Descripció dels algorismes i estructures de dades

GENÍS RIERA PÉREZ (genis.riera.perez)

MAURICIO IGNACIO CONTRERAS PINILLA (mauricio.ignaci.contreras)

ALEXANDER GIMÉNEZ ORTEGA (alexander.gimenez)

Codi de projecte: 7.2

Versió del lliurament: 3.0

Índex

1. Prefaci

Aquest document descriu en detall tots aquells algorismes i estructures de dades que hem estudiat, implementat i aplicat durant el procés de desenvolupament del projecte, gràcies als quals hem pogut resoldre els problemes inherents a la bona realització del projecte.

Una vegada descrits aquells algorismes relacionats amb les estratègies de joc (la intel·ligència artificial pròpiament dit), els hem enfrontat entre ells mitjançant un banc de proves empíriques basat en la disputa de partides. Durant les proves hem enregistrat les dades obtinguts referents al guanyador de cada partida, amb quin color de fitxes ha disputat cada partida cada estratègia, els temps de resposta per torn de cada estratègia durant cada partida, i els temps de duració de cada partida i de la prova. Amb aquestes dades hem calculat tot un seguit de valors més informatius tals com la mitjana dels temps de resposta per torn per partida de cada estratègia, que ens ajudaran en la posterior comparació dels resultats i presa de decisió per classificar cada estratègia segons el seu nivell de dificultat.

Per tenir una visió més àmplia i robusta sobre la valoració de les estratègies, cada membre del grup hem disputat partides contra totes les estratègies per valorar, d'una manera no tant analítica, el seu grau de dificultat, basant-nos en les nostres pròpies impressions en relació al comportament de les estratègies.

Al finalitzar totes les proves hem classificat cada estratègia segons el seu nivell de dificultat, justificant aquesta decisió amb tots els resultats, comparacions i impressions obtingudes durant les proves.

Per acabar, també descrivim tots aquells algorismes que, tot i no estar relacionats amb el nucli principal del projecte (com és la intel·ligència artificial del joc), també han estat necessaris pel desenvolupament d'altres funcionalitats importants del projecte, sense els quals algunes funcionalitats no tindrien sentit o deixarien de ser operatives.

2. Estratègies de joc implementades

2.1 Primera estratègia de joc

Per a poder tenir operativa la funcionalitat de jugar una partida contra la màquina (cas d'ús Jugar Partida), cal implementar una estratègia de joc exclusiva per al joc de taula Gomoku que actuï com a intel·ligència artificial. Aquesta s'encarregarà d'emular el comportament d'un jugador qualsevol del joc Gomoku, fent moviments que segueixen una certa lògica i amb el mateix objectiu que el de qualsevol jugador humà: guanyar la partida.

Per a implementar aquesta intel·ligència artificial ens hem basat en els coneixements que aporta la Teoria de Jocs aplicada a la computació en el camp de la intel·ligència artificial. En concret, l'estratègia de joc d'aquesta intel·ligència artificial està implementada seguint l'algorisme Minimax i aplicant l'optimització de la poda $\alpha - \beta$ per a obtenir un algorisme més eficient, ja que evita recórrer branques de l'arbre de moviments que són supèrflues de generar i revisar perquè no aporten informació rellevant.

No obstant, per a que l'execució del Minimax sigui satisfactòria i segueixi la filosofia de joc que representa, es necessita una funció d'avaluació que guiï l'algorisme a escollir el millor camí de l'arbre de moviments possibles generat. Aquesta és la missió principal de la funció d'avaluació. Aquesta funció d'avaluació està especialment implementada per al joc de taula Gomoku i, a grans trets, el que fa és avaluar (donar una puntuació numèrica) un tauler de Gomoku segons la seva distribució de fitxes durant una partida. Aquesta distribució de fitxes definida a causa de la disputa d'una partida es coneix com **l'estat del tauler**, i per a que la funció d'avaluació funcioni correctament cal garantir que aquesta distribució satisfà una disposició vàlida de les fitxes sobre el tauler, és a dir, que no hi ha cap fitxa o conjunt de fitxes que estiguin col·locades sobre el tauler de manera que es violi el reglament del Gomoku (veure el Manual d'usuari per a més informació sobre el reglament).

Entrant més en detall sobre com s'avalua un cert estat de tauler, la funció d'avaluació es basa en el recompte **d'estructures potencialment guanyadores** que té cada jugador. Cada estructura potencialment guanyadora està valorada amb una puntuació. Les que més punts tenen són aquelles que estan més a prop d'aconseguir l'objectiu final (muntar una línia de cinc fitxes seguides d'un mateix color). Una línia de cinc val la màxima puntuació (`Integer.MAX_VALUE` o `Integer.MIN_VALUE` en funció dels interessos del jugador controlat per aquesta intel·ligència artificial). Aleshores, el que cal comptar és, per cada fitxa d'un color trobada en el tauler, totes les seves estructures potencialment guanyadores, després multiplicar-les per la seva valoració establerta, i finalment sumar totes les puntuacions de cada estructura. Aquest càlcul s'ha de realitzar per cada jugador que disputa la partida en el tauler donat, és a dir, tindrem la suma total de totes les estructures potencialment guanyadores per al jugador amb fitxes negres, i la suma homòloga per al jugador amb fitxes blanques. Per acabar haurem de restar les dues sumes anteriors per conèixer quant de bo és l'estat que presenta el tauler per als interessos del jugador controlat per aquesta intel·ligència artificial. Depenent de quin color de fitxes tingui aquest jugador, haurem de restar la suma total de les estructures potencialment guanyadores pertanyents a les fitxes del seu color menys la suma total homòloga de les fitxes del jugador oponent.

Aleshores, si descrivim tots els passos que fa la funció d'avaluació per a avaluar un estat d'un tauler, obtenim la següent seqüència d'accions, ordenades per ordre de realització:

1. Recórrer el tauler posició a posició en busca de fitxes.
2. Quan en troba una, compta totes les estructures potencialment guanyadores que estan formades al voltant seu.
3. Un cop les ha comptat i analitzat totes (distingir de quines estructures es tracten), es multipliquen per la seva puntuació corresponent.

4. Suma les puntuacions de les estructures que es formen a partir d'aquesta fitxa, i el resultat es torna a sumar en una variable que acumula la puntuació total segons el color de la fitxa que s'està analitzant. Si és negre, vol dir que les estructures comptades pertanyen al jugador amb fitxes negres, i això significa que la suma parcial cal acumular-la a la variable que compta la puntuació total de les fitxes negres. Exactament hi ha el cas homòleg per al cas que la fitxa analitzada sigui de color blanc.
5. Tots els passos anteriors es van repetint en forma de bucle per a cada fitxa trobada al tauler.
6. Un cop finalitzada la lectura del tauler, es resten les sumes de puntuacions totals depenent del color de les fitxes amb les que juga el jugador controlat per aquesta intel·ligència artificial. Si juga la partida amb fitxes negres, l'avaluació del tauler es calcula restant la puntuació total de fitxes negres menys la puntuació total de fitxes blanques (fitxes de l'oponent). I viceversa si el jugador controlat per aquesta intel·ligència artificial disputa la partida amb fitxes blanques. És a dir, l'avaluació sempre serà la suma de puntuacions totals del color de les fitxes del jugador controlat per aquesta intel·ligència artificial menys la suma de puntuacions totals del color de les fitxes del seu oponent.

Per a realitzar correctament el recompte d'estructures potencialment guanyadores, és necessari que la funció d'avaluació sàpiga què és una estructura d'aquest tipus, quines estructures potencialment guanyadores diferents ha de tenir en compte a l'hora de comptar-les, i quina puntuació atorga'ls-hi de forma que unes estructures valguin més o menys segons el seu ordre de potencialitat per assolir l'objectiu de muntar una línia de cinc fitxes seguides i poder guanyar la partida. Abans de descriure la llista amb les diferents estructures potencialment guanyadores que té en compte la funció d'avaluació, és imprescindible conèixer la següent terminologia sobre diferents tipus d'estructures potencialment guanyadores:

- **Estructura simple:** línia d'un cert nombre de fitxes seguides i del mateix color.
- **Estructura complexa**: Fusió de dues estructures simples formades a partir d'una fitxa central, que representa el punt de connexió de les dues estructures simples.
- **Estructura oberta:** Estructura els extrems de la qual no estan bloquejats per cap fitxa de l'oponent ni es troben just als límits del tauler.
- **Estructura semioberta:** Estructura que un dels seus extrems no està bloquejats per cap fitxa de l'oponent ni es troba just als límits del tauler.

Un cop entesa la terminologia anterior, a continuació es llisten totes les estructures potencialment guanyadores que la funció d'avaluació té en consideració, juntament amb la seva puntuació corresponent:

- Estructura simple de cinc fitxes seguides del mateix color (estructura guanyadora). Puntuació: ∞ (`Integer.MAX_VALUE` en termes d'implementació).
- Estructura simple oberta de quatre fitxes. Puntuació: 10.000.
- Estructura complexa 4x4 semioberta (dues estructures simples semiobertes de quatre fitxes). Puntuació: 10.000.
- Estructura complexa d'una línia de quatre fitxes semioberta més una línia de tres fitxes oberta. Puntuació: 10.000.
- Estructura oberta o semioberta de quatre fitxes quasi seguides (conté una casella buida enmig de l'estructura). Puntuació: 10.000
- Estructura complexa 3x3 oberta (dues estructures simples obertes de tres fitxes). Puntuació: 5.000.
- Estructura simple oberta de tres fitxes quasi seguides (conté una casella buida enmig de l'estructura). Puntuació: 5000.
- Estructura complexa d'una línia de tres fitxes semioberta més una línia de tres fitxes oberta. Puntuació: 1.000.
- Estructura simple semioberta de quatre fitxes. Puntuació: 500.
- Estructura simple oberta de tres fitxes. Puntuació: 200.

- Estructura complexa 2x2 oberta (dues estructures simples obertes de dues fitxes). Puntuació: 100.
- Estructura simple semioberta de tres fitxes. Puntuació: 50.
- Estructura complexa 2x2 semioberta (dues estructures simples semiobertes de dues fitxes). Puntuació: 10.
- Estructura simple oberta de dues fitxes. Puntuació: 5.
- Estructura simple semioberta de dues fitxes. Puntuació: 3.

Per a conèixer més en detall com s'implementa aquesta funció d'avaluació es pot consultar el fitxer `IAGomoku.java`, el codi font del mètode `funcioAvaluacio` i tots els mètodes de classe privats que crida per garantir l'execució amb èxit. En el propi codi es troben comentades totes les variables i estructures de dades que la funció d'avaluació fa servir per gestionar tota aquesta heurística (anàlisis i determinació d'estructures potencialment guanyadores simples i complexes, gestió de les seves puntuacions corresponents, sumes parcials i totals de les puntuacions envers al color de les fitxes que es tracten en cada moment durant la lectura del tauler i anàlisi de situacions matemàticament o quasi guanyadores, en les quals l'avaluació de l'estat del tauler és la puntuació màxima o un valor molt proper a aquest).

