontal* en l'algorisme $2D$ depèn del nombre de columnes de la malla (2^c). La comunicació *vertical* és proporcional a r (logaritme del nombre de files 2^r). Com veurem a continuació, si s'escullen adequadament els valors de r i c podem reduir considerablement el cost total de la comunicació respecte a l'algorisme $1D$. A continuació es descriu aquest algorisme amb més detall.

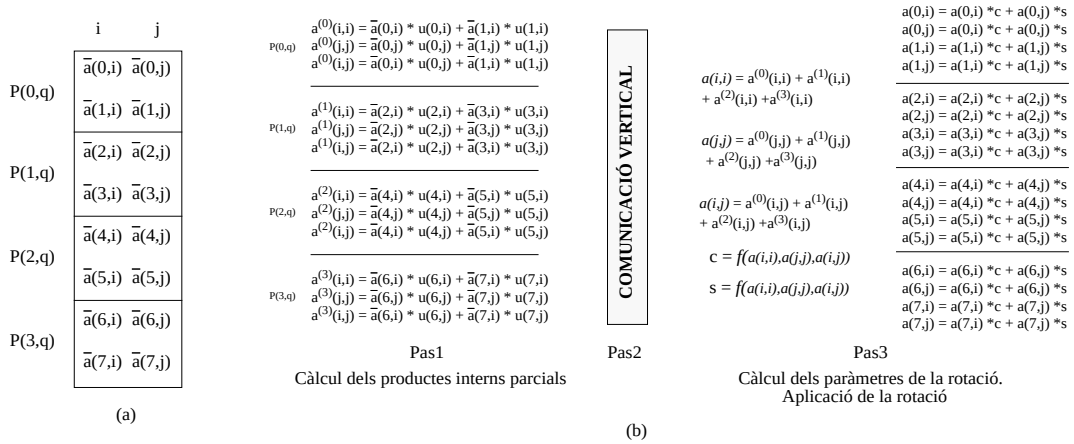


Figura 3.5 Càlcul i aplicació d'una rotació

3.4.1 Distribució de les dades

La distribució de les dades que requereix l'algorisme $2D$ és la que es mostra a la figura 3.4. Inicialment, les matrius $\bar{A}_0$ i U_0 es descomposen en 2^{c+1} blocs de $m/2^{c+1}$ columnes consecutives. Cada parella de blocs de dades (d'$\bar{A}_0$ i els corresponents blocs d'U_0) es distribueixen entre els 2^r nodes que pertanyen a una mateixa columna de la malla. Com a resultat d'aquesta distribució, a cada node se li assignen $m/2^r \times m/2^c$ elements cadascun. Dos subblocs de la matriu $\bar{A}$ i dos de la matriu U.

3.4.2 Reorganització dels càlculs

Per veure com es realitzen les operacions a l'hora de calcular i aplicar una rotació ens centrarem en el càlcul de la rotació $R_(i,j)$ corresponent a l'aparellament de les columnes (i,j). Suposarem que les columnes i i j estan distribuïdes entre tots els processadors de la columna q d'acord amb la distribució descrita en la secció anterior. Si $m = 8$ i $r = 2$ els 8 elements de les dues columnes i i j queden distribuïts entre els 4 processadors de la columna q tal i com es mostra en la figura 3.5(a). En la figura tan sols es mostren els elements d'$\bar{A}$.

Les operacions que han de realitzar tots els processadors de la columna q per tal de calcular i aplicar la rotació són les següents (figura 3.5(b)):

- $Pas1$. Primer que tot, cada processador $P(p,q)$ de la columna q, on $p \in [0..2^r-1]$, calcula els valors $a^{(p)}(i,j)$,$a^{(p)}(i,i)$ i $a^{(p)}(j,j)$ realitzant els productes interns parcials corresponents als trossos de columna disponibles en el processador.

$$a^{(p)}(i,i) = \sum_{k=p\times m/2^r}^{(m/2^r)(p+1)-1} \bar{a}_{k,i} \times u_{k,i}$$
$$a^{(p)}(j,j) = \sum_{k=p\times m/2^r}^{(m/2^r)(p+1)-1} \bar{a}_{k,j} \times u_{k,j}$$
$$a^{(p)}(i,j) = \sum_{k=p\times m/2^r}^{(m/2^r)(p+1)-1} \bar{a}_{k,i} \times u_{k,j}$$

- $Pas2$. Els processadors de la columna cooperen per tal de calcular els valors de $a(i,j)$,$a(i,i)$ i $a(j,j)$, acumulant els valors parcials calculats al $Pas1$. Al final d'aquest segon pas, tots els processadors de la columna coneixen $a(i,j)$,$a(i,i)$ i $a(j,j)$ i poden calcular els paràmetres de la rotació $R_{(i,j)}$. És en aquest pas 2 on apareix la communicació *vertical* entre els nodes de la mateixa columna.

- $Pas3$. Per últim, tots els processadors de la columna calculen de forma simultània els valors dels paràmetres c i s de la rotació i apliquen la rotació actualitzant els elements de les columnes afectades per la rotació $R_{(i,j)}$.

Després de realitzar totes les rotacions assignades a la columna de processadors, els nodes intercanvien amb els seus veïns en la fila els blocs de columnes segons el patró de comunicació corresponent a l'ordenació de Jacobi que s'estigui aplicant. Aquest intercanvi correspon a la comunicació *horitzontal* en la que com ja s'ha indicat anteriorment en l'algorisme $2D$ s'aplicarà l'ordenació d'Eberlein descrita en el capítol 1.

Tal i com s'ha descrit l'algorisme $2D$, la comunicació *vertical* té granularitat baixa, es generen molts missatges de tamany molt petit (només tres elements). Aquesta

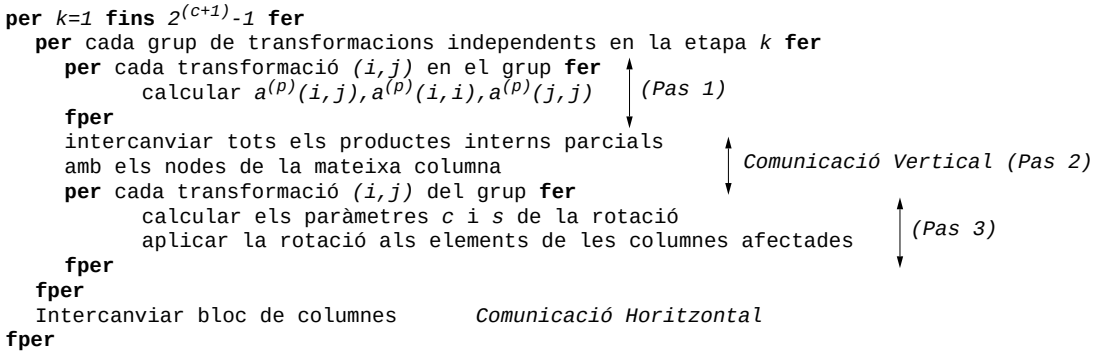


Figura 3.6 Codi per completar un escombrat.

característica pot fer que el temps de comunicació creixi considerablement si el cost d'iniciar un missatge (*startup*) és significatiu.

Amb l'objectiu de reduir al màxim el nombre de missatges que es generen en la comunicació *vertical*, les rotacions que cada columna de processadors ha de calcular i aplicar s'agrupen en grups de rotacions independents. Això suposa que per cada grup de rotacions independents:

- Es realitza el $pas1$ per totes les rotacions del grup de rotacions independents.
- Es realitza un únic intercanvi entre els processadors de la columna, comunicació *vertical*, per acumular els productes interns parcials de totes les rotacions del grup.
- Es realitza el $pas3$ per totes les rotacions del grup.

Ara en la comunicació *vertical* els processadors s'intercanvien un nombre més petit de missatges més grans, que contenen els productes parcials de vàries rotacions independents.

En la figura 3.6, es mostra el pseudocodi de l'algorisme que executarà cada node de la malla per completar un escombrat tenint en compte que en la comunicació *vertical* es treballa a nivell de grup de transformacions independents.

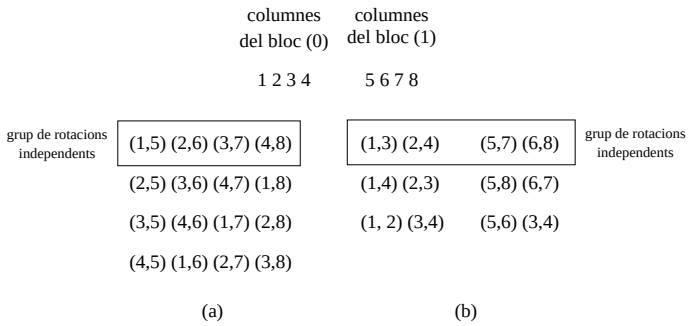


Figura 3.7 Generació de grups de rotacions independents. Com a resultat de la combinació de les columnes dels dos blocs (a) i com a resultat de la combinació de les columnes d'un mateix bloc (b)

Tenint en compte que dues rotacions, $R_{(i,j)}iR_{(m,n)}$, són independents si i, j, m i n són diferents entre sí, en la figura 3.7 es mostra un exemple de quins grups de rotacions independents es podrien formar si cada bloc té un tamany igual a quatre columnes. Cada columna s'identifica per un número i una rotació s'identifica per una parella de números (columnes). En la figura 3.7(a) es mostren els grups de rotacions independents que resulten de combinar totes les columnes d'un bloc amb totes les columnes de l'altre bloc. Aquests aparellaments s'han de realitzar en qualsevol etapa d'un escombrat. Addicionalment, en la primera etapa cada columna s'aparella amb totes les columnes del mateix bloc. En la figura 3.7(b) es mostra una possible organització en grups de rotacions independents d'aquests aparellaments addicionals per la primer etapa.

En general, en cada etapa es realitzen un total de $m/2^{c+1}$ grups de $m/2^{c+1}$ rotacions independents. A mes a més en la primera etapa com a resultat de la combinació de les columnes del mateix bloc es realitzen $m/2^{c+1} - 1$ grups de $m/2^{c+1}$ rotacions independents.

3.4.3 Esquema de la comunicació *vertical* i *horitzontal*

Primer, descriurem l'esquema de la comunicació *vertical* per tal de calcular els paràmetres d'un grup de transformacions independents. El problema de la comunicació *vertical*

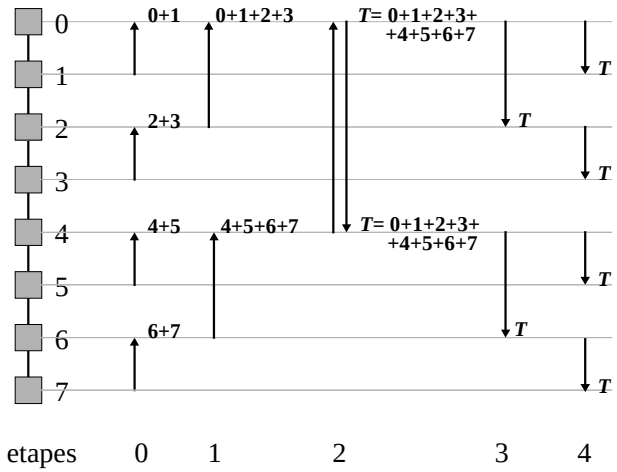


Figura 3.8 *Comunicació vertical* en una malla

consisteix, de fet, en el problema de la suma de varis vectors distribuïts entre una línia de nodes (columna de la malla).

Cadascun dels 2^r nodes d'un columna de la malla guarda un vector de $3m/2^{c+1}$ valors, que corresponen als productes interns parcials de les $m/2^{c+1}$ transformacions independents que resulten de la combinació de les columnes del grup. Tots aquests vectors s'han d'acumular de forma que al final de la comunicació *vertical* tots els nodes tenen una còpia del vector resultant, el qual conté els productes interns totals.

En la figura 3.8 es mostra un exemple de com es realitzaria la comunicació entre els nodes d'una mateixa columna suposant una línia de 2^3 nodes ($r = 3$) amb un model de comunicació *wormhole*. En aquesta figura els números identifiquen els nodes dintre la columna i a més a més identifiquen els vectors que s'han d'accumular. Les fletxes indiquen la comunicació entre els nodes. T fa referència al vector resultant. A continuació, es descriurà amb més detall l'algorisme per realitzar la comunicació *vertical*. Expressarem com $< n_{r-1}, ..., n_1, n_0 >$ la representació binària de l'enter n ($n \in [0, 2^r - 1]$).

L'algorisme consta de $2r - 1$ fases, numerades de 0 fins a $2r - 2$. En una fase i, amb $i \in [0, r - 2]$, tan sols els nodes amb etiqueta n que compleixen $n_{i-1} = ... =$

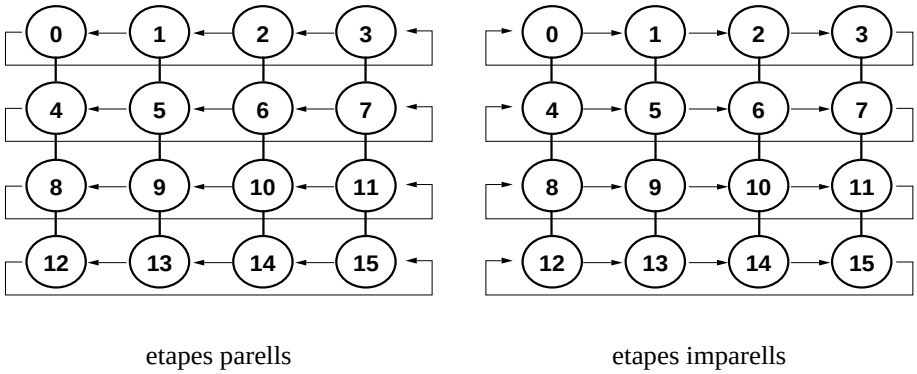


Figura 3.9 *Comunicació horitzontal* en una malla

$n_0 = 0$ estan actius. En particular, un node amb etiqueta $< n_{r-1}, ..., n_{i+1}, 1, 0, ..., 0 >$ envia el seu vector amb els productes interns parcials al node que té per etiqueta $< n_{r-1}, ..., n_{i+1}, 0, 0, ..., 0 >$, el qual, després de rebre el missatge, acumula el vector rebut al vector que ja tenia.

En la fase $r-1$, els nodes amb etiquetes 0 i 2^{r-1} intercanvien entre ells els seus vectors parcials i els acumulen. En aquest moment, ambdós nodes, 0 i 2^{r-1}, tenen una còpia del vector resultant T.

Finalment, en les últimes fases, $2r-1-i$ amb $i \in [0, r-2]$, els nodes tals que la seva etiqueta té el patró $< n_{r-1}, ..., n_{i+1}, 0, 0, ..., 0 >$ envien als nodes amb etiqueta $< n_{r-1}, ..., n_{i+1}, 1, 0, ..., 0 >$ una còpia del vector resultant T.

És important observar que totes les comunicacions que s'han de realitzar en cadascuna de les fases que s'acaben de descriure es poden realitzar en paral.lel, sense cap tipus de conflicte a l'hora d'accedir als enllaços.

Respecte a la comunicació *horitzontal*, aquesta es realitza entre els nodes d'una mateixa fila de la malla. En aquesta comunicació, les columnes de les matrius $\bar{A}$ i U s'intercanvien entre els nodes, de manera que cada node rep un nou subbloc de columnes per tal de generar les rotacions de l'etapa següent. Tots els nodes participen en la comunicació *horitzontal* (agrupats per files de la matriu) i apliquen el mateix patró de comunicació

que l'utilitzat per l'algorisme unidimensional, descrit anteriorment. Concretament els nodes intercanvien missatges de tamany igual a $m^2/2^d$ elements, que corresponen als subblocs de columnes de les matrius $\bar{A}$ i U. Tots els nodes es comuniquen en paral.lel. En la figura 3.9 es mostra un esquema de com es realitzaria la comunicació *horitzontal* en una malla de 16 nodes (4×4).

3.5 ANÀLISI DE L'ALGORISME $2D$ EN MALLES *CONFIGURABLES*

En aquesta secció s'avalua l'eficiència de l'algorisme $2D$ quan s'executa en una arquitectura $configurable$, es a dir considerem qualsevol valor de r (files) i c (columnes) sempre i quan es compleixi $r + c = d$.

3.5.1 Model analític

A continuació s'analitza amb més detall cadascuna de les parts d'aquest codi que s'ha descrit a 3.6.

Per completar un escombrat són necessàries $2^{c+1} - 1$ etapes. En cada etapa, cada columna de processadors cooperen en el càlcul i aplicació de totes les rotacions que resulten de combinar cadascuna de les columnes d'un bloc amb totes les columnes de l'altre bloc. En la primera etapa de cada escombrat ($k = 1$) cada columna de processadors és responsable de calcular i aplicar $m(m-2^c)/2^{2c+1}$ rotacions que resulten de combinar cada columna d'un bloc amb totes les columnes del mateix bloc i totes les columnes de l'altre bloc. Les rotacions resultants poden agrupar-se en $(m-2^c)/2^c$ grups amb $m/2^{c+1}$ rotacions independents cada grup. En la resta d'etapes per completar l'escombrat, on $k \in [2, 2^{c+1} - 1]$, cada columna de processadors és responsable de realitzar $m^2/2^{2c+2}$ rotacions que resulten de combinar cadascuna de les columnes d'un bloc amb totes les columnes de l'altre bloc. Les rotacions resultants poden agruparse en $m/2^{c+1}$ grups de $m/2^{c+1}$ rotacions independents cada grup.

El nombre d'operacions que cada node de la malla ha de realitzar per calcular els productes interns parcials d'una única rotació, $a^{(p)}(i,j)$,$a^{(p)}(i,i)$ i $a^{(p)}(j,j)$, és igual a $6 \times (m/2^r)$. Per tant, el nombre d'operacions per calcular els productes interns d'un bloc de $m/2^{c+1}$ rotacions independents és igual a:

$$6\frac{m^2}{2^{d+1}}$$

La comunicació *vertical* (per cada grup de rotacions independents) consisteix en intercanviar amb tots els processadors de la mateixa columna un vector de $3 \times m/2^{c+1}$ números (3 valors parcials per cada rotació del grup).

Després de la comunicació *vertical*, per cada rotació $R(i,j)$ del grup de rotacions independents s'han de calcular, a partir de $a(i,j)$, $a(i,i)$ i $a(j,j)$, els paràmetres c i s de cada rotació i a continuació aplicar cada rotació als elements de les columnes corresponents. El cost de calcular els paràmetres c i s es considera despreciable. Si el cost d'aplicar una rotació és de $12 \times (m/2^r)$ operacions, el nombre d'operacions necessàries que s'han de realitzar a l'hora d'aplicar el grup de rotacions independents que s'han calculat és:

$$12\frac{m^2}{(2^{d+1})}$$

Per últim, la comunicació *horitzontal* consisteix en intercanviar missatges de tamany igual a dos blocs de columnes, un bloc corresponent a les columnes de la matriu $\bar{A}$ i un altre bloc corresponent a les columnes de la matriu U. El tamany de cada bloc és de $m/2^r \times m/2^{c+1}$ elements. Per tant, en cada fase de comunicació *horitzontal* s'han d'intercanviar missatges de tamany igual a $m^2/2^d$ elements.

El cost total de càlcul de l'algorisme $2D$ no varia respecte al cost de l'algorisme $1D$ amb el mateix nombre de nodes. El cost és igual a:

$$t_R = (m-1)\frac{m}{2^{c+1}}\frac{18m}{2^r}T_c = \frac{9m^3 - m^2}{2^d}T_c \tag{3.2}$$

El terme $(m-1)$ correspon al nombre de grups de rotacions independents d'un escombrat, el terme $m/2^{c+1}$ és el nombre de transformacions independents per grup i per últim, el terme $18m/2^r$ és el nombre d'operacions per calcular els productes interns (fase 1) i aplicar una transformació (fase 3).

El cost de la comunicació *vertical* (incloent la suma dels vectors parcials) es pot calcular a partir de l'expressió següent:

$$t_V = (m-1)((2r-1)(T_s + \frac{3m}{2^{c+1}} \times T_e) + r(\frac{3m}{2^{c+1}} \times T_c)) \tag{3.3}$$

Aquesta expressió s'ha calculat considerant que el nombre de rotacions independents per escombrat és igual a $(m-1)$ i el cost de la comunicació *vertical* per grup detallada en la secció 1.3.2.

El cost de la comunicació *horitzontal* és el següent:

$$t_H = (2^{c+1} - 1)(T_s + \frac{m^2}{2^d} \times T_e) \tag{3.4}$$

Finalment, el cost de l'algorisme $2D$ quan s'executa en una malla $configurable$ és igual a:

$$t_{2D} = t_R + t_V + t_H$$

3.5.2 Anàlisi del rendiment

En aquesta secció es realitza un estudi del rendiment que s'obté amb l'algorisme $2D$ un cop mapejat en una malla *configurable*. Les gràfiques que es presenten a continuació poden dividir-se en dues seccions:

- Influència dels paràmetres T_e, m i T_s en el rendiment.
- Escalabilitat de l'algorisme proposat.

L'anàlisi s'ha realitzat a partir del model analític descrit en la secció anterior.

Influència dels paràmetres T_e, m i T_s

En totes les gràfiques que es presenten a continuació es considera un valor fix de d (concretament $d = 12$) i es presenta la relació dels temps d'execució dels dos algorismes, t_{1D}/t_{2D}, per tots els valors possibles d'r ($r \in [0, d]$). El cost inclou temps de càlcul i temps de comunicació dels dos algorismes ($1D$ i $2D$).

Per totes les gràfiques el cost del paràmetre T_c és fix i igual a 1. Les gràfiques intenten mostrar quin impacte tenen els paràmetres T_e , T_s i m en la configuració òptima de la malla. Per cada cas, es mantenen fixos dos dels tres paràmetres i al tercer se li donen valors diferents per tal de veure quin impacte té en la configuració òptima de la malla.

Veiem quines conclusions poden obtenir-se de les gràfiques:

- En les gràfiques de la figura 3.10 s'observa que el paràmetre T_e (temps de transmissió d'un enter en coma flotant) influeix decisivament en l'eficiència de l'algorisme. Hem assignat valors al paràmetre T_e dintre d'un rang bastant gran. Aquests valors indiquen la relació de magnituds entre el temps de realitzar una operació aritmètica en coma flotant i el temps d'enviar un número en coma flotant per la xarxa d'interconnexió d'un node a un altre.

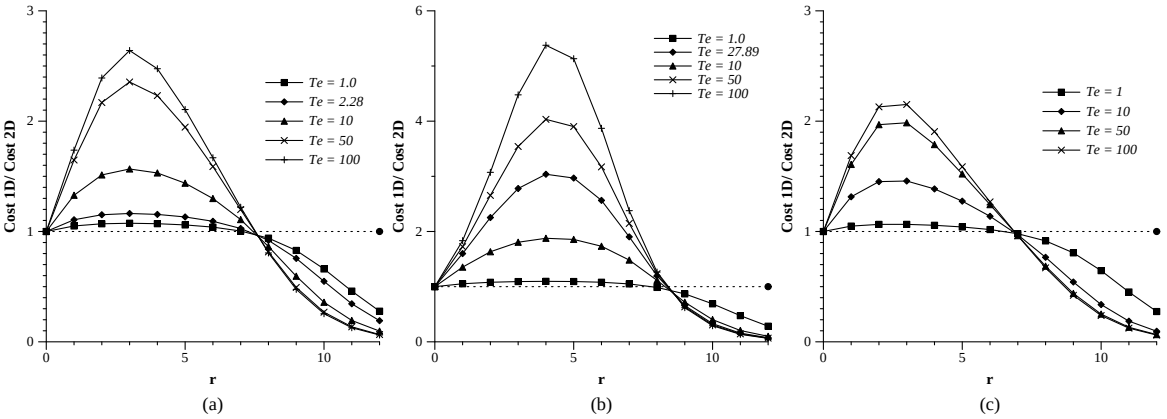


Figura 3.10 Influència de T_e. En (a) s'ha considerat la relació de valors $T_s = T_e \times 650$ corresponents a l'*intel Paragon*. En (b) s'ha considerat la relació de valors $T_s = T_e \times 70$ corresponents al $T3D$. En (c) s'ha considerat la relació de valors $T_s = T_e \times 1000$, els quals no corresponen a cap màquina concreta.

En la gràfica de la figura 3.10(a) s'avalua quin impacte en el rendiment té la variació del paràmetre T_e en un sistema amb una relació de paràmetres $T_s = 650 \times T_e$, que correspon a la relació de paràmetres de l'*intel Paragon* (capítol 1). Es pot observar, en la corba que correspon a $T_e = 2,28$ (paràmetres de l'*intel Paragon*), que podem aconseguir una reducció del cost total fins a un 20%. Per valors més grans de T_e s'aconsegueixen reduccions del cost més grans.

En la gràfica de la figura 3.10(b) s'avalua quin impacte en el rendiment té la variació del paràmetre T_e en un sistema amb una relació de paràmetres igual a $T_s = 70 \times T_e$, que correspon a la relació de paràmetres del *Cray T3D* (capítol 1). Es pot observar, en la corba que correspon a $T_e = 27.89$ (paràmetres del *Cray T3D*), que es pot aconseguir una reducció de fins a un 300%. De nou, per valors més grans de T_e s'aconsegueix reduir encara més el cost de l'algorisme.

En la gràfica de la figura 3.10(c) s'avalua quin impacte en el rendiment té la variació del paràmetre T_e en un sistema amb una relació de paràmetres igual a $T_s = 1000 \times T_e$, que no correspon a cap màquina real. En aquest cas es pot observar que per valors ronablement petits de T_e ja s'aconsegueix reduir el cost considerablement, fins a un 40% amb $T_e = 10$.

En les tres gràfiques s'ha considerat un tamany del problema igual a 2^{13}, el tamany del problema mínim que es pot resoldre en un sistema amb 2^{12} nodes.

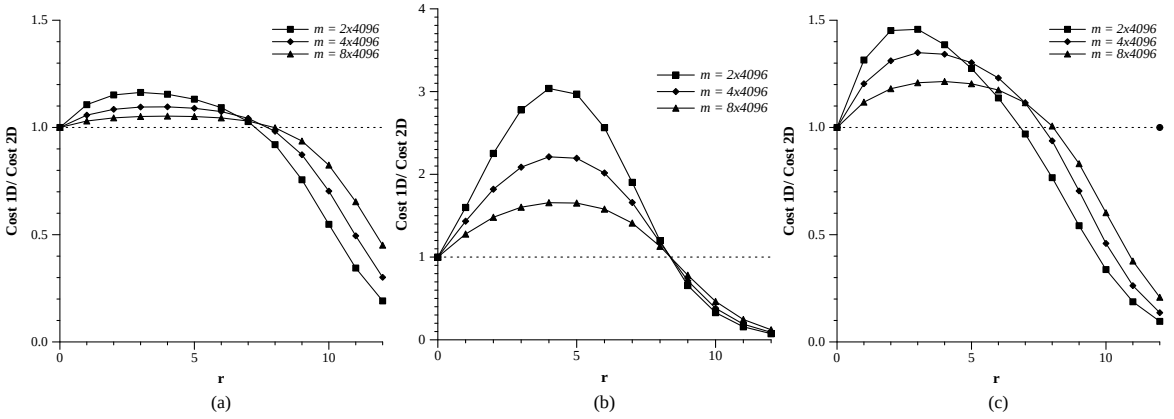


Figura 3.11 Influència del tamany del problema. En (a) s'han considerat valors de T_e, T_s i T_c de l'*intel Paragon* igual a $2, 28$, 1500 i 1 respectivament. En (b) s'han considerat valors de T_e, T_s i T_c del *Cray T3D* igual a $27, 89$, 1974 i 1 respectivament. En (c) s'han considerat valors de T_e, T_s i T_c hipotètics els quals no corresponen a cap màquina concreta i que són 10, 10000 i 1 respectivament.

- En les gràfiques de la figura 3.11 s'observa que el benefici que podem obtenir és bastant sensible als diferents valors m (tamany de la matriu). En aquest cas, però, el benefici decreix quan el tamany d'm s'incrementa, al contrari que en el cas del paràmetre T_e.

  En la gràfica de la figura 3.11(a) s'han considerat valors de T_e, T_s i T_c igual a $2, 28$, 1500 i 1 respectivament, que corresponen a les relacions de paràmetres de l'*intel Paragon* especificats en el capítol 1 . En aquest cas el rendiment varia d'un 20% fins a un 5%.

  En la gràfica de la figura 3.11(b) s'han considerat valors de T_e, T_s i T_c igual a $27, 85$, 1974 i 1 respectivament, valors que coincideixen els paràmetres d'un *Cray T3D* especificats en el capítol 1. En aquest cas el rendiment varia d'un 300% fins a un 50%.

  Per últim, en la gràfica de la figura 3.11(c) s'han considerat valors de T_c,T_e i T_s igual a 1, 10 i 10000 respectivament. Valors que no corresponen a cap màquina real. En aquest cas el rendiment varia d'un 50% fins a un 20%.

- En les gràfiques de la figura 3.12 es pot observar que són necessaris valors molt grans de T_s (temps d'iniciar una operació de comunicació, *startup*) per tal de veure l'impacte d'aquest paràmetre en el rendiment del sistema.

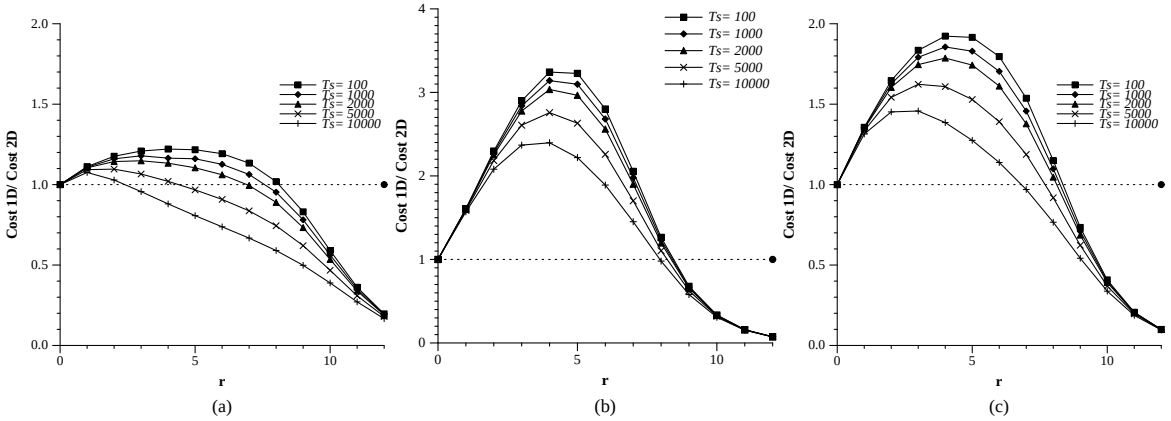


Figura 3.12 Influència del T_s. En (a) s'han considerat valors de T_e i T_c de l'*intel Paragon* igual a $2,28$ i 1 respectivament. En (b) s'han considerat valors de T_e i T_c del *Cray T3D* igual a $27,89$ i 1 respectivament. En (c) s'han considerat valors de T_e i T_c hipotètics els quals no corresponen a cap màquina concreta i que són 10 i 1 respectivament.

