ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Τεχνητή Νοημοσύνη

6η διάλεξη (2026-27)

Ίων Ανδρουτσόπουλος

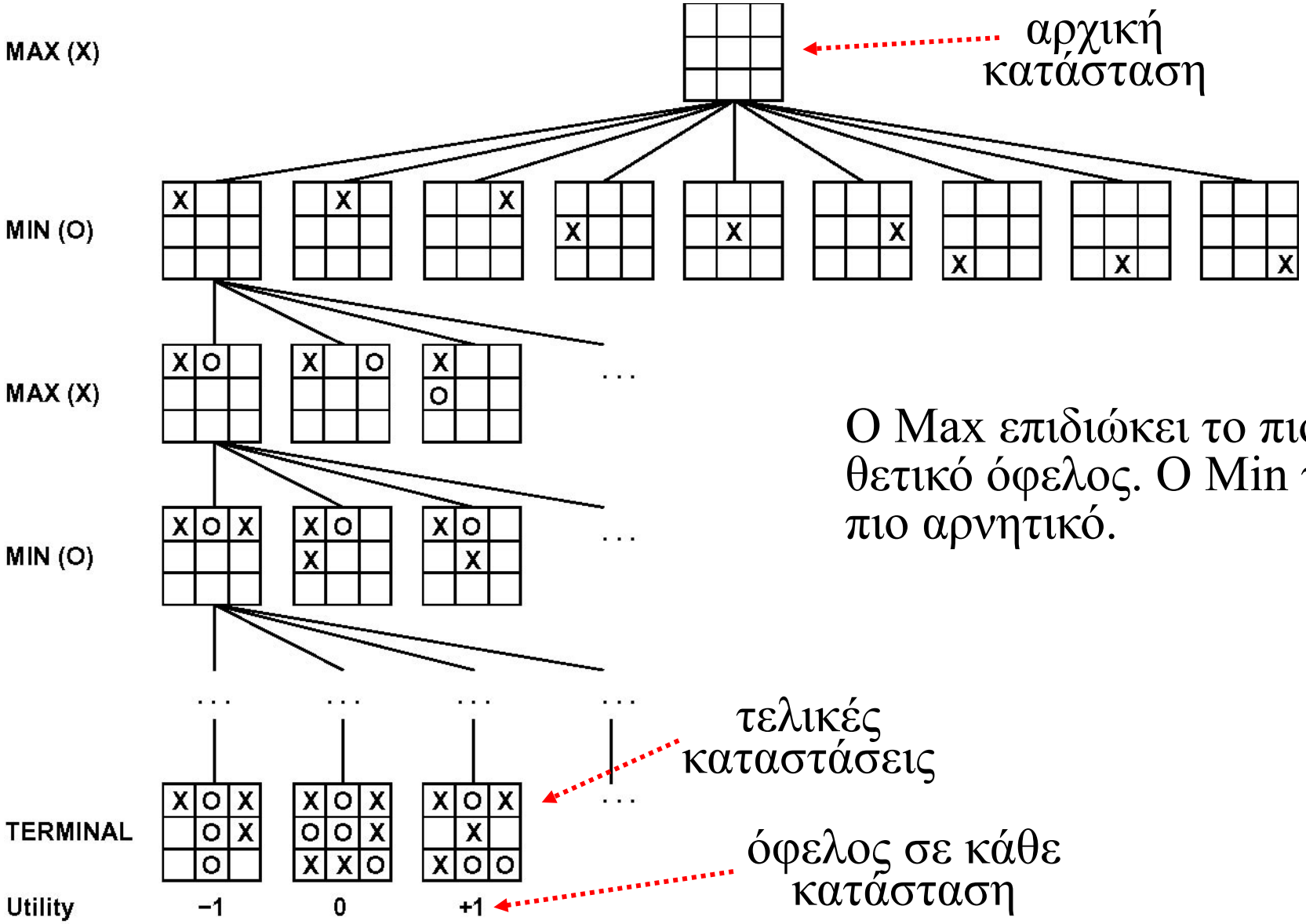
http://www.aueb.gr/users/ion/

Οι διαφάνειες αυτής της διάλεξης βασίζονται στα βιβλία *Τεχνητή Νοημοσύνη* των Βλαχάβα κ.ά., 3η έκδοση, Β. Γκιούρδας Εκδοτική, 2006 και *Artificial Intelligence – A Modern Approach* των S. Russel και P. Norvig, 2η και 4η έκδοση, Prentice