2.2. Segona estratègia de joc

Aquesta segona estratègia està basada, com la primera, en l'algorisme Minimax amb l'optimització de poda $\alpha - \beta$ i emprant la mateixa funció d'avaluació (veure la **secció 2.1** del present document), però afegeix dues optimitzacions addicionals que permeten augmentar el límit de la profunditat en la cerca de l'arbre de possibles moviments sense perdre massa eficiència pel que fa als costos temporals de l'algorisme.

La primera optimització està relacionada amb la potencialitat de la poda $\alpha - \beta$. Per aprofitar al màxim la tala de branques supèrflues de l'arbre és condició necessària que els nodes d'un mateix nivell estiguin ordenats segons si són més o menys prometedors des del punt de vista del jugador controlat per aquesta intel·ligència artificial. Si aconseguim ordenar-los tindrem que el node de més a l'esquerre (el primer que es visita) serà el més prometedor i, si el segon node ja és pitjor que l'anterior (en funció dels valors de α i β prenen en el nivell de profunditat en qüestió), aleshores es podarà tota la seva branca, i així successivament per la resta de nodes del mateix nivell. Això és així atès que en el Minimax l'arbre es recorre d'esquerre a dreta, explorant tots els nodes que la poda $\alpha - \beta$ no ha tallat. Per tant, només haurem expandit el subarbre que penja del node més prometedor del nivell actual. Per dur a terme aquesta ordenació cal aplicar la funció d'avaluació recorrent l'arbre per nivells (en amplada) en primer lloc en comptes de fer-ho per profunditat, aplicar la funció d'avaluació als nodes terminals i anar propagant els resultats cap als nivells superiors durant la tornada enrere com passa en la primera estratègia. Si ho fem en amplada, estem aplicant la funció d'avaluació a un nombre major de nodes respecte la primera estratègia, ja que a cada nivell, per cada node, l'estem aplicant. No obstant, l'avantatge és que ara tenim un estudi preliminar de cada node per tenir una idea de quant de bo és assolir un determinat estat de tauler, i això permet ordenar els nodes abans d'explorar-los en profunditat mitjançant les crides recursives corresponents.

Però no ens podem quedar només amb la millor jugada, perquè aquest estudi per nivells de cada node no és definitiu. Pot passar que un estat de tauler tingui una valoració molt bona al principi, però al cap d'uns torns faci perdre la partida al jugador controlat per aquesta intel·ligència artificial. És a dir, cal reafirmar que un cert estat de tauler és realment prometedor explorant-lo a més profunditat i comparant-lo amb altres possibles estats prometedors que, tot i que al principi no ho siguin tant com el primer node (una vegada estan ordenats), a la llarga poden conduir al jugador a una situació molt més favorable. Per aquest motiu el que fem és expandir, per a cada nivell, els 12 millors nodes una vegada han estat avaluats i ordenats en funció del valor que han obtingut a través de la funció d'avaluació. Hem escollit els 12 nodes més representatius després de fer exhaustives proves i comprovar que, amb aquest valor, aconseguim una estratègia prou robusta (en termes decisoris) sense sacrificar massa eficiència, assolint un cert compromís entre aquestes dues variables. En comparació amb l'estratègia anterior, en aquesta deixem d'expandir tots els possibles moviments per aconseguir augmentar la profunditat d'exploració amb similars temps d'execució. Després de fer exhaustives proves hem vist que per mantenir un cost temporal raonable per a la segona estratègia, la profunditat la podem augmentar en un nivell, passant de 3 a 4 respecte la primera estratègia. Si l'augmentem més, el temps d'execució augmenta de forma exponencial, convertint-se en una situació desesperant. D'altra banda, com que no explorem totes les possibilitats, estem perdent informació de la partida, per la qual cosa és possible que no explorem moviments que a priori puguin semblar molt poc o gens prometedors, però que al final poden acabar propiciant la victòria. És un risc que, vistos els resultats durant les proves empíriques, podem prendre sense temor a obtenir un algorisme poc robust o molt ineficient en temps d'execució.

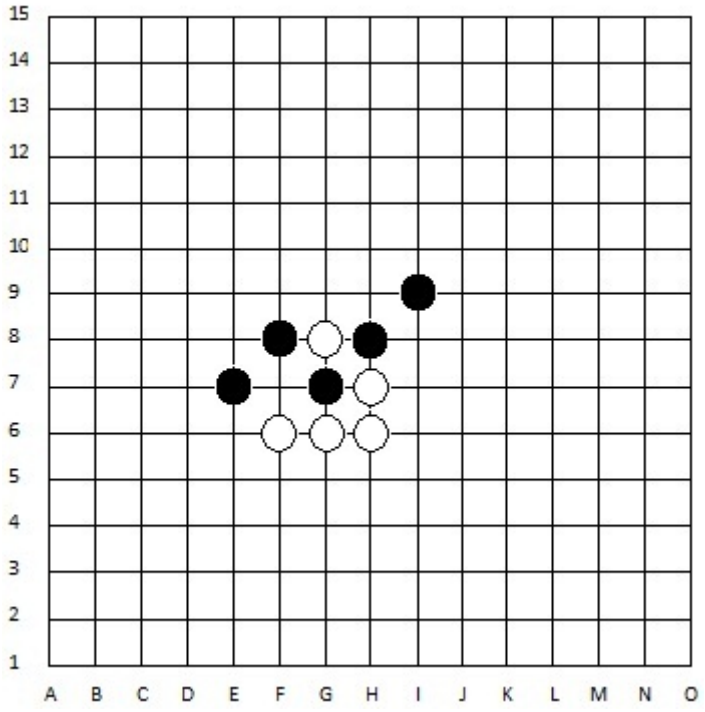
Per a poder ordenar els nodes ens cal alguna estructura de dades que ens permeti emmagatzemar informació de cada node (estat de tauler) d'un nivell de l'arbre de possibles moviments, i que a la vegada es trobin ordenats segons la seva avaluació. Per això hem escollit utilitzar una cua de prioritat, dins la qual emmagatzemem, per cada node generat d'un nivell, la seva avaluació (una vegada li hem aplicat la funció d'avaluació), i la posició del tauler de l'última fitxa col·locada que ha generat aquest estat. Aquesta informació està encapsulada dins la classe `Informacio`, declarada com

a classe interna a la classe `IAGomokuOptimitzada.java`. A més, cal implementar la interfície de Java `Comparator` per a definir com es comparen dos objectes de la classe `Informacio`, ja que aquesta classe no és un tipus primitiu i, per tant, no té definida l'operació de comparació que nosaltres volem aplicar per mantenir la coherència d'ordre dels elements dins la cua de prioritat. Aquí ens adonem que necessitem dos comparadors segons si el nivell on ens trobem és maximitzant (jugador controlat per aquesta intel·ligència artificial), o és minimitzant (oponent del jugador controlat per aquesta intel·ligència artificial). Si estem en un nivell maximitzant, volem que els elements de la cua de prioritat s'ordenin de major a menor valor d'avaluació; de més a menys prometedors des del punt de vista del jugador controlat per aquesta intel·ligència artificial. En canvi, si estem en un node minimitzant cal ordenar els elements de menor a major valor d'avaluació; de més a menys prometedors des del punt de vista de l'oponent del jugador controlat per aquesta intel·ligència artificial (mantenint la filosofia de l'algorisme base Minimax que suposa que l'oponent sempre farà el millor moviment des del seu punt de vista). Les dues classes que defineixen aquests comparadors són la classe `ComparadorInformacio` i la classe `ComparadorInformacioInvers` respectivament, declarades com a classes internes de la classe `IAGomokuOptimitzada.java`.

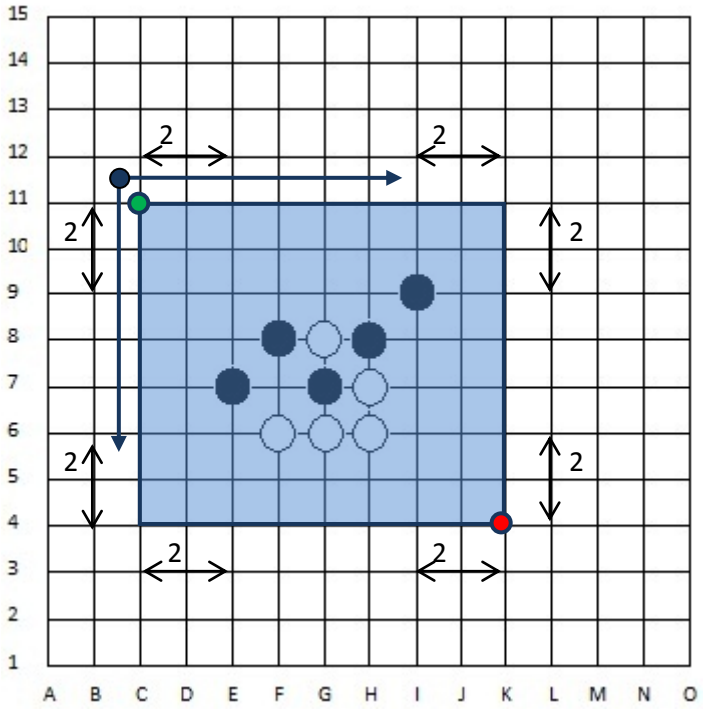
Aleshores, una vegada hem avaluat tots els nodes d'un mateix nivell i els hem guardat a la cua de prioritat emprant la comparació adequada, ara hem d'expandir (mitjançant les crides recursives inherents a Minimax) els 12 millors estats generats fins al moment, que justament són els 12 primers elements de la cua de prioritat.

La segona optimització està basada en situar dins del tauler on es troba **el focus central de la partida**, això és, en quina zona concreta del tauler s'estan col·locant les fitxes durant el transcurs del tauler. Partint de la idea que moure una fitxa molt allunyada del focus central de la partida és malgastar un torn i que, per tant, tots els estats que es poden generar a partir d'aquest per força seran sempre desfavorables, el que fem és delimitar un subtauler dins del tauler original, que englobi totes les fitxes dipositades sobre el tauler més un marge de dues caselles per permetre les jugades nul·les (deixar interseccions buides). D'aquesta manera aconseguim ometre totes aquelles caselles vàlides per generar possibles moviments, que si seguim aquesta suposició, estem segurs que no aportaran cap mena d'informació útil perquè no són moviments prometedors. Estalviem, sobretot al tram inicial de la partida, mirar un nombre elevat de caselles supèrflues, i evitem malgastar recursos computacionals en possibilitats irrellevants. Per entendre el funcionament d'aquesta optimització anem a mostrar-ho de forma gràfica.