En les tres gràfiques s'observa un decrement del rendiment quan s'incrementa T_s considerablement i a més a més l'increment de T_s modifica la configuració òptima de la malla, la qual tendeix a reduir el nombre de files.

Els valors de T_c i T_e són 1 i $2,28$ respectivament en la gràfica 3.12(a), que es corresponen als valors dels paràmetres de l'*intel Paragon*.

Els valors de T_c i T_e són 1 i 27.85 respectivament en la gràfica 3.12(b), que corresponen als valors dels paràmetres d'un *Cray T3D*.

Per últim, en la gràfica de la figura 3.12(c) s'han considerat valors de T_c,T_e i T_s iguals a 1, 10 i 10000 respectivament, valors que no pertanyen a cap màquina real.

En les tres gràfiques es considera un tamany de matriu igual a 2^{13}, el tamany de problema mínim que és possible resoldre en un sistema amb 2^{12}nodes.

- La configuració òptima generalment té menys files que columnes ja que el cost de la comunicació *vertical* creix ràpidament quan s'incrementa el nombre de files del sistema.

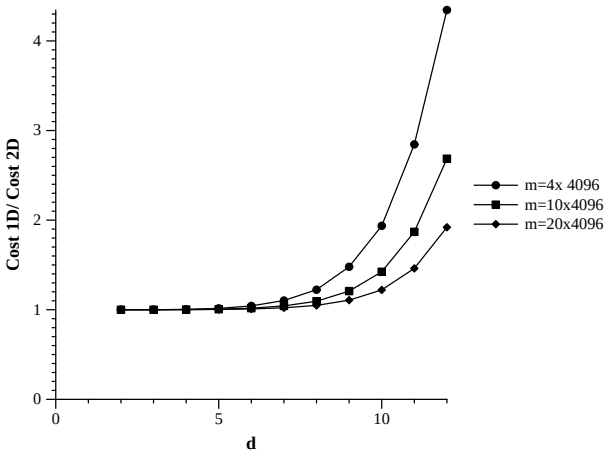


Figura 3.13 Anàlisi de l'escalabilitat

Escalabilitat de l'algorisme $2D$

En la gràfica de la figura 3.13 es mostra quin és el comportament de l'algorisme $2D$ quan el sistema escala. Aquesta gràfica mostra la relació t_{1D}/t_{2D} per valors fixats de T_s, T_e i variant els paràmetres d i m (en aquest cas m s'escala respecte al nombre de nodes del sistema). Per un valor determinat de d, es considera la configuració òptima de l'algorisme $2D$. La gràfica mostra que el benefici que es pot obtenir amb l'algorisme $2D$ creix quan s'incrementa el nombre de nodes del sistema. Aquest increment és considerablement gran per mides de problema petits i quan d augmenta.

3.6 ANÀLISI DE L'ALGORISME $2D$ EN MALLES $FIXES$ $2D$

En la secció anterior s'ha modelitzat l'algorisme $2D$ en un sistema $configurable$. Aquesta anàlisi ens permetia determinar quina és la configuració òptima (nombre de files i columnes que minimitza el temps d'execució) donat un nombre determinat de nodes (potència de 2). Molt probablement, si el que volem és utilitzar tots els nodes d'un sistema multicomputador, la topologia és $fixa$ i no és possible modificar-la. Així doncs, a continuació veurem com podem mapejar l'algorisme $2D$ descrit detalladament en la secció 2.3.2, en un multicomputador amb una topologia bidimensional regular quadrada

de $2^d = 2^{d/2} \times 2^{d/2}$ nodes. També es construeix el model del cost total de l'algorisme proposat i es realitza la seva avaluació a partir del model construït.

3.6.1 Mapeig de l'algorisme $2D$ en la malla

A l'hora de mapejar l'algorisme $2D$ en una malla, la solució immediata és considerar l'algorisme $2D$ amb $r = c = d/2$ i mapejar cadascun dels nodes de l'algorisme $2D$ en el node corresponent de la malla. Els resultats obtinguts per malles $configurables$ suggereix que aquesta pot no ser la solució òptima. Hem cregut necessari, doncs, considerar totes les possibles solucions i analitzar-les. El procediment correcte és considerar el mapeig de l'algorisme $2D$ ($2^r \times 2^c$ amb qualsevol valor de r i c sempre i quan es compleixi $r + c = d$) en la malla i construir un model analític del temps d'execució per l'algorisme $2D$ un cop mapejat. Aquest model ens permetrà determinar quins valors d'r i c són els òptims, els que minimitzen el temps d'execució de l'algorisme $2D$ en la malla.

S'han considerat dues formes diferents de mapejar l'algorisme $2D$ en la malla. El primer mapeig que es presenta l'hem anomenat Horizontal Communication Preserving, abreujat HCP. Aquest mapeig es caracteritza per què cada fila de l'algorisme $2D$ es mapeja en una o vàries files contínues de la malla, utilitzant el mapeig "ondulat" per files descrit en la secció 1.2. Amb aquest mapeig es manté la localitat de la comunicació *horitzontal* de l'algorisme $2D$ un cop mapejat en la malla, de manera que el cost es pot modelitzar de la mateixa manera que en el cas de les malles $configurables$, l'expressió t_H de la secció 1.4.2. Per una altra banda, en aquest mapeig el cost de la comunicació *vertical* pot incrementar-se respecte al cost de la comunicació *vertical* de la malla $configurable$, degut a que vàries columnes de l'algorisme $2D$ es mapegen en la mateixa columna de la malla de forma intercalada, i a l'hora de realitzar la comunicació *vertical* pot haver-hi conflictes a l'hora d'accedir als enllaços. Per exemple, en la figura 3.14(a) tenim dues columnes de l'algorisme $2D$ mapejades en una mateixa columna de la malla. En la primera columna de la malla trobem que s'han mapejat de forma intercalada dues columnes de l'algorisme $2D$, una columna està formada pels nodes pintats de negre i l'altra columna està formada per la resta de nodes de la columna de la malla (nodes en blanc i en gris). Quan les dues columnes hagin de realitzar la comunicació *vertical* hi

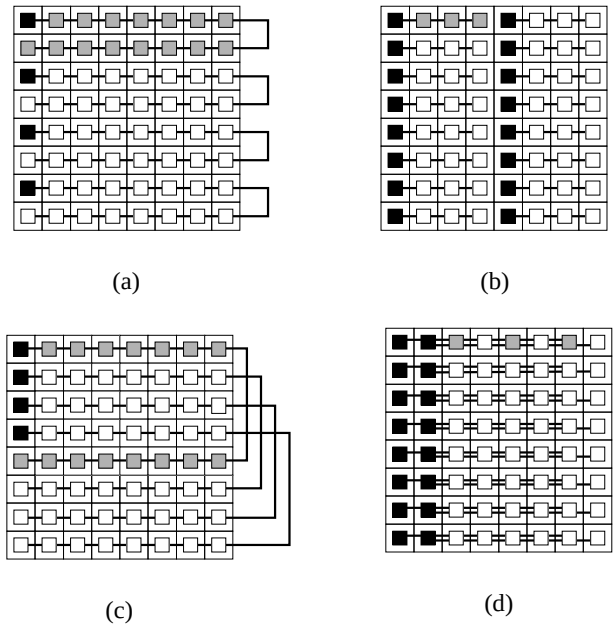


Figura 3.14 Un exemple de (a) mapeig HCP amb $r = 2$, (b) mapeig HCP amb $r = 4$, (c) mapeig VCP amb $r = 2$, i (d) mapeig VCP amb $r = 4$. En tots els casos $d = 6$

haurà conflicte d'accés als enllaços ja que els camins de les dues columnes de l'algorisme $2D$ es solapen en la malla.

En les figures 3.14(a) i 3.14(b) es mostren dos exemples de com es realitzaria el mapeig HCP. En aquestes figures les línies gruixudes identifiquen les files de l'algorisme $2D$, els quadrats negres identifiquen els nodes situats més a l'esquerra de cada fila de l'algorisme $2D$, els quadrats blancs identifiquen els nodes situats més a l'esquerra de cada columna i per últim els quadrats grisos identifiquen la resta dels nodes de la malla que formen part de la primera fila de l'algorisme $2D$.

El segon mapeig que es proposa l'hem anomenat Vertical Communication Preserving, abreujat VCP. En aquest mapeig, cada columna de l'algorisme $2D$ es mapeja en una o més columnes contínues de la malla per tal de mantenir la localitat de la comunicació *vertical*. L'algorisme de mapeig que s'aplica és el mapeig "ondulat" per columnes. En aquest cas el cost de la comunicació *horitzontal* s'incrementa respecte a la comunicació *horitzontal* de la malla $configurable$ degut a què vàries files de l'algorisme $2D$ estan

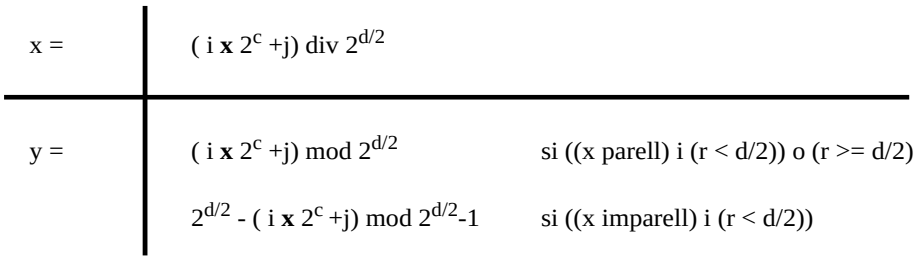


Figura 3.15 Definició del mapeig HCP en la malla

intercalades en la malla, i per tant això provoca l'aparició de conflictes d'accés als enllaços de la malla a l'hora de realitzar la comunicació *horitzontal* entre les diferents files intercalades.

En les figures 3.14(c) i 3.14(d) es mostren dos exemples de com es realitzaria el mapeig VCP. En aquestes figures les línies gruixudes identifiquen les files de l'algorisme $2D$, els quadrats negres identifiquen els nodes situats més a l'esquerra de cada fila de l'algorisme $2D$, els quadrats blancs identifiquen els nodes situats més a l'esquerra de cada columna i per últim els quadrats grisos identifiquen la resta dels nodes de la malla que formen part de la primera fila de l'algorisme $2D$.

Dels dos mapejos que s'han descrit, aplicarem el mapeig HCP i descartarem el mapeig VCP ja que en les proves realitzades s'ha pogut comprovar que pel rang de valors dels paràmetres T_e, T_s i T_c que s'han considerat, amb el mapeig HCP sempre s'obté un cost de comunicació més petit que amb el mapeig VCP.

Aquests resultats semblen bastant lògics tenint en compte que el pes de la comunicació *horitzontal* és més gran que el pes que pot tenir la comunicació *vertical*, i amb el mapeig HCP la comunicació horitzontal es manté exactament igual que pel cas d'una malla $configurable$ (per tant mínima) a cost d'un petit increment de la comunicació *vertical*.

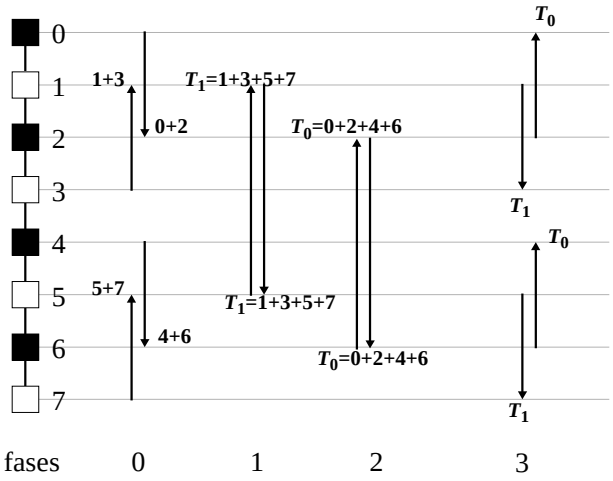


Figura 3.16 Exemple de l'algorisme per implementar $IVC_{k,l}$, amb $k = 1$ i $l = 3$

3.6.2 Model analític

El model analític que es desenvolupa a continuació es correspon amb l'algorisme $2D$ quan es mapeja en la malla segons l'algorisme de mapeig HCP. En el mapeig HCP, cada fila de l'algorisme $2D$ es mapeja en una o vàries files contínues de la malla. Quan $r < d/2$, com a l'exemple de la figura 3.14(a), per garantir la relació de veïns dels nodes que formen part d'una mateixa fila, s'utilitza un mapeig "ondulat" per files, que mapeja una fila de l'algorisme $2D$ en un total de $2^{d/2-r}$ files contínues de la malla. Quan $r > d/2$, com en l'exemple de la figura 3.14(b), vàries files de l'algorisme $2D$ es mapegen en una fila de la malla.

En general, amb el mapeig HCP cada node (i, j) de l'algorisme $2D$ ($i \in [0, 2^r - 1], j \in [0, 2^c - 1]$) es mapeja en el node (x, y) de la malla ($x, y \in [0, 2^{d/2} - 1]$), d'acord amb les expressions de la taula 3.15.

A l'hora de construir el model del cost de comunicació cal recordar que el mapeig HCP no modifica el cost de la comunicació *horitzontal*, respecte al cost de la comunicació *horitzontal* de la malla $configurable$ (expressió 3.4 en la secció 3.5.1) Quant a la comunicació *vertical* cal distingir tres situacions diferents:

- Cas 1: $r = d/2$

  La comunicació *vertical* es pot realitzar tal i com s'ha descrit per la malla $configurable$, ja que en aquest cas els nodes veïns en l'algorisme $2D$ també són nodes veïns un cop es mapegen en la malla. A partir d'ara, aquest cas de comunicació *vertical* l'anomenarem Contiguous Vertical Communication, abreujat com $CVC_{d/2}$. El subíndex indica que en la comunicació *vertical* hi participen un total de $2^{d/2}$ nodes de la malla.

- Cas 2: $r < d/2$

  En aquest cas, $2^{d/2-r}$ columnes de l'algorisme $2D$ es troben intercalades en cadascuna de les columnes de la malla. La figura 3.14(a) mostra un exemple d'aquesta situació. L'operació que es requereix per realitzar la comunicació *vertical* en aquest cas l'anomenarem Interleaved Vertical Communicaction, abreujat $ICV_{k,l}$. Els subíndexos indiquen que les columnes de la malla estan formades per 2^l nodes i que un total de 2^k columnes de l'algorisme $2D$ estan intercalades en cada columna de la malla. Per exemple, la comunicació *vertical* que s'ha de realitzar en l'exemple concret de la figura 3.14(a) s'expressaria com $IVC_{1,3}$ ja que cada columna de la malla està formada per 2^3 nodes i en cada columna de la malla s'intercalen 2^1 columnes de l'algorisme $2D$.

  A l'hora d'implementar $ICV_{k,l}$ s'utilitza un algorisme una mica diferent a l'utilitzat anteriorment pel cas CVC. La figura 3.16 mostra un exemple de l'algorisme que es proposa pel cas concret de $ICV_{1,3}$. Els quadrats blancs i negres permeten distingir les dues columnes de l'algorisme $2D$ que es troben intercalades en la columna de la malla. En general, quan tenim més de dues columnes intercalades en una columna de la malla, aquestes s'organitzen en grups de 2. Cal recordar que el nombre de files i columnes dels algorismes sempre són potències de 2 i per tant sempre podrem aparellar totes les columnes de l'algorisme $2D$. Les comunicacions corresponents a cada parella de columnes poden realitzar-se de forma simultània, en direccions oposades en la columna de la malla, utilitzant el mateix esquema que pel cas de CVC. Tan sols les comunicacions corresponents a la fase central del cas CVC no poden realitzar-se simultàniament.

  Així doncs, la comunicació *vertical* pel cas $IVC_{k,l}$ es realiza en un total de $2 \times (l - k)2^{k-1}$ fases, ja que tenim 2^{k-1} parelles de columnes, cada columna de l'algorisme

$2D$ consta d'un total de 2^{l-k} nodes i la comunicació *vertical* de cada parella de columnes pot realitzar-se amb $2(l-k)$ fases.

- Cas 3: $r > d/2$

  En aquest cas, cada columna de l'algorisme $2D$ es troba repartida entre un total de $2^{r-d/2}$ columnes no contínues de la malla. Cadascuna d'aquestes columnes es troba intercalada per un total de 2^{d-r} columnes de la malla que han estat assignades a unes altres tantes columnes de l'algorisme $2D$. Un exemple d'aquest cas és el de la figura 3.14(b). La comunicació *vertical* en aquest cas es completa en dues fases. En la primera fase es realitza un $CVC_{d/2}$ en cada columna de la malla. A continuació, un $IVC_{d-r,d/2}$ es realitza en cada fila de la malla.

El cost de la comunicació *vertical* d'un escombrat complet és igual a:

$$\begin{array}{ll} (m-1) & t_{CVC_{d/2}} \qquad (r = d/2) \\ (m-1) & t_{IVC_{d/2-r,d/2}} \qquad (r < d/2) \\ (m-1)(t_{CVC_{d/2}} + t_{CVC_{d-r,d/2}}) & \qquad (r > d/2) \end{array} \tag{3.5}$$

on:

$$t_{CVC_k} = (2k-1)(T_s + \frac{3m}{2^{c+1}} \times T_e) + k(\frac{3m}{2^{c+1}} \times T_c)$$
$$t_{IVC_{k,l}} = 2^{k-1}(2(l-k)(T_s + \frac{3m}{2^{c+1}} \times T_e) + (l-k)(\frac{3m}{2^{c+1}} \times T_c))$$

Les expressions de 3.5 s'han obtingut considerant que el nombre de transformacions independents en cada escombrat és $(m-1)$ i el cost de la comunicació *vertical* es realitza per grups de rotacions independents. Finalment, el temps total d'execució de l'algorisme pot expressar-se:

$$t_{2D} = t_R + t_H + t_V$$

on les expressions de t_R i t_H són les que s'han avaluat pel cas de la malla $configurable$ en la secció 2.4.2 i t_V correspon a l'expressió 3.5.

3.6.3 Anàlisi del rendiment

En aquesta secció es realitza un estudi de l'eficiència de l'algorisme $2D$ un cop mapejat en una malla regular $2D$. Aquesta anàlisi es realitzarà a partir dels models analítics desenvolupats en les seccions anteriors. Les gràfiques que es presenten poden dividir-se en tres seccions:

- La influència d'alguns paràmetres en el rendiment de l'algorisme.
- L'escalabilitat dels algorismes proposats.
- L'eficiència del mapeig utilitzat, el mapeig HCP.

Influència dels paràmetres T_e, m i T_s

Les gràfiques que es mostren a continuació es corresponen a l'anàlisi de l'algorisme quan es mapeja en una malla regular, suposant un sistema amb un nombre de nodes igual a 2^{12} ($d = 12$). En les gràfiques es mostra la relació t_{1D}/t_{2D} per tots els valors possibles del paràmetre r ($r \in [0, d]$), sent 2^r el nombre de files de l'algorisme $2D$, i variant els valors dels paràmetres T_e, m i T_s respectivament. El paràmetre T_c es fix i igual a 1.

En aquesta secció, on es realitza una anàlisi de la malla $fixa$, s'han tingut en compte tan sols dos sistemes. El primer correspon a un sistema amb uns valors de T_c, T_e i T_s iguals a 1, 2, 28 i 1500 que coincideixen amb els paràmetres de l'$intel\ Paragon$ que té una topologia en malla, i el segon correspon a un sistema hipotètic amb uns valors de T_c, T_e i T_s iguals a 1, 10 i 1000 amb la idea de considerar un sistema amb un rang de valors més gran.

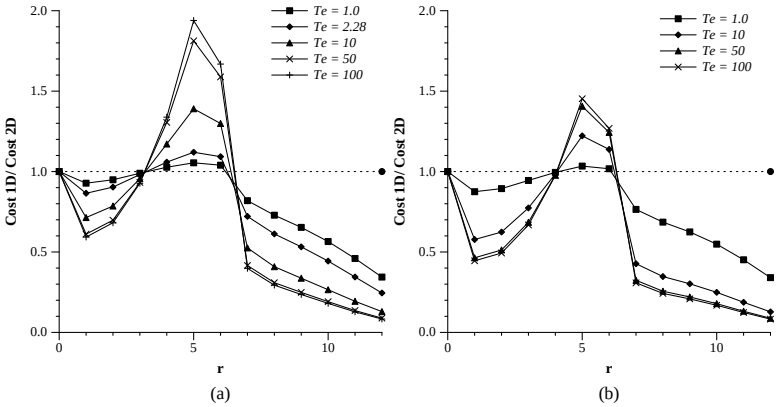


Figura 3.17 Impacte del paràmetre T_e. En la gràfica (a) s'ha considerat la relació de paràmetres de l'*intel Paragon* $T_s = 650 \times T_e$. En la gràfica (b) s'ha considerat la relació de paràmetres $T_s = 1000 \times T_e$.

- En les gràfiques de la figura 3.17 s'observa, de nou, que el paràmetre T_e (temps de transmissió d'un enter en coma flotant) influeix decisivament en l'eficiència de l'algorisme. Hem assignat valors al paràmetre T_e dintre un rang bastant gran. Aquests valors indiquen la relació de magnituds entre el temps de realitzar una operació aritmètica en coma flotant i el temps d'enviar un número en coma flotant per la xarxa d'interconnexió d'un node a un altre.

  En la gràfica de la figura 3.17(a) s'avalua quin impacte té la variació del paràmetre T_e en el rendiment del sistema. S'ha considerat un sistema amb una relació de paràmetres $T_s = 650 \times T_e$ (corresponent a la relació de paràmetres de l'*intel Paragon*). Es pot observar, en la corba amb $T_e = 2,28$, que podem reduir el cost un 10% en el cas de tenir un sistema amb un valor de paràmetres iguals a l'*intel Paragon*. Mentre que si mantenim la relació $T_s/T_e = 650$ i incrementem el valor de T_e el rendiment del sistema creix considerablement.

  En la gràfica de la figura 3.17(b) s'avalua la variació del paràmetre T_e en un sistema amb una relació de paràmetres $T_s = 1000 \times T_e$, considerant un rang de valors més gran. Es pot observar que per valors de T_e grans la reducció del cost total és d'un 50%, mentre que per valors de T_e més petits la reducció és més petita; per $T_e = 10$ es redueix el cost total en un 20%.

  En les tres gràfiques s'ha considerat un tamany del problema igual a 2^{13}, el tamany del problema mínim que es pot resoldre en un sistema amb 2^{12} nodes.

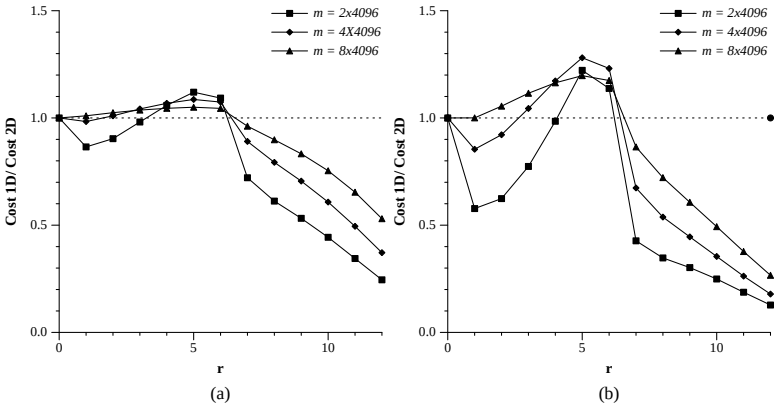


Figura 3.18 Impacte del paràmetre m. En la gràfica (a) s'ha considerat els paràmetres de l'*intel Paragon*, $T_s = 1500$ i $T_e = 2,28$. En la gràfica (b) s'ha considerat els paràmetres $T_c = 1$, $T_s = 10000$ i $T_e = 10$.

- En les gràfiques de la figura 3.18 s'observa que el benefici que podem obtenir és bastant sensible als diferents valors d'm (tamany de la matriu). En aquest cas, però, el benefici decreix quan el tamany d'm s'incrementa, al contrari que en el cas del paràmetre T_e.

  La gràfica de la figura 3.18(a) correspon al cas en què els valors dels paràmetres T_e, T_s i T_c és igual a $2,28$, 1500 i 1 respectivament, valors que coincideixen amb la relació de paràmetres d'un *intel Paragon*. En aquest cas el rendiment varia d'un 12% fins a un 5%.

  En la gràfica de la figura 3.18(b) s'han considerat valors de T_c,T_e i T_s iguals a 1, 10 i 10000 respectivament. En aquest cas, considerant tamanys de problema igual que en la gràfica 3.18(a), el rendiment varia d'un 30% fins a un 20%.

- En les gràfiques de la figura 3.19 es pot observar quin és l'impacte de T_s en el rendiment del sistema.

  En les dues gràfiques 3.19(a) (amb paràmetres de l'*intel Paragon*) i 3.19(b) (amb $T_e = 10$) s'observa un comportament equivalent del sistema, per una part hi ha una reducció del benefici quan s'incrementa T_s i a més a més l'increment de T_s modifica la configuració òptima de la malla, la qual tendeix a reduir el nombre de files.

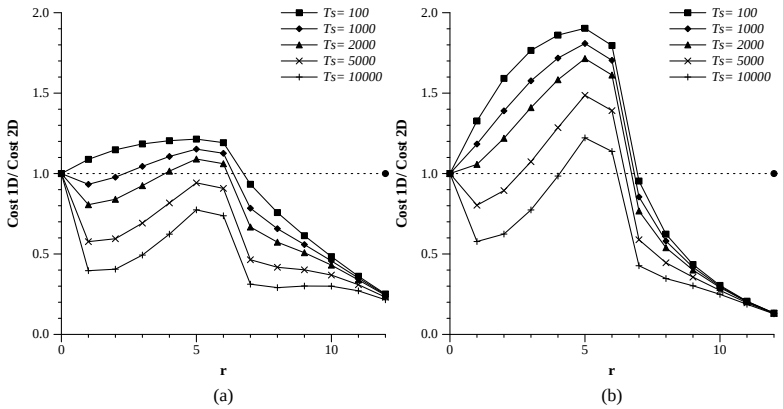


Figura 3.19 Impacte del paràmetre T_s. En la gràfica (a) s'ha considerat els paràmetres de l'*intel Paragon*, $T_c = 1$ i $T_e = 2,28$. En la gràfica (b) s'han considerat els paràmetres $T_c = 1$ i $T_e = 10$.

En general es pot observar que en el cas de malles fixes el rendiment del sistema és més petit que en el cas de malles configurables. Això és degut al cost que afegeix l'algorisme de mapeig HCP que s'ha aplicat.

Anàlisi de l'escalabilitat de l'algorisme $2D$

La gràfica de la figura 3.20 mostra quin és el comportament de l'algorisme $2D$ quan la malla escala.

Aquesta gràfica mostra la relació t_{1D}/t_{2D} per uns valors determinats dels paràmetres T_e i T_s i variant els paràmetres m i d. El valor del paràmetre m s'escala respecte al paràmetre d. En aquest cas, donat un valor de d, es considera el valor òptim d'r per determinar el temps d'execució en cada cas, t_{2D}. La gràfica ens mostra quin és l'increment de benefici que podem obtenir amb l'algorisme $2D$ quan s'incrementa el nombre de nodes del sistema. Aquest increment és particularment gran per valors d'm petits i per valors de T_e grans.

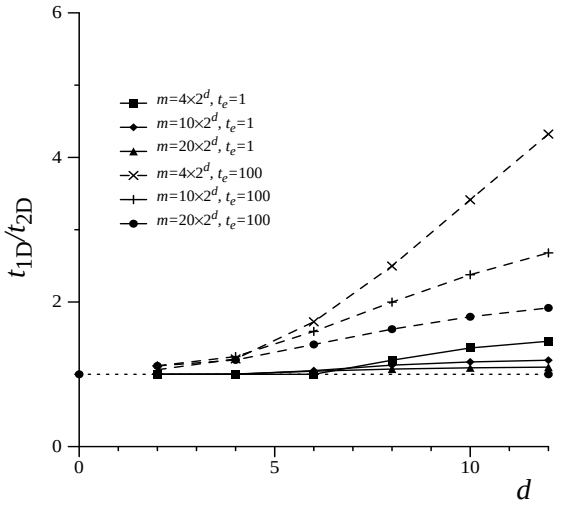


Figura 3.20 Anàlisi de l'escalabilitat de l'algorisme $2D$ quan es mapeja en una malla $2D$ $fixa$.

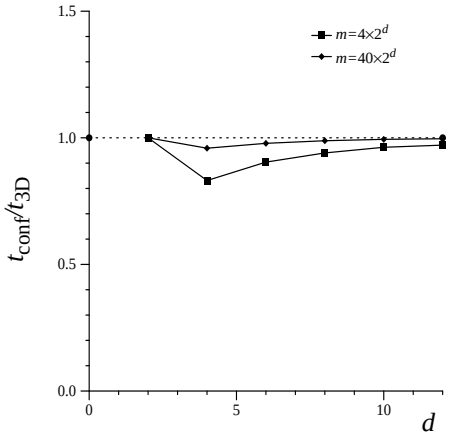


Figura 3.21 Comparació del cost de l'algorisme $2D$ quan s'executa en una malla i s'aplica el mapeig HCP amb el cost de l'algorisme en un entorn $configurable$

Avaluació de l'eficiència del mapeig HCP

El mapeig HCP s'ha escollit per tal de conservar el cost de la comunicació *horitzontal* exactament igual que en el cas de tenir malles $configurables$, concretament corre-

spon al terme T_H de l'expressió T_{2D} (expressió 3.4) de la secció 3.5.1. Per una altra banda, aquest mapeig probablement produirà un increment de la comunicació *vertical* respecte al cas de les malles $configurables$, que correspon al terme T_V de l'expressió t_{2D} (expressió 3.3).

El que volem veure és quin és el cost afegit que inclou el fet d'aplicar l'algorisme de mapeig HCP per mapejar l'algorisme $2D$ en la malla. Recordem que amb aquest mapeig s'aconsegueix que els nodes d'una mateixa fila en la malla es mapegin a distància 1 (la comunicació *horitzontal* es mantingui exactament igual que en una malla $configurable$) i encara que els nodes d'una mateixa columna de la malla poden ser mapejats a distància més gran que 1.