Hall, 2003 και 2020. Τα περισσότερα σχήματα των διαφανειών προέρχονται από αντίστοιχες διαφάνειες του δεύτερου βιβλίου.

Τι θα ακούσετε σήμερα

- Αναζήτηση με αντιπάλους.
- Παιχνίδια μηδενικού και μη μηδενικού αθροίσματος.
- Αλγόριθμος MiniMax.
- Πριόνισμα α-β.
- Παιχνίδια με παράγοντα τύχης.

Παιχνίδια αντιπάλων

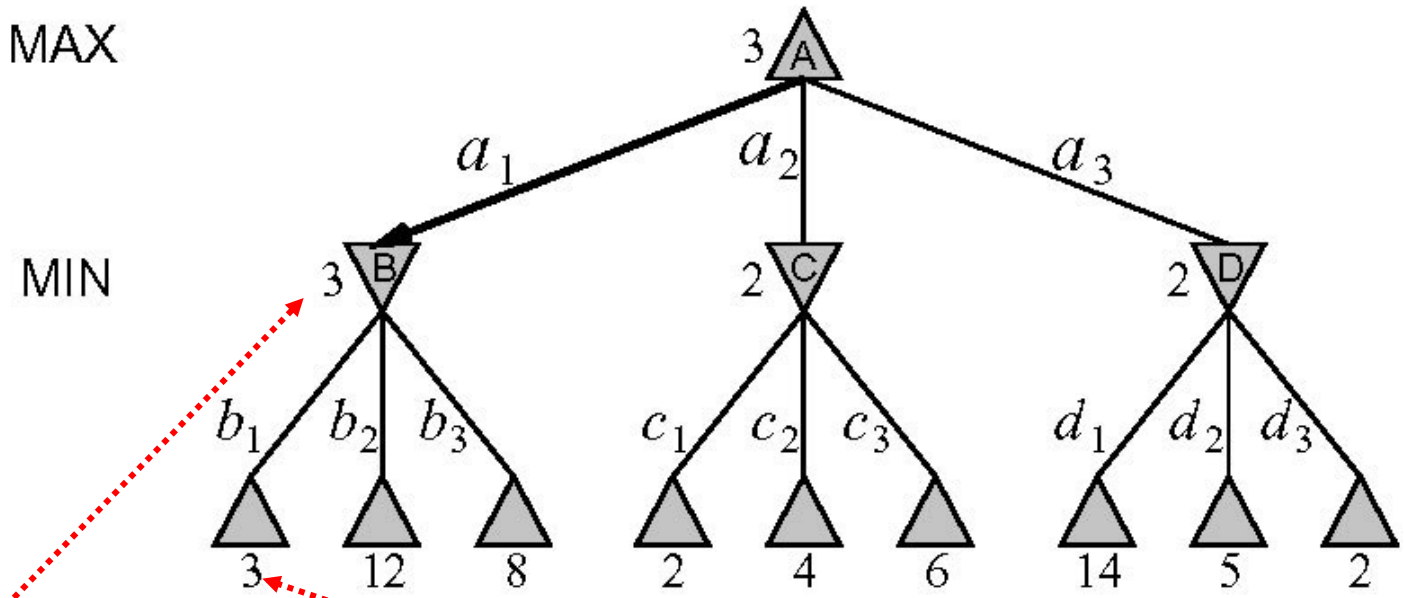


Ο Max επιδιώκει το πιο θετικό όφελος. Ο Min το πιο αρνητικό.