Per exemple, suposem que ens trobem davant del següent estat del tauler (enmig d'una partida en joc):



En comptes de generar tots els possibles moviments de tot el tauler per expandir-los durant les crides recursives, només generem els possibles moviments per al següent subtauler (delimitat per la quadrícula blava):



És a dir, no generem tots els possibles moviments vàlid partint de la casella A15 fins a la casella O1, sinó que en funció de la situació del focus central de les fitxes, en aquest cas, només es generen les possibilitats vàlides que van des de la casella C11 (escaire superior esquerre del subtauler, marcat de color verd) fins a la casella K4 (escaire inferior dret del subtauler, marcat en vermell). Com es pot apreciar, el nombre de caselles a considerar és molt més reduït, i això és significant en els temps d'execució de l'algorisme, ja que amb aquesta optimització els aconseguim reduir de forma perceptible. Cal fixar-se també en què sempre es deixa un marge de dues caselles per a tots els costats del subtauler respecte la disposició de les fitxes pel motiu prèviament explicat (veure les cotes en l'esquema anterior).

El mètode encarregat de calcular, donat un tauler en un cert estat d'una partida, el subtauler que conté el focus central de joc (conté totes les fitxes col·locades sobre el tauler en aquest instant) s'anomena:

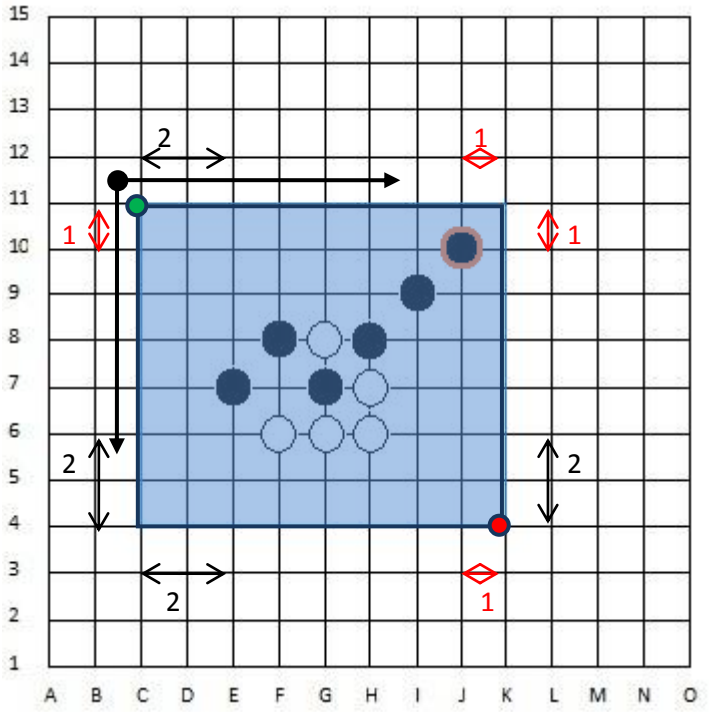
```
private int[] ubicaFocus( TaulerGomoku tauler )
```

Donat un objecte de la classe `TaulerGomoku`, ens retorna un *array* anomenat `limits` de quatre posicions que indiquen les posicions dels punts que delimiten el subtauler: `limits[0]` i `limits[1]` ens donen la fila i la columna respectivament de la casella que marca l'escaire superior esquerre que delimita el subtauler (marcat de color verd en l'esquema anterior). Per altra banda, `limits[2]` i `limits[3]` ens donen la fila i la columna respectivament de la casella que marca l'escaire inferior dret que delimita el subtauler (marcat de color vermell en l'esquema anterior).

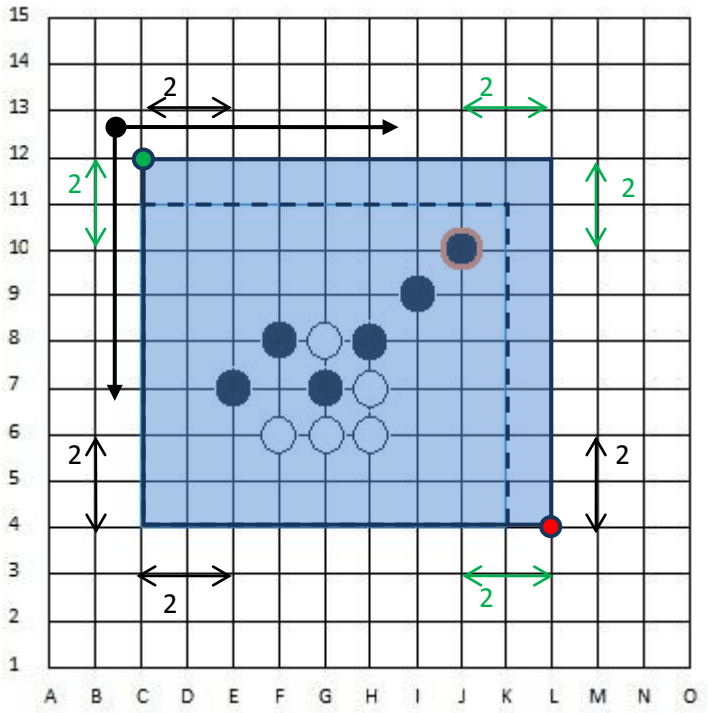
A més, per no haver de calcular novament el subtauler a cada crida recursiva (sempre que col·loquem noves fitxes al tauler), hem definit un altra mètode que actualitza el subtauler definit d'un tauler global. Aquesta actualització es basa en l'última fitxa col·locada sobre el tauler: si aquesta es troba dins del subtauler però no compleix les condicions dels marges de dues caselles, o simplement es troba fora del subtauler, aleshores cal ampliar els límits del subtauler. Aquest mètode s'anomena:

```
private int[] actualitzaFocus( int[] limits, int fila, int
columna, int mida )
```

Donat un subtauler ja definit d'un tauler original, la fila i la columna de l'última fitxa col·locada sobre el tauler global, i la mida del tauler global, actualitzem els límits del subtauler a considerar. Per entendre-ho millor ens basarem en un exemple gràfic. Partint de la situació del tauler i del subtauler definits a l'esquema anterior, ara col·loquem la següent fitxa a la casella J10 (encerclada de color taronja):



Tot i que la fitxa recentment col·locada es troba dins del subtauler ja definit, podem veure com alguns límits no compleixen la condició del marge de dues caselles (cotes marcades amb color vermell). Per tant, per garantir novament un subtauler correcte (tots els marges estan a distància de dues caselles respecte les fitxes més externes del nucli de la partida) hem de modificar el límits de color verd (escaire superior esquerre) i vermell (escaire inferior dret). Per fer-ho hauríem d'actualitzar la fila del límit superior esquerre (canviar el punt verd de la casella C11 a C12), i la columna del límit inferior dret (canviar el punt vermell de la casella K4 a L4). El resultat final, una vegada actualitzats els límits del subtauler, seria el següent:



Veient que l'actualització consta només de comparacions i operacions d'escalat dels valors per a la fila i la columna dels punts que delimiten l'àrea del subtauler, és més útil i eficient anar cridant aquest mètodes durant tot el procés recursiu en comptes de fer servir el mètode de calcular un subtauler tota l'estona. Aquest últim mètode només el cridem just abans de començar a fer servir les crides recursives pertanyents a l'algorisme Minimax.

Per a conèixer més en detall com s'implementa aquesta segona estratègia es pot consultar el fitxer `IAGomokuOptimitzada.java`, dins el qual es troba la implementació dels mètodes, classes i codi addicional referents a les dues optimitzacions afegides. A més, tant les variables com les estructures de dades emprades i blocs concrets de codi estan degudament comentats per poder entendre amb més facilitat com funciona l'algorisme.

2.3. Tercera estratègia de joc

Aquesta estratègia es va plantejar des del principi amb l'objectiu d'aconseguir un algorisme de joc relativament senzill a la vegada que ràpid, entenent que sacrificaríem obtenir un anàlisis exhaustiu (en contraposició a les dues primeres estratègies) a canvi de rendiment.

La principal idea darrere d'aquesta tercera estratègia és que, donat un tauler, els millors moviment a priori seran els que ho són tant pel propi jugador com pel seu oponent. És a dir, les posicions profitoses pel nostre contrincant també ho són per nosaltres, i a l'inrevés.

El que tenim doncs és que a cada torn es realitza l'anàlisi tant pel jugador com per l'oponent, assignant una puntuació a cada casella que pugui jugar un paper en la creació d'una o més línies de 5 fitxes (destacant-se de forma natural doncs les estructures bidimensionals, per exemple).

L'altra idea que ha motivat aquesta intel·ligència artificial ha estat que, si aquesta és capaç de sobreviure a la partida els torns suficients amb una tàctica prou defensiva, les finestres d'oportunitat per amenaçar al contrincant i realitzar moviments guanyadors aniran augmentant en freqüència degut a la major concentració de fitxes al tauler.

Podem descriure l'algorisme amb els següents passos per a cada torn en què es crida al mètode de computació de moviment:

1. Es reinicien els anàlisis, és a dir, es posen a zero totes les puntuacions de les caselles.

2. Es recorre el tauler de forma seqüencial i per cada casella no buida que es troba, es realitzen les següent operacions (actualitzant l'anàlisi del jugador o de l'oponent segons correspongui):

a. Es comprova si la casella pot ser part d'una línia de 5 en cada una de les direccions possibles.

b. En cas afirmatiu s'apliquen les puntuacions corresponents a tota la seva zona d'influència (conjunt de posicions vàlides amb les que pot crear línies). Hi ha la possibilitat de matissar lleugerament els punts segons la distància respecte la fitxa analitzada en qüestió.

3. S'escull de totes les posicions del tauler la que tingui la puntuació més alta sumant tant l'anàlisi del jugador que representa la intel·ligència artificial com el del seu oponent.

Com a resultat obtenim una heurística que en general es comportarà de forma molt defensiva encara que, en certs punts, decisions una mica més agressives podrien finalitzar la partida ràpidament en favor d'aquesta.

En quant a les estructures de dades utilitzades, totes les estructures han estat implementades mitjançant *arrays* de tipus simple d'una o dues dimensions. El mecanisme d'elecció del millor moviment, però, ha estat implementat amb una cua de prioritats amb un comparador personalitzat que donat dues posicions, utilitza com a criteri la suma dels punts de l'anàlisi del jugador i de l'oponent per a aquella posició.

Totes les puntuacions i ponderacions són fàcilment configurables mitjançant les estructures de dades utilitzades (excepte al càlcul de la potencialitat d'una línea, que tot i així els valor estan concentrats en un punt concret del mètode).

Per a més detalls es pot consultar el propi codi de la classe `IAGomokuAlternativa`. Cal notar que encara que vam realitzar algunes proves substituint parts de codi per altres aparentment més eficients, correctes o sofisticades (per exemple, una determinació més precisa de la zona d'influència d'una fitxa), o amb diferents coeficients i valors, les

proves que feiem a cada pas només feien que reafirmar-nos en el funcionament de la implementació entregada.