Dit d'una altra forma, ens preguntem si interessa buscar un mapeig alternatiu al HCP amb el que s'obtingui un increment de benefici més gran que amb el mapeig HCP. Considerem el temps d'execució de l'algorisme t_{2D} una fita inferior del temps d'execució de l'algorisme $2D$ en el sistema considerat (malles $2D$ $fixes$). Llavors el que ens interessa és que el temps d'execució de l'algorisme $2D$ un cop mapejat en la malla aplicant el mapeig HCP sigui proper a la fita inferior considerada.

En la gràfica de la figura 3.21 es mostra la relació t_{Conf}/t_{2D}. Els valors de t_{Conf} i t_{2D} s'han obtingut considerant el valor òptim del paràmetre r. Com es pot veure en la gràfica que s'ha obtingut, amb el mapeig HCP es pot obtenir un benefici molt proper a l'òptim especialment per tamanys de problema grans. Per tant podem considerar que el mapeig HCP és suficientment bò.

3.7 ANÀLISI DE L'ALGORISME $2D$ EN MALLES $FIXES$ $3D$

En aquesta secció, es modelitza l'algorisme $2D$ quan es mapeja en un cub regular de tres dimensions amb un total de 2^d nodes, estructurats de manera regular $2^d = 2^{d/3} \times 2^{d/3} \times 2^{d/3}$.

z =	(i **x** 2^c +j) div $2^{2d/3}$	
y =	((i **x** 2^c +j) mod $2^{2d/3}$) div $2^{d/3}$	
x =	(i **x** 2^c +j) div $2^{2d/3}$	si ((y parell) i (r >= 2d/3))
	$2^{2d/3}$ - (i **x** 2^c +j) div $2^{2d/3}$-1	si ((y imparell) i (r < 2d/3))

Figura 3.22 Definició del mapeig HCP en el cub

3.7.1 Mapeig de l'algorisme $2D$ en el cub

Com en el cas de la malla, a l'hora de mapejar l'algorisme $2D$ en el cub és possible aplicar el mapeig HCP i el mapeig VCP. Pels mateixos motius que pel cas anterior, s'aplicarà el mapeig HCP i es descartarà el mapeig VCP.

A continuació es defineix formalment com es realitza el mapeig HCP en un cub. Donat un node (i, j) $(i \in [0, 2^r - 1], j \in [0, 2^c - 1])$ de l'algorisme $2D$ aquest el mapeja en el node (x, y, z) $(x, y, z \in [0, 2^{d/3} - 1])$ del cub d'acord a les expressions que es poden trobar en la taula 3.22.

3.7.2 Model analític

Igual que en el cas de la malla, en el cub el cost de la comunicació *horitzontal* quan s'aplica el mapeig HCP coincideix amb el cost de la comunicació *horitzontal* en malles $configurables$. En aquest cas, si es compleix $r \geq d/3$, llavors cada fila de l'algorisme $2D$ ocupa vàries files del cub en una determinada cara del cub, exactament de la mateixa manera que quan aplicàvem l'algorisme de mapeig HCP per mapejar l'algorisme $2D$ en la malla. Quan $r < d/3$, cada fila de l'algorisme $2D$ es mapeja en vàries cares del cub consecutives. De nou, s'utilitza el mapeig "ondulat" per files per tal d'aconseguir que els nodes veïns en una fila en l'algorisme $2D$ també siguin veïns en el cub, figura 3.23.

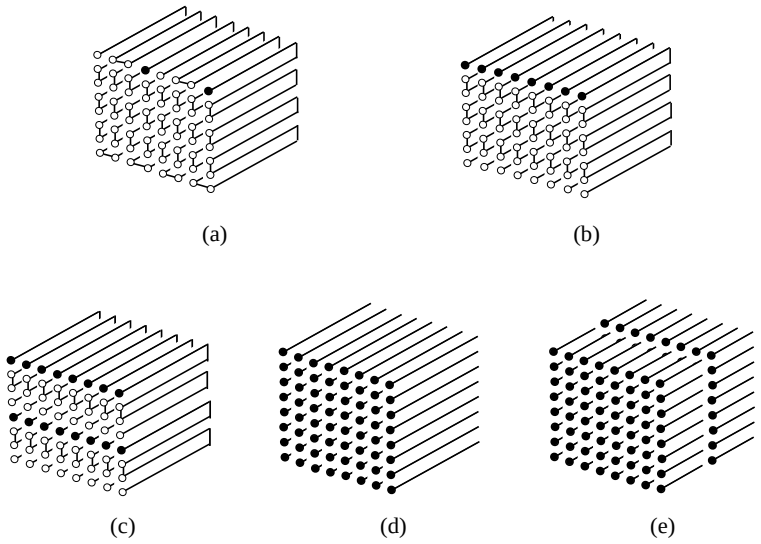


Figura 3.23 Mapeig HCP en el cub quan (a) $r < d/3$, (b) $r = d/3$, (c) $d/3 < r < 2d/3$, (d) $r = 2d/3$, i (e) $r > 2d/3$

Respecte a la comunicació *vertical* podem trobar cinc situacions diferents segons sigui la relació de files i columnes de l'algorisme $2D$ amb la dimensió del cub. En la figura 3.23 es poden veure un exemple de cadascuna de les cinc possibles situacions. En aquesta figura, els cercles en negreta fan referència al node situat més a l'esquerra de cada fila de l'algorisme $2D$, els cercles blancs fan referència a la resta de nodes que formen part d'una mateixa fila i que ajuden a comprendre com funciona el mapeig HCP en el cub i les línies uneixen els nodes que formen part de les files de l'algorisme $2D$.

Les cinc diferents situacions que podem trobar són les següents:

- Cas 1: $r < d/3$ (figura 3.23(a))

  Cada fila de l'algorisme $2D$ es mapeja en vàries cares del cub consecutives. La comunicació *vertical* es resol realitzant l'algorisme $ICV_{d/3-r,d/3}$ en la dimensió 2 del cub.

- Cas 2: $r = d/3$ (figura 3.23(b))

Cada fila de l'algorisme $2D$ es mapeja en una única cara del cub. La comunicació *vertical* es realitza aplicant l'operació $CVC_{d/3}$ en la dimensió 2 del cub.

- Cas 3: $d/3 < r < 2d/3$ (figura 3.23(c))

  Cada fila de l'algorisme $2D$ es mapeja en vàries files del cub, però no ocupa una cara completa del cub. La comunicació *vertical* es realitza aplicant l'algorisme $IVC_{2d/3-r,d/3}$ en la dimensió 1 del cub i l'algorisme $CVC_{d/3}$ en la dimensió 2 del cub.

- Cas 4: $r = 2d/3$ (figura 3.23(d))

  Cada fila de l'algorisme $2D$ es mapeja en una única línia del cub. La comunicació *vertical* es realitza aplicant l'algorisme $CVC_{d/3}$ en la dimensió 1 del cub i l'algorisme $CVC_{d/3}$ en la dimensió 2 del cub.

- Cas 5: $r > 2d/3$ (figura 3.23(e))

  Vàries línies de l'algorisme $2D$ es mapegen en una única línia del cub. La comunicació *vertical* es realitza aplicant l'algorisme $IVC_{d-r,d/3}$ en la dimensió 0, l'algorisme $CVC_{d/3}$ en la dimensió 1 del cub i l'algorisme $CVC_{d/3}$ en la dimensió 2 del cub.

Finalment, el cost de la comunicació *vertical* pot resumir-se en les següents expressions:

$$
\begin{array}{lll}
(m-1) & t_{ICV_{d/3-r,d/3}} & \mathrm{r} < \mathrm{d}/3 \\
(m-1) & t_{CVC_{d/3}} & \mathrm{r} = \mathrm{d}/3 \\
(m-1) & t_{ICV_{2d/3-r,d/3}} + t_{CVC_{d/3}} & \mathrm{d}/3 < \mathrm{r} < 2\mathrm{d}/3 \\
(m-1) & t_{CVC_{d/3}} + t_{CVC_{d/3}} & \mathrm{r} = 2\mathrm{d}/3 \\
(m-1) & t_{ICV_{d-r,d/3}} + t_{CVC_{d/3}} + t_{CVC_{d/3}} & \mathrm{r} > 2\mathrm{d}/3
\end{array}
$$

on les expressions analítiques de $t_{IVC_{k,l}}$ i t_{CVC_k} poden trobar-se en la secció 1.5.2. El temps total d'execució de l'algorisme $2D$ un cop mapejat en el cub és igual a:

$$t_{3D} = t_R + t_H + t_V$$

Les expressions t_R i t_H coincideixen amb les calculades pel cas de la malla $configurable$ en la secció 2.4.2.

3.7.3 Anàlisi del rendiment

En aquesta secció es realitza un estudi de l'eficiència de l'algorisme $2D$ un cop mapejat en una malla regular de tres dimensions. Aquesta anàlisi es realitzarà a partir dels models analítics desenvolupats en la secció anterior. Les gràfiques que es presenten poden dividir-se en tres grups en els que s'avaluen tres aspectes diferents:

- La influència d'alguns paràmetres en el rendiment de l'algorisme.
- L'escalabilitat dels algorismes proposats.
- L'eficiència del mapeig utilitzat, el mapeig HCP.

Influència dels paràmetres T_e, m i T_s

Les gràfiques que es mostren a continuació es corresponen a l'anàlisi de l'algorisme quan es mapeja en un cub regular, suposant un sistema amb un nombre de nodes igual a 2^{12} ($d = 12$). En les gràfiques es mostra la relació t_{1D}/t_{2D} per tots els valors possibles del paràmetre r ($r \in [0, d]$), sent 2^r el nombre de files de l'algorisme $2D$, i variant els valors dels paràmetres T_e, m i T_s respectivament. El paràmetre T_c és fix i igual a 1.

En aquesta secció, on es realitza una anàlisi del cub en un entorn fix, s'han tingut en compte tan sols dos sistemes. El primer correspon a un sistema amb uns valors de T_c, T_e i T_s iguals a 1, 27, 89 i 1974 que coincideixen amb la relació de paràmetres d'un $Cray\ T3D$ que té una topologia toroidal de tres dimensions, i el segon correspon a un sistema hipotètic amb uns valors de T_c, T_e i T_s iguals a 1, 10 i 1000 amb la idea de considerar un sistema amb un rang de valors més gran.

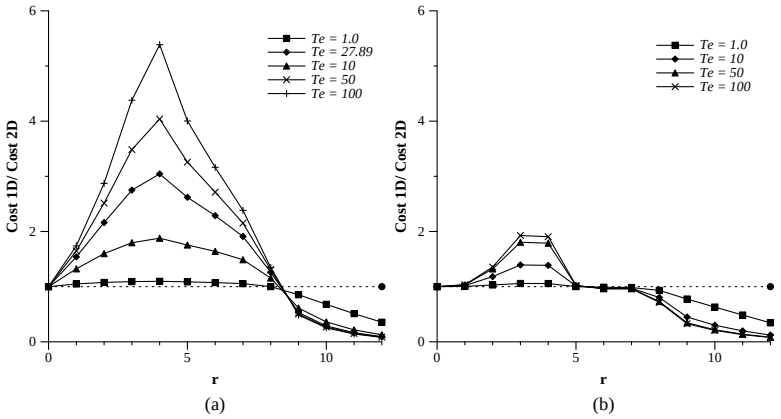


Figura 3.24 Impacte del paràmetre T_e. En la gràfica (a) s'ha considerat la relació de paràmetres del *Cray T3D* igual a $T_s = 70 \times T_e$. En la gràfica (b) s'ha considerat la relació de paràmetres $T_s = 1000 \times T_e$.

- En les gràfiques de la figura 3.24 s'observa que el paràmetre T_e (temps de transmissió d'un enter en coma flotant) influeix decisivament en l'eficiència de l'algorisme. Hem assignat valors al paràmetre T_e dintre d'un rang bastant gran. Aquests valors indiquen la relació de magnituds entre el temps de realitzar una operació aritmètica en coma flotant i el temps d'enviar un número en coma flotant per la xarxa d'interconnexió d'un node a un altre.

  En la gràfica de la figura 3.24(a) s'avalua quin impacte té la variació del paràmetre T_e en el rendiment del sistema. S'ha considerat un sistema amb una relació de paràmetres $T_s = 70 \times T_e$ (que coincideixen amb la relació de paràmetres del *Cray T3D*). Es pot observar, en la corba amb $T_e = 27,89$, que podem reduir el cost considerablement, més d'un 100%.

  En la gràfica de la figura 3.10(b) s'avalua la variació del paràmetre T_e en un sistema amb una relació de paràmetres $T_s = 1000 \times T_e$, considerant un rang de valors més gran.

  En les dues gràfiques s'ha considerat un tamany del problema igual a 2^{13}, el tamany del problema mínim que es pot resoldre en un sistema amb 2^{12} nodes.

- En les gràfiques de la figura 3.25 s'observa que el benefici que podem obtenir és bastant sensible als diferents valors d'm (tamany de la matriu). En aquest cas,

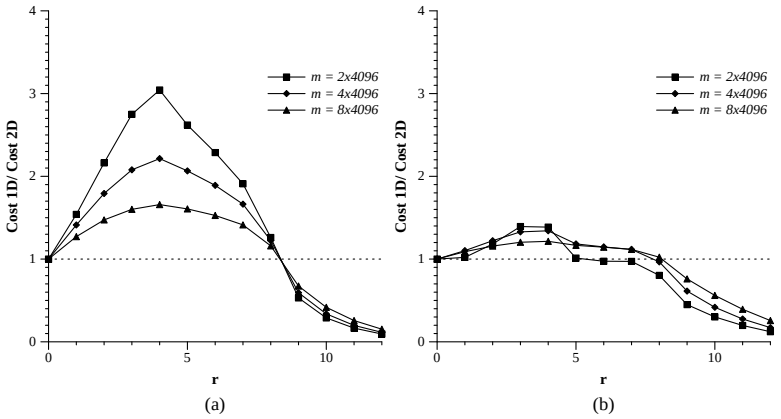


Figura 3.25 Impacte del paràmetre m. En la gràfica (a) s'han considerat els paràmetres del $Cray\ T3D$, $T_c = 1$, $T_s = 1974$ i $T_e = 27,89$. En la gràfica (b) s'han considerat els paràmetres $T_c = 1$, $T_s = 10000$ i $T_e = 10$.

però, el benefici decreix quan el tamany d'm s'incrementa, al contrari que en el cas del paràmetre T_e.

La gràfica de la figura 3.25(a) corresponen al cas en què els valors dels paràmetres T_e, T_s i T_c és igual a $27,85$, 1974 i 1 respectivament (*Cray T3D*. En aquest cas el rendiment varia d'un 300% fins a un 50%.

Per últim, en la gràfica de la figura 3.11(b) tenint un rang de valors més gran s'han considerat valors de T_c, T_e i T_s igual a 1, 10 i 10000 respectivament. En aquest cas el rendiment varia d'un 40% fins a poc més d'un 20%.

- En les gràfiques de la figura 3.26 es pot observar que són necessaris valors grans de T_s (temps d'iniciar una operació de comunicació, *startup*) per a què l'impacte en el rendiment sigui notable.

  En les dues gràfiques 3.26(a) ($Cray\ T3D$) i 3.26(b) (amb $T_e = 10$) s'observa un comportament equivalent. Per una part hi ha una reducció del benefici quan s'incrementa T_s i a més a més l'increment de T_s modifica la configuració òptima de la malla, la qual tendeix a reduir el nombre de files.

  És fàcil veure que en el cas de malles $3D$ fixes són necessaris valors de T_s molt més gran que en el cas de malles $2D$ fixes per veure l'impacte negatiu en el rendiment del sistema.

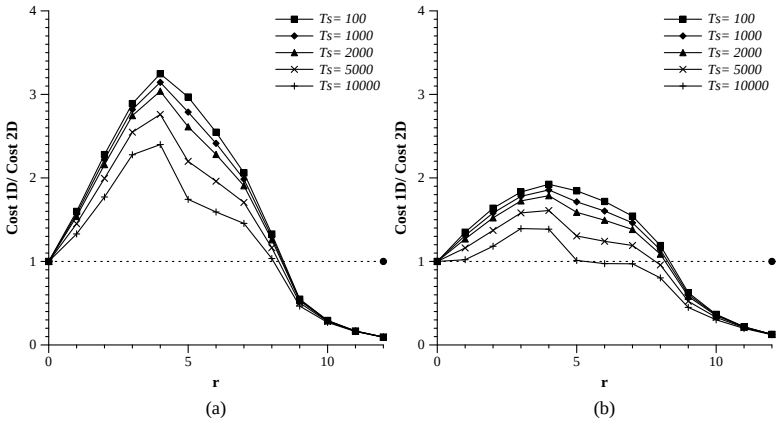


Figura 3.26 Impacte del paràmetre T_s. En la gràfica (a) s'han considerat els paràmetres del $Cray\ T3D$, $T_c = 1$ i $T_e = 27,89$. En la gràfica (b) s'han considerat els paràmetres $T_c = 1$ i $T_e = 10$.

És important assenyalar que en el cas de malles $3D$ fixes el rendiment del sistema és bastant similar al cas de malles $configurables$. En aquest cas, el rendiment de l'algorisme $2D$ quan es mapeja en malles $3D$ fixes és millor que el rendiment que s'obté quan es mapeja en malles $2D$ fixes.

Aquest comportament és degut a què, com ja s'ha observat pel cas de malles configurables, l'organització òptima dels nodes era generalment en malles rectangulars i les malles rectangulars es mapegen més eficientment en malles/torus $3D$ que en malles $2D$. En molts casos s'aconsegueix un mapeig a distància 1 de la malla òptima configurable en la malla $3D$, és a dir totes les comunicacions, *vertical* i *horitzontal*, es realitzen exactament con en la malla $configurable$.

Anàlisi de l'escalabilitat de l'algorisme $2D$

Les gràfiques de la figura 3.27 mostren l'escalabilitat de l'algorisme $2D$ quan es mapeja en el cub. Aquestes gràfiques mostren la relació t_{1D}/t_{3D} per uns valors determinats dels paràmetres T_e i T_s i variant els paràmetres m i d. El valor del paràmetre m s'escala respecte al paràmetre d. En aquest cas, donat un valor de d, es considera el valor òptim d'r per determinar el temps d'execució en cada cas, t_{3D}. Les gràfiques ens mostren quin és l'increment de benefici que podem obtenir amb l'algorisme $2D$ quan s'incrementa el

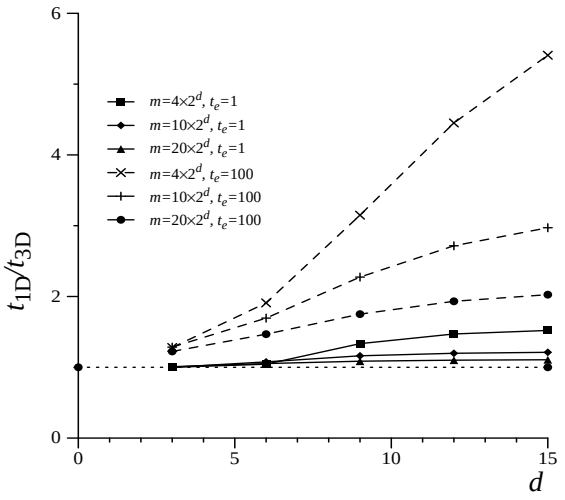


Figura 3.27 Anàlisi de l'escalabilitat de l'algorisme $2D$ quan es mapeja en el cub.

nombre de nodes del sistema. Aquest ingrement és particularment gran per valors d'm petits i per valors de T_e grans. Per últim podem observar que l'increment de benefici és bastant més gran pel cas del cub que pel de la malla (que es mostra a la gràfica 3.20).

Avaluació de l'eficiència del mapeig HCP

El mapeig HCP s'ha escollit per tal de conservar el cost de la comunicació *horitzontal* exactament igual que en el cas de tenir malles $configurables$, concretament es correspon al terme t_H de l'expressió t_{2D} (expressió 3.4) de la secció 3.5.1. Per una altra banda, aquest mapeig probablement produirà un increment de la comunicació *vertical* respecte al cas de les malles $configurables$, que correspon al terme t_V de l'expressió t_{2D}.

En aquest apartat es realitza una anàlisi equivalent a la que s'ha realitzat per malles $2D$ $fixes$. En la gràfica de la figura 3.28 es mostra la relació t_{2D}/t_{3D}. Els valor de t_{2D} i t_{3D} s'han obtingut considerant el valor òptim del paràmetre r. Com es pot veure en les gràfiques que s'han obtingut, de nou amb el mapeig HCP es pot obtenir un benefici

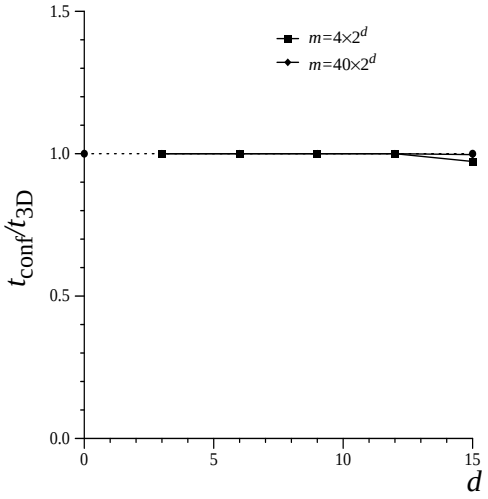


Figura 3.28 Comparació del cost de l'algorisme $2D$ quan es mapeja en un cub (aplicant el mapeig HCP) amb el cost de l'algorisme $2D$ quan s'executa en un entorn $configurable$.

molt proper a l'òptim. Amb aquesta anàlisi queda clar que en molts casos el mapeig HCP en malles/torus $3D$ aconsegueix un cost igual al cost de les malles configurables.

3.8 ANÀLISI DE L'ALGORISME $2D$ EN TOPOLOGIES TOROIDALS REGULARS DE DUES I TRES DIMENSIONS

El mapeig de l'algorisme $2D$ en un torus regular de dues i de tres dimensions és exactament igual que en el cas de la malla. L'anàlisi de l'algorisme és totalment equivalent a l'anàlisi desenvolupada pel cas de les malles; és per aquesta raó que tan sols s'especificaran aquells punts en el que difereixen totes dues anàlisis.

L'única diferència entre una malla regular multidimensional i un torus regular multidimensional consisteix en què en el cas d'un torus existeixen enllaços que connecten directament els nodes extrems, enllaços que no existeixen en una malla. Aquests en-

llaços no afecten a la comunicació *horitzontal* ja que amb l'algorisme de comunicació Eberlein que utilitza l'algorisme $2D$ aplicat en la malla ja s'aconseguia un cost de comunicació mínim totalment equivalent al cost de comunicació en el cas que s'executés en un anell (com seria el cas del torus). Respecte a la comunicació *vertical* veiem que en aquest cas podem modificar el patrò de comunicació convenientment per tal de treure profit d'aquests enllaços addicionals.

A continuació es descriu l'algorisme de comunicació *vertical* que es proposa pel cas d'una topologia toroidal. Les operacions de comunicació es reorganitzen de tal manera que la comunicació *vertical* s'aconsegueix finalitzar en $2r-2$ fases, una menys que en el cas de la malla. Per aconseguir aquesta reducció es segueix els següent esquema de comunicació:

Expressarem com $< n_{r-1}, ..., n_1, n_0 >$ la representació binària de l'enter n ($n \in [0, 2^r - 1]$).

Les $2r-2$ fases es numeren de 0 fins a $2r-3$. En una etapa i, amb $i \in [0, r-3]$, tan sols els nodes amb etiqueta n que compleixen $n_{i-1} = ... = n_0 = 0$ estan actius. En particular, un node amb etiqueta $< n_{r-1}, ..., n_{i+1}, 1, 0, ..., 0 >$ envia el seu vector amb els productes interns parcials al node que té per etiqueta $< n_{r-1}, ..., n_{i+1}, 0, 0, ..., 0 >$, el qual, després de rebre el missatge, acumula el vector rebut al vector que ja tenia.

En la fase $r-2$ tots els nodes involucrats en aquesta etapa envien i reben dades. Els nodes que estan actius en aquesta etapa són els nodes amb les etiquetes 0, $2^r/4$, $2^r/2$ i $3 \times 2^r/4$. Els nodes 0 i $2^r/4$ (i $2^r/2$ i $3 \times 2^r/4$) s'intercanvien els corresponents subproductes i els acumulen.

En la fase $r-1$, de nou són els mateixos nodes de l'etapa anterior els que s'intercanvien els subproductes. Els nodes amb etiquetes 0 i $3 \times 2^r/4$ (i els nodes $2^r/4$, $2^r/2$) intercanvien entre ells els seus vectors parcials i els acumulen. En aquest moment, tots quatre nodes tenen una còpia del vector resultant T.

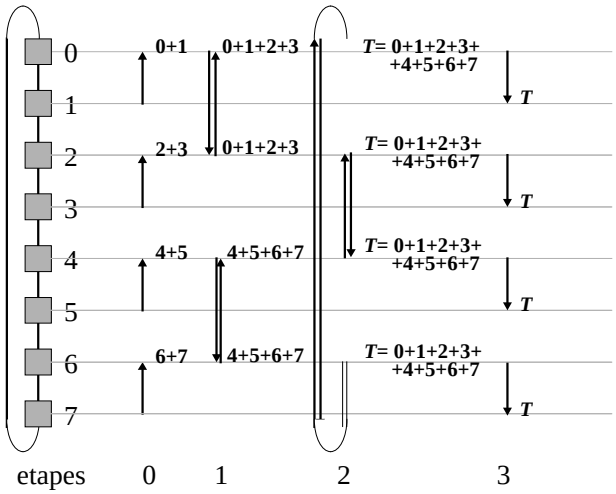


Figura 3.29 *Comunicacio vertical* en un torus

Finalment, en les últimes fases, $2r-2-i$ amb $i \in [0, r-3]$, els nodes tals que la seva etiqueta té el patró $< n_{r-1}, ..., n_{i+1}, 0, 0, ..., 0 >$ envien als nodes amb etiqueta $< n_{r-1}, ..., n_{i+1}, 1, 0, ..., 0 >$ una còpia del vector resultant T.

En la figura 3.29 es mostra un exemple de l'esquema que s'acaba de descriure, suposant una línia de 2^3 nodes ($r = 3$) amb un model de comunicació *wormhole*. En aquesta figura els números identifiquen els nodes dintre la columna i a més a més identifiquen els vectors que s'han d'acumular. Les fletxes indiquen la comunicació entre els nodes. T fa referència al vector resultant. Per analogia a les malles, aquest esquema l'anomenarem Contiguous Vertical Communication for Toruses, abreujat $CVCT$.

L'impacte que pot tenir l'aplicació d'aquest algorisme $CVCT$ en lloc de l'algorisme CVC (per malles) en el cost total de comunicació és mínim. A efectes pràctics en cada etapa de comunicació *vertical* d'un grup de rotacions independents ens estalviem un únic missatge. És per aquest motiu que els resultats de l'avaluació de l'algorisme $2D$ quan s'executa en un torus $2D$ i $3D$ regular són gairebé e iguals que els que s'han obtingut de l'avaluació de l'algorisme $2D$ quan s'executa en la malla, i per aquest motiu no els presentem en aquesta secció.

3.9 RESUM DE CONTRIBUCIONS

En la literatura s'han proposat algorismes per resoldre el problema de valors i vectors propis en una topologia en malla multidimensional. Per una part podem trobar algorismes per malles els quals apliquen el mètode de Jacobi *bilateral*, que com ja s'ha vist en el primer capítol no és el mètode més òptim en un entorn multicomputador degut als requeriments de comunicació del mateix mètode. Per una altra banda també s'han proposat algorismes específics per sistòlics, que a part d'aplicar el mètode de resolució de Jacobi *bilateral*, són algorismes que consideren les comunicacions gairebé despreciables, aspecte que en un entorn paral.lel considerat en aquest treball (multicomputadors, DSM i xarxes d'estacions de treball) no és aplicable.

Els algorismes que hem trobat en la literatura que apliquen el mètode de Jacobi *unilateral* (mètode més òptim en el cas d'entorns multicomputadors) estan pensats per una topologia *unidimensional*, en línia o en anell s'identifiquen en el capítol com algorismes $1D$.

En aquest capìtol s'ha proposat un algorisme pensat per una topologia bidimensional, anomenat algorisme $2D$, el qual aplica el mètode de Jacobi *unilateral*. Aquest algorisme s'ha avaluat en tres entorns diferents:

- Un entorn sense cap tipus de restricció en la disposició dels nodes del sistema, anomenat entorn $configurable$.
- En un entorn en el que els nodes estan topològicament disposats en malla bidimensional regular, amb el mateix nombre de nodes en les dues dimensions igual a una potència de 2.
- En un entorn en el que els nodes estan topològicament disposats en malla tridimensional regular, amb el mateix nombre de nodes en les tres dimensions igual una a potència de 2.

S'ha vist que depenent del valor de T_c, T_e i T_s el cost de comunicació es pot reduir considerablement amb l'algorisme $2D$ respecte al cost de comunicació de l'algorisme $1D$, sigui quin sigui l'entorn escollit.

Relacionatdes amb aquest treball s'han realitzat les següents publicacions:

- "A Jacobi-based Algorithm for Computing Symmetric Eigenvalues and Eigenvectors in a Two-dimensional Mesh" D. Royo, M. Valero-García i A. González; *Euromicro Workshop on Parallel and Distributed Processing* , Gener del 1998, Madrid.
- "Implementing the One-Sided Jacobi Method on a 2D/3D Mesh Multicomputer"; D. Royo, M. Valero-García i A. González; en procés de revisió en una revista internacional.

4

MAPEIG DE L'ALGORISME BR EN MALLES I TORUS DE DUES DIMENSIONS

Resum

En aquest capítol s'analitzen els algorismes resultants de mapejar l'algorisme BR en una malla i un torus de dues dimensions. Per realitzar el mapeig s'ha aplicat el mètode CALMANT, *desenvolupat en [7]. Abans, però, s'ha realitzat una extensió del mètode per cobrir totes les necessitats de comunicació de l'algorisme BR, que en el mètode original no es contemplaven.*

El contingut del capítol s'organitza de la manera següent: primer es descriu la metodologia CALMANT, a continuació s'aplica la metodologia a l'algorisme BR i es proposen els algorismes d'encaminament de missatges que són òptims (minimitzen el cost de la comunicació) i per últim es realitza una avaluació de l'algorisme resultant un cop mapejat en els diferents entorns considerats -malles i torus bidimensionals quadrats (amb el mateix nombre de nodes igual a una potència de 2 *en cadascuna de les dues*

dimensions) amb una arquitectura one − port i all − port-. Es compara l'eficiència de l'algorisme BR un cop mapejat en la malla i el torus amb l'algorisme $2D$ *mapejat en malles i torus regulars de dues dimensions que s'ha descrit en el capítol* 3. *Es veurà que per sistemes amb un nombre de nodes de fins a* 256 *l'algorisme BR té un cost de comunicació més petit que l'algorisme* $2D$ *quan s'executa en malles* $2D$ *i* $3D$ *fixes. Per segons quins valors dels paràmetres de comunicació aquesta reducció pot ser de fins a un* 40% *respecte al cost de la comunicació de l'algorisme* $2D$.

