Ros Roger, Alexandre

Març 2023

13. Chess. Consider a board game such as Go or Chess, where the state space of possible positions on an $n \times n$ board is exponentially large and there is no guarantee that the game only lasts for a polynomial number of moves. Assume for the moment that there is no restriction on visiting the same position twice.

1. Show that, given a configuration of an $n \times n$ board, deciding whether the first player has a *forced* win strategy is in `EXP`. (`CHESS` and `GO` are well known `EXP`-complete problems).
2. Show that if we can guarantee that the game only lasts a polynomial number of moves, then such problem will be in `PSPACE`.

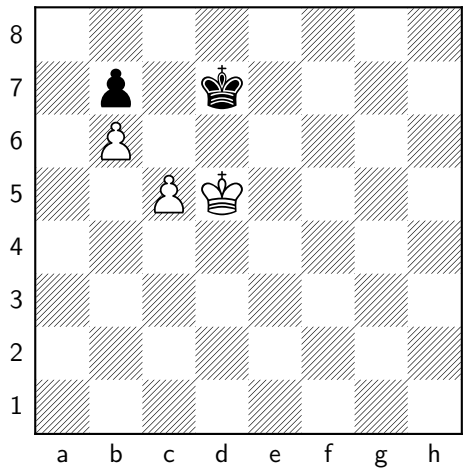
Solució a l'apartat 1.
Per a demostrar que `CHESS` $\in$ `EXPTIME`, primerament hem d'entendre a quin problema en concret ens referim. El problema consisteix en, donada una configuració legal d'un tauler d'escacs $n \times n$, determinar si les blanques (sense pèrdua de generalitat) tenen una estratègia guanyadora forçada.

Naturalment, els escacs són jugats en un escaquer de 8×8. Quan generalitzem el joc en un escaquer $n \times n$, estem suposant que el nombre de peces excepcionant els dos ♔ incrementa en una potència fraccionària de n.

Primerament, calcularem el nombre de configuracions possibles en el tauler $n \times n$. Donat que en els escacs hi ha sis peces diferents (♙ ♘ ♗ ♖ ♕ ♔) per cada jugador, el nombre de configuracions possibles ve limitat per 13^{n^2} [1], doncs cada casella pot estar ocupada per una de les dotze peces, o bé no estar pas ocupada. (Nota: moltes d'aquestes configuracions no són legals).

Donat que el nombre de configuracions és exponencial respecte la mida de l'entrada, podem dissenyar un algorisme que ens decidirà en temps exponencial el problema. Partint de la configuració inicial $\mathfrak{C}_0$, generarem totes les possibles configuracions que poden seguir immediatament a la inicial. Farem un recorregut d'aquest graf i determinarem si les blanques poden forçar una seqüència $\mathfrak{C}_0 \hookrightarrow \mathfrak{C}_1 \hookrightarrow \cdots \hookrightarrow \mathfrak{C}^*$ on $\mathfrak{C}^*$ és una posició on les blanques han guanyat, tenint en compte que les negres poden respondre amb qualsevol moviment vàlid.

Tot i així, hem de tenir en compte repeticions de les configuracions. Nosaltres considerarem que la repetició de la mateixa configuració (i mateix torn de jugador) és taules immediates (cicle). És possible que les blanques, per tal de guanyar, forcin a que la mateixa posició es repeteixi (però sent el torn de les negres). Això es coneix com a triangulació o pèrdua de "tempo". El següent problema n'és un exemple, on les blanques juguen i guanyen:



L'algorisme exponencial.

Donat que per diferents branques la mateixa configuració pot repetir-se, haurem de dissenyar un algorisme recursiu similar al Min-Max que implementi memorització (PD) per a que la complexitat temporal sigui linial respecte el nombre de configuracions diferents, prèviament calculada.

On S és una pila per a detectar cicles (evitar recursió infinita) i M és la memòria (buides per a la crida inicial):

L'algorisme es presenta a la pàgina 3 per causes d'espai

L'algorisme `ExpCHESS` farà un pruning de les branques ja visitades mitjançant la memorització. D'aquesta manera, aconseguim reduir de manera dràstica la complexitat temporal. No resulta difícil d'observar que aquest algorisme té un temps $O(C)$ on C és el nombre màxim de subrutines / configuracions que cridarem. Donat [1], l'algorisme `ExpCHESS` és $O(13^{n^2})$ i per tant `CHESS` $\in$ `EXPTIME`. Q.E.D.

Solució al puzzle.

1 ♔e5! ♔c6 2 ♔d4 ♔d7 3 ♔d5 ♔c8 4 ♔e6! ♔d8 5 ♔d6 ♔c8 6 ♔e7 ♔b8 7 ♔d7 ♔a8 8 c6

Les blanques forcen la mateixa posició inicial, ara amb el torn de les negres, després de la jugada 3. Les negres estan en zugzwang, i acaben perdent la partida davant la promoció d'un peó de blanques. Les caselles e5 i d4 serveixen de triangulació.

Solució a l'apartat 2.

Ara ens donen la garantia que el joc dels escacs en un tauler $n \times n$ podrà com a màxim tenir $p(n)$ moviments, on $p(n)$ és un polinomi. Seguidament demostrarem que, en aquest cas, `POLYCHESS` $\in$ `PSPACE`.

Podem oblidar-nos de l'algorisme `ExpCHESS` proposat anteriorment, doncs l'ús de la memorització implica emmagatzemar l'estat final de cada configuració, evidentment no estariem pas a `PSPACE`.

Aquest nou algorisme farà el mateix que l'anterior, però sense la memorització. Donat que el joc només pot durar un nombre polinòmic de jugades, això significa que en el "game-tree" la profunditat màxima de l'arbre serà $p(n)$. A més, com que S i la pila de la recursió (recordem que aquest algorisme recorre l'arbre en profunditat) només emmagatzemen la seqüència de configuracions que hem pres des de l'estat inicial i aquesta seqüència pot ésser com a màxim $p(n)$, l'algorisme només consumirà l'ordre de $O(p(n))$ cel·les de memòria, i per tant `POLYCHESS` $\in$ `PSPACE`. Q.E.D.

Algorithm 1 `ExpCHESS`: Can white force a win?

Input
C Board configuration
$w|b$ Turn of play

if $\langle C, w|b\rangle \in S$ **then** ▷ Suppose S is a global stack that initially was empty
return false
else
push $\langle C, w|b\rangle$ to S
end if

if $\langle C, w|b\rangle \in M$ **then**
pop from S
return $M[\langle C, w|b\rangle]$
end if

if w **then**
if white is checkmated or draw **then**
pop from S
set $M[\langle C, w|b\rangle] =$ false
return false
end if

for every possible configuration D reachable from C via white move **do**
Run `ExpCHESS` with input D and b
if `ExpCHESS` returns true **then**
pop from S
set $M[\langle C, w|b\rangle] =$ true
return true
end if
end for
pop from S
set $M[\langle C, w|b\rangle] =$ false
return false
end if

if b **then**
if black is checkmated **then**
pop from S
set $M[\langle C, w|b\rangle] =$ true
return true
end if
for every possible configuration D reachable from C **do**
Run `ExpCHESS` with input D and w
if `ExpCHESS` returns false **then**
pop from S
set $M[\langle C, w|b\rangle] =$ false
return false
end if
end for
pop from S
set $M[\langle C, w|b\rangle] =$ true
return true
end if