3. Proves empíriques entre estratègies

3.1. Descripció de les proves

Per a poder enfrontar en igualtat de condicions cadascuna de les estratègies entre elles hem dissenyat una metodologia de prova que a continuació passem a definir:

- Les proves es basen en la disputa de 50 partides seguides entre dues de les estratègies de joc.
- Les dues estratègies enfrontades durant una prova mai han estat les mateixes (no hem enfrontat una estratègia contra ella mateixa per motius conceptuals).
- Un enfrontament entre dues estratègies distingeix a dos contrincants: el contrincant local (primera estratègia que apareix en la declaració de l'enfrontament) i el contrincant visitant (l'estratègia restant). Per exemple, si la prova enfronta la primera estratègia contra la segona estratègia, aleshores la primera estratègia és el contrincant local, i la segona estratègia és el contrincant visitant.
- Per garantir un equilibri durant la realització de cada prova per a les dues estratègies implicades, en les primeres 25 partides el contrincant local serà qui jugui amb les fitxes negres. Les darreres 25 partides ho serà el contrincant visitant.
- Totes les proves s'han dut a terme mitjançant un programa responsable d'executar una bateria de 50 partides seguides, incloent el registre i control de les dades produïdes durant el transcurs de les proves. Aquest programa s'anomena `ProvaJugadorsMaquina` i el seu codi font es pot consultar al fitxer `ProvaJugadorsMaquina.java`, El fitxer executable es troba al subdirectori EXE del directori arrel (al fitxer `index.txt` del mateix subdirectori s'explica com funciona el programa per si es desitja fer més proves). El programa s'ha executat sobre el següent entorn de treball i condicions (ordinador):

  - Processador Intel® Core™2 Duo E8400, 6 MB de memòria cache, 3GHz, 1333 MHz FSB (Front Side Bus) (feu clic aquí per a més informació).
  - 4 GB de memòria RAM DDR2 800MHz, dos mòduls DIMM de 2 GB cadascun per utilitzar l'arquitectura Dual Channel.
  - Placa Base Asus P5QL-E (feu clic aquí per a més informació).
  - Disc dur magnètic de 500GB de 7200 rpm (revolucions de gir del disc per minut) connectat a una interfície de comunicació SATA a 3Gb/s.
  - Sistema Operatiu Windows 7 Ultimate Edition de 32 bits.
- Durant la realització de les proves, l'ordinador descrit en el punt anterior no estava executant altres tasques (programes executant-se a segon pla).

Un fet important a tenir en compte és que, com que les estratègies sempre mouen al centre del tauler quan comencen la partida atès que és el moviment més òptim, pot donar-se el cas que el transcurs entre partides sigui similar. Aquest fet podria desvirtuar els resultats obtinguts, però per altre banda, si forcem a que el primer moviment sigui aleatori (sempre dins dels límits del tauler evidentment), estaríem modificant el comportament de les estratègies implementades, i aleshores les proves no reflectirien el seu veritable comportament.

3.2. Segona estratègia contra primera estratègia

Una vegada hem enfrontat la primera estratègia contra la segona mitjançant la prova ja descrita (veure la **secció 3.1** del present document) hem obtingut els següents resultats:

Resultats		**Primera estratègia**	**Segona estratègia**
Fitxes negres	**Victòries**	9	18
	Derrotes	16	7
Fitxes blanques	**Victòries**	7	16
	Derrotes	18	9
Total victòries		16	34
Total derrotes		34	16
Total empats		0	0

Els temps de duració (en segons) de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	Duració	Partida	Duració
1	108,43	**26**	205,74
2	109,32	**27**	110,63
3	108,03	**28**	25,11
4	33,20	**29**	25,06
5	33,62	**30**	175,40
6	109,28	**31**	38,84
7	109,54	**32**	38,90
8	33,27	**33**	30,67
9	33,23	**34**	174,80
10	109,01	**35**	38,96
11	39,00	**36**	25,06
12	109,59	**37**	38,88
13	105,38	**38**	205,36
14	109,28	**39**	25,12
15	109,02	**40**	110,80
16	109,39	**41**	38,87
17	33,28	**42**	17489
18	110,05	**43**	37,70
19	108,77	**44**	39,17
20	33,29	**45**	206,01
21	33,07	**46**	205,39
22	108,23	**47**	174,27
23	109,24	**48**	39,03
24	108,28	**49**	36,72
25	108,38	**50**	31,37

Duració total de la prova (hores:minuts:segons): **01:12:53**.

La mitjana dels temps de resposta per torn (en segons) per a la primera estratègia de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	**Mitjana de temps per torn**	**Partida**	**Mitjana de temps per torn**
1	0,62	**26**	1,82
2	1,04	**27**	0,46
3	0,75	**28**	2,25
4	2,88	**29**	1,57
5	5,17	**30**	1,11
6	0,52	**31**	4,58
7	0,65	**32**	4,39
8	7,18	**33**	3,47
9	7,51	**34**	4,64
10	1,61	**35**	0,67
11	4,41	**36**	36,06
12	0,68	**37**	4,39
13	17,08	**38**	0,51
14	4,32	**39**	33,26
15	0,69	**40**	1,06
16	1,39	**41**	1,14
17	0,56	**42**	0,70
18	0,80	**43**	5,41
19	1,03	**44**	0,85
20	17,60	**45**	29,28
21	43,81	**46**	4,08
22	0,70	**47**	16,95
23	4,58	**48**	28,27
24	1,11	**49**	14,54
25	3,97	**50**	0,57

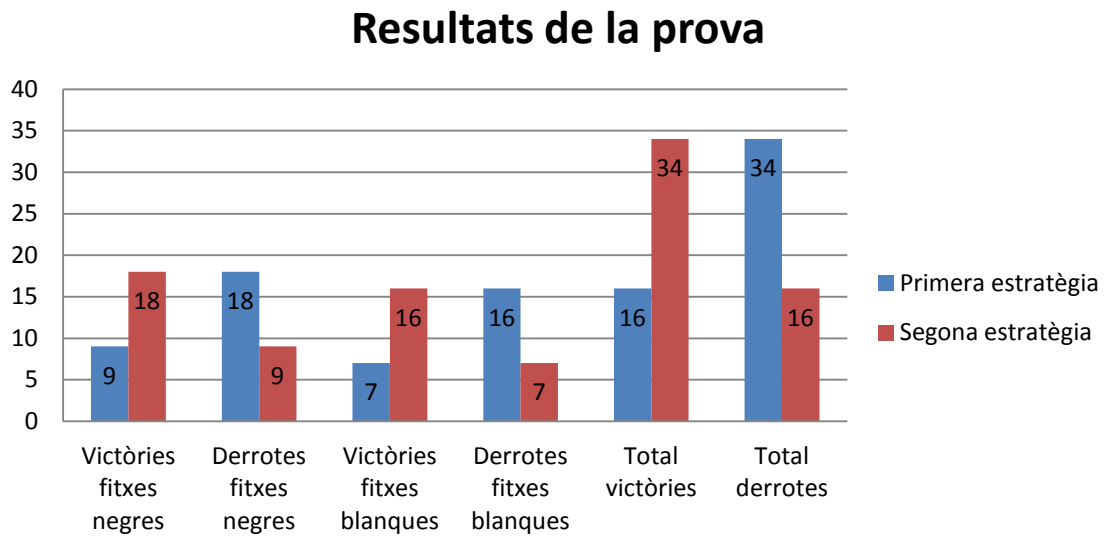
Mitjana global de temps de resposta per torn de la primera estratègia: **3,42 segons**.

La mitjana dels temps de resposta per torn (en segons) per a la segona estratègia de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	**Mitjana de temps per torn**	**Partida**	**Mitjana de temps per torn**
1	1,57	**26**	1,84
2	2.68	**27**	0,60
3	1,86	**28**	1,50
4	4,59	**29**	3,76
5	1,70	**30**	2,34
6	0,80	**31**	2,20
7	1,42	**32**	2,80
8	1,90	**33**	2,34
9	1,79	**34**	1,40
10	2,48	**35**	2,16
11	4,17	**36**	1,75
12	1,57	**37**	3,61
13	1,59	**38**	0,58
14	2,19	**39**	1,22
15	1,71	**40**	2,32
16	2,23	**41**	2,25
17	1,96	**42**	1,85
18	1,89	**43**	2,36
19	2,63	**44**	2,43
20	2,26	**45**	3,26
21	2,47	**46**	1,75
22	1,74	**47**	2,62
23	1,58	**48**	1,59
24	2,06	**49**	1,87
25	2,57	**50**	2,26

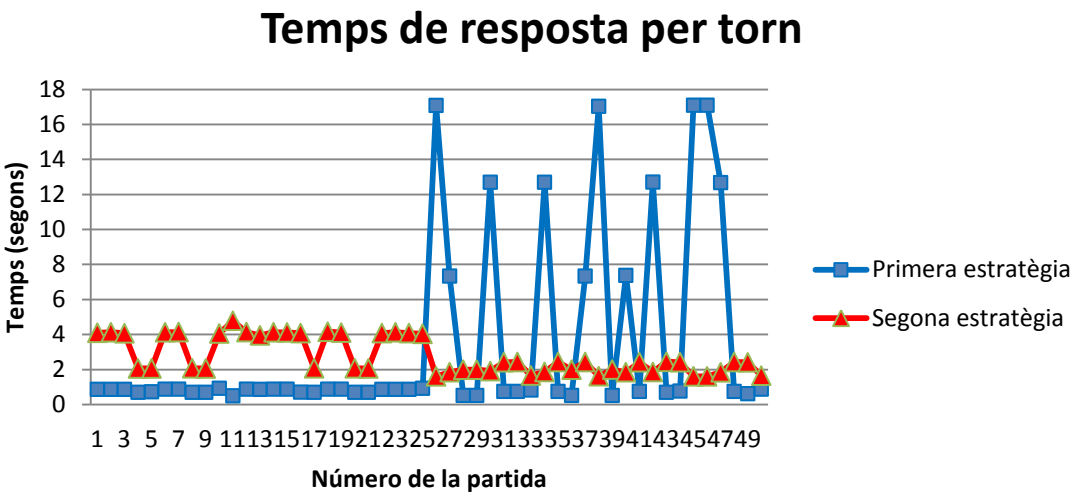
Mitjana global de temps de resposta per torn de la segona estratègia: **2,79 segons**.

Tots aquests resultats els podem expressar gràficament mitjançant els següents gràfics, que ens ajuda a tenir una visió general dels resultats de la prova:



Es pot apreciar com la segona estratègia domina en tots els àmbits per un marge de diferència ampli respecte la primera estratègia. Per tant, vistos els resultats es pot concloure que la segona estratègia és més robusta en termes decisoris i de comportament que la primera. El risc que es corre només explorant els 12 nodes a priori més prometedors a cada crida recursiva (en la segona estratègia), resulta ser eficaç vistos els resultats de la prova. És a dir, que la pèrdua d'informació per no explorar tots els nodes, però augmentant un nivell de profunditat més respecte la primera estratègia han esdevingut ser dos factors claus per a la millora de la robustesa de la segona estratègia.