4.1 INTRODUCCIÓ

En el grup de treball s'ha desenvolupat una metodologia que hem anomenat *CALMANT* [7], *CC-cube Algorithms on Meshes ANd Tori*, que ens permet mapejar de forma sistemàtica i eficient un algorisme de tipus CC-cub en un multicomputador amb una topologia en hipercub, malla o torus multidimensional. Aquesta metodologia s'ha aplicat a algorismes paral.lels com la FFT [9] i Complete Exchange [10].

A continuació es descriu breument la metodologia $CALMANT$. El mètode s'aplica a l'algorisme BR en un entorn de malles i torus regulars i bidimensionals. En aquest treball, per a algunes operacions de comunicació que s'han de realitzar es proposen nous algorismes d'encaminament de missatges que són més senzills que els algorismes d'encaminament originalment proposats en la descripció del mètode [7][25]. Per últim s'avalua l'algorisme resultant, i com es veurà aquest aconsegueix en sistemes petits reduir la comunicació considerablement respecte al cost de comunicació generat per l'algorisme $2D$ descrit en el capítol 3.

4.2 METODOLOGIA CALMANT

Una de les contribucions d'aquest treball és l'ampliació de la metodologia CALMANT per poder aplicar-la a l'algorisme BR (per la resolució del problema dels valors i vectors propis). A continuació es fa una breu descripció d'aquesta metodologia.

4.2.1 Visió general

La metodologia $CALMANT$ ens permet executar de forma eficient algorismes pensats inicialment per multicomputadors amb una topologia en hipercub (algorismes CC-cub) en multicomputadors amb una topologia en malla o torus.

La metodologia es composa de dues fases (figura 4.1). En la primera fase s'aplica la segmentació de les comunicacions a l'algorisme en qüestió (de tipus CC-cub), amb el

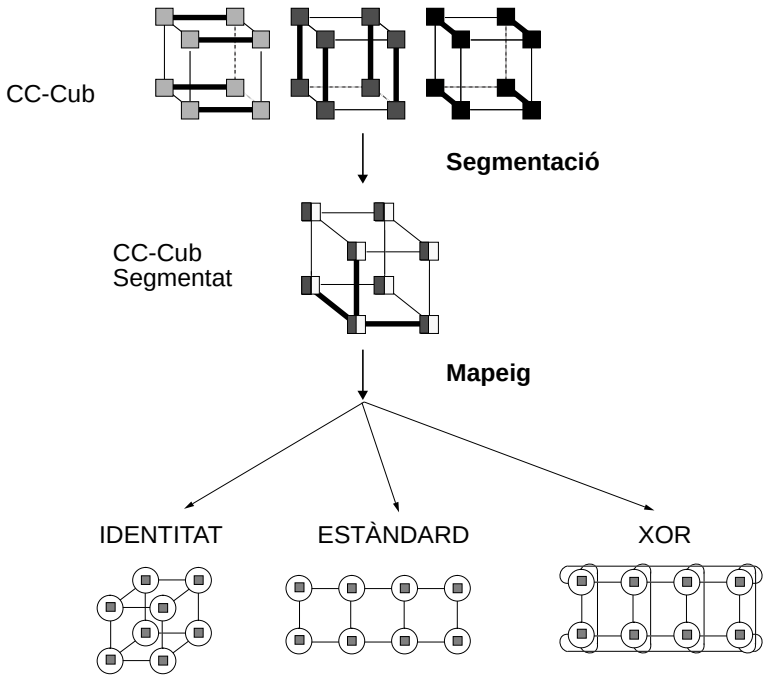


Figura 4.1 Visió general del mètode

que s'aconsegueix un increment potencial del paral.lelisme de les comunicacions. Un cop segmentat, l'algorisme s'ha de mapejar en l'arquitectura on s'executarà, procés que correspon a la segona fase del mètode.

Finalment, un cop decidit el mapeig que s'aplica en cada cas, s'ha de determinar quin és l'encaminament dels missatges per tal d'aconseguir finalitzar cadascuna de les operacions de comunicació amb el nombre mínim de cicles.

El concepte de CC-cub i la tècnica de la segmentació de les comunicacions s'han descrit en el capítol 2. A continuació es descriuen els algorismes de mapeig que s'apliquen depenent de la topologia (hipercub, malla o torus) i quins són els algorismes d'encaminament de missatges que es proposen per realitzar cadascuna de les operacions de comunicació.

4.2.2 Mapeig d'un CC-cub en un hipercub, en una malla i un torus

En la segona fase de la metodologia CALMANT es mapegen els diferents processos de l'algorisme CC-cub segmentat en els nodes del multicomputador.

Un algorisme paral.lel composat per un conjunt de processos els quals s'intercanvien informació entre ells, sempre es pot representar com un graf en el que els processos podrien ser els nodes i els intercanvis entre el processos estarien representats per les arestes. En el cas d'un algorisme CC-cub, les arestes que determinen l'intercanvi entre els diferents processos defineixen una topologia en hipercub.

El mapeig d'algorismes CC-cub en multicomputadors amb una topologia en hipercub és immediat, ja que el graf de l'algorisme coincideix amb el graf de la topologia del multicomputador. En aquest cas direm que s'aplica el mapeig identitat.

En el cas de les malles i dels torus el mapeig no és immediat. Pel cas de les malles, s'aplica el mapeig *standard*, proposat a [45]. En el cas dels torus, s'aplica el mapeig Xor, proposat a [25]. No es desenvolupa una descripció formal i precisa d'aquests dos mapejos ja que no són aportacions del treball. A continuació tan sols es descriuen les característiques de cadascun d'ells rellevants per al seguiment del treball que es desenvolupa en aquest capítol.

Mapeig *standard*

Matic presenta a [45] un estudi del mapeig *standard* d'algorismes CC-cub en malles i torus bidimensionals.

A l'hora de realitzar el mapeig d'un algorisme en una topologia determinada no sempre és possible aconseguir que els processos que són veïns en el graf original es mapegin en nodes veïns de la nova topologia. El mapeig *standard* aconsegueix un mapeig amb una dilatació regular en la malla. És a dir, tot els processos que inicialment estan connectats en una determinada dimensió en el CC-cub es mapegen a la mateixa distància en la

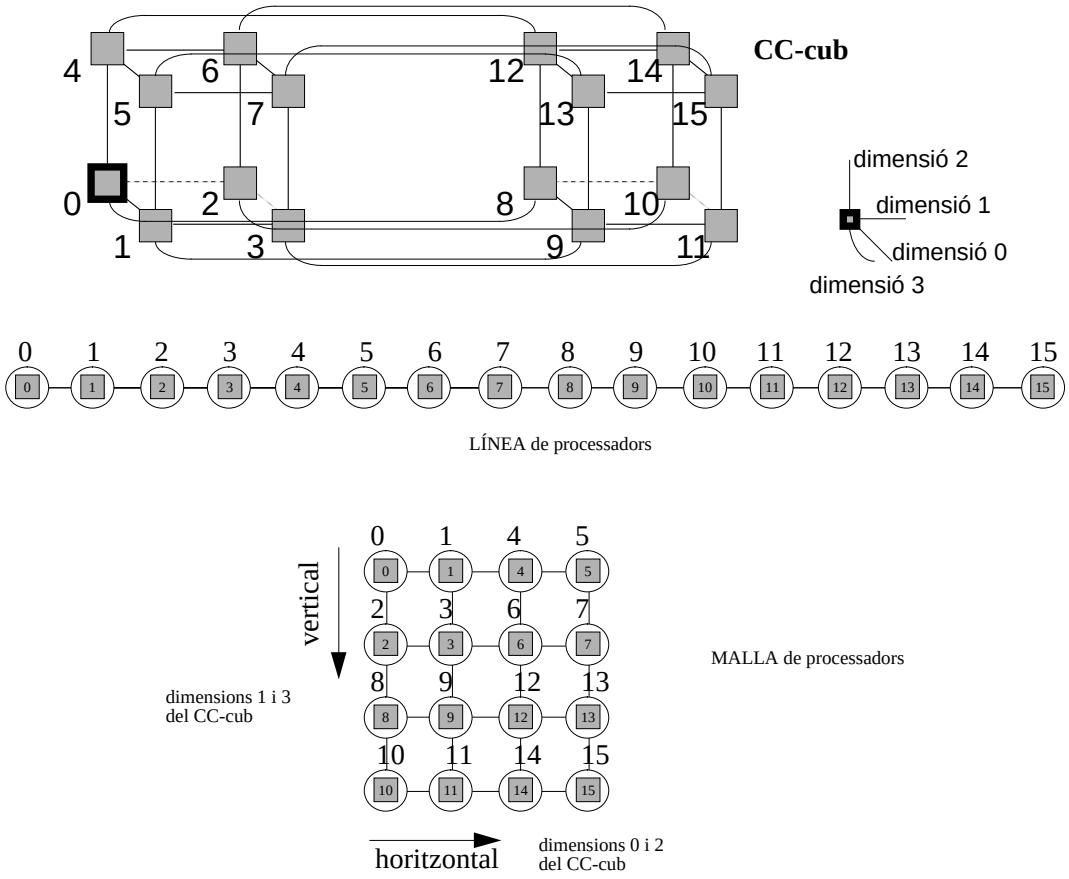


Figura 4.2 Mapeig *standard*

malla (distància ≥ 1). D'aquesta manera s'aconsegueix una comunicació equilibrada, és a dir, el cost de comunicació és el mateix per qualsevol parella de processos que eren veïns en una determinada dimensió de l'hipercub.

En la figura 4.2 es mostra un exemple de com un algorisme CC-cub que consta de 16 processos, de dimensió 4, que s'intercanvien informació segons les arestes que s'han dibuixat (les quals defineixen una topologia de comunicació en hipercub), es mapeja en una línia de 16 nodes aplicant el mapeig *standard*. Un cop realitzat el mapeig, tots els processos que s'intercanvien informació per la dimensió 0 estan mapejats a distància 1 en la línia, tots els processos que s'intercanvien informació per la dimensió 1 estan mapejats a distància 2, tots els processos que s'intercanviaven informació per

la dimensió 2 estan tots a distància 4 i finalment tots els processos que s'intercanvien informació per la dimensió 3 estan a distància 8.

En general, amb aquest mapeig s'aconsegueix que la distància mínima entre dos processos que intercanvien informació en l'algorisme CC-cub un cop mapejat en la malla sigui igual a $2^0 = 1$ i la distància màxima sigui igual a $2^{d/n-1}$, sent 2^d el número de nodes del sistema i n el número de dimensions de la malla (per la línia $n = 1$).

En aquest treball tan sols es consideren malles multidimensionals regulars en les que el nombre de nodes és el mateix en totes les dimensions i igual a una potència de 2. En el cas de malles de més d'una dimensió, les dimensions de l'hipercub en el CC-cub es mapegen de forma alternada en cadascuna de les dimensions de la malla. En general, per una malla amb 2^d nodes i n dimensions, on cada dimensió té un nombre de nodes 2^{d-n}, la dimensió i de l'hipercub es mapeja en la dimensió j de la malla si $(i+1) \bmod 2^n = j$, amb $0 \leq i < d$ i $0 \leq j < n$.

En la figura 4.2(b) es mostra un exemple per una malla de 2^4 nodes i dues dimensions, les dimensions de l'hipercub parells, 0 i 2, es mapegen en la dimensió *horitzontal* de la malla i les dimensions de l'hipercub imparells, 1 i 3, es mapegen en la dimensió emphvertical de la malla.

Mapeig Xor

Quan el multicomputador té una topologia en torus, s'aplica el mapeig Xor. Aquest mapeig proposat a [25] és més eficient que el mapeig *standard* (pel cas del torus) ja que aconsegueix distàncies mitges més petites tenint en compte els enllaços *wraparound* del torus.

De nou, amb aquest mapeig s'aconsegueix una dilatació regular en el torus. En la figura 4.3 es mostra com es realitzaria el mapeig d'un CC-cub de dimensió 4 en un anell amb 16 nodes quan s'aplica el mapeig xor. En aquest cas, el processos que s'intercanvien informació per la dimensió 0 estan mapejats a distància 1 en l'anell, tots els processos que s'intercanvien informació per la dimensió 1 estan mapejats a distància 2, tots els

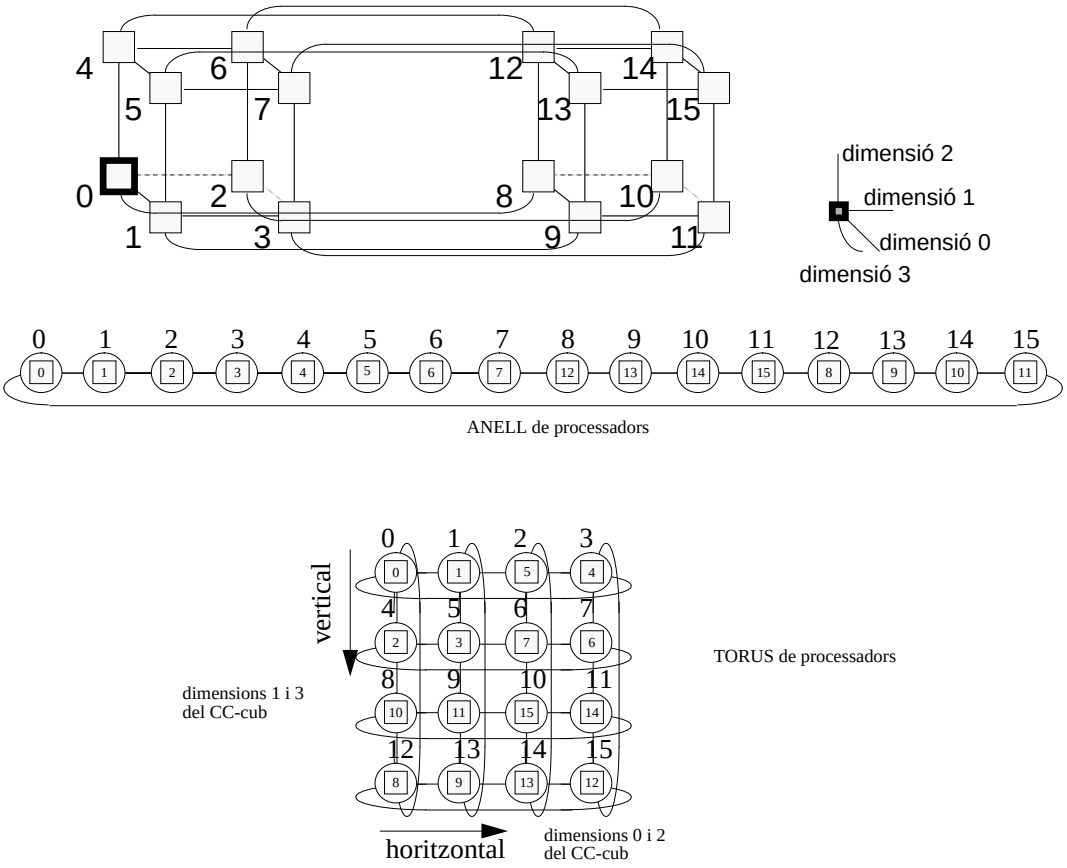


Figura 4.3 Mapeig Xor

processos que s'intercanviaven informació per la dimensió 2 estan tots a distància 4 i finalment tots els processos que intercanvien informació per la dimensió 3 estan tots a distància 4 de nou ja que utilitzen l'enllaç *wraparound*.

Amb aquest mapeig s'aconsegueix que la distància mínima entre dos processos que intercanvien informació en l'algorisme CC-cub un cop mapejat en la malla sigui igual a $2^0 = 1$ i la distància màxima sigui igual a $2^{d/n-2}$, sent 2^d el número de nodes del sistema i n el número de dimensions del torus, en el cas de l'anell $n = 1$.

En aquest treball tan sols es consideren torus multidimensionals en els que el nombre de nodes és el mateix en totes les dimensions i igual a una potència de 2. En el cas de torus de més d'una dimensió, les dimensions de l'hipercub es mapegen de forma

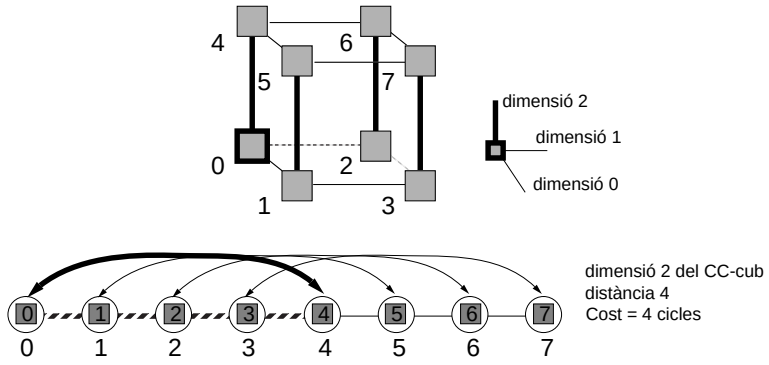


Figura 4.4 Gestió de missatges

alternada en cadascuna de les dimensions de la malla. En general, per una malla amb 2^d nodes i n dimensions, on cada dimensió té un nombre de nodes 2^{d-n}, la dimensió i de l'hipercub es mapeja en la dimensió j de la malla si $(i+1) \bmod 2^n = 0$, amb $0 \leq d$ i $0 \leq j < n$.

En la figura 4.3 es mostra un exemple per una malla de 2^4 nodes i dues dimensions; les dimensions de l'hipercub parells, 0 i 2, es mapegen en la dimensió *horitzontal* i les dimensions de l'hipercub imparells, 1 i 3, en la dimensió *vertical* tal i com es realitza en una malla.

4.2.3 Encaminament de missatges

En l'algorisme CC-cub original, amb una topologia de comunicació en hipercub, tots els processos que s'intercanvien informació es troben a distància 1 i per tant tots els processos poden realitzar en paral.lel una operació de comunicació amb un altre procés veí en una mateixa dimensió de l'hipercub.

Acabem de veure que un cop mapejat un algorisme CC-cub en una malla o un torus, les distàncies entre processos que s'han d'intercanviar informació són més grans que 1 per la majoria dels casos. Per exemple, en la figura 4.4 es pot veure com queden mapejats els processos d'un CC-cub de dimensió 3 en una línia de 8 nodes. Els processos que es comunicaven per la dimensió 2 en el CC-cub es troben a distància 4 en la línia.

En aquest exemple, si tots els processos han d'intercanviar informació amb el procés veí segons la dimensió 2, totes les operacions de comunicació per cada parella de processos s'han de realitzar seqüencialment. Això és degut a què els enllaços ocupats en una operació de comunicació entre dos processos per la dimensió 2, no permet solapar cap altra operació de comunicació entre qualsevol altra parella de processos que siguin veïns en la mateixa dimensió.

Per exemple, suposem que s'estan intercanviant un missatge la parella de processos (0,4). Mentre es realitza la transmissió entre aquests dos processos, no es possible enviar cap altre missatge ja que els enllaços que estan ocupant els processos (en la figura amb un traç ratllat) són necessaris per qualsevol altra operació de comunicació que s'hagin de realitzar entre la resta de parelles de processos veïns en la dimensió 2 del CC-cub, (1,5), (2,6) i (3,7). El mateix passa sigui quina sigui la parella de processos que s'intercanviïn la informació. És necessari, doncs, realitzar les operacions de comunicació de forma seqüencial una darrera l'altra.

Es defineix *cicle*, com el cost d'enviar un missatge d'un procés a un altre qualsevol del CC-cub.

El cost per realitzar l'operació de comunicació de l'exemple de la figura 4.4 és de 4 cicles, sempre i quan considerem que el cost d'enviar un missatge és d'un cicle i els missatges que s'intercanvien dos processos veïns es solapen perquè s'estan enviant en sentit oposat i els enllaços són *full duplex*.

En general, el cost total degut a què tots els processos enviïn un missatge al seu veí en una mateixa dimensió és proporcional a la distància a la que s'han mapejat en la malla o el torus. Aquest fet fa que s'incrementi el cost de la comunicació.

Un cop mapejades les dimensions de l'hipercub en la malla, el cost degut a que tots els processos enviïn un missatge al seu veí en una dimensió qualsevol i $(0 \leq i < d)$ suposa un nombre de cicles igual a $2^{i/2}$ si i és parell i un nombre de cicles igual a $2^{(i-1)/2}$ cicles si i és imparell.

En els cas del torus, el cost degut a què tots els processos enviïn un missatge al seu veí en una dimensió qualsevol i ($0 \leq i < d-2$) suposa un nombre de cicles igual a $2^{i/2}$ si i és parell i un nombre de cicles igual a $2^{(i-1)/2}$ cicles si i és imparell. Per $i = d-1$ i $i = d-2$ enviar un missatge té un cost igual a $2^{(d-4)/2}$ cicles.

Tenint en compte aquestes expressions, a continuació es construiex un model del cost de la comunicació de l'algorisme BR un cop mapejat en la malla/torus sense aplicar la segmentació de les comunicacions. Aquest model s'utilitzarà més tard per veure quina és la reducció del cost de la comunicació que s'aconsegueix amb l'algorisme segmentat en front de l'algorisme sense segmentar.

El cost total de comunicació (nombre de cicles, que en aquest cas coincideix amb el nombre de missatges que s'intercanvien les diferents parelles de processos del CC-cub) de l'algorisme BR quan es mapeja en una malla regular de dues dimensions -$2^{d/2}$ files i $2^{d/2}$ columnes.

$$T_{ComMNs} = \sum_{k=0}^{d/2-1} 2^k [\sum_{i=0}^{d-2k-1} 2^i + \sum_{i=0}^{d-(2k+1)-1} 2^i] + 2^{d/2-1} + 2 \times \sum_{i=0}^{d/2-1} 2^i$$

On el primer terme correspon al cost de comunicació degut a les etapes d'intercanvi, el segon terme correspon al cost de comunicació de l'última transició i el tercer i últim terme fa referència al cost de comunicació de les etapes de divisió. L'expressió es pot simplificar com:

$$T_{ComMNs} = (3 \times 2^{d/2})(2^{d/2} - 1) + 2^{d/2-1} \tag{4.1}$$

En el cas de què l'algorisme es mapegi en una topologia toroidal el cost de la comunicació és la mateixa que en el cas de la malla excepte en les dimensions $d-1$ i $d-2$, que en el cas del torus es mapegen a una distància que és igual a la meitat de la distància de la malla (degut a que s'utilitzen els enllaços *wraparound*). El cost de la comunicació es pot expressar com:

$$T_{ComTNs} = T_{ComMNs} - 7 \times 2^{\frac{d}{2}-2}$$

L'últim terme substreu el cost afegit de considerar el mapeig en la malla. Finalment podem expressar el cost total de comunicació com:

$$T_{ComTNs} = [\frac{2^d + 2^{d-1}}{2^{d/2-1}} \times (2^{d/2} - 1) - 6 \times 2^{d/2-2}] \quad (4.2)$$

Podem observar en la figura 4.4, que mentre els processos (0,4) s'estan enviant un missatge, els processos 5, 6 i 7 estan lliures i per tant hi hauria la possibilitat que els processos (5,7), veïns en la dimensió 1, intercanviessin un missatge o bé ho fessin els processos (6,7) veïns en la dimensió 0. Per què això sigui possible, en cada operació de comunicació un procés ha de poder intercanviar dades amb diferents processos que siguin veïns en dimensions diferents de l'hipercub. Aquest solapament no es possible realitzar-lo en el CC-cub sense segmentar, ja que en cada operació de comunicació del CC-cub un procés tan sols ha d'intercanviar un únic missatge amb un altre procés en una determinada dimensió de l'hipercub.

Com ja hem vist en el capítol 2, la tècnica de la segmentació de comunicacions ens permet incrementar el paral.lelisme de comunicacions permetent que un procés en cada operació de comunicació tingui la possibilitat d'enviar més d'un missatge en paral.lel a través de diferents dimensions de l'hipercub. Per tant, aplicarem aquesta tècnica per poder solapar missatges i així aconseguir reduir el cost de la comunicació de l'algorisme CC-cub quan es mapeja en la malla o el torus.

4.3 APLICACIÓ DEL MÈTODE CALMANT A L'ALGORISME BR

En aquesta secció s'analitzen les operacions de comunicació que resulten després de mapejar l'algorisme BR segmentat en malles i torus de dues dimensions. El mapeig es

realitza aplicant el mètode *CALMANT* descrit a [7]. El mètode s'amplia per cobrir les necessitats de comunicació de l'algorisme BR un cop mapejat en la malla o el torus.

S'ha de fer una puntualització respecte a l'algorisme BR quan es mapeja en la malla/torus bidimensional. Com ja es va veure en el primer capítol, quan s'aplica l'algorisme BR no tots els escombrats segueixen el mateix patró d'utilització de les dimensions de l'hipercub. Això era així per tal d'aconseguir que l'ordre d'aplicació de les transformacions fos igual en cada escombrat. A continuació es mostra un exemple de quina seria la utilització de les dimensions en els d primers escombrats per $d = 3$.

Primer escombrat: 010201020101002
Segon escombrat: 121012101212110
Tercer escombrat: 202120212020001

En aquest capítol s'aplica l'algorisme BR una mica modificat, que l'hem anomenat BR simplificat. Al llarg d'aquest capítol quan ens referim a l'algorisme BR en realitat fem referència a l'algorisme BR simplificat. L'algorisme BR simplificat aplica sempre en tots els escombrats la mateixa sequència de dimensions, la qual coincideix amb la sequència de dimensions utilitzada en el primer escombrat en l'algorisme BR original. En l'algorisme BR simplificat, l'ordre d'aplicació de les transformacions per completar un escombrat es repeteix exactament cada d escombrats i no en cada escombrat com en l'algorisme BR original. Aquesta modificació no afecta per res el nombre d'etapes en completar un escombrat. Per veure, si aquesta desició afecta a la convergència de l'algorisme hem realitzat el mateix anàlisi de convergència que s'ha descrit en el capítol 2 (aplicat per analitzar la convergència de les ordenacions que allí es proposaven). Els resultats que s'han obtingut de l'anàlisi es mostra en la taula de la figura 4.5. Dels resultats obtinguts podem concloure que amb aquesta modificació l'algorisme segueix sent igual de vàlid que l'algorisme original, ja que respecte a la convergència té un comportament equivalent a l'algorisme original.

m	P	*BR*	BR simplificat
8	4	3.66	3.86
	2	3.36	3.36
16	8	4.73	4.63
	4	4.26	4.50
	2	4.00	4.00
32	16	5.06	5.10
	8	5.03	5.13
	4	5.00	5.10
	2	4.66	4.66
64	32	5.96	6.00
	16	6.00	6.00
	8	5.86	5.96
	4	5.73	5.73
	2	5.00	5.00

Figura 4.5 Anàlisi de la convergència del BR simplificat

El fet de segmentar l'algorisme fa que en una operació de comunicació un procés tingui la possibilitat d'enviar 2 o més missatges a altres processos veïns en vàries dimensions de l'hipercub. El mètode $CALMANT$ que es proposa en [11] considera operacions de comunicació en les que els processos envien a processos veïns en dimensions consecutives de l'hipercub. Per aquestes operacions de comunicació es proposen algorismes d'encaminament que són òptims - és a dir, permeten finalitzar les operacions de comunicació amb el nombre mínim de cicles possible per qualsevol combinació de topologia (malla/torus/hipercub) i arquitectura ($one-port$/ $all-port$) - o bé gairebè òptims.

Quan s'aplica el mètode $CALMANT$ a l'algorisme BR apareixen noves operacions de comunicació, en les que els processos intercanvien dades amb qualsevol altre procés del CC-cub i per tant no necessàriament són veïns en dimensions consecutives de l'hipercub. En un principi, els algorismes d'encaminament proposats en la metodologia original es poden aplicar a qualsevol operació de comunicació que resulten en el cas de l'algorisme BR. S'ha fet una anàlisi més acurat d'aquests algorismes i com a resultat de l'anàlisi s'ha proposat el següent:

- Pel subconjunt d'operacions de comunicació en les que els processos involucrats són veïns en la mateixa dimensió de la malla, s'apliquen els algorismes d'encaminament proposats en la metodologia obtenint resultats òptims en quant que -minimitzen el nombre de cicles per completar l'operació de comunicació-.
- Pel subconjunt d'operacions de comunicació en les que els processos involucrats no són veïns en la mateixa dimensió de la malla/torus, s'han proposat nous algorismes d'encaminament, els quals també són òptims en quant que -minimitzen el nombre de cicles per completar l'operació de comunicació- i a més a més són molt més simples que els algorismes d'encaminament ja proposats.

4.3.1 Identificació dels tipus de subproblemes

El mètode CALMANT s'aplica a l'algorisme BR segmentat amb un grau de segmentació $Q = 2$ perquè és el que permet obtenir millors resultats. La segmentació de les comunicacions tan sols es pot aplicar en les fases d'intercanvi. Tots els processos del CC-cub intercanvien les dades amb els processos veïns en les dimensions segons la seqüència D_e^{BR} $(1 \leq e < d)$, definida en el capítol 2 (sent d la dimensió de l'hipercub).

Recordem que a l'hora de mapejar els processos del CC-cub en la malla s'aplica el mapeig *standard* i en mapejar-los en un torus s'aplica el mapeig Xor, de manera que les dimensions parells de l'hipercub, $< 0, 2, 4.. >$, queden mapejades en la dimensió horitzontal de la malla/torus i les dimensions imparells de l'hipercub, $< 1, 3, 5... >$ queden mapejades en la dimensió vertical de la malla/torus.

Analitzem quina és la seqüència d'utilització de les dimensions de l'hipercub en una etapa d'intercanvi qualsevol. Per a això veiem el cas concret amb $e = 6$.