Παιχνίδια αντιπάλων

- Συνάρτηση **οφέλους:** Αξιολογεί κάθε **τελική** κατάσταση. Επιστρέφει αριθμητική τιμή.
  - Στο σκάκι +1 αν κερδίζει ο Max, –1 αν κερδίζει ο Min. Γενικότερα, π.χ. συνάρτηση του χρηματικού κέρδους.
- Παιχνίδια **μηδενικού αθροίσματος**:
  - Σε κάθε τελική κατάσταση, το άθροισμα του οφέλους των δύο παικτών είναι 0.
  - Π.χ. στο σκάκι, όταν κερδίζει ο ένας (+1), χάνει ο άλλος (–1).
- Αρχικές απλοποιήσεις:
  - Μόνο **δύο αντίπαλοι**. Παίζει ένας παίκτης κάθε φορά.
  - **Πλήρως γνωστό** περιβάλλον και συνέπειες κινήσεων.
  - **Δεν υπάρχουν τυχαίες** μεταβάσεις (π.χ. ζάρια).

Τιμή Minimax ενός κόμβου



Όφελος αν φτάσουμε σε αυτή την τελική κατάσταση.

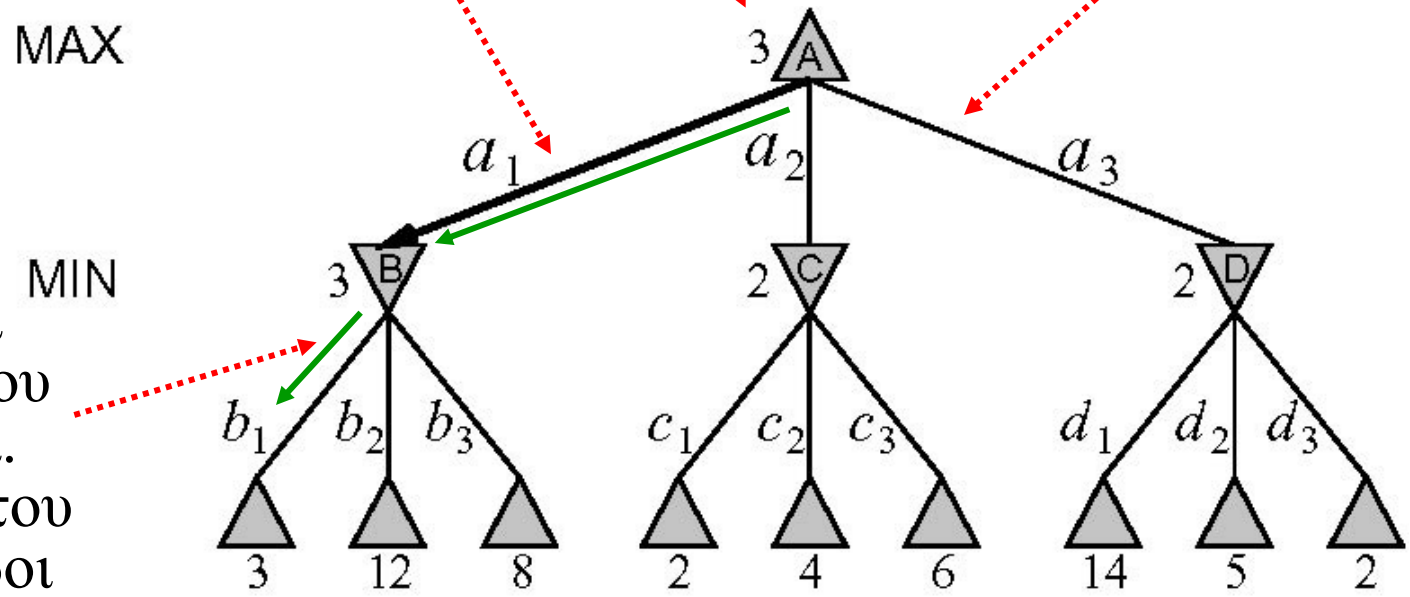
Αν το παιχνίδι φτάσει σε αυτή την κατάσταση, θα τελειώσει με όφελος 3, γιατί ο Min θα επιλέξει στην επόμενη κίνηση την τελική κατάσταση με το μικρότερο όφελος.