Si ara expressem gràficament les mitjanes dels temps de resposta per torn per a totes les partides de la prova obtenim el següent gràfic:



En aquest gràfic podem apreciar com els temps de resposta per torn de la segona estratègia són més constants que no pas els de la primera estratègia. Això és degut a la incorporació de les dues optimitzacions addicionals en la segona estratègia (ordenació de nodes i delimitar focus centra de la partida), en comparació amb la implementació d'una sola optimització pel cas de la primera estratègia (poda α – β). En moltes partides veiem com la primera estratègia triga molt poc (sobretot quan aquesta juga amb fitxes blanques), però en d'altres (sobretot quan juga en negres) els temps augmenten descaradament. En altres paraules, és molt inestable i els seus temps de resposta dependran molt de la situació del tauler i de com es vagin generant els possibles moviments per a que la poda α – β s'aprofiti en més o menys magnitud. Aquesta dependència tan forta amb l'atzar (situacions del tauler que afavoreixen o no la poda α – β sense ordenar la visita de cada node en funció de certs criteris) provoquen aquests desajustaments en els temps de resposta. A més, el temps mitjà global de resposta per torn també reflexa molt bé que la segona estratègia és més eficient, però per un marge ajustat: 2,79 segons en mitjana enfront els 3,42 segons de la primera estratègia. La segona estratègia també sembla mostrar uns temps de resposta inferiors, en mitjana, quan juga amb fitxes blanques que no pas quan ho fa amb fitxes negres, segons el gràfic anterior.

3.3. Tercera estratègia contra segona estratègia

Una vegada hem enfrontat la tercera estratègia contra la segona mitjançant la prova ja descrita (veure la **secció 3.1** del present document) hem obtingut els següents resultats:

Resultats		**Tercera estratègia**	**Segona estratègia**
Fitxes negres	**Victòries**	15	0
	Derrotes	10	25
Fitxes blanques	**Victòries**	25	10
	Derrotes	0	15
Total victòries		40	10
Total derrotes		10	40
Total empats		0	0

Els temps de duració (en segons) de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	Duració	Partida	Duració
1	31,08	**26**	63,99
2	43,70	**27**	63,50
3	30,79	**28**	64,32
4	31,68	**29**	65,56
5	31,13	**30**	63,97
6	31,90	**31**	63,91
7	30,67	**32**	64,78
8	32,33	**33**	63,77
9	32,33	**34**	63,49
10	30,44	**35**	63,69
11	31,11	**36**	63,49
12	30,69	**37**	65,24
13	55,26	**38**	64,09
14	30,67	**39**	65,06
15	30,66	**40**	63,82
16	30,88	**41**	62,65
17	36,25	**42**	65,24
18	36,67	**43**	64,93
19	31,82	**44**	64,04
20	55,39	**45**	63,25
21	36,03	**46**	65,14
22	31,74	**47**	63,24
23	43,80	**48**	70,97
24	31,15	**49**	64,45
25	30,74	**50**	63,58

Duració total de la prova (hores:minuts:segons): **00:41:20**.

La mitjana dels temps de resposta per torn (en segons) per a la tercera estratègia de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	**Mitjana de temps per torn**	**Partida**	**Mitjana de temps per torn**
1	0,00	**26**	0,00
2	0,00	**27**	0,00
3	0,00	**28**	0,00
4	0,00	**29**	0,00
5	0,00	**30**	0,00
6	0,00	**31**	0,00
7	0,00	**32**	0,00
8	0,00	**33**	0,00
9	0,00	**34**	0,00
10	0,00	**35**	0,00
11	0,00	**36**	0,00
12	0,00	**37**	0,00
13	0,00	**38**	0,00
14	0,00	**39**	0,00
15	0,00	**40**	0,00
16	0,00	**41**	0,00
17	0,00	**42**	0,00
18	0,00	**43**	0,00
19	0,00	**44**	0,00
20	0,00	**45**	0,00
21	0,00	**46**	0,00
22	0,00	**47**	0,00
23	0,00	**48**	0,00
24	0,00	**49**	0,00
25	0,00	**50**	0,00

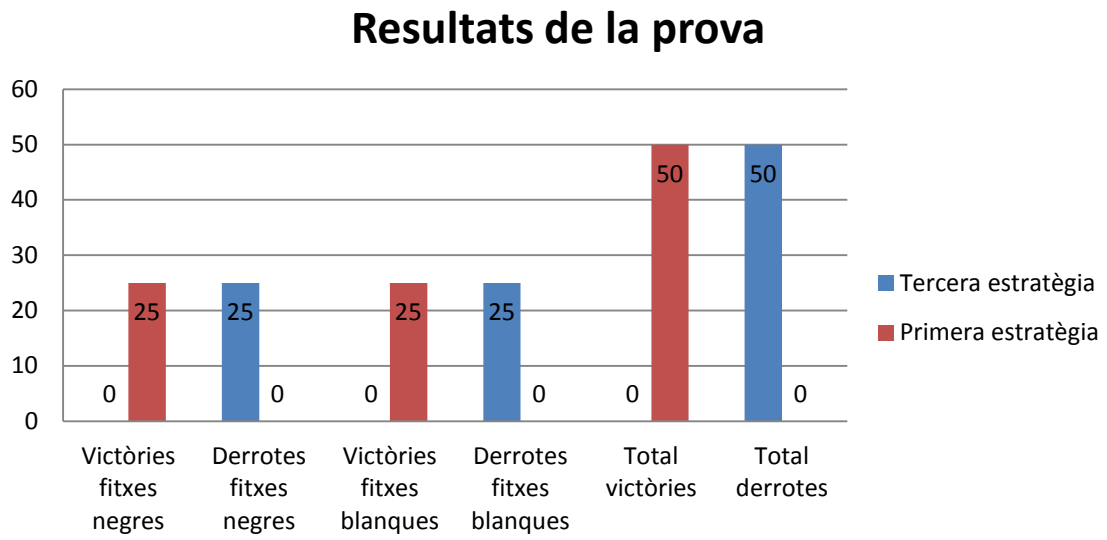
Mitjana global de temps de resposta per torn de la primera estratègia: **0,00 segons**.

La mitjana dels temps de resposta per torn (en segons) per a la segona estratègia de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	**Mitjana de temps per torn**	**Partida**	**Mitjana de temps per torn**
1	2,58	**26**	3,04
2	3,11	**27**	3,02
3	2,36	**28**	3,06
4	2,43	**29**	2,98
5	2,59	**30**	3,04
6	2,27	**31**	3,04
7	2,35	**32**	2,94
8	2,30	**33**	3,03
9	2,30	**34**	3,02
10	2,33	**35**	3,03
11	2,59	**36**	3,02
12	2,35	**37**	2,96
13	3,24	**38**	3,05
14	2,35	**39**	2,95
15	2,35	**40**	3,04
16	2,37	**41**	3,03
17	2,78	**42**	3,10
18	2,82	**43**	3,09
19	2,27	**44**	3,05
20	3,25	**45**	3,01
21	2,00	**46**	2,96
22	2,26	**47**	3,01
23	3,12	**48**	2,84
24	2,59	**49**	3,07
25	2,36	**50**	3,02

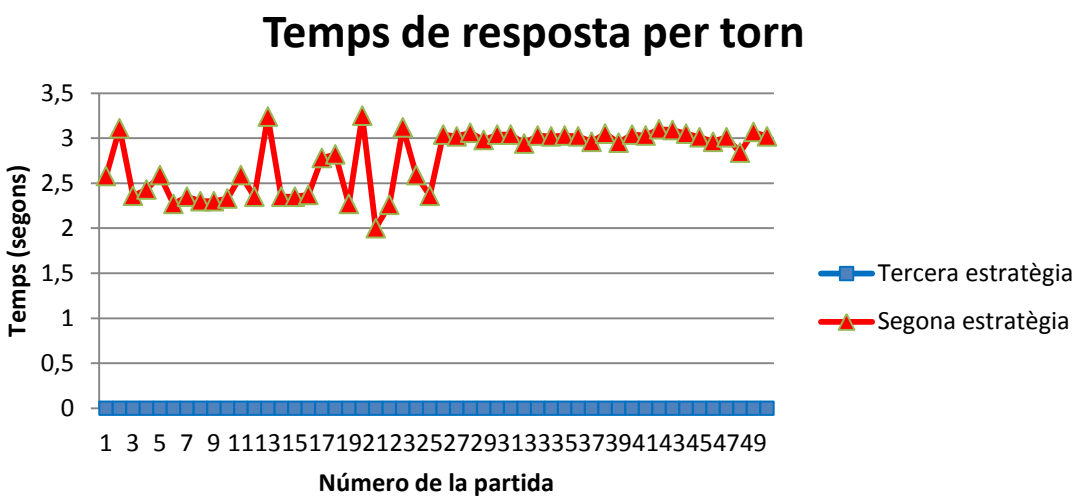
Mitjana global de temps de resposta per torn de la segona estratègia: **2,77 segons**.

Tots aquests resultats els podem expressar gràficament mitjançant els següents gràfics, que ens ajuda a tenir una visió general dels resultats de la prova:



Es pot apreciar com els resultats han estat molt contundents. La tercera estratègia ha estat imbatible jugant amb fitxes blanques, i superior quan ha jugat amb fitxes negres. No obstant, aquests resultats cal agafar-los amb prudència degut a que el començament de totes les partides sempre és el mateix (s'inicia la partida amb la primera fitxa al centre del tauler). Aquest fet ha pogut provocar que la majoria de les partides fossin quasi idèntiques o simètriques, i per això el resultat d'aquestes podia ser previsible. El que queda patent és que la tercera estratègia sap jugar molt bé a la defensiva (quan juga amb blanques), o bé que la segona estratègia és massa ofensiva i descuida la seva defensa. En qualsevol cas, els resultats queden patents i tampoc els podem menysprear. Sembla ser que jugar a la defensiva, a mig termini, comporta arribar a un estat molt favorable el qual es converteix en victòria quasi segura. En comparació amb la prova anterior (veure **secció 3.2** del present document), aquí la pèrdua d'informació que pateix la segona estratègia a l'hora d'explorar possibles moviments per guanyar eficiència és un risc que ha acabat passant factura.

Si ara expressem gràficament les mitjanes dels temps de resposta per torn per a totes les partides de la prova obtenim el següent gràfic:



Els temps de resposta també són molt clars. Mentre que la segona estratègia assoleix uns temps estables i molt semblants als de la prova anterior (veure la **secció 3.2** del present document), els temps de resposta de la tercera estratègia són immediats, de l'ordre dels centenars de microsegons. Per aquest motiu, segons l'escala del gràfic i de les dades recollides, aquests s'han aproximat als zero segons en totes les partides. Una dada curiosa és que la segona estratègia sembla ser més estable en els temps de resposta quan ha jugat amb fitxes de color negre. Aquestes diferències en els temps de resposta es deu a la diferència de concepte existent entre els dues estratègies enfrontades: mentre que la segona estratègia es basa en algorismes de cerca exhaustiva i poda de branques en arbres (en anglès coneguts com *branch and bound*), la tercera estratègia implementa una heurística estàtica que no genera totes les possibilitats de moviment per a torn futurs, la qual cosa suposa un cost computacional molt inferior i, evidentment, una estratègia molt eficient.