$D_{e=6}^{BR} =< 0102010301020104010201030102010501020103010201040102010301020 10 >$

Recordem com organitza l'algorisme les operacions de comunicació i de càlcul. Per simplificar l'explicació i les expressions aritmètiques que es presenten a continuació

suposarem que en una operació de càlcul de l'algorisme original sense segmentar s'ha de realitzar el càlcul d'un conjunt de dades que denotarem per N. En el cas de l'algorisme BR aquesta N correspondria al càlcul d'$m/2^{d+1} \times m/2^{d+1}$ rotacions independents, i en la primera etapa a més a més ha de realitzar $m/2^{d+1} \times (m/2^{d+1} - 1)$ rotacions independents addicionals (capítol 1).

Com s'acaba d'indicar, quan no s'aplica la segmentació de les comunicacions en la primera etapa un procés calcula un conjunt de dades, N, i a continuació les intercanvia amb el procés veí en la dimensió 0; en la segona etapa es realizen nous càlculs (de tamany igual a N) i l'intercanvi es realitza amb el veí en la dimensió 1. Els següents intercanvis es realitzen per la dimensió i segons l'ordre en què apareixen les dimensions en la seqüència D^{BR}.

Quan apliquem la segmentació de les comunicacions amb un grau de segmentació igual a $Q = 2$, les operacions de càlcul i comunicació es reorganitzen de la forma següent: en la primera etapa els processos calculen la meitat de les dades $N/2$ i les intercanvien amb el procés veí en la dimensió 0; en la segona fase els processos poden calcular la segona meitat de les dades corresponents a la primera etapa i la primera meitat de les dades corresponents a la segona etapa i a continuació intercanvien els dos blocs de dades (de tamany igual a $N/2$) amb els processos veïns en les dimensions 0 i 1; en la tercera etapa l'intercanvi es realitza amb els veïns en la dimensió 1 i 0 i així fins que es finalitza l'etapa d'intercanvi.

En el cas que en una operació de comunicació un procés hagi d'enviar un únic missatge al procés veí en la dimensió i ho denotarem com $subproblema < i >$. En el cas que en una operació de comunicació un procés hagi d'enviar un missatge al procés veí en la dimensió i i un altre missatge al procés veí en la dimensió j, ho denotarem com $subproblema < i, j >$. N seguirà considerant-se com el tamany del missatge quan no s'aplica la segmentació de les comunicacions, i en el cas que s'apliqui, el tamany del missatge és igual a N/Q sent Q el grau de segmentació. En el nostre cas es consideraran missatges de tamany igual a $N/2$ ja que el grau de segmentació considerat és 2.

Podem observar que quan $Q = 2$ en la seqüència $D_{e=6}^{BR}$ apareixen els següents tipus de subproblemes:

$< 0 >$	Corresponent a les etapes de pròleg i a l'epíleg
$< 0, 1 >$	apareix 2^{e-1} cops
$< 0, 2 >$	apareix 2^{e-3} cops
$< 0, 3 >$	apareix 2^{e-4} cops
$< 0, 4 >$	apareix 2^{e-5} cops
$< 0, 5 >$	apareix 2^{e-6} cops

En general, en una etapa d'intercanvi qualsevol e amb un grau de segmentació igual a $Q = 2$ troben els següents subproblemes:

$< 0 >$	Corresponent a les etapes de pròleg i a l'epíleg
$< 0, 1 >$	apareix 2^{e-1} cops
$< 0, i >$	amb $1 < i < e$, apareix $2^{e-(i+1)}$ cops

Així doncs, podem distingir quatre tipus de subproblemes diferents:

- Subproblema $< 0 >$.

  En aquest cas s'envia un únic missatge al procés veí en la dimensió 0.

- Subproblema $< 0, 1 >$.

  De la manera que es mapegen els processos del CC-cub, els processos que eren veïns en la dimensió 0 i en la dimensió 1 estan a distància 1 en la malla, en la dimensió *horitzontal* i en la dimensió *vertical* respectivament. En aquest cas totes les comunicacions són locals.

- Subproblema $< 0, i >$ amb i parell.

  En aquest cas tenim que cada procés del CC-cub ha d'enviar dos missatges a dos processos veïns en la mateixa dimensió de la malla, dimensió *horitzontal*.

- Subproblema $< 0, i >$ amb i imparell $i \geq 3$.

  En aquest cas tenim que cada procés del CC-cub ha d'enviar dos missatges a dos processos veïns en les dues dimensions de la malla, el veí en la dimensió 0 per la dimensió *horitzontal* de la malla i el veí en la dimensió i del CC-cub per la dimensió *vertical*.

4.3.2 Algorismes d'encaminament òptims per cada subproblema i per cada escenari

Com acabem de veure podem trobar quatre tipus d'operacions de comunicació diferents. A continuació es realitza una anàlisi més detallada de cadascun d'aquests subproblemes.

Subproblema $< 0 >$

En el cas de les operacions del tipus $< 0 >$ no és necessari cap tipus d'algorisme d'encaminament ja que els processos veïns a distància 0 en l'hipercub es mapegen en la malla a distància 1, per tant totes les comunicacios són locals i no apareix cap tipus de conflicte. Aquesta operació de comunicació es realitza en un cicle, que es pot modelitzar com:

$$T_s + \frac{N}{2} T_e$$

Subproblema $< 0, 1 >$

De la manera que es mapegen els processos del CC-cub, els processos que eren veïns en la dimensió 0 es mapegen en la dimensió *horitzontal* de la malla/torus a distància

1 i els processos que eren veïns en la dimensió 1 es mapegen en la dimensió *vertical* de la malla/torus a distància 1. Totes les comunicacions són locals.

En el cas de malles i torus amb una arquitectura $one - port$, en un cicle els processos s'intercanvien dades amb el procés veí en la dimensió 0 del CC-cub i en el cicle següent els processos s'intercanvien dades amb el procés veí en la dimensió 1 del CC-cub. El cost total de la comunicació és igual al cost d'enviar dos missatges i es pot modelitzar com:

$$2 * (T_s + \frac{N}{2}T_e)$$

sent $T_s + N/2T_e$ el cost d'enviar un missatge amb un tamany igual a $N/2$ degut a que s'està aplicant la segmentació de les comunicacions amb un grau de segmentació igual $Q = 2$.

En el cas de malles i torus amb una arquitectura $all - port$, en un cicle els processos poden intercanviar en paral.lel dades amb els dos processos veïns, en la dimensió 0 i en la dimensió 1, ja que no existeix cap tipus de conflicte en els enllaços que s'utilitzen en les diferents operacions de comunicació. El cost total de la comunicació es pot modelitzar com:

$$2 * T_s + \frac{N}{2}T_e \tag{4.3}$$

En l'expressió 4.3 es pot veure que el cost d'inicialitzar els dos missatges no és possible solapar-lo (d'aquí el factor 2 que multiplica a T_s), mentre que el cost d'enviar els dos missatges pels diferents enllaços es solapa totalment.

Subproblema $< 0, i >$ amb i parell, $i > 0$.

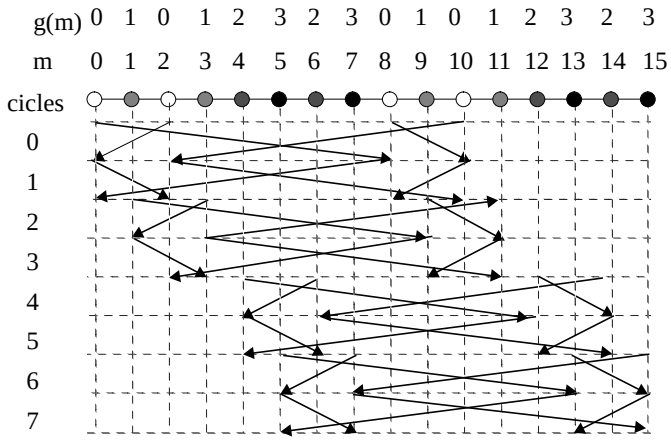
En aquest cas ens trobem que tots els processos del CC-cub han d'enviar dos missatges a dos processos veïns en dues dimensions de l'hipercub que han estat mapejats en la mateixa dimensió de la malla, dimensió *horitzontal*.

A continuació es descriu l'algorisme d'encaminament proposat per realitzar l'operació de comunicació, $< i, j >$, en la que cada procés de l'algorisme CC-cub envia dos missatges a dos nodes veïns en les dimensions i i j respectivament els quals han estat mapejats en la mateixa dimensió de la malla (dimensió *horitzontal* o dimensió *vertical*). L'algorisme d'encaminament proposat és vàlid per qualsevol i i j (considerarem que $i > j$ sense perdre generalitat). Aquesta condició inclou el cas en què $i \geq 2$ amb i parell i $j = 0$, que és el que ens interessa resoldre per l'algorisme BR.

L'algorisme es basa en el fet que per a que tots els processos de l'hipercub enviïn un missatge al procés veí en la dimensió i són necessaris $2^{i/2}$ cicles. D'aquests cicles un procés tan sols està ocupat un cicle i la restà està sense fer res. Aquests cicles es podrien aprofitar per enviar un altre missatge a un altre veí j sempre i quan $j < i$ (tal i com s'ha mostrat en l'exemple de la figura 4.4).

En l'algorisme d'encaminament proposat els nodes de cada fila de la malla s'organitzen en $2^{(i-1)}$ grups, numerats de 0 fins a $2^{(i-1)} - 1$. Tots els nodes d'un mateix grup k utilitzen els cicles $2k$ i $2k + 1$ per realitzar l'intercanvi d'informació per les dimensions i i j. La manera en què cada node s'assigna a un grup és molt important per tal d'evitar conflictes en l'accés als enllaços. Per especificar el grup al que pertany un node m, primer que tot definirem la funció $D(h, b)$ que retorna el número natural un cop eliminat el bit b-èssim de la representació binària del número h. Per exemple, $D(5, 1) = 3$. El node m s'assigna al grup $g(m)$ segons l'expressió següent:

$$g(m) = D(m \bmod 2^{i/2}, j)$$



(a)

mi	mj	cicle 2k	cicle 2k+1
0	0	enviar(i), rebre(j)	enviar(j), rebre(i)
0	1	enviar(j), rebre(i)	enviar(i), rebre(j)
1	0	enviar(j), rebre(i)	enviar(i), rebre(j)
1	1	enviar(i), rebre(j)	enviar(j), rebre(i)

(b)

Figura 4.6 Patró de comunicació pel subproblema $< 0, 4 >$

En la figura 4.6 es mostra un exemple de com es realitzaria l'encaminament dels missatges per completar l'operació de comunicació $< 0, 4 >$. En cada fila de la malla els processos veïns en la dimensió 0 de l'hipercub es mapegen a distància 1 i els processos veïns en la dimensió 4 de l'hipercub es mapegen a distància 4. Els nodes es classifiquen en quatre grups, $2^{\lfloor 4/2 \rfloor}$, identificats per diferents tonalitats de gris. En la figura es mostra quin és el patró de comunicació aplicat per tots els nodes. Aquest esquema determina, per cada node de la línia, quan (en quin cicle) ha d'intercanviar les seves dades i amb qui. El patró que es mostra en la figura 4.6(a), és el mateix per tots els grups, però cada grup realitza l'intercanvi de dades en cicles diferents. En el cas general, tal i com s'ha dit anteriorment, el grup de nodes k s'intercanvia les dades en els cicles $2k$ i $2k+1$. Les operacions que realitza un node m que pertany al grup k en els cicles corresponents, depèn del valors dels bits i-èssim i j-èssim del número binari corresponent a m. En la taula de la figura 4.6(b) es mostra quines operacions realitza cada node dintre d'un grup per completar l'operació de comunicació $< 0, 4 >$, considerada en l'exemple. En aquesta taula, $enviar(i)/rebre(j)$ significa que s'envia/rep un missatge al/del veí en la dimensió i/j. Els valors d'm_j i m_i fan referència als bits i-èssim i j-èssim del número m.

Amb aquest esquema s'aconsegueix solapar totalment l'operació de comunicació que es realitza amb processos veïns en la dimensió j amb l'operació de comunicació que es realitza amb els processos veïns en la dimensió i del CC-cub original.

El cost per realitzar aquesta operació de comunicació en una arquitectura $one-port$ es pot expressar com:

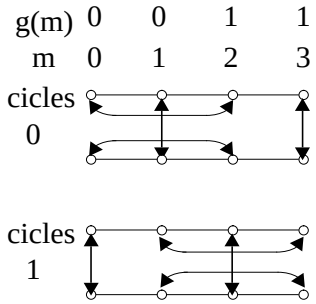
$$T_{Comi2j} = 2^{i/2} \times (T_s + N/2T_e)$$

El cost de la comunicació en una arquitectura $all-port$ és exactament igual que en una arquitectura $one-port$ ja que el cost de la comunicació en aquest cas està limitat pel cost d'enviar el missatge entre els processos mapejats a la distància més gran (i). A [25][10] es demostra que les operacions de comunicació d'aquest tipus no poden resoldre's amb menys cicles i per tant aquestes són òptimes.

Subproblema $< 0, i >$ amb i imparell $i \geq 3$.

En aquest cas tenim que cada procés del CC-cub ha d'enviar dos missatges a dos processos veïns en les dues dimensions de la malla, el veí en la dimensió 0 del CC-cub per la dimensió *horitzontal* de la malla i el veí en la dimensió i del CC-cub per la dimensió *vertical* de la malla. Seria possible aplicar l'esquema proposat pel cas anterior, però per a què sigui factible les diferents files de la malla han d'aplicar l'esquema en ordre diferent (alteració de l'ordre dels cicles d'operació) per a què no hi hagin conflictes en el cas d'un sistema $one-port$. A continuació es proposa un nou algorisme d'encaminament molt més simple en el que totes les files i columnes de la malla poden aplicar exactament el mateix esquema de comunicació.

Inicialment es generen grups de nodes tal i com s'ha descrit pel cas anterior, o sigui com si les dimensions 0 i i estiguessin mapejades en la mateixa dimensió de la malla. L'única diferència respecte a l'algorisme d'encaminament proposat pel cas anterior és l'ordre en què s'envien els missatges en cada un dels cicles necessaris per completar una operació de comunicació. En la figura 4.7 es mostra un exemple pel cas $< 0, 3 >$. En



(a)

mi	mj	2k	2k+1
0	0	enviar(j), rebre(j)	enviar(i),rebre(i)
0	1	enviar(i),rebre(i)	enviar(j), rebre(j)
1	0	enviar(j), rebre(j)	enviar(i),rebre(i)
1	1	enviar(i),rebre(i)	enviar(j), rebre(j)
		<0,3>	<0,3>

(b)

Figura 4.7 Analisi de l'algorisme d'encaminament per l'operació $< 0,3 >$ en un model one-port

4.7(a) s'indica el patró de comunicació dels dos cicles per separat, cada línia de punts blancs representa un grup de quatre nodes situats en la mateixa fila de la malla. En la figura 4.7(b) es mostra l'ordre en què s'envien els missatges en els diferents cicles cada un dels nodes d'un grup. Aquest esquema s'aplica per qualsevol $i > 1$.

De nou l'algorisme d'encaminament proposat és vàlid per malles i torus i per arquitectures $one-port$ i $all-port$. El cost per realitzar aquesta operació de comunicació sigui quin sigui l'entorn considerat es pot expressar com:

$$T_{Comi2j} = 2^{(i-1)/2} \times (T_s + N/2T_e)$$

A [25][10] es demostra que tots els algorismes d'encaminament proposats en aquest treball són òptims, es a dir, sempre s'aconsegueix finalitzar una operació de comunicació amb el nombre mínim de cicles possible.

4.4 AVALUACIÓ

4.4.1 Model analític de l'algorisme BR segmentat quan es mapeja en una malla bidimensional

En aquesta secció es desenvolupa el model analític del cost de la comunicació de l'algorisme BR segmentat amb un grau de segmentació $Q = 2$, quan aquest es mapeja en una malla quadrada aplicant l'algorisme de mapeig *standard*, descrit al principi d'aquest capítol.

Analitzem amb més detall la seqüència de dimensions de l'ordenació BR per completar un escombrat per un CC-cub de dimensió $d = 6$ (es detallen les dimensions utilitzades en cada etapa $e = 0, 1, 2, 3, 4, 5$ i les dimensions utilitzades en cadascuna de les etapes de divisió). Tenim que:

$$
\begin{array}{rll}
D_{d=6}^{BR} = & < & 0102010301020104010201030102010501020103010201040102010301020105 \\
& 5 & 01020103010201040102010301020105 \\
& 4 & 010201030102010 \\
& 3 & 0102010 \\
& 2 & 010 \\
& 1 & 0 \\
& 0 & \\
& 5 & >
\end{array}
$$

Amb una finestra de dimensió $Q = 2$ trobem els següents subproblemes (els quals s'obtenen de segmentar les etapes d'intercanvi):

$< 0 >$	Que apareix (d-1)*2 cops
$< 0, 1 >$	Que apareix $2^{d-5} + 2^{d-4} + 2^{d-3} + 2^{d-2} + 2^{d-1}$ cops

$< 0, 2 >$ Que apareix $2^{d-5} + 2^{d-4} + 2^{d-3} + 2^{d-2}$ cops

$< 0, 3 >$ Que apareix $2^{d-5} + 2^{d-4} + 2^{d-3}$ cops

$< 0, 4 >$ Que apareix $2^{d-5} + 2^{d-4}$ cops

$< 0, 5 >$ Que apareix 2^{d-5} cops

En general, per un hipercub de dimensió d parell, el nombre de subproblemes del tipus $< 0, i >$ que apareixen en totes les etapes del nucli necessàries per completar un escombrat són:

$$< 0, i > \quad \text{apareix } 2^1 + 2^2 + ... + 2^{d-i} \text{ cops}$$

En les fases de pròleg i epíleg s'han d'enviar un total de $(d-1) * 2$ missatges. Respecte a les etapes de divisió, s'ha d'enviar un missatge per cadascuna de les dimensions de l'hipercub $0, 1..d-1$. I per últim s'ha de realitzar l'última transició, en la que s'envia un missatge per la dimensió $d-1$.

Considerem de nou que el tamany dels missatges en l'algorisme sense segmentar és igual a N. En les etapes d'intercanvi de l'algorisme segmentat amb un grau de segmentació igual a 2 els missatges són de tamany $N/2$, mentre que els missatges de les etapes de divisió (en les que no es pot aplicar la segmentació) són de tamany igual a N, sabent que el cost de realitzar cada operació de comunicació és el que s'ha modelitzat en la secció anterior en termes de T_s i T_e. A continuació es desenvolupa el model del cost de comunicació de l'algorisme BR-segmentat (amb un grau de segmentació igual a $Q = 2$) quan es mapeja en malles amb una arquitectura $one - port$.

L'expressió que ens dóna el nombre de missatges degut a l'operació de comunicació $< 0, 1 >$ és igual a :

$$\sum_{i=0}^{d-2} 2^{i+1} \times 2(T_s + N/2T_e) = 4(2^{d-1} - 1)(T_s + \frac{N}{2}T_e)$$

on el terme 2^{i+1} fa referència al nombre de parelles $< 0, 1 >$ que trobem en totes les fases d'intercanvi i el terme $2(T_s + N/2T_e)$ fa referència al cost de realitzar una operació $< 0, 1 >$ en el model $one - port$.

L'expressió que ens dóna el nombre de missatges degut a les operacions $< 0, i >$ amb $i > 1$ és:

$$\sum_{i=1}^{d/2-1} [\sum_{k=0}^{d-2i-1} 2^{k+1} + \sum_{k=0}^{d-(2i+1)-1} 2^{k+1}] \times 2^i(T_s + \frac{N}{2}T_e)$$
$$= [3 \times 2^d \times \frac{2^{(d/2)-1} - 1}{2^{(d/2)-1}} - 4 \times (2^{d/2} - 2)](T_s + \frac{N}{2}T_e)$$

On el terme $[\sum_{k=0}^{d-2i-1} 2^{k+1} + \sum_{k=0}^{d-(2i+1)-1} 2^{k+1}]$ fa referència al nombre de parelles $< 0, i >$ amb $i > 1$ i el terme $2^i(T_s + N/2T_e)$ fa referència al cost de realitzar una operació $< 0, i >$

Les etapes de pròleg i epíleg tenen un cost igual a:

$$2(d-1)(T_s + \frac{N}{2} * T_e)$$

El cost de l'última etapa d'intercanvi és igual al cost d'enviar un missatge de tamany N a un veí a distància 0, igual a $T_s + N * T_e$.

El cost degut a les etapes de divisió és igual a:

$$\sum_{i=0}^{d/2-1} 2 \times 2^i(T_s + N * T_e) = 2(T_s + NT_e)(2^{d/2} - 1)$$

i per últim, el cost degut a l'última transició és igual a:

$$2^{d/2-1}(T_s + N * T_e)$$

El cost total de la comunicació es pot expressar com:

$$\begin{aligned} T_{embBRSmO} &= [2(d+1) - 4 \times 2^{d/2} + 2^d[\frac{2^{(d/2)-1}-1}{2^{(d/2)-1}} + 2]](T_s + \frac{N}{2}T_e) + \\ &+ (\frac{5}{2} \times 2^{d/2} - 1)(T_s + NT_e) \end{aligned}$$

L'única diferència entre aquest model i el model per arquitectures $all - port$ és la modelització de l'operacio de comunicació $< 0, 1 >$, que en el cas del model $all - port$ es pot solapar la transmissió dels dos missatges. L'expressió que ens dóna el nombre de missatges degut a l'operació de comunicació $< 0, 1 >$ en aquest cas és igual a:

$$\sum_{i=0}^{d-2} 2^{i+1} \times (T_s + N/2T_e) = (2^d - 2)(T_s + \frac{N}{2}T_e)$$

Amb el que s'aconsegueix reduir el cost degut als missatges de tipus $< 0, 1 >$ a la meitat respecte al que s'obté en el cas de la malla. El cost de la comunicació es pot expressar com:

$$\begin{aligned} T_{embBRSmA} &= [2(d+1) + 2 - 4 \times 2^{d/2} + 2^d[\frac{2^{(d/2)-1}-1}{2^{(d/2)-1}} + 1]](T_s + \frac{N}{2}T_e) + \\ &+ (\frac{5}{2} \times 2^{d/2} - 1)(T_s + NT_e) \end{aligned}$$

Avaluació de la reducció del cost de la comunicació quan s'aplica la segmentació de les comunicacions

En la gràfica de la figura 4.8 es mostra quina és la reducció de la comunicació de l'algorisme BR mapejat en la malla bidimensional quan es segmenta (amb $Q = 2$) respecte al cost del mateix algorisme quan es mapeja en la malla però no s'aplica la segmentació de les comunicacions (amb $Q = 1$). S'observa que per matrius petites i

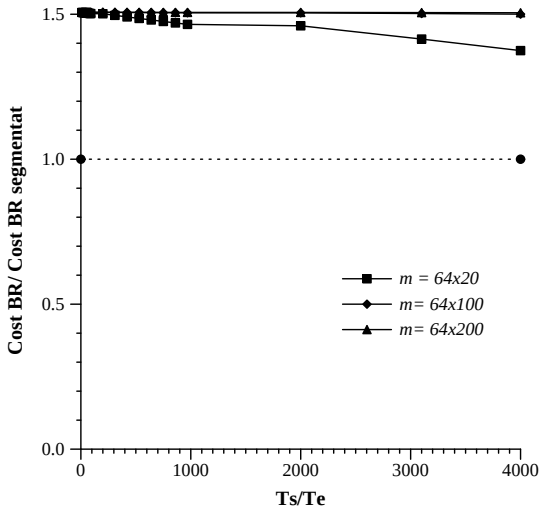


Figura 4.8 Reducció del cost de la comunicació degut a la segmentació de les comunicacions

valors grans de la relació dels paràmetres T_s/T_e la reducció és petita (tan sols fins a un 20%), i a mesura que el tamany de la matriu va augmentant la reducció del cost de la comunicació és d'un 50% del cost de l'algorisme sense segmentar per qualsevol valor de la relació T_s/T_e.

Anàlisi del rendiment en una malla regular de dues dimensions respecte a l'algorisme $2D$

S'ha comprovat que per un model $one-port$ l'algorisme $2D$ descrit en el capítol 2 sempre té un cost de comunicació inferior a l'algorisme BR segmentat i mapejat en la malla.

En les gràfiques de la figura 4.9 s'analitza quin és el rendiment de l'algorisme BR segmentat (amb $Q = 2$) quan es mapeja en una malla bidimensional quadrada amb una arquitectura $all-port$. En les gràfiques es mostra la relació del cost de comunicació $T_{2D}/T_{embBRSmO}$ per diferents valors de T_s/T_e.

En les gràfiques de la figura 4.9 es pot observar que amb l'algorisme BR segmentat es pot reduir el cost de la comunicació fins a gairebé un 21% per malles amb 16 nodes

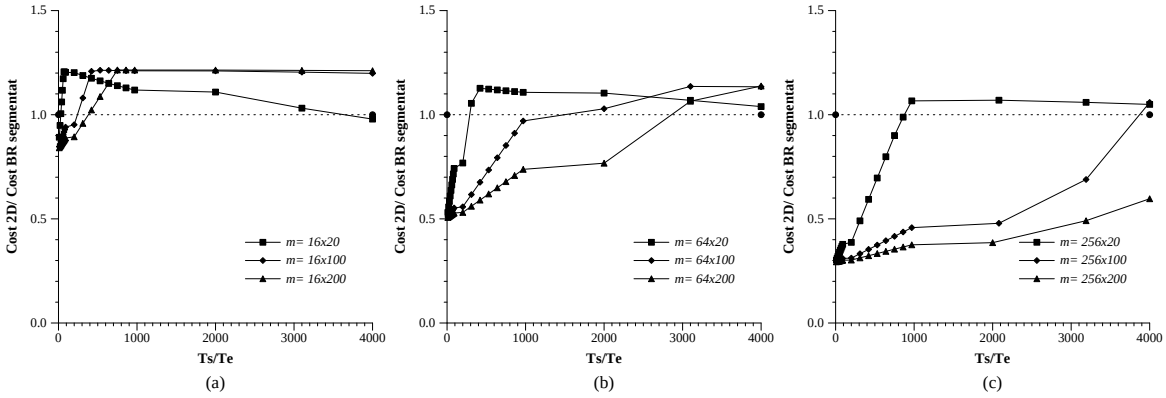


Figura 4.9 Anàlisi de l'algorisme BR segmentat quan es mapeja en una malla $2D$ fixa amb una arquitectura $all - port$. En una malla amb 16 nodes (a). En una malla amb 64 nodes (b) i en una malla amb 256 nodes (c).

(gràfica 4.9(a)), i fins a un 13% en malles amb 64 nodes (gràfica 4.9(a)) i fins un 8% en malles amb 256 nodes (gràfica 4.9(c)). A mesura que augmenta el tamany del problema, la relació T_s/T_e ha de ser una mica més gran per obtenir un rendiment màxim. Per malles quadrades amb un nombre de nodes més gran que 256, el nombre de missatges que es generen amb l'algorisme BR es dispara i fa que l'algorisme $2D$ tingui costos de comunicació considerablement més petits que l'algorisme BR segmentat.

4.4.2 Model analític de l'algorisme BR segmentat quan es mapeja en un torus bidimensional

En aquesta secció es desenvolupa el model analític del cost de la comunicació de l'algorisme BR segmentat amb un grau de segmentació $Q = 2$, quan aquest es mapeja en un torus regular bidimensional aplicant l'algorisme de mapeig Xor descrit al principi d'aquest capítol.

Si es realitza una anàlisi equivalent al realitzat pel cas de les malles $2D$, s'obtenen els mateixos tipus de subproblemes que pel cas de la malla, i l'única diferència respecte al model desenvolupat per una malla és que en el torus les dimensions $e-1$ i $e-2$, 4 i 5 en l'exemple per $D^{BR}_{d=6}$, es mapegen a la mateixa distància que les dimensions $e-3$ i $e-4$, 3 i 2 en l'exemple per $D^{BR}_{d=6}$, igual a $2^{(e-4)/2}$. Per tant, els models analítics pel cas

d'un torus són gairebé els mateixos que pel cas de la malla, amb l'única diferència que en el cas del torus el cost de la comunicació entre els processos veïns en les dimensions $e-1$ i $e-2$ és la meitat respecte del cost de la comunicació en una malla degut a què en el torus s'utilitzen els enllaços *wraparound*.

Tenint en compte aquests canvis les expressions del cost de la comunicació en un torus amb una arquitectura $one-port$ pot expressar-se quan $d \geq 4$ com:

$$\begin{aligned} T_{embBRStO} &= [2(d+1) - 4 \times 2^{d/2} - 4 \times 2^{\frac{(d-2)}{2}} + 2^d[\frac{2^{(d/2)-1}-1}{2^{(d/2)-1}} + 2]](T_s + \frac{N}{2}T_e) + \\ &+ (\frac{5}{2} \times 2^{d/2} - \frac{3}{2} \times 2^{\frac{(d-2)}{2}} - 1)(T_s + NT_e) \end{aligned}$$

Quan $d = 2$ el cost de la comunicació coincideix amb el cost de comunicació de la malla.

En el cas d'una arquitectura $all-port$ el cost de la comunicació per $d \geq 4$ pot expressar-se com:

$$\begin{aligned} T_{embBRStA)} &= [2(d+1) + 2 - 4 \times 2^{d/2} - 4 \times 2^{\frac{(d-2)}{2}} + 2^d[\frac{2^{(d/2)-1}-1}{2^{(d/2)-1}} + 1]](T_s + \frac{N}{2}T_e) + \\ &+ (\frac{5}{2} \times 2^{d/2} - \frac{3}{2} \times 2^{\frac{(d-2)}{2}} - 1)(T_s + NT_e) \end{aligned}$$

Quan $d = 2$ el cost de la comunicació de nou coincideix amb el cost de comunicació de la malla.

Avaluació de la reducció del cost de la comunicació quan s'aplica la segmentació de les comunicacions

En aquest cas el comportament és exactament igual al cas de la malla.