Τιμή Minimax ενός κόμβου

Αν φτάσουμε σε αυτή την κατάσταση, το παιχνίδι θα τελειώσει με όφελος 3, γιατί ο Max θα επιλέξει την κίνηση που οδηγεί στο μεγαλύτερο όφελος.

Αν ο Max κάνει αυτή την κίνηση, το παιχνίδι θα τελειώσει με όφελος 3.

Αν ο Max κάνει αυτή την κίνηση, το παιχνίδι θα τελειώσει με όφελος 2.



Μονοπάτι κινήσεων που θα επιλεγεί. Κατά μήκος του όλοι οι κόμβοι έχουν την **ίδια τιμή**.

Τιμή Minimax ενός κόμβου

- **minimax-value(n)** =
  - utility(n), αν n τερματικός κόμβος,
  - $\max_{s \in succ(n)}$minimax-value(s), αν στον n παίζει ο Max,
  - $\min_{s \in succ(n)}$minimax-value(s), αν στον n παίζει ο Min,
  - όπου succ(n) οι κόμβοι των καταστάσεων-παιδιών.
- **Αν φτάσει** το παιχνίδι στον **κόμβο n**, το παιχνίδι **θα τελειώσει με όφελος minimax-value(n)**.
  - Αν επιλέξουν στη συνέχεια και οι δύο παίκτες τις κινήσεις που τους συμφέρουν περισσότερο, δηλαδή αυτές με τις μέγιστες ή ελάχιστες, αντίστοιχα, τιμές Minimax.
  - Αν επιλέγουμε σύμφωνα με τις τιμές Minimax (με πλήρες δέντρο) και ο αντίπαλος όχι, το παιχνίδι θα τελειώσει με όφελος ίσο με minimax-value(n) ή καλύτερο (για εμάς).

Αλγόριθμος Minimax με DFS

- Εξερευνούμε **όλο το χώρο αναζήτησης** από την κατάσταση στην οποία βρισκόμαστε με **DFS**.
  - Δεν σταματάμε όταν φτάσουμε σε τελική κατάσταση.
- Κατά την εξερεύνηση υπολογίζουμε:
  - το **όφελος στα φύλλα**,
  - την **τιμή Minimax** των μη τερματικών κόμβων.
- **Ποια κίνηση επιλέγουμε** (από τη ρίζα, όπου παίζουμε εμείς); Αυτή που μας οδηγεί στο βέλτιστο όφελος.
  - Αν ο αντίπαλος δεν επιλέξει τις βέλτιστες κινήσεις, το όφελός μας θα είναι μεγαλύτερο (για πλήρες δέντρο, χωρίς ευρετικές).
- Πολυπλοκότητα:
  - **Χώρος: O(bm)** (ή O(m) για DFS με οπισθοδρόμηση).
  - **Χρόνος: $O(b^m)$**.
  - Συνήθως **δεν είναι εφικτό** να φτιάξουμε το **πλήρες δέντρο**.