3.4. Tercera estratègia contra primera estratègia

Una vegada hem enfrontat la tercera estratègia contra la segona mitjançant la prova ja descrita (veure la **secció 3.1** del present document) hem obtingut els següents resultats:

Resultats		**Tercera estratègia**	**Primera estratègia**
Fitxes negres	**Victòries**	0	25
	Derrotes	25	0
Fitxes blanques	**Victòries**	0	25
	Derrotes	25	0
Total victòries		0	50
Total derrotes		50	0
Total empats		0	0

Els temps de duració (en segons) de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	Duració	Partida	Duració
1	59,79	**26**	68,71
2	13,08	**27**	68,49
3	158,16	**28**	68,60
4	13,87	**29**	69,25
5	14,46	**30**	68,89
6	13,11	**31**	68,90
7	13,13	**32**	69,00
8	13,75	**33**	68,59
9	59,57	**34**	69,84
10	13,12	**35**	68,84
11	14,25	**36**	68,83
12	14,59	**37**	69,00
13	59,51	**38**	68,87
14	13,04	**39**	68,85
15	13,03	**40**	68,47
16	12,03	**41**	68,62
17	14,00	**42**	68,79
18	57,60	**43**	68,65
19	14,59	**44**	69,01
20	14,90	**45**	68,97
21	59,28	**46**	69,25
22	14,00	**47**	69,14
23	60,14	**48**	68,82
24	12,05	**49**	68,70
25	13,06	**50**	68,74

Duració total de la prova (hores:minuts:segons): **00:41:21**.

La mitjana dels temps de resposta per torn (en segons) per a la tercera estratègia de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	**Mitjana de temps per torn**	**Partida**	**Mitjana de temps per torn**
1	0,00	**26**	0,00
2	0,00	**27**	0,00
3	0,00	**28**	0,00
4	0,00	**29**	0,00
5	0,00	**30**	0,00
6	0,00	**31**	0,00
7	0,00	**32**	0,00
8	0,00	**33**	0,00
9	0,00	**34**	0,00
10	0,00	**35**	0,00
11	0,00	**36**	0,00
12	0,00	**37**	0,00
13	0,00	**38**	0,00
14	0,00	**39**	0,00
15	0,00	**40**	0,00
16	0,00	**41**	0,00
17	0,00	**42**	0,00
18	0,00	**43**	0,00
19	0,00	**44**	0,00
20	0,00	**45**	0,00
21	0,00	**46**	0,00
22	0,00	**47**	0,00
23	0,00	**48**	0,00
24	0,00	**49**	0,00
25	0,00	**50**	0,00

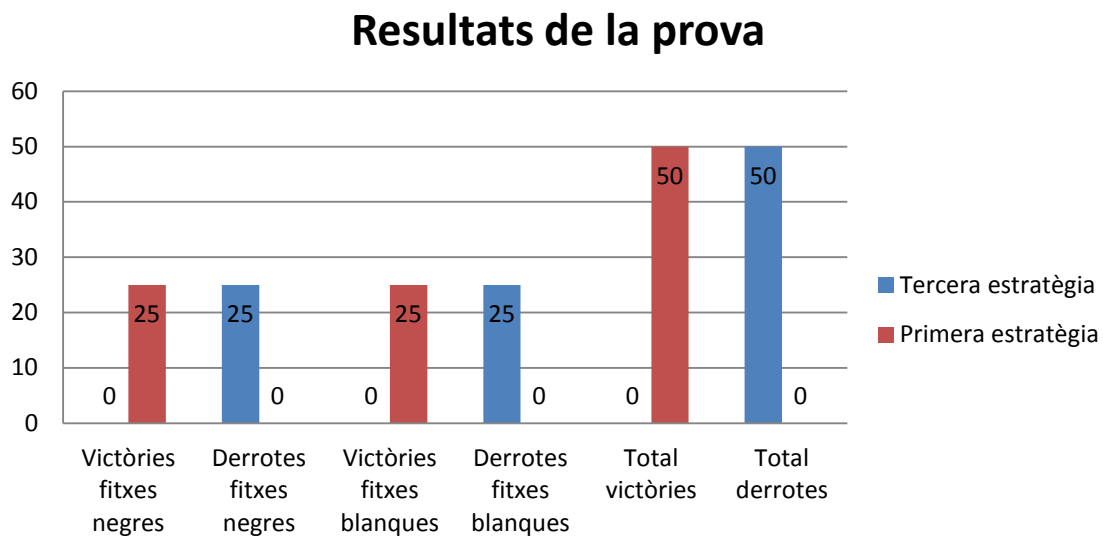
Mitjana global de temps de resposta per torn de la primera estratègia: **0,00 segons**.

La mitjana dels temps de resposta per torn (en segons) per a la primera estratègia de cadascuna de les 50 partides que conformen la prova han estat els següents:

Partida	**Mitjana de temps per torn**	**Partida**	**Mitjana de temps per torn**
1	4,59	**26**	5,28
2	0,93	**27**	5,26
3	8,78	**28**	5,27
4	1,06	**29**	5,32
5	1,11	**30**	5,29
6	0,93	**31**	5,29
7	0,93	**32**	5,30
8	0,97	**33**	5,27
9	4,58	**34**	5,37
10	0,93	**35**	5,29
11	1,09	**36**	5,29
12	1,12	**37**	5,30
13	4,57	**38**	5,29
14	0,92	**39**	5,29
15	0,92	**40**	5,26
16	1,00	**41**	5,27
17	1,07	**42**	5,28
18	4,11	**43**	5,27
19	1,11	**44**	5,30
20	1,14	**45**	5,30
21	4,55	**46**	5,32
22	1,07	**47**	5,31
23	4,62	**48**	5,29
24	1,00	**49**	5,28
25	0,93	**50**	5,28

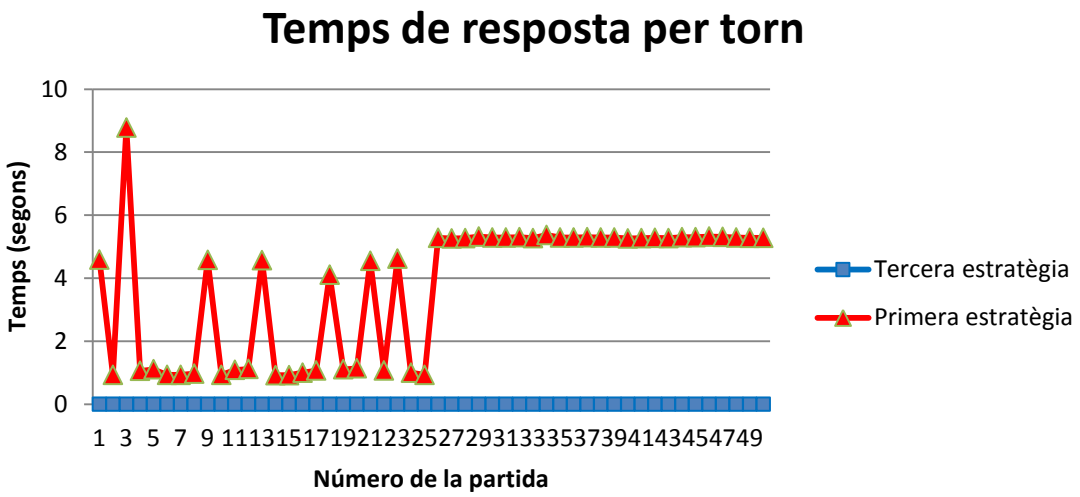
Mitjana global de temps de resposta per torn de la segona estratègia: **3,73 segons**.

Tots aquests resultats els podem expressar gràficament mitjançant els següents gràfics, que ens ajuda a tenir una visió general dels resultats de la prova:



Si en l'anterior prova (veure la **secció 3.3** del present document) els resultats havien estat molt contundents, en aquesta prova encara ho han estat més. La primera estratègia ha guanyat totes les partides de la prova. No obstant, aquests resultats també els hem de llegir entre línies si prenem com a handicap que totes les partides s'inicien amb la primera fitxa al centre del tauler, i que aquest fet pot provocar partides no del tot aleatòries entre sí. Però sense tenir en compte aquest fet, es veu clarament com la primera estratègia, tot i no explorar amb tanta profunditat com ho fa la segona estratègia l'arbre de generació de moviments, aconsegueix millors resultats segurament gràcies a que no perd informació al respecte (explora fins la profunditat màxima tots els possibles moviments que no han estat podats per l'optimització $\alpha - \beta$). Aquesta prova sembla reafirmar que no corre cap risc quan es juga contra la tercera estratègia és el més sensat per guanyar-la.

Si ara expressem gràficament les mitjanes dels temps de resposta per torn per a totes les partides de la prova obtenim el següent gràfic:



Els temps reflecteixen el que ens hem trobat en les anteriors proves realitzades (veure les **seccions 3.3** i **3.2** del present document). La tercera estratègia té un temps de resposta immediat a causa de la seva naturalesa conceptual, i la primera estratègia és força inestable quan juga amb fitxes blanques, però es manté molt estable quan ho fa amb fitxes negres, al voltant dels 5 segons. En mitjana global, triga 3,73 segons segons els resultats de la taula anterior. Per tant, podem reafirmar que, sense cap mena de dubte, la tercera estratègia és la més eficient de totes les estratègies implementades en el projecte.

4. Proves empíriques entre humans i estratègies

4.1. Primera estratègia contra jugadors humans

En la majoria de partides que els integrants del grup hem disputat contra la primera estratègia hem vist que juga de manera bastant robusta. Pren bones decisions en la majoria dels casos, tant ofensivament com defensivament. Hem detectat que aquesta és més ofensiva que no pas defensiva, sobretot si juga amb fitxes negres atès que sempre intentar dur la iniciativa de l'atac. Sempre que pot intenta deixar una intersecció buida (provoca el moviment nul) per despistar l'adversari i, si aquest no veu les seves intencions, en el següent torn sentencia la partida. També cal notar que, quan la partida ha arribat a un estat matemàticament perdut per aquesta estratègia, fa un moviment aleatori atès que, per molt que intenti anar a la defensiva, ja no pot evitar la derrota. Pot sembla un aspecte contradictori i poc coherent a simple vista, però si es pensa bé en termes decisoris no importa el pròxim moviment si el resultat, faci el que faci, ja està adjudicat. Aquest comportament és degut a la filosofia de l'algorisme que fa funcionar aquesta estratègia, el Minimax, en suposar que l'adversari sempre realitzarà el moviment més òptim. Per tant, es podria donar el cas que un jugador fes un moviment que suposés la seva victòria matemàtica abans d'hora, però que no s'hagués adonat de la potencialitat del seu últim moviment, i aquesta estratègia respongués amb un moviment que el jugador humà no entengués. Però és un risc que cal córrer si es vol adoptar aquesta filosofia de joc. En una mesura molt més inferior, aquest comportament també ve definit per la implementació adoptada en la funció d'avaluació pertanyent a aquesta estratègia. Per a més informació sobre les implementacions citades anteriorment es pot consultar la **secció 2.1** del present document.

Pel que fa als temps de resposta, en la majoria dels casos són prou ràpids, com es pot observar en els estudis empírics ja comentats (veure la **secció 3** del present document). No obstant, hi ha determinades situacions en que el temps de resposta assoleix uns pics molt considerables, pels motius ja explicats (veure la **secció 3** del present document).