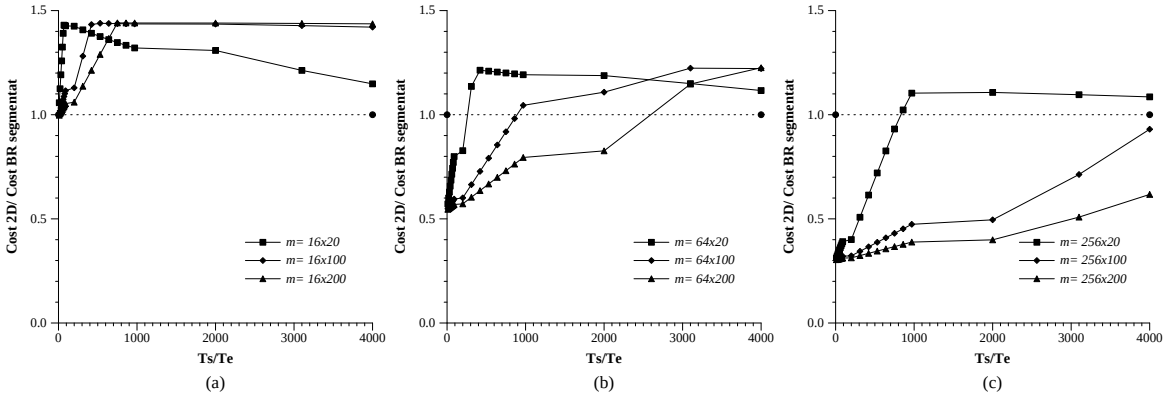


Figura 4.10 Anàlisi de l'algorisme BR segmentat quan es mapeja en un torus $2D$ fix amb una arquitectura $all-port$. En un torus amb 16 nodes (a), en un torus amb 64 nodes (b) i en un torus amb 256 nodes (c).

Anàlisi del rendiment en un torus regular de dues dimensions respecte a l'algorisme $2D$

En les gràfiques de la figura 4.10 s'analitza quin és el rendiment de l'algorisme BR segmentat (amb $Q = 2$) quan es mapeja en un torus bidimensional quadrat amb una arquitectura $all-port$. En les gràfiques es mostra la relació del cost de comunicació $T_{2D}/T_{embBRStO}$ per diferents valors de T_s/T_e.

En les gràfiques de la figura 4.10 es pot observar que amb amb l'algorisme BR segmentat es pot reduir el cost de la comunicació fins a quasi un 40% per torus amb 16 nodes (gràfica 4.9(a)), fins a un 20% en torus amb 64 nodes (gràfica 4.9(a)) i fins un 11% en malles amb 256 nodes (gràfica 4.9(c)). A mesura que augmenta el tamany del problema, la relació T_s/T_e ha de ser una mica més gran per obtenir un rendiment màxim. Per torus quadrats amb un nombre de nodes més gran que 256, el nombre de missatges que es generen amb l'algorisme BR es dispara i fa que l'algorisme $2D$ tingui costos de comunicació considerablement més petits que l'algorisme BR segmentat.

4.5 RESUM DE CONTRIBUCIONS

En el grup de treball s'ha desenvolupat una metodologia, anomenada $CALMANT$, que ens permet mapejar de forma eficient en una topologia en malla o una topologia toroidal un algorisme CC-cub (algorisme amb una topologia de comunicacions en hipercub). En aquest capítol s'ha aplicat la metodologia $CALMANT$ a l'algorisme BR, realitzant-se una extensió a subproblemes del tipus $< i, j >$ (amb $j \geq i + 1$) i proposant-se un nou algorisme d'encaminament de missatges per resoldre un tipus d'operació de comunicació que s'ha de realitzar a l'hora d'aplicar l'algorisme, molt més senzill que el que havia estat proposat inicialment en la metodologia.

El cost de la comunicació de l'algorisme resultant s'ha comparat amb el cost de l'algorisme $2D$ proposat en el capítol 3 en un entorn amb una topologia en malla bidimensional regular i en un entorn amb una topologia toroidal bidimensional regular. En els dos casos, s'ha avaluat per una arquitectura $one-port$ i per una arquitectura $all-port$.

S'ha vist que en el cas d'arquitectures $one-port$ l'algorisme $2D$ té un cost de comunicació inferior a l'algorisme BR-$segmentat$ un cop mapejat en la malla o el torus.

En el cas d'arquitectures $all-port$, per malles i torus amb un nombre de nodes inferior o igual a 256 l'algorisme BR-$segmentat$ té un cost de comunicació inferior, entre un 10% i un 45% menys, que l'algorisme $2D$. Per tamanys de problema petits i nombre de nodes petit (*16* i *64*), la reducció és important per gairebé qualsevol valor de T_e i T_s. A mesura que el tamany del problema augmenta es necessiten relacions de T_e i T_s més grans. Aquest comportament s'accentua per sistemes amb un nombre de nodes més gran de *64*.

5

ANÀLISI D'ESCALABILITAT ORIENTADA A L'USUARI FINAL DEL SISTEMA

Resum

L'escalabilitat és una propietat desitjable dels sistemes paral.lels. Tothom té una noció intuïtiva del concepte d'escalabilitat; un sistema es considera escalable si és capaç d'utilitzar de forma eficient un nombre creixent de recursos del sistema.

En la literatura és possible trobar diverses metodologies d'anàlisi de l'escalabilitat, isoeficiència, temps fix, memòria fixa, etc. Metodologies que si s'apliquen a un determinat sistema donen resultats diversos i inclús contradictoris. Davant d'aquesta diversitat de metodologies i resultats és important tenir clar per una part què és el que interessa a l'usuari que realitza l'anàlisi i per una altra part quin tipus d'anàlisi realitza el mètode escollit; d'aquesta forma sabrem com interpretar de forma correcta els resultats obtinguts.

En aquest capítol es proposa una nova metodologia d'anàlisi de l'escalabilitat orientada a l'usuari final del sistema; més concretament es proposen tres mètodes diferents per realitzar l'anàlisi segons les necessitats de l'usuari. En última instància l'anàlisi que es proposa s'ha realitzat amb l'idea d'ajudar a l'usuari del sistema a decidir si és interessant o no incrementar el nombre de nodes del sistema.

En la primera part del capítol es fa una petita introducció on es descriu l'estat de l'art del problema de l'escalabilitat. A continuació es presenta un exemple on s'analitza l'escalabilitat d'un sistema aplicant el mètode de la isoeficiència *i aplicant un mètode proposat en aquest treball. En l'exemple es veu clarament les conclusions tant diferents que es poden obtenir segons el mètode utilitzat. A continuació es defineix formalment la metodologia proposada en aquest treball, orientada a l'usuari final del sistema. Seguint aquesta metodologia, es presenten tres mètodes d'anàlisi de l'escalabilitat. Finalment, en l'última secció es presenta un exemple d'aplicació d'aquests nous mètodes d'escalabilitat.*

5.1 INTRODUCCIÓ

L'escalabilitat és una propietat desitjada en un sistema multicomputador. El concepte d'escalabilitat és bastant intuitïu; un sistema es considera escalable si és capaç d'utilitzar de forma eficient un nombre creixent de recursos del sistema [17], [35].

En la literatura és possible trobar diverses metodologies d'anàlisi de l'escalabilitat [36], [6], [60], isoeficiència [26], temps fix, memòria fixa [17], etc. Metodologies que si s'apliquen a un determinat sistema donen resultats diversos i inclús contradictoris [17], [55]. Davant d'aquesta diversitat de metodologies i resultats és important tenir clar per una part què és el que interessa a l'usuari que realitza l'anàlisi i per una altra part quin tipus d'anàlisi realitza el mètode escollit; d'aquesta forma sabrem com interpretar de forma correcta els resultats obtinguts [30].

Vegem quines questions planteja el concepte *escalabilitat*:

- Qui pot estar interessat en l'anàlisi de l'escalabilitat? Els resultats de l'anàlisi de l'escalabilitat poden ser útils per als dissenyadors de l'arquitectura, dissenyadors del sistema o per als usuaris finals. Quin és l'objectiu de l'anàlisi de l'escalabilitat ? Dissenyar noves arquitectures, màrketing, comparar diferents sistemes, determinar quin increment de benefici reporta incrementar el nombre de nodes del sistema...
- Què ens interessa escalar i com? Bàsicament el terme es pot aplicar a l'arquitectura, a un algorisme o bé a un sistema.

  En el context d'aquest treball un sistema es defineix com la combinació d'una arquitectura i un algorisme concrets.
- Com determinem si un sistema és escalable? Fixant-nos en l'eficiència, en l'$Speed-up$, veient en quina proporció augmenta el tamany del problema o quina precisió dels resultats es pot obtenir...

Escalar una màquina significa incrementar la seva potència. Això s'aconsegueix fent alguna part de la màquina més gran, més sofisticada, amb més potència o més ràpida

(memòria. processador, sistema d'IO...). Des del punt de vista del nostre interès en màquines paral.leles caracteritzem la mida de la màquina amb el nombre de nodes. A l'hora d'escalar una màquina considerem que el node (processador, memòria, I/O) es manté exactament igual, i per tant escalar una màquina es traduirà en incrementar el nombre de nodes, nodes exactament iguals als nodes que hi havien a la màquina abans d'escalar.

En aquest treball ens centrem en l'anàlisi de l'*escalabilitat* de sistemes (arquitectura + algorisme) des del punt de vista de l'usuari final. Des d'aquest punt de vista, l'anàlisi de l'escalabilitat ha de donar informació suficient per tal que es pugui respondre la pregunta seguent: "És una bona inversió incrementar el nombre de nodes del meu sistema?" Malgrat que s'han proposat en la literatura diversos mètodes d'anàlisi de l'escalabilitat, considerem que cap d'ells permet respondre aquesta pregunta, ja que els diferents mètodes no han estat proposats pensant en les necessitats que pot tenir un usuari final del sistema. Malgrat que s'han proposat molts mètodes d'anàlisi de l'escalabilitat (adreçar-se a [36]), cap dels mètodes proposats han estat pensats des d'aquest punt de vista.

En tots els mètodes proposats es poden distingir tres components:

- Model d'escalat. Indica com utilitza l'usuari l'increment de nodes del sistema.
- Figura de mèrit. És la mesura que vol millorar l'usuari quan incrementa el nombre de nodes del sistema.
- Caracterització de l'escalabilitat. És la funció que ens permet mostrar el comportament de la figura de mèrit quan el nomdre de nodes del sistema va incrementant-se. És necessari definir un criteri clar per determinar si un sistema és escalable (molt, poc o no gens). Per exemple, el criteri per determinar l'escalabilitat podria definir-se com: si la funció és positiva el sistema és escalable i si la funció és zero el sistema no és escalable. A l'hora de representar la funció que caracteritza l'escalabilitat és important que ens permeti determinar fàcilment si el sistema escala i en quina mesura.

Alguns dels mètodes més àmpliament utilitzats que s'han proposat pera l'anàlisi de l'escalabilitat són el mètode de la *isoeficiència* [26] i el mètode CMP [16], [17]. Veiem com podem identificar en aquests mètodes cadascun dels components que acabem de definir.

En el mètode de la *isoeficiència*, alhora que s'incrementa el nombre de nodes del sistema el tamany del problema també augmenta per tal de mantenir constant l'eficiència del sistema a un valor predeterminat. Aquest seria el model d'escalat d'aquest mètode. La figura de mèrit és el tamany del problema. Per caracteritzar l'escalabilitat amb aquest mètode s'utilitza la funció $f_E(p)$, que determina quin tamany de problema és necessari per obtenir una eficiència del sistema igual a E amb un nombre de nodes igual a p. Un sistema és escalable si és possible trobar un problema no massa gran que ens permeti obtenir una eficiència del sistema igual a E.

El mètode CMP ha estat proposat en [17], i aquest aplica el model d'escalabilitat limitat per la memòria. Amb aquest model d'escalat, incrementar el nombre de nodes del sistema s'utilitza per poder augmentar el tamany del problema tant com sigui possible, amb l'única limitació de la capacitat de memòria de cada node. La figura de mèrit és l'$speed-up$ escalat. Si la funció $speed-up$ es defineix com:

$$speedup(p,n) = \frac{T_1(n)}{T_p(n)}$$

on p és el nombre de processadors, n és el tamany del problema i $T_k(n)$ és el temps per resoldre un problema de tamany n amb k processadors. Definim l'*speedup escalat* com:

$$speedupescalat(p,n') = \frac{T_1(n')}{T_p(n')}$$

L'*speedup escalat* es defineix com l'*speedup* d'un sistema amb p nodes que resol un tamany de problema n' igual al problema més gran que es pugui resoldre tenint en

compte la memòria dels p nodes. L'única limitació en aquest model és la quantitat de memòria que té cada node.

L'escalabilitat es caracteritza per la funció que defineix el creixement asimptòtic de l'$speed-up$ escalat, quan el nombre de nodes creix cap a infinit.

Aquests mètodes (CPM i isoeficiència) han estat utilitzats pera l'anàlisi d'alguns sistemes concrets dels que s'han obtingut resultats interessants. Aquests reusltats interessants per arquitectes del sistema són de molt poca utilitat per als usuaris finals del sistema. Això és degut a que alguns dels components dels dos mètodes (model d'escalat i figura de mèrit) no reflecteixen l'interès de l'usuari final a l'hora d'utilitzar el sistema paral.lel.

Per exemple, incrementar el tamany del sistema amb l'objectiu de mantenir l'eficiència no és de molta utilitat a un usuari final del sistema, al qual molt possiblement li interessarà incrementar el tamany del sistema amb l'objectiu de resoldre un problema més gran per obtenir resultats més precisos o amb l'objectiu de reduir al màxim el temps d'execució. De la mateixa manera es podria qüestionar que la figura de mèrit $speed-up$ escalat pot ser útil als arquitectes del sistema però pot ser de poca utilitat als usuaris finals. Generalment, els usuaris finals estan interessats en altres figures de mèrit com el temps d'execució o la mida del problema.

Amb aquests arguments, podem concloure que els resultats obtinguts com a resultat de l'aplicació dels mètodes CPM i *isoeficiència* poden ser de poc interès als usuaris finals. El primer perquè no utilitza una figura de mèrit adequada i el segon perquè aplica un model d'escalat no massa útil a l'usuari final del sistema.

Molt del material que es presenta a continuació és ben conegut. La contribució d'aquest treball a l'anàlisi de l'escalabilitat és l'esforç realitzat per transformar aquest material en una eina útil per a l'usuari final del sistema. En termes dels tres elements que caracteritzen un mètode d'anàlisi de l'escalabilitat la principal contribució de la metodologia proposada és:

- Tan sols s'utilitzen figures de mèrit i models d'escalat que són rellevants a l'usuari final. En particular, s'utilitzen figures de mèrit com el temps d'execució i el tamany del problema, i models d'escalat com el model de mida de problema fix, limitat pel temps d'execució o bé limitat per la memòria. A més a més, tan sols s'utilitzen combinacions de figura de mèrit i model d'escalat que tinguin sentit.
- Es proposa una nova forma de caracteritzar l'escalabilitat d'un sistema que permet realitzar una anàlisi ràpida del comportament del sistema.

Seguint la metodologia que es descriu a continuació és possible proposar diversos mètodes d'anàlisi de l'escalabilitat. En aquest capítol es proposen tres mètodes, útils per diferents usuaris finals del sistema. Com es veurà més endavant, les conclusions que es poden obtenir amb aquests mètodes són diferents i més útils que altres mètodes proposats. Per il.lustrar això, en la secció 2 es presenta un exemple en el que s'analitza l'escalabilitat d'un sistema paral.lel (càlcul de la FFT en un hipercub) aplicant el mètode de la *isoeficiència* i aplicant un mètode orientat a l'usuari el qual combina el model d'escalat limitat pel temps i la figura de mèrit tamany del problema. Es veu que els resultats obtinguts són considerablement diferents depenent del mètode aplicat. Després d'aquest exemple, en la secció 3 es descriu formalment la metodologia proposada. Es presenten tres mètodes diferents d'anàlisi de l'escalabilitat orientats a l'usuari, els quals descriuen tres formes diferents d'utilitzar un nombre creixent de nodes. Finalment, en la secció 4 es mostra un exemple d'aplicació d'aquests nous mètodes. L'objectiu d'aquest exemple no és realitzar l'anàlisi d'escalabilitat del sistema sinó mostrar la facilitat amb què es pot determinar l'evolució d'un sistema (si és o no escalable, i en quina mesura) aplicant la metodologia proporcionada.

5.2 MOTIVACIÓ DE LA NOVA METODOLOGIA

En [27] es presenta l'anàlisi d'escalabilitat de diversos algorismes per calcular la FFT en un hipercub aplicant el mètode de la *isoeficiència*. En aquesta secció es revisen els resultats d'aquesta anàlisi pel cas concret de l'algorisme $binary - exchange$ en un hipercub. A continuació es presenta un mètode d'anàlisi alternatiu pel mateix sistema

paral.lel, basat en un mètode proposat en aquest treball. Com veurem més tard, les conclussions que s'obtenen de les dues anàlisis són molt diferents i inclús contradictoris.

5.2.1 Algorisme $binary - exchange$ per al càlcul de la FFT en un hipercub

Suposem que hem de calcular l'FFT d'una seqüència d'n punts ($n = 2^r$) en un hipercub de dimensió d, amb $p = 2^d$. L'algorisme $binary - exchange$ és una implementació paral.lela de l'algorisme $radix - two\ Cooley - Tukey$ per resoldre el problema que ens ocupa. A continuació es descriu l'algorisme $binary - exchange$.

L'element i-èssim de la seqüència d'entrada s'emmagatzema en el node i/p. L'algorisme $binary - exchange$ consisteix en r iteracions. En cadascuna de les d primeres iteracions, cada procés (node) es comunica amb un dels processos veïns en l'hipercub (nodes veïns) i a continuació realitza una sèrie de càlculs. Més concretament, l'etapa de comunicació consisteix en intercanviar 2^{r-d-1} elements complexos (operands d'entrada) i com a resultat generar dos nous nombres complexos de la nova seqüència. En cadascuna de les $r - d$ últimes iteracions cada procés calcula 2^{r-d-1} parelles de nombres complexos (per realitzar aquest càlcul no és necessari cap tipus de comunicació).

A partir d'ara, denotarem com operació elemental al càlcul de cadascuna de les parelles de nombres complexos. El temps necessari per completar una operació elemental es defineix com T_c (per exemple, el temps per calcular una parella de nombres complexos és igual a $2 \times T_c$). El cost d'enviar un missatge amb una longitud igual a N nombres reals a un procés veí en l'hipercub es pot expresar tal i com s'ha modelitzat en el capítol 1.

El cost total de l'algorisme $binary - exchange$ quan s'executa en un hipercub és igual a:

$$T_p = r(2^{r-d}T_c) + d(T_s + 2^{r-d}T_e)$$

El primer terme es correspon al cost de realitzar el càlcul de 2^{r-d-1} parelles de nombres complexos, tenint en compte que el cost d'una parella és igual a $2 \times T_c$, en cadascuna de les etapes de què consta l'algorisme, igual a r. El segon terme modelitza el cost de la comunicació de les d primeres etapes. En cadascuna de les etapes de comunicació s'intercanvien 2^{r-d-1} parelles de nombres complexos.

Suposem que cada node del multicomputador disposa d'una capacitat de memòria local igual a R nombres reals ($R = 2^s$). També es considerarà que la FFT més gran que pot resoldre's en un sistema amb 2^d nodes té 2^{d+s-1} nombres complexos. Si el sistema no tingués memòria virtual no seria possible calcular FFT's més grans. Si en tingués, problemes més grans generarien moltes fallades de pàgines, aspecte que no es té en compte en el model anterior.

Finalment, és important assenyalar que si la seqüència d'entrada inicial consta d'n punts complexos llavors el sistema més gran que pot utilitzar-se per resoldre l'FFT aplicant l'algorisme $binary - exchange$ és igual a $n/2$.

5.2.2 Anàlisi de l'escalabilitat aplicant el mètode de la *isoeficiència*

A continuació es presenten breument les conclusions que s'obtenen de l'anàlisi de l'escalabilitat quan s'aplica el mètode d'*isoeficiència* proposat a [27] al sistema paral.lel composat per l'algorisme emphbinary-exchange quan s'executa en un multicomputador amb una topologia en hipercub. Amb el mètode de la *isoeficiència* l'escalabilitat d'un sistema paral.lel es caracteritza per la funció que determina la mida del problema que es requereix per mantenir l'eficiència a un valor predeterminat quan s'incrementa el nombre de nodes del sistema. La mida del problema, W, que considerem en el nostre treball es defineix com el nombre d'operacions elementals que es requereixen per resoldre el problema aplicant el millor algorisme seqüencial. Aquesta definició de treball és una mica diferent de la que es dóna en l'article [27], on W no és el nombre d'operacions sinó el temps que es tarda en executar-les. Aquesta petita diferència no afecta per res el resultat de les comparacions.

A partir d'ara, t_e^i fa referència al temps que passa el node i realitzant càlculs útils i t_o^i fa referència al temps que passa el node i en comunicació, més el temps que passa un node realitzant qualsevol altra activitat que no existiria en cas que el problema es resolgués en un únic node utilitzant el millor algorisme seqüencial conegut.

Tenim que $W \times T_c = \sum_i t_e^i$.

En el cas del càlcul de la FFT, es pren l'algorisme de $radix-two\ Cooley-Tukey$ com el millor algorisme seqüencial. En aquest cas tenim que $W = nr$.

Si T_p es defineix com el temps de resoldre el problema utilitzant un sistema paral.lel amb p nodes, l'eficiencia paral.lela $E(p)$ es defineix com:

$$E(p) = \frac{W \times T_c}{T_p \times p}$$

El temps extra degut a l'algorisme paral.lel, anomenat T_o, es defineix com $T_o = \sum_i t_o^i$. Si $T_p = t_e^i + t_0^i$ per qualsevol i, tenim que $W \times T_c + T_o = p \times T_p$ i per tant podem expressar l'eficiència paral.lela com:

$$E(p) = \frac{1}{1 + (T_o/W) \times T_c}$$

És obvi que si volem mantenir l'eficiència paral.lela E a un cert valor quan s'incrementa el nombre de nodes del sistema p, el ratio $(T_o/W) \times T_c$ ha de mantenir-se constant. Degut a que generalment T_o s'incrementa amb p, el ratio $(T_o/W) \times T_c$ es mantindrà constant tan sols si la mida del problema W s'incrementa. En particular s'ha de complir que:

$$W = K\frac{T_o}{T_c}$$

on $K = E/(1-E)$.

El grau d'escalabilitat del sistema es caracteritza per la funció $f_E(p)$ la qual determina el valor de W que es requereix per mantenir una determinada eficiència E per un nombre de nodes p. Aplicant l'anàlisi de la isoficiència a l'algorisme $binary-exchange$ en l'hipercub s'obtenen les següents expressions:

$$\begin{aligned} t_o^i &= d(T_s + 2^{r-d}T_e) \\ T_o &= \sum_i t_o^i = pd(T_s + 2^{r-d}T_e) \\ W &= nr \end{aligned}$$

Si $W = KT_o/T_c$ llavors tenim:

$$nr = \frac{Kpd(T_s + 2^{r-d}T_e)}{T_c} \tag{5.1}$$

Per analitzar l'escalabilitat del sistema, veiem quina influència tenen cadascun dels dos termes, sabent que l'increment de la mida del problema estarà determinat pel terme més gran. Analitzem amb més detall cadascun dels dos termes. Per la restricció en W degut al primer terme de l'expressió 5.1 hem de:

$$W = nr = K\frac{T_s}{T_c}pd = O(p \log p) \tag{5.2}$$

En aquest cas quan el sistema creix en un nombre de p nodes, el tamany del problema ha de créixer en una relació $O(p \log p)$ per mantenir l'eficiència del sistema.

Per la restricció en W degut al segon terme de l'expressió 5.1 hem de:

$$\begin{aligned} r &= K\frac{T_e}{T_c}d \rightarrow \\ W &= nr = n \times K\frac{T_e}{T_c}d \rightarrow \end{aligned}$$

$$W = nr = p^{KT_e/T_c}K\frac{T_e}{T_c}d = O(p^{KT_e/T_c}\ \log\ p) \tag{5.3}$$

En aquest cas quan el sistema creix en un nombre de p nodes, el tamany del problema ha de créixer en una relació $O(p^{KT_e/T_c}\ \log\ p)$ per mantenir l'eficiència del sistema.

Finalment, el tamany del problema que ens permetrà mantenir l'eficiència constant dependrà de la relació de valors de T_e i T_c i de K i haurà de crèixer amb $O(p^M\ \log\ p)$ sent $M = Max(1, KT_e/T_c)$.

De les expressions 5.2 i 5.3 es pot concloure que els valors T_e i T_c tenen un gran pes a l'hora de determinar el grau d'escalabilitat del sistema. En particular, quan $KT_e/T_c < 1$, el creixement de la mida del problema ve determinat per l'expressió 5.2. Per una altra part, tenint en compte que $K = E/(1-E)$, per mantenir l'eficiència per sota d' $E = T_c/(T_c + T_e)$ es requereix que W creixi en $O(p\ \log\ p)$, que pot considerar-se un creixement moderat. Mentre que quan $KT_e/T_C > 1$, el creixement de W ve determinat per l'expressió 5.3. Per mantenir una eficiència més gran que $E = T_c/(T_c + T_e)$ es necessari que W tingui un creixement $O(p^{KT_e/T_c} log\ p)$. En aquest cas podríem dir que el sistema és poc escalable.

També pot ocòrrer que l'eficiència no pugui mantenir-se. Això passa quan el problema necessari per mantenir l'eficiència no es pot resoldre en el sistema per falta de memòria. En particular, podria tindre's en compte que la mida de l'FFT més gran que pot resoldre's en un hipercub amb 2^d nodes amb una capacitat de memòria de 2^s paraules (reals de doble precisió) per node és 2^{d+s-1}. De l'expressió 5.3 es dedueix que l'eficiència es pot mantenir tan sols quan es compleix:

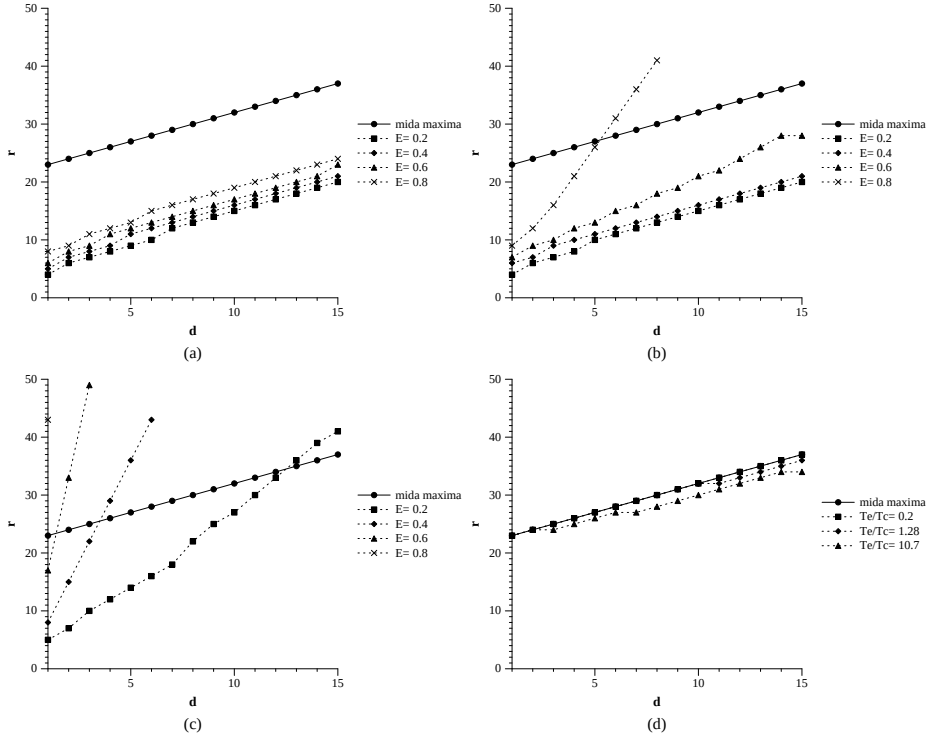


Figura 5.1 Tamany de l'FFT (2^r) necessari per mantenir constant l'eficiència a E quan s'incrementa el nombre de nodes del sistema (2^d)

$$d \leq \frac{s-1}{K(T_e/T_c)-1} \tag{5.4}$$

Per il.lustrar aquestes conclussions, en la figura 5.1 es mostren alguns resultats que s'han obtingut dels models analítics que s'han descrit en aquesta secció. En les gràfiques es mostra el tamany de la FFT que es necessita per mantenir l'eficiència donat un nombre de nodes. En particular, es mostren en l'eix X els valors de d i els valors de r en l'eix Y (en lloc del tamany del problema, que es calcularia a partir de l'expressió $r2^r$). S'han considerat quatre valors diferents de l'eficiència: $E = 0.2$, 0.4, 0.6 i 0.8. els valors de K considerats són $0,25$, $0,66$ $1,5$ i 4. Finalment es consideren tres valors diferents per la relació T_e/T_c que són: $0,2$, $1,28$ i $10,7$. Aquests són els valors utilitzats a [27] i que es corresponen als multicomputadors $nCUBE1$, $nCUBE2$ i $iPSC/RX$ respectivament.

En cada gràfica, una de les curves (mida-màxima) mostra la mida de l'FFT més gran que es pot resoldre en un sistema suposant una memòria per node de $2^s = 2^{24}$ nombres reals per node (128 MBytes per node). Aquesta curva ens permet determinar de forma ràpida i clara els casos en que l'eficiència no es pot mantenir ja que el problema que es necessita no cap en la memòria del sistema.

En la figura 5.1(a) es mostra el cas en què $T_e/T_c = 0,2$. En aquest cas, l'eficiència del sistema es manté amb un creixement d'r proporcional a d, ja que $KT_e/T_c < 1$ per tots els valors de K.

En la figura 5.1(b) es pot observar que, quan $T_e/T_c = 1.28$ i $K = 4$ ($E = 0.8$) el sistema és poc escalable ($KT_e/T_c = 5.12$). A més a més, per valors de d més grans que 5 l'eficiència no pot mantenir-se, tal i com es pot comprovar en l'expressió 5.4. Per la resta de valors d'E el sistema és escalable.

Finalment, en la figura 5.1(c) es veu que quan $T_e/T_c = 10.7$ el sistema es escalable tan sols per $E = 0.2$ i tan sols per valors de $d \geq 13$.

És important assenyalar que variant el tamany de la memòria per node no s'obté cap conclusió nova de l'anàlisi.

5.2.3 Anàlisi de l'escalabilitat aplicant el mètode limitat pel temps d'execució

En aquesta secció es proposa un mètode d'anàlisi d'escalabilitat alternatiu de l'algorisme $binary-exchange$ en un hipercub, utilitzant el model d'escalat limitat pel temps d'execució descrit a [31] i el tamany del problema com la figura de mèrit. Aquest mètode és un exemple particular del que hem anomenat un mètode d'anàlisi de l'escalabilitat orientat a l'usuari. Aquest es un dels mètodes que es descriuran formalment en la secció següent.

En el model d'escalat limitat pel temps d'execució, quan s'incrementa el nombre de nodes l'usuari incrementa la mida del problema tot el possible sempre i quan no s'ex-

cedeixi un temps d'execució prefixat T. L'anàlisi de l'escalabilitat determina fins quan es pot incrementar el tamany del problema. Sota aquest model, un sistema és escalable si l'increment del tamany del problema és proporcional a l'increment del nombre de nodes.