4.2. Segona estratègia contra jugadors humans

Les impressions que desprèn aquesta estratègia quan els integrants del grup hem disputat partides contra ella són molt similars a les exposades en la primera estratègia (veure la **secció 4.1** del present document). La causa d'aquesta similitud és que aquesta estratègia també es basa en la mateixa filosofia de joc i en la mateixa funció d'avaluació. Intenta dur la dinàmica de la partida sempre que pot, jugant a despistar l'oponent amb la jugada del moviment nul (deixar una intersecció buida). Però no sempre pren les mateixes decisions que la primera estratègia per les optimitzacions que implementa (veure la **secció 2.2** del present document), sobretot per l'augment d'un nivell més de profunditat que se li aplica. Tot i així, els membres del grup tenim la impressió que aquesta estratègia és una mica més difícil de guanyar, que és més robusta en la presa de decisions. La diferència és lleugera, però aquesta impressió és unànime entre el grup.

Pel que fa als temps de resposta, aquests es mantenen molt estables en totes les situacions estudiades. Dóna la impressió que és un xic més lenta que la primera estratègia, però les proves empíriques demostren que els temps, en mitjana, són bons i molt regulars. No pateix pics de retard en el còmput del moviment més òptim que pateix la primera estratègia gràcies a les optimitzacions que implementa.

4.3. Tercera estratègia contra jugadors humans

Després que tots els integrants del grup haguem disputat partides contra la tercera estratègia, hem vist que aquesta té un comportament contrari al de les altres estratègies. Aquesta estratègia és de caire defensiu: qualsevol petita amenaça del rival la detecta ràpidament i intenta tapar les seves opcions de victòria. En la majoria de les partides que hem disputat sembla que pren bones decisions defensives, tapant sempre d'una manera prou intel·ligent com per evadir la formació d'un cúmul de fitxes de l'oponent. D'aquesta manera assegura que la partida augmenti en nombre de torns jugats i s'ompli

substancialment el tauler de fitxes, disminuint progressivament les opcions a formar estructures complexes que optin a la victòria. La seva robustesa es fonamenta en el fet que, a mesura que avança la partida i aquesta estratègia va tapant opcions del rival, arribarà un moment en què es formaran cúmuls de fitxes d'aquesta estratègia que formaran estructures potencialment o matemàticament guanyadores, les quals li asseguren assolir l'objectiu del joc i derrotar al seu oponent. No obstant, no sempre pren les millors decisions, sobretot en aquells casos en què podria anar a l'atac i prefereix fer moviments de caire defensiu, perdent oportunitats de victòria. I a vegades també ocorre que no sempre defensa la opció més perillosa del rival, sobretot quan el tauler està ple de fitxes, però aquests errors decisoris són molt poc freqüents en general.

Pel que fa als temps de resposta, sense cap mena de dubte aquesta estratègia és la més eficient de totes les que hem implementat en el projecte. La seva resposta en tots els casos és immediata, imperceptible per a l'ésser humà (estem parlant que triga en decidir un moviment en uns temps de l'ordre de la desena a la centena de microsegons). Aquests temps de resposta s'han pogut assolir gràcies a l'heurística implementada (veure la **secció 2.3** del present document).

5. Conclusions i classificació de les estratègies

5.1. Classificació final segons la dificultat

Una vegada hem enfrontat les diferents estratègies implementades entre elles i els membres del grup ens hem enfrontat també contra cadascuna d'elles, estem en les millors condicions per discernir quina d'elles és l'estratègia considerada de dificultat fàcil, mitjana o difícil. Com que els resultats analítics quan s'han enfrontat entre elles no són massa clars (veure la **secció 3** del present document), perquè no hi ha hagut cap que guanyi les altres dues de manera contundent, i pel handicap dels inicis de partida força predictibles, ens hem basat més en les impressions que desprenen quan juguen contra jugadors humans.

La **tercera estratègia** l'hem considerada com l'estratègia de **dificultat fàcil** del projecte atès que, una vegada es coneix com contrarestar la seva manera de defensar, i aprofitant que no sempre pren les millors decisions defensives quan està envoltada de més d'una opció perillosa, es pot guanyar-la.

La **primera estratègia** l'hem considerat com la de **dificultat mitjana**, atès que de forma general costa més de vèncer. Sembla ser que normalment pren decisions més robustes i, a més, com que té un comportament més ofensiu, és més probable que en un dels seus intents per anar a l'atac aconsegueixi la victòria. A més, vistos els resultats quan hem enfrontat la primera estratègia contra la segona estratègia (veure la **secció 3.4** del present document), podem observar com la primera estratègia sembla ser més bona que la tercera estratègia.

I finalment hem decidit que la **segona estratègia** és la de **dificultat difícil** dins del nostre projecte. Si la comparem amb la primera estratègia en els resultats analítics (veure la **secció 3.2** del present document), veiem com la segona estratègia guanya més vegades, i dóna la impressió de ser una mica més difícil de vèncer respecte la primera estratègia. Aquesta percepció és molt fina, però consensuada entre tot el grup. A més també l'hem considerat com l'estratègia difícil perquè és més eficient que la primera estratègia (veure la **secció 3.2** del present document).

Després d'arribar a tals conclusions, no podem negar que, a base de jugar i jugar partides contra aquestes estratègies, al final un és capaç d'aprendre com es comporten realment i guanyar-les de tant en tant. No són perfectes, però l'estratègia que nosaltres hem considerat fàcil potser per alguns jugadors els pot semblar que és complicada de vèncer, i en canvi l'estratègia que nosaltres hem considerat com a difícil els pot semblar més senzilla de guanyar. Això reflexa el fet que la classificació que nosaltres hem establert no té perquè coincidir amb el parer de tots els jugadors que s'enfronten a aquestes estratègies, creant-se d'aquesta manera divergència d'opinions entre el col·lectiu de jugadors. Però la nostra intenció ha estat dedicar un nombre considerable d'hores en l'estudi exhaustiu de les tres estratègies per comparar-les i poder arribar a uns resultats clarividents que evidenciessin els motius pels quals hem decidit classificar les estratègies d'aquesta manera, fent servir arguments el més objectius i crítics possible.

Per finalitzar, la relació entre les classes implementades i les estratègies a les quals fan referència dins del projecte és la següent:

- La tercera estratègia és la que està implementada dins la classe `IAGomokuAlternativa`, i és la considerada com la de dificultat fàcil.
- La primera estratègia és la que està implementada dins la classe `IAGomoku`, i és la considerada com la de dificultat mitjana.
- La segona estratègia és la que està implementada dins la classe `IAGomokuOptimitzada`, i és la considerada com la de dificultat difícil.

6. Altres algorismes implementats

6.1. Algorisme de xifrat Vigenère

Aquest algorisme ens proporciona un mètode prou segur i estable per garantir la seguretat del sistema durant la funcionalitat de registrar usuaris i els possibles atacs que pugui patir per part d'usuaris amb caràcter maliciós.

Aquest és un algorisme d'encriptació basat en un xifrat de substitució simple i polialfabètic. Donat un **diccionari de caràcters** ben definit, aleshores el que permet fer aquest algorisme és xifrar **una cadena de caràcters** vàlida (els caràcters de la qual formen part del diccionari) a una altra cadena illegible de caràcters (els quals també estan presents en el diccionari) mitjançant **una clau** durant el procés de xifrat. La clau també és una cadena de caràcters vàlida per al diccionari definit. És a dir, que donat un missatge a xifrar i una clau, amb la precondició que els caràcters que les formen es troben en el diccionari, podem xifrar el missatge per evitar que el seu contingut sigui revelat a terceres persones. Aquesta és la filosofia en la qual es basa la seguretat del nostre projecte. A continuació expliquem amb detall com funciona aquest algorisme a través d'un exemple senzill:

Suposem que tenim definit un diccionari de caràcters. Aquest diccionari engloba els caràcters en majúscula del nostre abecedari: A, B, C, D, ..., Z (sense comptar els caràcters ç ni ll). En total tenim un diccionari format per 26 caràcters. Aleshores nosaltres volem xifrar el següent missatge (cadena de caràcters): **GOMOKU**. I la clau (també és una cadena de caràcters) que farem servir per el nostre propòsit serà la següent: **PROP**. Primer cal comprovar que es compleixen les dues següents premisses:

1. Els caràcters que formen el missatge a xifrar i la clau han de pertànyer al conjunt de caràcters que defineixen el diccionari.
2. La longitud del missatge a xifrar ha de ser igual o superior a la longitud de la clau, entenen com a longitud el nombre de caràcters que formen aquestes dues cadenes.

A simple vista podem veure que les dues premisses es compleixen (tots els caràcters presents a la cadena GOMOKU i a la cadena PROP pertanyen al diccionari prèviament definit, i la cadena GOMOKU és més llarga que la cadena PROP, 6 caràcters contra 4 caràcters). Com que les dues premisses es compleixen, diem que aquest missatge es pot xifrar seguint l'algorisme de Vigenère. El procés de xifrat és el següent:

1. Amb el diccionari definit prèviament hem de crear el que s'anomena la **taula de Vigenère**, gràcies a la qual podrem fer tot el procés de xifrat (substitució de caràcters). Aquesta taula té tantes files i columnes com la mida (nombre de caràcters) del diccionari, en aquest cas serà de 26 files per 26 columnes. La taula s'ompliria de la següent manera:

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

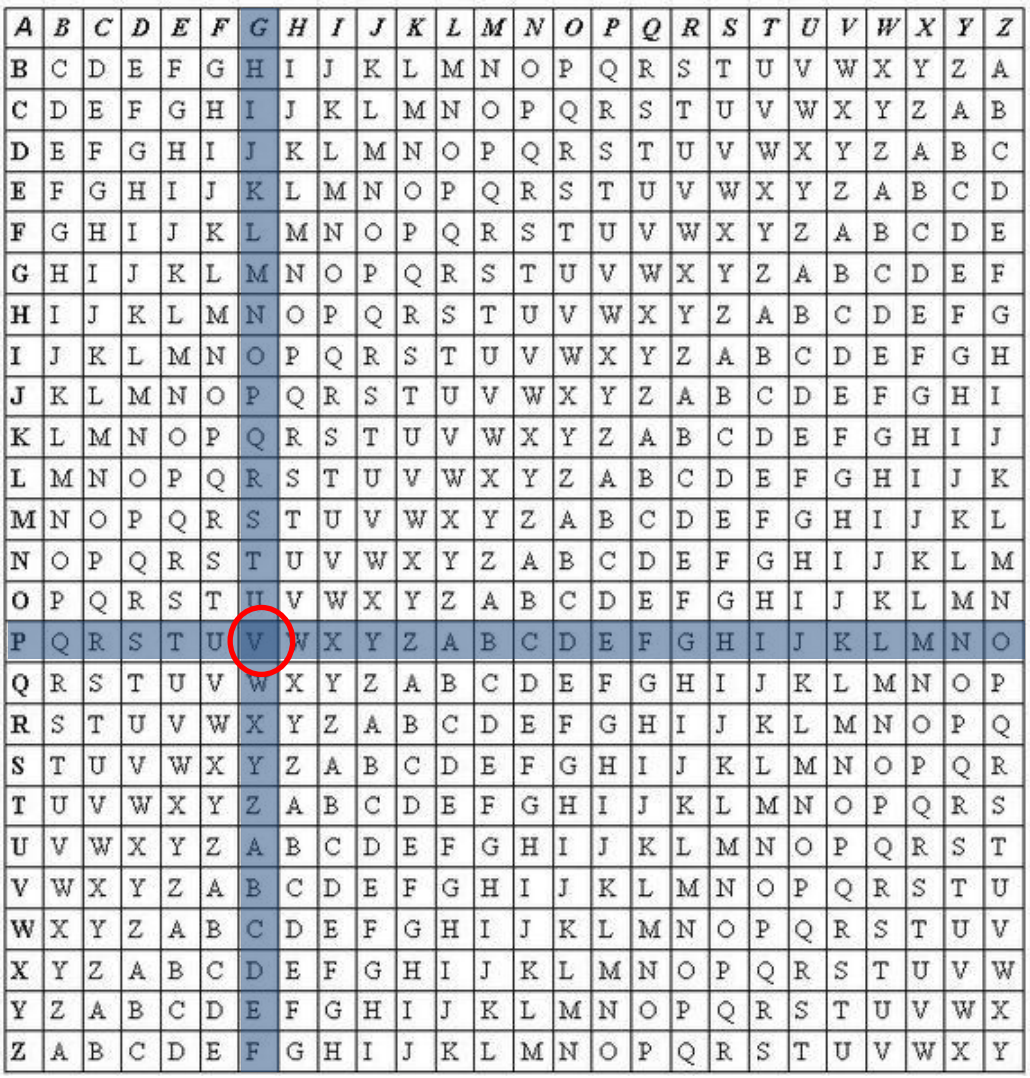
És a dir, si la formem d'esquerre a dreta, la primera fila la ompliríem seqüencialment començant pel primer caràcter del diccionari fins l'últim (A, B, C, .., Z) en el mateix ordre en què apareixen al diccionari. La següent fila la començaríem a omplir a partir del segon caràcter pertanyent al diccionari (B en aquest cas), i aniríem omplint seqüencialment la fila en el mateix ordre que estan definits els caràcters al diccionari, i quan arribem a l'últim caràcter continuaríem omplint la fila des del primer caràcter del diccionari, i així fins arribar a omplir tota la fila (en la segona fila seria: B, C, D, .., Z, A). Per a la tercera fila començaríem pel caràcter C i l'aniríem omplint: C, D, E, ..., Z, A, B. I així successivament fins a tenir la taula completada, del mateix aspecte que l'anterior.

2. Aleshores el següent pas és replicar la clau de la mateixa manera que hem replicat el diccionari durant la formació de les files per a la taula de Vigenère, fins que la clau assoleixi una longitud igual a la del missatge a xifrar. En el nostre exemple, la clau final seria **PROPPR**:

G	**O**	**M**	**O**	**K**	**U**
P	**R**	**O**	**P**	**P**	**R**

L'únic que fem és anar concatenant caràcter a caràcter la cadena inicial de la clau fins que aquesta tingui la mateixa longitud que el missatge que volem xifrar (aniríem formant la nova clau amb la seqüència PROPPROPPROP... fins a tenir una clau de la mateixa longitud que la del missatge a xifrar).

3. Ara hem de localitzar el primer caràcter del missatge a xifrar a la taula, a partir de la primera fila (en aquest cas el caràcter G es troba, dins la fila 1, a la columna 7). Fem el mateix procediment pel primer caràcter de la clau, però aquest l'hem de cercar a partir de la primera columna (en aquest cas el caràcter P es troba, dins la primera columna, a la fila 16). Una vegada hem localitzat el primers caràcter del missatge a xifrar i el de la clau, dibuixem la intersecció, i on es creuin serà el caràcter xifrat per a la primera posició de la cadena xifrada. En el nostre exemple el caràcter codificat seria V, vegem-ho en la següent imatge:



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Després caldria fer el mateix procediment de conversió per al segon caràcter del missatge a xifrar (el caràcter O) amb el segon caràcter de la clau (el caràcter R), donant com a resultat de la conversió el caràcter F, i així successivament fins a

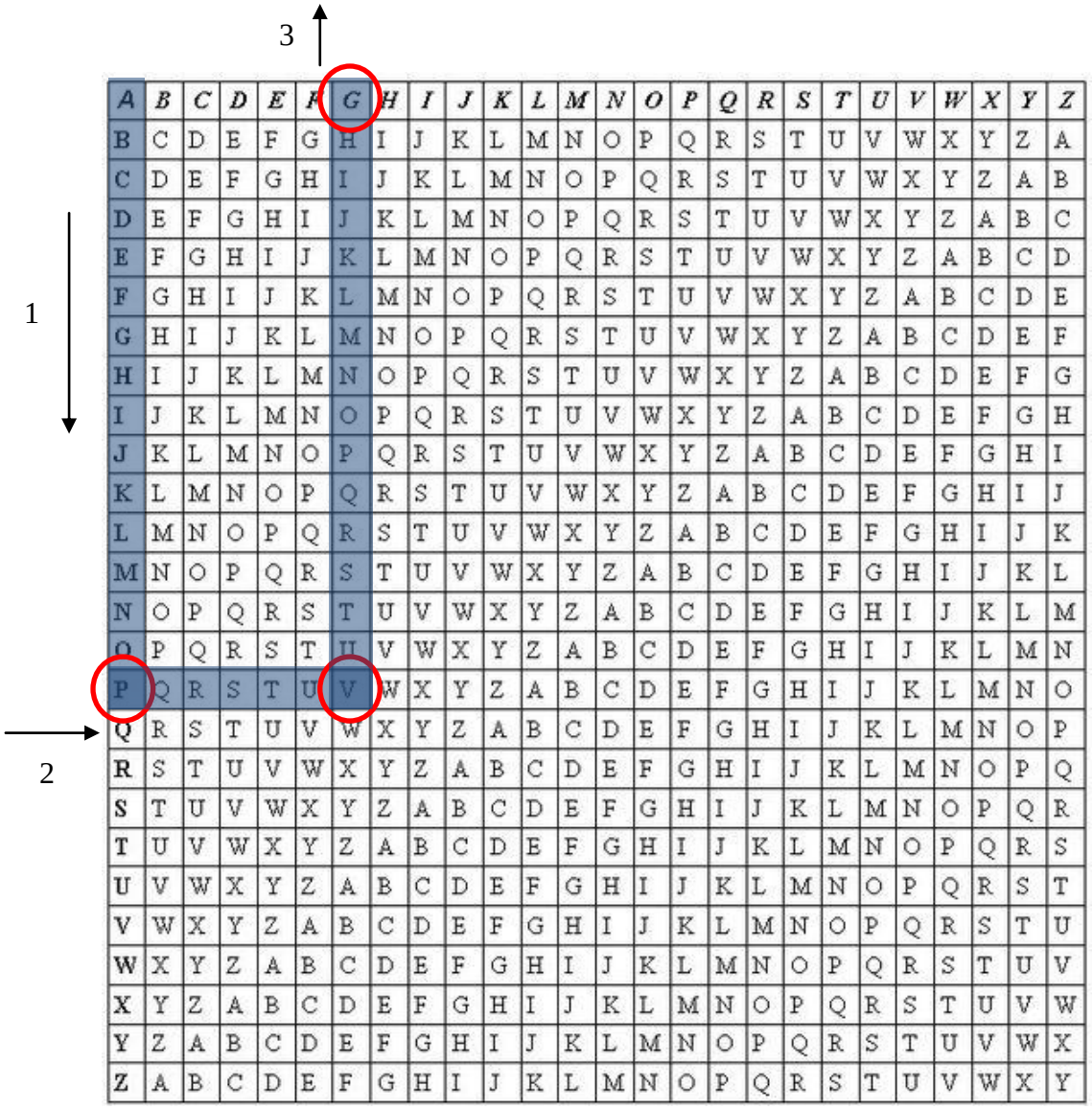
convertir tots els caràcters del missatge a xifrar. I el resultat obtingut serà una nova cadena de caràcters que representa el missatge xifrat. En el nostre exemple la codificació final seria com la que presentem a continuació:

Missatge a xifrar	**G**	**O**	**M**	**O**	**K**	**U**
Clau replicada	**P**	**R**	**O**	**P**	**P**	**R**
Missatge xifrat	**V**	**F**	**A**	**D**	**Z**	**L**

Per al procés de desencriptació el procés és molt similar però a la inversa. Ara suposem que tenim un **missatge xifrat** (partint de l'exemple anterior, el missatge que volem desxifrar seria **VFADZL**), **la clau** amb la qual s'ha xifrat el missatge original (el desxifrat, el que volem conèixer) (segons el nostre exemple la clau seria, una vegada replicada, **PROPPR**), i coneixem per complet **el diccionari** amb el qual ha estat encriptat el missatge. Aleshores, després d'haver formulat la taula de Vigenère que servirà coma a taula de conversió, caldria seguir els següents passos:

1. Cerquem el primer caràcter de la clau (caràcter P) a partir de la primera columna de la taula.
2. Després recorrem d'esquerre a dreta la fila en busca del primer caràcter del missatge a desxifrar (caràcter V).
3. Una vegada localitzat, determinem a quina columna de la taula pertany: aquesta ens dirà el caràcter desxifrat (en aquest cas seria el caràcter G).

Si repliquem aquests tres passos per a cadascun dels caràcters que formen el missatge xifrat, acabarem obtenint el missatge desxifrat, l'original (en el nostre exemple seria el missatge **GOMOKU**). El següent dibuix exemplifica el procés anterior per al cas del primer caràcter a desxifrar (els nombres de les fletxes fan referència a la numeració dels passos descrits anteriorment):



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

La implementació d'aquest algorisme es pot consultar al fitxer `Vigenere.java`, el qual està perfectament documentat per poder entendre el seu funcionament, així com quines estructures de dades s'utilitzen per a dur a terme aquesta tasca de forma correcta. Notar que en el projecte no es fa servir justament el diccionari que hem definit per formalitzar l'exemple, sinó que el diccionari emprat està definit a partir de la taula de caràcters ASCII. El nostre diccionari de caràcters és un subconjunt d'aquesta taula, el rang del qual va des del caràcter número 33 fins al caràcter 126, obtenint d'aquesta manera un diccionari format per 94 caràcters. La següent taula mostra quins són els caràcters pertanyents al diccionari:

DEC	HEX	OCT	CHAR	DEC	HEX	OCT	CH	DEC	HEX	OCT	CH	DEC	HEX	OCT	CH
0	0	000	NUL	32	20	040		64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	80	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Índex en decimal asssociat al caràcter

Caràcter que representa

Si el missatge a xifrar, o la clau contenen algun caràcter que no es trobi dins d'aquest rang, aleshores el missatge no es podrà xifrar. Al mateix temps, si la longitud de la clau és superior a la del missatge a xifrar, en quest cas tampoc podrem xifrar o desxifrar el missatge a tractar.