En l'anàlisi de l'escalabilitat que segueix, s'assumeix que el temps de referència T és el temps que es requeriex per resoldre el problema més gran possible en un únic node. Suposant que la memòria d'un node té una capacitat de $R = 2^s$ nombres reals, el problema més gran (FFT) que es pot resoldre té un tamany igual a 2^{s-1} i per tant el temps T és igual a:

$$T = (s-1)2^{s-1}T_c$$

Si tenim un sistema amb 2^d nodes, el tamany ($n = 2^r$) de l'FFT més gran que pot resoldre's sense excedir un temps T es pot calcular a partir de l'expressió:

$$T_p = T \rightarrow$$

$$r2^{r-d}T_c + dT_s + d2^{r-d}T_e = (s-1)2^{s-}T_c \tag{5.5}$$

La restricció de W degut al terme $r2^{r-d}T_c$ en 5.5 ve donat per la següent expressió:

$$W = r2^r = (s-1)2^{d+s-1} = O(p) \tag{5.6}$$

El terme dT_s en l'expressió 5.5 no suposa cap restricció a W.

La restricció de W degut al terme $d2^{r-d}T_e$ en l'expressió 5.5 ve donat per la següent expressió:

$$2^r = \frac{(s-1)2^{d+s-1}T_c}{dT_e} \rightarrow$$

$$r = d + s - 1 + \log_2 \frac{(s-1)T_c}{dT_e} \rightarrow$$

$$W = r2^r = (s-1)2^{s-1}\frac{T_c}{T_e}2^d(1 - \frac{L(d)}{d}) \tag{5.7}$$

on:

$$L(d) = \log \frac{dT_e}{(s-1)T_c} + 1 - s$$

L'expressió 5.6 es correspon al cas ideal (escalabilitat perfecta) en el que quan s'incrementa el nombre de nodes del sistema hi ha un increment proporcional del tamany del problema sense passar del temps màxim d'execució T.

L'expresió 5.7 reflexa quina és la diferència entre el cas ideal i el cas real. Quan $L(d) < 0$ estem en el cas ideal en el que $W = O(p)$. Aquest cas passa quan:

$$d \leq \frac{(s-1)T_c}{T_e}2^{s-1} \tag{5.8}$$

Per una altra banda, quan $L(d) > 0$, això és quan l'expressió 5.8 no es compleix, no és possible incrementar la mida del problema de forma proporcional a l'increment de

nodes del sistema. En aquest cas, també tenim un límit en el nombre de nodes en que podem incrementar el nostre sistema. Per avaluar aquest límit, hem de tenir en compte què en un sistema amb 2^d nodes amb una capacitat de memòria per node de 2^s nombres reals, el problema més petit (FFT) que pot resoldre's té una longitud de 2^{d-s+1}. Per un cert valor de d, el càlcul de l'FFT més petita pot tenir un temps més gran que T. Aquest valor de d limita la mida del sistema. Aquest límit es pot obtenir a partir de l'expressió:

$$d + s - 1 + \log \frac{(s-1)T_c}{dT_e} \geq d - s + 1$$

La part esquerra de la inequació correspon al tamany de l'FFT que té un temps T en el sistema paral.lel. Aquest tamany ha de ser més gran o igual que el tamany més petit del problema (FFT). En cas contrari, el tamany més petit de la FFT tindria un temps de resolució més gran que T. Partint d'aquesta expressió es pot obtenir el següent límit de d:

$$d \leq \frac{(s-1)T_c}{T_e} 2^{2s-2} \tag{5.9}$$

En qualsevol cas, es pot observar que per un rang raonable de valors considerats pels paràmetres d, T_e/T_c i s, la condició 5.8 i 5.9 sempre es compleixen amb un valor de $W = O(p)$. Per tant podem considerar que el sistema és escalable en qualsevol dels casos.

En la gràfica 5.1(d) es mostren aquestes conclusions.

Per concloure aquesta secció, hem mostrat que l'anàlisi de l'escalabilitat amb els mètodes de la *isoeficiència* i el mètode limitat pel temps d'execució s'obtenen resultats molt

diferents respecte a l'escalabilitat d'un mateix sistema. Per una altra banda, el resultat que s'obté de l'anàlisi limitat pel temps és més rellevant per l'usuari. La definició de mètodes d'anàlisi de l'escalabilitat de sistemes que puguin ser d'interès per l'usuari final del sistema són la principal motivació de la metodologia de l'escalabilitat que es descriu en la secció següent.

Per tots els mètodes que es proposen s'ha escollit una combinació dels dos components, anàlisi de l'escalabilitat i figura de mèrit, tenint en compte les prioritats i necessitats dels usuaris del sistema. D'aquesta manera, la informació que s'obté de l'anàlisi amb aquests mètodes, ajuda a l'usuari a decidir si és interessant i rendible incrementar el nombre de nodes del seu sistema. Es proposa una manera comú a tots els mètodes, de caracteritzar l'escalabilatat que permet una anàlisi ràpida i global de l'evolució del sistema.

5.3 METODOLOGIA D'ANÀLISI DE L'ESCALABILITAT

5.3.1 Caracterització d'un sistema paral.lel

En l'entorn d'aquest treball, un sistema paral.lel s'ha definit com una màquina paral.lela que resol un problema aplicant un algorisme paral.lel específic. El sistema paral.lel es caracteritza per una sèrie de funcions que es construeixen a partir dels paràmetres del sistema i dels paràmetres del problema. Les funcions es descriuen a continuació.

En aquest treball es consideren dintre de les màquines paral.leles els multicomputadors (formats per un conjunt de nodes connectats per una xarxa d'interconnexió). Un multicomputador es caracteritzarà a partir dels paràmetres T_c, T_s i T_e definits en el capítol 1. A més a més, considerarem R com la mida de la memòria per node del sistema, mesurada en nombre de valors en coma flotant que pot emmagatzemar. Suposem que no existeix memòria virtual. Recordant el que s'ha dit ja en el capítol 1, un node inverteix un temps igual a $T_s + N \times T_e$ per enviar un missatge amb N números en coma flotant a qualsevol altre node del sistema (*wormhole*).

El problema que s'ha de resoldre es caracteritza pel paràmetre n, que fa referència a la mida de les estructures d'entrada (per exemple, el tamany de la matriu $n \times n$). La mida del problema, $W(n)$, es defineix com el nombre d'operacions en punt flotant necessàries per resoldre un problema amb un tamany de dades igual a n. En aquest treball, suposem que la mida del problema depèn exclusivament d'n i, per tant, incrementar el tamany del problema és equivalent a incrementar el tamany de les dades. Altres autors han assenyalat que l'increment de la mida del problema pot dependre d'altres paràmetres diferents a la quantitat de dades [58], [5]. Així, per exemple, uns altres paràmetres que podrien tenir-se en compte a l'hora d'incrementar la mida del problema serien la precisió del resultat o el tamany dels intervals. Encara que no s'ha realitzat en aquest treball, sembla bastant factible extendre les propostes realitzades a una concepte més general de la mida del problema.

El sistema paral.lel es caracteritza per cinc funcions que es descriuen a continuació:

- $T(p, n)$ $\Rightarrow$ Temps per resoldre el problema amb un tamany de les dades igual a n en un sistema amb p nodes.
- $\Gamma(n)$ $\Rightarrow$ Màxim nombre de nodes que podem utilitzar en el sistema paral.lel per resoldre un problema amb un tamany de n.
- $M(n)$ $\Rightarrow$ Mínim nombre de nodes que es requereixen per poder resoldre un problema amb un tamany de dades igual a n.
- $\Gamma^{-1}(p)$ $\Rightarrow$ Tamany de dades del problema més petit que pot resoldre's en un sistema amb p nodes.
- $M^{-1}(p)$ $\Rightarrow$ Tamany de dades del problema més gran que pot resoldre's en un sistema amb p nodes.

5.3.2 Caracterització d'un mètode d'anàlisi orientat a l'usuari final del sistema

Un mètode orientat a l'usuari es caracteritza bàsicament per dos components: el model d'escalat i la figura de mèrit.

- El model d'escalat defineix la forma en què l'usuari utilitza un nombre creixent de nodes. Per exemple, l'usuari pot incrementar la mida del sistema amb l'objectiu de resoldre un problema de mida fixa més ràpidament (escalat de mida problema fix [36]). Alternativament, l'usuari pot incrementar el tamany del sistema amb l'objectiu d'incrementar al màxim possible el tamany del problema per tal d'aconseguir resultats més precisos (escalat de tamany de memòria fix [58]).
- La figura de mèrit és la mètrica que l'usuari vol millorar quan s'incrementa el tamany del sistema. La figura de mèrit està molt relacionada amb el model d'escalat.

L'evolució de la figura de mèrit determina l'escalabilitat d'un sistema quan s'incrementa la mida del sistema depenent del model d'escalat que s'apliqui. Qualsevol mètode d'anàlisi de l'escalabilitat hauria de proporcionar un criteri per determinar si un sistema és més o menys escalable. La metodologia que es proposa en aquest treball aplica un criteri específic per determinar l'escalabilitat del sistema, el qual es descriu a continuació.

Definim $\Delta F(p, m)$ com:

$$\Delta F(p, m) = \frac{F((1+m)p) - F(p)}{F(p)}$$

On $\Delta F(p)$ és l'increment de la figura de mèrit quan es parteix d'un sistema amb p nodes i s'incrementa a $(1+m)p$ nodes (suposa un increment del $m * 100\%$). Per exemple, si $\Delta F(100, 0.8) = 0.3$, significa que la figura de mèrit $F(p)$ s'ha incrementat un 30% quan passem d'un sistema amb 100 nodes a un nou sistema amb 180 nodes, o sigui un increment del 80% en la mida del sistema.

L'escalabilitat d'un sistema es caracteritza amb la funció $H(p, m)$, que es defineix com:

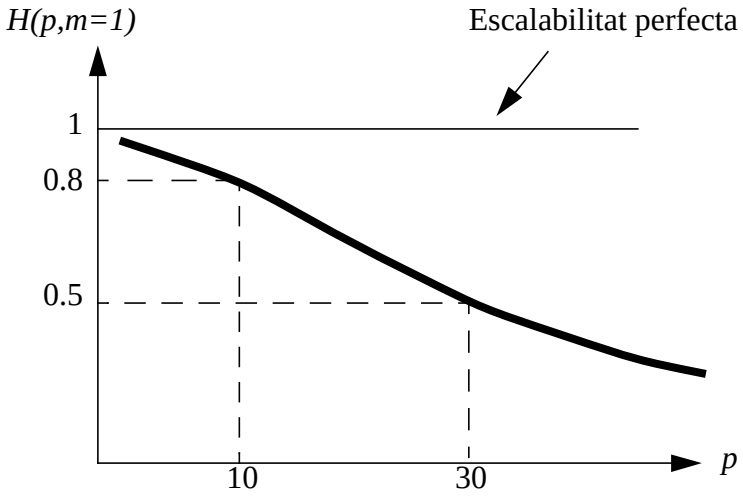


Figura 5.2 Un exemple de la funció $H(p,m)$, per un valor fix de m

$$H(p,m) = \frac{\Delta F(p,m)}{m}$$

De nou, com a exemple, si $\Delta F(100, 0.8) = 0.3$ llavors $H(100, 0.8) = 0.375$. La funció $H(p,m)$ relaciona l'increment de la figura de mèrit amb l'increment del nombre de nodes. Quan $H(p,m) = 1$ (cas ideal) significa que l'increment del sistema s'amortitza totalment en termes d'increment de la figura de mèrit. Hi ha la possibilitat que $H(p,m) > 1$, que reflexaria una situació de superescalabilitat. Per últim, $H(p,m) < 0$ indica que l'increment en el número de nodes del sistema implica una reducció en la figura de mèrit. Una anàlisi ràpida de la funció $H(p,m)$ ens dóna una visió general de l'escalabilitat del nostre sistema.

En l'exemple de la figura 5.2, la gràfica de $H(p,m)$ mostra quin impacte té en la figura de mèrit el fet de doblar el nombre de nodes del sistema. Si passem de 10 a 20 nodes hi ha un increment del 80% de la figura de mèrit. Mentre que si passem de 30 a 60 nodes la figura de mèrit s'incrementa tan sols en un 50%.

La caracterització de l'escalabilitat es proposa amb la idea de proporcionar una visió ràpida i clara del comportament del sistema a l'hora d'escalar. Això s'obté projectant

la funció $H(p, m)$ en un espai tridimensional. L'escalabilitat perfecta ve representada per una superfície plana amb una alçada igual a 1. Quan la superfície cau a partir d'un cert nombre de nodes indica que l'escalabilitat decreix, o sigui, s'està obtenint menys benefici del màxim possible.

És quan es representa la funció $H(p, m)$ quan les funcions $\Gamma(n)$, $M(n)$, $\Gamma^{-1}(n)$ i $M^{-1}(n)$ entren en joc. Aquestes funcions limiten el nombre de nodes i el tamany del problem. En particular, d'acord a les definicions que s'han donat de les funcions es construeixen els següents límits:

$$\begin{aligned} \Gamma^{-1}(p) \leq \quad & n \quad \leq M^{-1}(p) \\ M(n) \leq \quad & p \quad \leq \Gamma(n) \end{aligned}$$

La primera expressió indica que el tamany del problema, n, no pot ser mai més petit que el problema més petit que es pot resoldre amb un sistema amb p nodes i no pot ser mai més gran que el tamany del problema més gran que es pugui executar en un sistema amb p nodes, considerant en qualsevol dels casos que el tamany de memòria per node és igual a R.

La segona expressió indica que el nombre de nodes, p, no pot ser mai més petit que el mínim nombre de nodes necessaris per resoldre un problema de tamany igual a n i que el nombre de nodes no pot ser mai més gran que el nombre de nodes màxim que poden utilitzar-se per resoldre un problema de tamany igual a n.

Amb aquestes restriccions podria passar que la funció $H(p, m)$ no es pugui representar per alguna parella de valors (p, m). Aquesta situació indica que el sistema no pot escalar més enllà d'un nombre de nodes. Donat un model d'escalat, podria ser que algunes de les restriccions no es puguin aplicar; per exemple, com veurem més tard, en els tres models exemples que es descriuen a continuació, les restriccions de p tan sols afecten a l'escalabilitat del sistema quan es considera el model limitat per la mida del problema.

5.3.3 Tres exemples de mètodes d'anàlisi d'escalabilitat orientats a l'usuari

A continuació es descriuen tres models diferents d'anàlisi de l'escalabilitat. Tots ells es basen en la metodologia descrita en la secció anterior. Cadascun dels mètodes defineix una manera diferent d'utilitzar un nombre creixent de nodes del sistema. Per cada mètode es descriu el model d'escalat i la figura de mèrit aplicada. Per tots ells, la funció $H(p.m)$ caracteritza l'escalabilitat, tenint en compte que les restriccions de p i n són diferents per cada cas.

Tamany de Problema Fix

En aquest cas, se suposa que el sistema s'utilitza per resoldre un problema de tamany fix (n) lo més ràpid possible (model d'escalat: problema de tamany fix [58]. La figura de mèrit en aquest cas és l'$speed-up$, que es defineix com:

$$F(p) = S(p,n) = \frac{T(1,n)}{T(p,n)} = \frac{W \times T_c}{T(p,n)}$$

on $T(p,n)$ és el temps que es necessita per resoldre un problema de tamany n utilitzant un sistema amb p nodes.

Amb aquest mètode d'escalabilitat, el nombre de nodes té les seguents limitacions:

$$M(n,R) \leq p \leq T(n)$$

Aquesta expressió significa que el nombre de nodes del sistema ha de ser més gran o igual que el nombre de nodes mínim necessari per resoldre un problema de tamany n. Per una altra banda, el nombre de nodes no pot superar el nombre màxim de nodes que pot utilizar-se per resoldre un problema de tamany n.

Temps Fix

En aquest cas, quan s'incrementa el nombre de nodes, l'usuari incrementa el tamany del problema tot el possible tenint en compte que no pot superar un temps T en la seva resolució (model d'escalat: limitat pel temps).

Definim $n_T(p)$ com el tamany del problema que tarda un temps T en resoldre's en un sistema amb p nodes. Això s'expressa com:

$$T_p(n_T(p)) = T$$

En general, és possible que la quantitat de memòria necessària per un problema de tamany igual a $n_T(p)$ sigui més gran que la quantitat de memòria total del sistema. Definim, doncs, $n_L(p)$ com el tamany del problema més gran que pot resoldre's en un sistema amb p nodes, que el temps de resolució no sigui més gran de T i que càpiga en la memòria disponible del sistema:

$$n_L(p) = min(M^{-1}(p, R), n_T(p))$$

La figura de mèrit amb aquest mètode es defineix com:

$$F(p) = W(n_L(p))$$

S'ha de tenir en compte que $n_L(p)$ ha de satisfer la següent condició:

$$T^{-1}(p) \leq n_L(p)$$

Aquesta condició no es satisfà quan el tamany del problema més petit que pot resoldre's en un sistema amb p nodes necessita un temps més gran que T. Aquesta condició estableix un límit en el nombre de nodes del sistema amb aquest model d'escalat. És important veure que la restricció imposada per $M^{-1}(p)$ s'ha tingut en compte en l'operador Min a l'hora de calcular $n_L(p)$.

Tamany de memòria Fixa

En aquest cas, quan s'incrementa el nombre de nodes l'usuari incrementa el tamany del problema tot el possible, sempre i quan no sobrepassi el tamany de la memòria total del sistema (model d'escalat: limitat pel tamany de la memòria). La figura de mèrit, $F(p)$, és simplement la mida del problema més gran que pot resoldre's en un sistema amb p nodes, amb un tamany de memòria d'R per nodes:

$$F(p) = W(M^{-1}(p, R))$$

El problema més gran ha de ser més gran o igual que el problema més petit que pot resoldre's en un sistema amb p nodes. En cas contrari, l'increment en el nombre de nodes del sistema no permet incrementar el tamany del problema. De nou, aquesta condició estableix un límit en el nombre de nodes del sistema. D'aquesta restricció s'obté l'expressió:

$$M^{-1}(p, R) \geq T^{-1}(p)$$

5.4 EXEMPLE D'APLICACIÓ DE LA NOVA METODOLOGIA

En aquesta secció es presenta un exemple d'aplicació de la metodologia proposada en aquest capítol per tal de fer una anàlisi de l'escalabilitat d'un sistema que calcula els

valors i vectors propis d'una matriu simètrica real. El sistema utilitza una línia de nodes i implementa el mètode de Jacobi *one − sided*. Aquest ja s'ha descrit en el primer capítol de forma detallada; a continuació es recorden alguns detalls d'aquest sistema.

És important assenyalar que no ens interessa realitzar una anàlisi exahustiva de l'escalabilitat d'aquest sistema; amb aquest exemple s'intenta il.lustrar que la metodologia proposada proporciona una informació molt clara del comportament del sistema quan aquest s'escala.

5.4.1 Descripció del sistema

El sistema calcula els valors i vectors propis d'una matriu, A, matriu real i simètrica de dimensió $n \times n$. La mida del problema (nombre d'operacions per escombrat) es pot expressar com:

$$W(n) = 9(n^3 - n^2)$$

En el sistema paral.lel, les matrius A i U es descomposen en $2p$ blocs de $n/(2p)$ columnes consecutives. Inicialment, a cada node del sistema se li assignen una parella de blocs d' A i els corresponents blocs de la matriu U. Cada escombrat consisteix en $2p-1$ etapes. En cada etapa, cada node realitza un nombre de transformacions sense necessitar cap tipus d'intercanvi entre els altres nodes de la línia. Al final de cada etapa, cada node intercanvia un bloc d'A i el corresponent bloc d'U amb un dels nodes de la línia. Aquesta comunicació es pot realitzar de forma simultània sense cap tipus de conflicte en els enllaços de la línia.

5.4.2 Caracterització del sistema

Les cinc funcions que caracteritzen el sistema són les següents:

- $T(p,n) = \frac{9(n^3-n^2)}{p} \times T_c + (2p-1)(T_s + \frac{n^2}{p} \times T_e)$

  En el primer terme d'aquesta expressió es pot veure que el càlcul d'un escombrat està perfectament distribuït entre els p nodes del sistema. El segon terme correspon al cost de la comunicació (intercanviar un bloc d'A i d'U) al final de cada etapa d'un escombrat.

- $\Gamma(n) = \frac{n}{2}$ i $\Gamma^{-1}(p) = 2p$

  Per obtenir aquestes expressions s'ha considerat que cada node tan sols té assignades dues columnes de la matriu A i les mateixes dues columnes de la matriu U, necessàries per què cada node pugui calcular i aplicar una única transformació en cada etapa.

- $M(n) = \frac{2n^2}{R}$ i $M^{-1}(p) =$ Min $(\sqrt{\frac{pR}{2}}, \frac{R}{4})$.

  Per obtenir aquestes expressions s'ha tingut en compte que les matrius A i U requereixen $2n^2$ elements (reals en coma flotant) de memòria i R és la quantitat de memòria per node del sistema. El segon terme de l'operador Min en $M^{-1}(p)$ s'ha obtingut tenint en compte que un node com a mínim ha d'emmagatzemar un nombre de 4 columnes.

5.4.3 Tamany de problema fix

Considerem el problema amb un tamany $n = M^{-1}(1)$ (el problema més gran que pot resoldre's en un únic node). Les gràfiques que corresponen a l'anàlisi de l'escalabilitat amb un model d'escalat de tamany de problema fix es mostren en la figura 5.3. En qualsevol cas, $T_c = 1$, $R = 2^{20}$ i m pren valors entre $[0,1]$, o sigui, un increment de fins a un 100%.

La gràfica de la figura 5.3(a) mostra el cas en què $T_e = 1$ i $T_s = 1000$. La gràfica mostra que l'escalabilitat del sistema està limitada pel nombre de nodes ja que $H(p,m)$ no està definida per valors de p més grans de 300. Pels punts en què el sistema està definit, el sistema escala bastant bé, ja que la funció $H(p,m)$ pren valors pròxims a 1. En la gràfica de la figura 5.3(b) es mostra com influeix en l'escalabilitat del sistema l'increment de T_e. En aquest cas, es pot observar com la superfície baixa ràpidament. Finalment, en

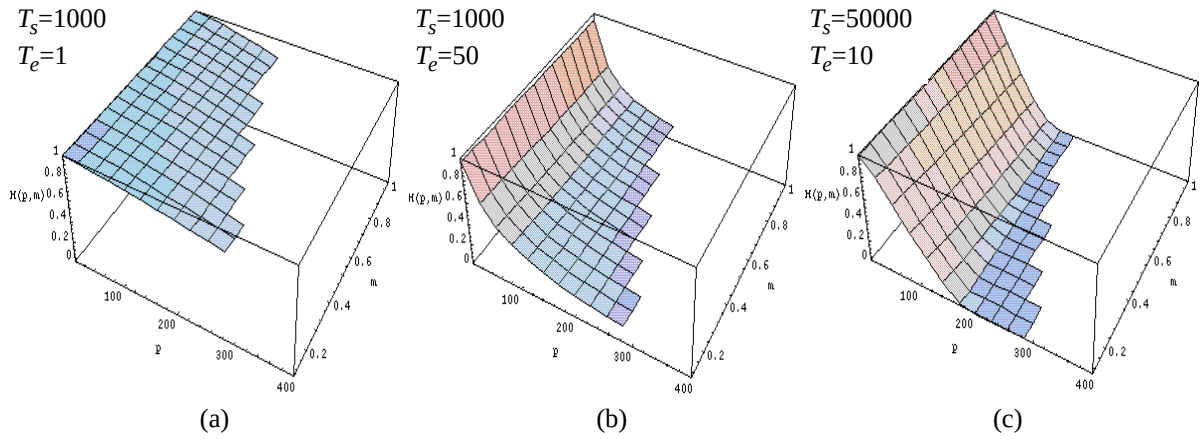


Figura 5.3 Anàlisi de l'escalabilitat amb un model d'escalat de tamany de problema fix per sistemes amb diferents valors de T_e i T_s.

la gràfica de la figura 5.3(c) es considera el cas en què $T_e = 10$ i $T_s = 50000$. En aquest cas, la funció $H(p, m)$ decreix per sota 0 per sistemes amb més de 200 nodes, indicant una reducció de la figura de mèrit. Això s'indica per una superfície d'alçada 0 (els valors negatius d'$H(p, m)$ es posen a 0 quan es mostra la gràfica).

5.4.4 Temps fix

Les gràfiques corresponents a l'anàlisi d'escalabilitat amb un model d'escalat de temps fix són 5.4. El temps de referència pot escollir-se de forma arbitrària, en aquest cas el temps T s'ha escollit com:

$$T = W(M^{-1}(1)) \times T_c$$

Aquest és el temps necessari per resoldre en un únic node el problema més gran possible que cap a la memòria d'un únic node. De nou, $T_c = 1$ i $R = 2^{20}$ en tots els casos. En la gràfica de la figura 5.4(a) es mostra l'escalabilitat del sistema quan $T_e = 1$ i

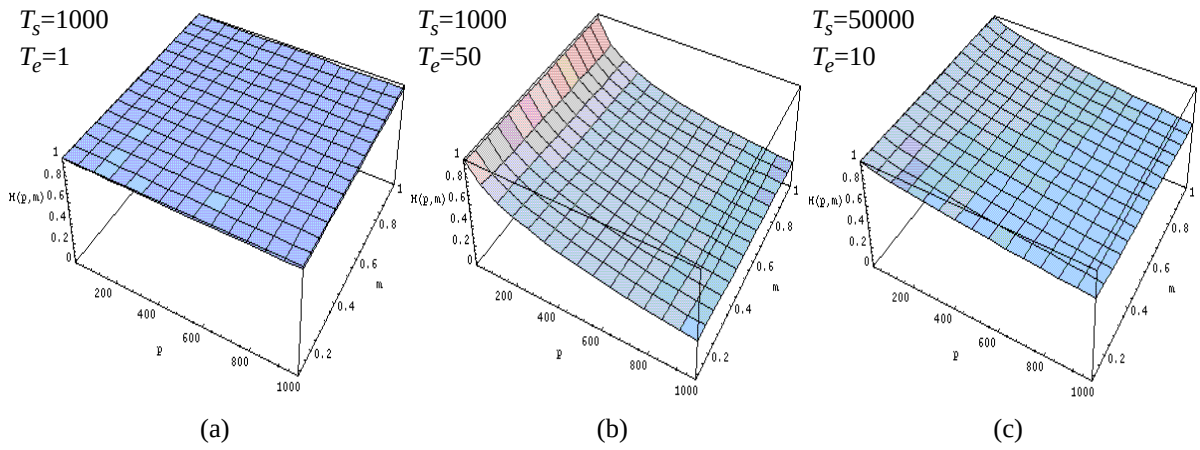


Figura 5.4 Anàlisi de l'escalabilitat amb un model d'escalat de temps fix per sistemes amb diferents valors de T_e i T_s.

$T_s = 1000$. El sistema és perfectament escalable pel rang de nodes que s'ha considerat (fins 1000). L'impacte d'incrementar el valor de $T_e = 50$ es mostra en la gràfica de la figura 5.4(b). L'escalabilitat decreix però es pot obtenir un increment de la figura de mèrit per un nombre de nodes bastant gran. Finalment, en la gràfica de la figura 5.4(c), que correspon als valors $T_e = 10$ i $T_s = 50000$, es veu que la figura de mèrit és poc sensible per valors grans de T_s, ja que la superfície es manté propera a 1 en tots els casos considerats.

És important assenyalar que les restriccions en el nombre de nodes i en el tamany del problema tindrien impacte en l'escalabilitat en el cas de considerar sistemes més grans.

5.4.5 Memòria fixa

La gràfica de la figura 5.5 mostra els resultats que s'han obtingut en l'anàlisi d'escalabilitat amb un model d'escalat limitat per la memòria. En aquest cas, els valors de T_c, T_s i T_e no són rellevants ja que la figura de mèrit depèn exclusivament de la quantitat de memòria en el sistema (considerem que $R = 2^{20}$ com en els apartats anteriors). La gràfica mostra una situació de superescalabilitat ja que el valor de la funció $H(p, m)$

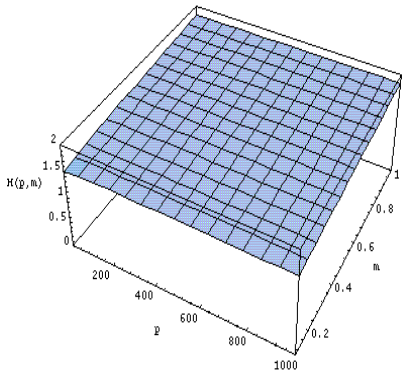


Figura 5.5 Anàlisi de l'escalabilitat amb un model d'escalat de memòria fixa.

està dintre del rang $[0, 2]$. Aquesta superescalabilitat s'atribueix al fet que mentre que els requeriments de memòria del sistema creixen amb n^2, la figura de mèrit $W(n)$ creix proporcional a n^3.

Respecte a les restriccions en el nombre de nodes i en el tamany del problema, igual que en el cas del model amb temps fix, el seu impacte es podria observar en sistemes amb un nombre de nodes molt més gran que els que s'han considerat.

5.5 RESUM DE CONTRIBUCIONS

En aquesta secció s'ha proposat una nova metodologia d'anàlisi de l'escalabilitat orientada a l'usuari final del sistema. S'ha realitzat una descripció formal de la metodologia:model d'escalat, figura de mèrit i caracterització de l'anàlisi (funció $H(p, m)$).

S'ha vist que aplicant la metodologia és possible obtenir diversos mètodes d'anàlisi de l'escalabilitat d'un sistema (segons els interessos concrets de l'usuari que realitza l'anàlisi) que proporcionen informació vàlida i rellevant a l'usuari del mateix i que permeten determinar quin és el benefici que s'obté quan s'incrementa el nombre de

nodes del sistema. Amb aquesta anàlisi l'usuari ha de ser capaç de decidir si és o no és rendible realitzar la inversió (incrementar el nombre de nodes).

A més a més s'ha proposat una manera de caracteritzar l'escalabilitat que ens permet veure ràpidament de forma visual quin és el comportament del sistema.

Relacionats amb aquest treball s'han realitzat les següents publicacions:

- "Un método para el Análisis de la Escalabilidad de Sistema Paralelos"; D. Royo, M. Valero-García i A. González; *VI Jornadas de Paralelismo*, 1995, Barcelona.
- "A methodology for User-Oriented Scalability Analysis"; D. Royo, M. Valero-García i A. González; *IEEE International Conference on Application Specific Systems, Architectures and Processors* , Juliol de 1997, Zurich.
- "A Framework for User-oriented Scalability Analysis"; D. Royo, C. Marí, M. Valero-García i A. González; *Report del DAC, UPC-DAC-1997-78* , 1997.

6

CONCLUSIONS I TREBALL FUTUR

En aquest treball s'han proposat un conjunt d'algorismes paral.lels per la resolució dels valors i vectors propis d'una matriu real i simètrica. S'han proposat algorismes per tres topologies diferents: hipercubs, malles multidimensionals i torus multidimensionals. Més concretament:

- S'han proposat un conjunt d'algorismes per hipercubs: *BR segmentat*, *α-optimal* i *Grau-4*. Algorismes que han donat lloc a tres noves ordenacions de Jacobi, *α minimum*, *BR-permutada* i *Grau-4*, que mitjançant l'aplicació de la tècnica de la segmentació de les comunicacions permeten explotar el paral.lelisme de les comunicacions d'una arquitectura en hipercub $multiple - port$.

  S'ha vist que amb aquests nous algorismes el cost de comunicació es redueix considerablement respecte al cost de comunicació de l'algorisme BR pel cas de tenir

una arquitectura *all-port*. Més exactament s'aconsegueixen reduccions en el cost de la comunicació de:

- fins un 50% amb l'algorisme BR segmentat.
- fins un 75% amb l'algorisme *Grau-4*. Aquest algorisme es pot aplicar a qualsevol tamany de problema.
- el cost de la comunicació es divideix per d (sent d la dimensió de l'hipercub) amb l'algorisme α-*optimal*. Aquest algorisme es pot aplicar tan sols a problemes amb un tamany gran.

- S'ha proposat un algorisme amb una topologia de comunicació en malla bidimensional, anomenat algorisme $2D$. Hem vist que quan aquest algorisme s'executa en multicomputadors amb una topologia en malla o torus multidimensional, per certs valors de T_e, T_s, T_c, d i m, té un cost total molt més petit que altres propostes que es poden trobar en la literatura. Aquesta anàlisi és vàlida per arquitectures *one-port* i *multiple-port*.

  L'algorisme s'ha avaluat en malles configurables, el qual ha permès determinar, donat un nombre de nodes 2^d, quina és la configuració òptima, $2^r \times 2^c = 2^d$, que minimitza el cost de la comunicació. Dels resultats de l'anàlisi s'ha vist que les configuracions òptimes són malles rectangulars, menys files que columnes. Aquesta conclusió pot ser bastant útil en el cas de arquitectures configurables, o en el cas de tenir que escollir subconjunts de nodes dintre un sistema.

  L'algorisme també s'ha avaluat per multicomputadors amb una configuració fixa regular de dues i tres dimensions, amb un nombre de files/columnes igual a una potència de 2.

  Dels resultats obtinguts es pot concloure que l'algorisme $2D$ té un cost més petit que altres algorismes que s'han proposat en la literatura (que degut a la seva topologia de comunicació els hem anomenat $1D$). La reducció del cost és més gran per relacions de T_e/T_c i T_c/T_s grans. Els resultats mostrats en el capítol 3 es pot veure que la reducció del cost varia entre un 20% a un 80% en configuracions $2D$ amb valors de $T_e/T_c \in [10, 100]$ i valors de $T_c/T_s \in [100, 2000]$. Pel cas de topologies $3D$ la reducció del cost varia entre un 40% i un 100% per valors de $T_e/T_c \in [10, 100]$ i valors de $T_c/T_s \in [100, 2000]$. En qualsevol cas, donat un nombre de nodes, els guanys són més significatius per matrius petites.

Els resultats que s'han obtingut en les malles configurables són molt semblants als resultats que s'han obtingut per malles i torus $3D$, això és degut a que la configuració òptima que s'ha obtingut en les malles configurables té un mapeig això és degut a que la configuració òptima que s'ha obtingut en les malles configurables té un mapeig directe en malles i torus $3D$, amb el que el cost afegit alhora de mapejar l'algorisme $2D$ en topologies d'aquest tipus és gairebé 0.

- S'ha proposat un nou algorisme que resulta del mapeig de l'algorisme *BR segmentat* en una malla i en un torus (aplicació i extensió de la metodologia *CALMANT*). S'ha vist que per malles i torus bidimensionals amb una arquitectura $multiple - port$ aquest nou algorisme pot tenir un rendiment millor que l'algorisme $2D$.

  En aquest cas, s'ha mostrat que per malles i torus amb una arquitectura $all - port$ de tamany petit (16 i 64 nodes) i de tamany mitjà (256 nodes), l'algorisme BR segmentat un cop mapejat en una malla o un torus bidimensional regular aplicant la metodologia CALMANT té un cost de comunicació de fins un 40% menys que el cost de la comunicació de l'algorisme $2D$ en les mateixes condicions. Aquest comportament s'accentua en el cas de matrius grans.

Tots els algorismes proposats en aquest treball s'han avaluat mitjançant models analítics construïts en funció dels paràmetres T_e, T_s, T_c, d i m. Dels resultats que s'han obtingut podem concloure que tots ells són algorismes prometedors i que seria interessant la seva implementació en màquines reals.

En l'última part del treball s'ha proposat una metodologia d'anàlisi de l'escalabilitat de sistemes paral.lels que proporciona informació vàlida a l'usuari final del sistema en funció dels seus requeriments, informació que li permet determinar si és interessant o no augmentar el nombre de nodes del sistema.

En aquest treball queden les següents línies obertes:

Respecte als algorismes proposats:

- Noves ordenacions

  - Burcar una ordenació equilibrada d'ordre més gran que 4.
  - Buscar una ordenació que sigui bona pels dos models d'implementació de la segmentació de comunicacions, *deep* i *shallow pipelining*.
  - Buscar una ordenació que sigui bona per resoldre el problema dels valors singulars. En aquest cas que a més a més de generar totes els aparellaments de columnes per completar un escombrat ordena les columnes de la matriu amb ordre creixent respecte a la seva norma.

- Algorisme *2D*
  - Generalitzar l'anàlisi de l'algorisme $2D$ a malles de qualsevol tamany; en el treball tan sols s'han considerat topologies regulars amb un nombre de files i columnes potència de 2.

- Mapeig dels algorismes per hipercubs en malles i torus multidimensionals
  - El en treball tan sols s'ha estudiat quin és el comportament de l'algorisme BR quan es mapeja en una malla/torus bidimensional.S'ha aplicat el mètode $CALMANT$ amb un grau de segmentació igual a 2. Interessaria doncs, fer un estudi extensiu per qualsevol valor de Q, per qualsevol algorisme proposat en el capìtol 2 i per malles/torus de qualsevol dimensió.

Respecte a la metodologia d'avaluació:

- Obtindre resultats en màquines reals dels algorismes proposats en el treball.
- Veure quins aspectes hi han que tenir en compte alhora d'executar els algorismes en sistemes tipus DSM i NOW.

Respecte a l'anàlisi de l'escalabilitat

- Desenvolupar alguna eina que ens permeti definir els models dels sistemes i que generi de forma automàtica les gràfiques corresponents a cada model d'escalat. A més a més que permeti la definició de nous models d'escalat.

- Buscar alternatives a la generació del model del sistema, per exemple, l'utilització de traces o bé del codi del programa que implementa l'algorisme que es vol avaluar). Actualment s'utilitzen models analítics.

A

ANNEX 1

A continuació es demostra que el valor del paràmetre α corresponent a la seqüència D_e^{p-BR}, per valors grans d'e, tendeix a $1,25$ cops el valor de la fita mínima que com recordarem s'expressava com $\lceil (2^e - 1)/e \rceil$.

Per simplificar les expressions suposarem que $e-1$ és potència de 2. Per qualsevol altre valor d'e, el valor d'α pren un valor intermig entre el valor d'α de les dues potències de 2 més pròximes a e en cadascun dels dos costats; el comportament asimptòtic que s'observa per valors potència de 2 també pot aplicar-se a qualsevol altre valor.

Sigui $e = 2^S + 1$. En aquest cas, la seqüència D_e^{p-BR} s'obté després d'aplicar S transformacions a la seqüència original D_e^{BR}. Per il.lustrar el desenvolupament següent utilitzarem com a exemple el cas particular $e = 17$. Aquest cas requereix 4 transformacions les quals es mostren a la figura A.1.

1st transformació	**2nd transformació**	**3rd transformació**	**4th transformació**
2nd 16-subsequència	2nd 15-subsequència	2nd 14-subsequència	2nd 14-subsequència
(0,15)	(0,7)	(0,3)	(0,1)
(1,14)	(1,6)	(1,2)	4th 14-subsequència
(2,13)	(2,5)	4th 14-subsequència	(2,3)
(3,12)	(3,4)	(4,7)	6th 14-subsequència
(4,11)	4th 15-subsequència	(5,6)	(6,7)
(5,10)	(8,15)	6th 14-subsequència	8th 14-subsequència
(6,9)	(9,14)	(12,15)	(4,5)
(7,8)	10,13)	(13,14)	10th 14-subsequència
	(11,12)	8th 14-subsequència	(14,15)
		(8,11)	12th 14-subsequència
		(9,10)	(12,13)
			14th 14-subsequència
			(8,9)
			18th 14-subsequència
			(10,11)

Figura A.1 Transformacions per generar D_{17}^{p-BR}. Cada parella fa referència a la transposició dels enllaços identificats pels mateixos números que constitueixen la tupla

Anomenarem $D_e^{p-BR}(k)$ a la seqüència que s'obté després de la transformació k, amb $k \in [0, S-1]$. En general, alguns dels elements de la seqüència $D_e^{p-BR}(k)$ es veuran afectats per altres transformacions que es realitzaran més tard, mentre que alguns altres ja no es veuran afectats per cap altra transformació més. Aquest segon grup d'elements els referenciarem com els elements que es fixen en la transformació k. Definirem $r_k(i)$ com el nombre de repeticions d'un enllaç i en la seqüència $D_e^{p-BR}(k)$ que han estat fixats després de la transformació k. Definirem $p_k(i)$ com el nombre de repeticions de l'enllaç i en la seqüència $D_e^{p-BR}(k)$ que es veuran afectats per futures transformacions.

Òbviament, el nombre de repeticions de l'enllaç i en la seqüència resultant D_e^{p-BR} és $r_0(i) + ... + r_{S-1}(i) + p_{S-1}(i)$. A partir d'ara, obtindrem les expressions analítiques

d'$r_k(i)$ i $p_k(i)$ que ens permetran realitzar l'anàlisi del comportament asimptòtic del valor d'α.

La primera transformació és una permutació que s'aplica a la segona subseqüència de dimensió $(e-1)$ i consisteix en una sèrie de transposicions tal i com s'han definit en el capítol 3. Les transposicions definides aparellen els elements (de 0 fins a $e-1$) de tal manera que l'element més freqüent s'intercanvia amb l'element menys freqüent; el segon element més freqüent s'intercanvia amb el segon element menys freqüent, i així fins que tots els elements queden aparellats. Com que la primera i segona subseqüències de dimensió $(e-1)$ són exactament iguals i la permutació tan sols s'aplica a la segona subseqüència, després d'aquesta primera permutació el nombre de repeticions de l'element que apareixia més cops es redueix a la meitat més la meitat del nombre de repeticions de l'element que apareixia menys cops abans d'aplicar la permutació (no es té en compte l'element central que tan sols apareix un sol cop en la seqüència i no és modificat per cap transformació). El nombre de repeticions de l'element menys freqüent s'incrementa en la mateixa quantitat de repeticions en què l'element més freqüent es redueix.

Qualsevol element $i \in [0,(e-1)/2-1]$ que apareix en la segona subseqüència de dimensió $(e-1)$ queda fixat després de la primera transformació. El mateix passa amb qualsevol element $i \in [(e-1)/2, e-2]$ que apareix en la primera subseqüència de dimensió $(e-1)$. A més a més, el nombre d'aparicions dels elements $i \in [0,(e-1)/2-1]$ en la segona subseqüència de dimensió $(e-1)$ és igual a $2^0, 2^1, ..., 2^{(e-1)/2-1}$ respectivament. El nombre d'aparicions dels elements $i \in [(e-1)/2, e-2]$ que apareixen en la primera subseqüència de dimensió $(e-1)$ és el mateix però en ordre invers.

El mateix esquema es repeteix un cop aplicada la segona permutació. En aquest cas, qualsevol element $i \in [0,(e-1)/2^2-1]$ de la segona subseqüència de dimensió $(e-2)$ queda fixat. El mateix passa en els casos següents:

- qualsevol element $i \in [(e-1)/2^2,(e-1)/2-1]$ en la primera subseqüència de dimensió $(e-2)$

- qualsevol element $i \in [(e-1)/2, 3(e-1)/2^2 - 1]$ en la tercera subseqüència de dimensió $(e-2)$
- qualsevol element $i \in [3(e-1)/2^2, e-2]$ en la quarta subseqüència de dimensió $(e-2)$

El nombre de cops que un element $i \in [0, (e-1)/2^2 - 1]$ en la segona subseqüència de dimensió $(e-2)$ queda fixat és igual a $2^{(e-1)/2-1}, ..., 2^{3(e-1)/2^2-2}$. Per a la resta d'intervals els elements es fixen amb el mateix nombre que en el segon interval però en alguns casos en ordre invertit.

Degut a la naturalesa recursiva de la seqüència D_e^{p-BR}, ens trobem que després de cada transformació $k \in [0, S-1]$, el nombre de repeticions dels elements que no pertanyen a l'interval $[0, (e-1)/2^{k+1}-1]$ que han estat fixats coincideix amb el nombre de repeticions dels elements que pertanyen a l'interval $[0, (e-1)/2^{k+1} - 1]$.

Per una altra banda, després de la primera transformació, qualsevol element que pertany a l'interval $[0, (e-1)/2 - 1]$ en la primera subseqüència de dimensió $(e-1)$ no serà modificat per cap altra permutació. Es compleix el mateix per qualsevol element que pertany a l'interval $[(e-1)/2, e-2]$ en la segona subseqüència. El nombre de repeticions dels elements del primer interval que apareixen en la primera subseqüència de dimensió $(e-1)$ és de $2^{e-2}, 2^{e-3}, ..., 2^{(e-1)/2}$ respectivament. El nombre de repeticions dels elements del segon interval que apareixen en la segona subseqüència de dimensió $(e-1)$ coincideix amb l'interval anterior però en ordre invertit. En general, qualsevol permutació que incrementi/decrementi el nombre de repeticions d'un element del primer interval, afectarà amb el mateix increment/decrement al mateix element en el segon interval ja que els dos intervals tenen el mateix nombre de repeticions.

De nou, degut a la naturalesa recursiva de la seqüència D_e^{p-BR}, ens trobem que després de cada transformació $k \in [0, S-2]$, el nombre de repeticions dels elements que no pertanyen a l'interval $[0, (e-1)/2^{k+1}-1]$ que seran modificats per futures permutacions coincideix amb el nombre de repeticions d'alguns altres elements j que pertanyen a $[0, (e-1)/2^{k+1} - 1]$. A més a més, el nombre de repeticions d'aquests dos elements i i

j s'incrementarà i decrementarà en la mateixa quantitat per qualsevol permutació que els afecti.

Com a resultat d'aquesta anàlisi podem assegurar que a l'hora d'obtenir les expressions d'$r_k(i)$ i $p_k(i)$ és suficient calcular-los pels elements del primer interval en cada transformació, $i \in [0, (e-1)/2^{k+1} - 1]$, ja que el valor que s'obté coincideix amb el valor d'un altre element considerat en un altre interval.

Lema 1.

$$p_k(i) = 2^{e-2-k-i} \qquad k \in [-1, S-1] \qquad i \in [0, \tfrac{e-1}{2^{k+1}} - 1] \tag{A.1}$$

Demostració. Per $k = -1$, o sigui, abans d'aplicar cap transformació, l'equació es compleix ja que coincideix amb el nombre de repeticions de cada dimensió en la seqüència D_e^{BR}. Qualsevol transformació k intercanvia cada dimensió $i \in [(e-1)/2^{k+1} - 1]$ amb la dimensió $(e-1)/2^k - 1 - i$, tan sols en la segona subseqüència de dimensió $(e-k-1)$.

Lema 2.

$$r_k(i) = 2^{e-\frac{e-1}{2^k}+i-k-1} \qquad k \in [0, S-1] \qquad i \in [0, \tfrac{e-1}{2^{k+1}} - 1] \tag{A.2}$$

Demostració. En la transformació k, qualsevol element i que pertany a l'interval $[0, (e-1)/2^{k+1} - 1]$ s'intercanvia amb l'element $(e-1)/2^k - 1 - i$ en la segona subseqüència de dimensió $(e-k-1)$. Per tant, el nombre de repeticions de l'element i en la segona subseqüència de dimensió $(e-k-1)$ que després d'aplicar la permutació queden en la mateixa subseqüència de dimensió $(e-k-1)$ és igual al nombre de repeticions de l'element $(e-1)/2^k - 1 - i$ en aquesta subseqüència abans de la permutació. Això és, $r_k(i) = p_{k-1}((e-1)/2^k - 1 - i)/2$, que s'obté a partir de l'expressió que es mostra en el lema.

Lema 3.

Després de la transformació k, el nombre màxim de repeticions de qualsevol element que no es veurà afectat per cap altra transformació és igual a:

$$N_k = 2^{e-\frac{e-1}{2^{k+1}}-k-2} \qquad k \in [0, S-1] \tag{A.3}$$

Demostració. És obvi que $N_k = \max r_k(i)$. Aplicant l'expressió del lema 2 s'obté l'expressió A.3.

Lema 4.

$$N_k \leq 2^{e-2S+k} \qquad k \in [0, S-2] \tag{A.4}$$

Demostració. Partint de la definició d'N_k que es dóna en el lema 3 amb l'expressió A.4, el lema 4 es pot demostrar amb la inequació següent:

$$\frac{e-1}{2^{k+1}} \geq 2S - 2k - 2 \tag{A.5}$$

Com que $e - 1 = 2^S$, la inequació A.5 es compleix si $2^{S-k-1} \geq 2S - 2k - 2$. El terme de l'esquerra creix més ràpid que el terme de la dreta quan decreix k, per aquest motiu és suficient per demostrar que la inequació és certa per valors de k grans. Si substituïm K per $S - 2$, s'obté el mateix valor en les dues parts de la inequació.

Lema 5.

$$\sum_{k=0}^{S-2} N_k \leq 2^{e-S-1} - 2^{e-2S} \tag{A.6}$$

Demostració. Aquesta expressió s'obté directament realitzant el sumatori de 0 fins a $S-2$ del terme N_k que s'ha obtingut en el lema 4.

Teorema 1.

El valor d'α corresponent a les seqüències D_e^{p-BR} està afitat per l'expressió següent:

$$\alpha < \frac{2^e}{e-1} + \frac{2^{e-2}}{e-1} - \frac{2^e}{(e-1)^2}$$

Demostració. El valor del paràmetre α està afitat pel sumatori dels termes següents: el valor màxim de $p_{S-1}(i)$,N_0,N_1,...,N_{S-1}. Així doncs, l'expressió que queda és:

$$\alpha \leq p_{s-1}(0) + N_{S-1} + \sum_{k=0}^{S-2} N_k$$

Si afegim el valor de $p_{S-1}(0)$ donat pel lema 1, el valor de N_{S-1} donat pel lema 3 i la fita superior donada en el lema 5, es compleix la fita màxima definida en aquest teorema.

Teorema 2.

La fita superior del paràmetre α que s'estableix en el teorema 1 tendeix a 1.25 cops la fita inferior donada per l'expressió $\lceil (2^e-1)/e \rceil$.

Demostració. Aquest teorema es demostra calculant el límit:

$$\lim_{e\to\alpha} \quad \frac{\frac{2^e}{e-1} + \frac{2^{e-2}}{e-1} - \frac{2^e}{(e-1)^2}}{\frac{2^e-1}{e}} =$$

$$= \lim_{e\to\alpha} \frac{\frac{2^e}{e-1}}{\frac{2^e-1}{e}} + \lim_{e\to\alpha} \frac{\frac{2^{e-2}}{e-1}}{\frac{2^e-1}{e}} + \lim_{e\to\alpha} \frac{\frac{2^e}{(e-1)^2}}{\frac{2^e-1}{e}} = 1 + 0.25 + 0 = 1.25$$

REFERÈNCIES

[1] W C. Athas and C. L. Seitz. Multicomputers: Message-passing concurrent computers. *IEEE Computer*, pages 9–24, agost 1988.

[2] M Berry and A. Sameh. Multiprocessor jacobi schemes for dense symmetric eigenvalue and singular value decompositions. In *Proceedings of ICPP 98*, pages 433, 440, 1986.

[3] C Bischof. The two-sided jacobi method on a hypercub. In *SIAM Proceedings of the Second Conference on Hypercub Multiprocessors*, 1987.

[4] R P. Brent and F. T. Luk. The solution of singular-value and symmetric eigenvalue problems on multiprocessor array. *SIAM J. Sci. Statist. Computing*, pages 69–84, 1985.

[5] D Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture. A Hardware/Software Approach.* Morgan Kaufmann Publishers INC., 1997.

[6] D E. Culler, R. M. Karp, D. Patterson, A. Sshsy, E. E. Santos, K. E. Schauser, R.Subramonian, and T.Von Eicken. Logp, a practical model of parallel computation. *Communications of the ACM*, 39(11):78–43, 1996.

[7] L Díaz de Cerio. *CALMANT: Un Método Sistemático para le ejecución de algorismos con topologia en hipercubo en multicomputadores.* PhD thesis, Universitat Politècnica de Catalunya, desembre 1998.

[8] L Díaz de Cerio, A. Gonzàlez, and M. Valero-García. Communication pipelining in hypercubes. *Parallel Processing Letters*, 6(4), desembre 1996.

[9] L Díaz de Cerio, M. Valero-García, and A. Gonzàlez. A study of the communication cost of the fft on torus multicomputers. *Report del DAC. UPC-DAC-94-25*, 1994.

[10] L Díaz de Cerio, M. Valero-García, and A. Gonzàlez. A systematic approach to developed efficient completed exchange algorithms for meshes and tori. *Report del DAC. UPC-DAC-97-29*, 1997.

[11] L Díaz de Cerio, M. Valero-García, A. Gonzàlez, and D. Royo. Calmant: un mètodo sistemàtico para la ejecución de algorismos con topologia en hipercubo en mallas y toros. *Report del DAC. UPC-DAC-97-36*, 1997.

[12] J J. Dongarra and T. Dunigan. *Message-Passing Performance of Various Computers.* Technical Report, University of Tennessee and Oak Ride National Laboratory, 1995.

[13] J Duato, S. Yalamanchili, , and L. Ni. *Interconnection Networks an Engineering Approach.* IEEE Computer Society Press, 1997.

[14] P J. Eberlein. On one-sided jacobi methods for parallel computation. *SIAM J. Algebraic Discrete Methods*, pages 790–796, 1987.

[15] P J. Eberlein and H. Park. Efficient implementation of jacobi algorithms and jacobi sets on distributed memory architectures. *Journal of Parallel and Distributed Computing*, (8):358–366, 1990.

[16] M Fienup and S. C. Kothari. Cmp: A memory-constrained scalability metric. In *7th. SIAM Conf. on Parallel Processing*, pages 848, 853, 1995.

[17] M A. Fienup and S. C. Kothari. A memory-constrained scalability metric. In *International Conference on Parallel Processing*, pages III–1, III–7, 1994.

[18] M J. Flynn. Very high-speed computing systems. In *Proceedings of the IEEE 54*, number 12, pages 1901–1909, desembre 1966.

[19] M J. Flynn. Some computer organizations and their effectiveness. In *IEEE Transaction on Computers*, number 9, pages 948–960, 1972.

[20] G Fox, M. Johnson, G. Lyzenga, J. Salmon, and D. Walke. *Solving Problems on Concurrent Processors, Volume I.* Prentice-Hall International Editions, 1988.

[21] G R. Gao and S. J. Thomas. An optimal parallel jacobi-like solution method for singular value decomposition. In *Proceedings of Int'l Conf. on Parallel Processing*, pages 47–53, 1988.

[22] Domingo Giménez, Vicente Hernández, , and Antonio M. Vidal. Efficient jacobi algorithms on multicomputers. *Lecture Notes in Computer Science*, (1041):247–256, 1996.

[23] Domingo Giménez, Vicente Hernández, Robert van de Geijn, and Antonio M. Vidal. A block jacobi method on a mesh of processors. *Concurrency: Practice and Experience*, 9(5):319–412, 1997.

[24] G H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore, MD: The John Hopkins Univ. Press, 1983.

[25] A Gonzàlez, M. Valero-García, and L. Díaz de Cerio. Executing algorithms with hypercube topology on torus multicomputers. *IEEE Trans. on Parallel and Distributed Systems*, 6(8):803–814, agost 1995.

[26] A Y. Grama, A. Gupta, and V. Kumar. Isoefficiency: Measuring the scalability of parallel algorithms and architectures. *IEEE Parallel and Distributed Technology*, pages 12–21, agost 1993.

[27] A Y. Grama, A. Gupta, and V. Kumar. The scalability of fft on parallel computers. *IEEE Transactions on Parallel and Distributed Systems*, 4(8):922–932, agost 1993.

[28] J P. Hayes and T. N. Mudge. Hypercube supercomputers. In *Proceeding of the IEEE*, number 12, pages 653–660, desembre 1989.

[29] J P. Hayes, T. N. Mudge, and Q. F. Stout. Architecture of a hypercube supercomputers. In *Proc. Int. Conf. on Parallel Processing*, pages 653–660, 1986.

[30] Mark D. Hill. What is scalability? *Computer Architecture News*, 18(4):18–21, 1990.

[31] K Hwang. *Advanced Computer Architecture, Parallelism, Scalability and Programmability*. Morgan Kaufmann Publishers INC., 1990.

[32] S L. Jhonsson and K. L. Krawitz. Cooley-tukey fft on the connection machine. *Parallel computing*, (18):1201–1221, 1992.

[33] A H. Karp. Programming for parallelism. *Computer*, pages 39–49, maig 1987.

[34] P Kermani and L. Kleinrock. Virtual cut-through: A new communication switching technique. *Computers Networks*, 3(4):267–286, 1979.

[35] V Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing. Design and Analysis Algorithms.* The Benjamin/Cummings Publishing Company, Inc., 1994.

[36] V Kumar and A. Gupta. Analysing scalability of parallel algorithms and architectures. *Journal of Parallel and Distributed Computing*, (22):379–391, 1994.

[37] M Lam. Software pipelining: an effective scheduling technique for vliw machines. In *Conference on Programming Language, Design and Implementation*, pages 318–328, 1988.

[38] F T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, Inc., 1992.

[39] L M. Li and P. K. McKinley. A survey of wormhole routing techniques in directed netwoks. *IEEE Computer*, 26(2):62–76, febrer 1993.

[40] F T. Luk. Architectures for computing eigenvalues and svds. *SPIE Vol. 614 Highly Parallel Signal Processing Architectures*, pages 24–33, 1986.

[41] F T. Luk and H. Park. A proof of convergence for two parallel jacobi svd algorithms. *IEEE Transactions on Computers*, 38(6):806–811, juny 1989.

[42] M Mantharam and P. J. Eberlein. Block-recursive algorithm to generate jacobi-sets. *Parallel Computing*, (19):481–496, 1993.

[43] M Mantharam and P. J. Eberlein. New jacobi sets for parallel computation. *Parallel Computing*, (19):437–454, 1993.

[44] W F. Mascharenhas. On the convergence of jacobi method for arbitrary orderings. *SIAM Journal Matrix ANAL. APPL.*, 16(4):1197–1209, octubre 1995.

[45] S Matic. Emulation of hypercub architecture on nearest-neighbor mesh-connected processing elements. *IEEE Trans. on Computers*, 39(5):698–700, maig 1990.

[46] P K. Mckinley, Y. Tsai, and D. F. Robinson. Collective communication in wormhole-routed massively parallel computers. *Computer*, pages 39–49, desembre 1995.

[47] nCUBE Corporation. *nCUBE 6400 Processor Manual.*

[48] L M. Ni and P. K. Mckinley. A survey of wormhole routing techniques in directed networks. *IEEE Computer*, 26(2):62–76, febrer 1993.

[49] D A. Patterson and J. L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface.* Morgan Kaufmann Publishers, Inc., 1998.

[50] J G. Peters and M. Syska. Circuit-switched broadcasting in torus networks. *IEEE Transactions on Parallel and Distributed Systems*, 7(3):39–49, març 1996.

[51] M J. Quinn. *Designing Efficient Algorithms for Parallel Computers.* McGraw-Hill Series in Computer Organization and Architecture, 1987.

[52] D Royo, A. Gonzàlez, and M. Valero-García. Jacobi orderings for multi-port hypercubes. In *12th. International Parallel Symposium and 9th. Symposium on Parallel and Distributed Processing*, pages 653–660, 1998.

[53] D Royo, A. Gonzàlez, and M. Valero-García. Low communication overhead jacobi algorithms for eigenvalue computation on hypercubes. *Acceptat pendent de publicació en el Journal on Supercomputing*, 1999.

[54] D Royo, M. Valero-García, and A. Gonzàlez. A jacobi-based algorithm for computing symmetric eigenvalues and eigenvectors in a two-dimensional mesh. In *Euromicro Workshop on Parallel and Distributed Processing*, pages 653–660, gener 1998.

[55] D Royo, M. Valero-García, A. González, and C. Marí. A methodology for user-oriented scalability analisys. In *IEEE Int. Conf. on Application Specific Systems, Architectures and Processors*, pages 304–315, juliol 1997.

[56] C L. Seitz. The cosmic cube. *Communications of the ACM*, 28(1):22–33, octubre 1985.

[57] G Shroff and R. Schreiber. On the convergence of the cyclic jacobi method for parallel block orderings. *SIAM Journal Matrix ANAL. APPL.*, 10(3):326–346, juliol 1989.

[58] J P. Singh, J. L. Hennessy, and A. Gupta. Scaling parallel programs for multiprocessors: Methodology and examples. *IEEE Computer*, pages 42–50, juliol 1993.

[59] J H. Wilkinson. *The Algebraic Eigenvalue Problem.* Claredon Press, 1965.

[60] X Zhang, Y. Yan, and K. He. Latency metric: an experimental method for measuring and evaluating parallel program and architecture scalability. *Journal of Parallel and Distributed Computing*, (22):392–410, 1994.

[61] B B. Zhou and R. P. Brent. A parallel ordering algorithm for efficient one-sided jacobi svd computations. *IEEE Parallel and Distributed Computing*, 42:1–10, 1997.