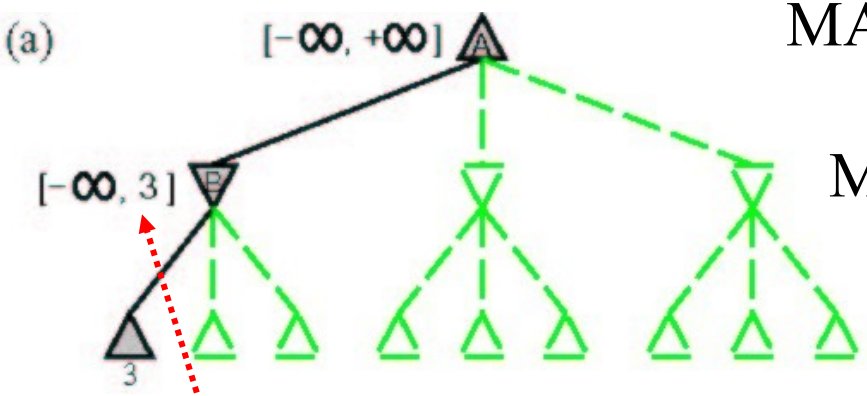
Minimax με DFS

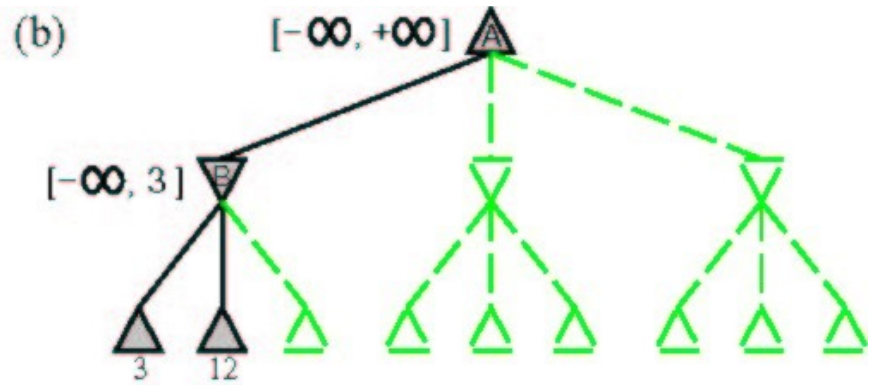
```
function minimax(state)
  v ← max-value(state) // Θεωρούμε ότι ο υπολογιστής είναι ο Max.
  return κίνηση που οδηγεί στην s ∈ succ(state) με τιμή v

function max-value(state)  // Τιμή Minimax ενός κόμβου όπου παίζει ο Max.
  if terminal-test(state) then return utility(state)
  v ← −άπειρο
  for s ∈ succ(state) // Αν χρησιμοποιούμε DFS με οπισθοδρόμηση, παράγουμε
                      // μόνο ένα παιδί τη φορά και επιστρέφουμε για το επόμενο.
    v ← max(v, min-value(s))  // Κρατάμε τη μεγαλύτερη τιμή Minimax των παιδιών.
  return v

function min-value(state) // Τιμή Minimax ενός κόμβου όπου παίζει ο Min.
  if terminal-test(state) then return utility(state)
  v ← +άπειρο
  for s ∈ succ(state)
    v ← min(v, max-value(s))  // Κρατάμε τη μικρότερη τιμή.
  return v
```

Πριόνισμα του δέντρου

MAX

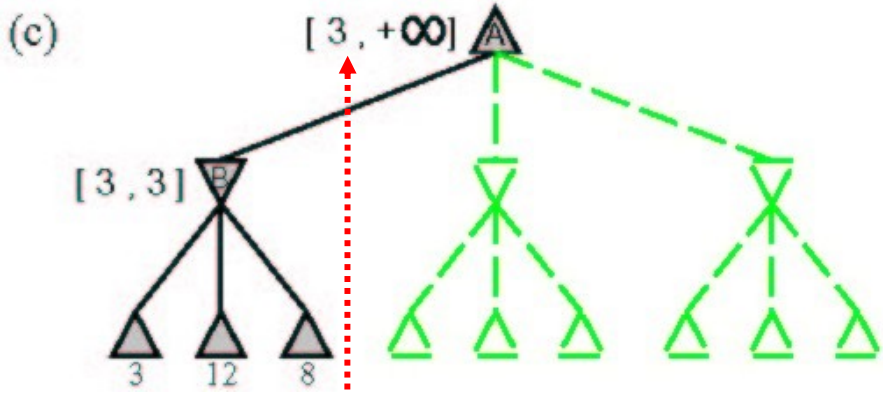
MIN

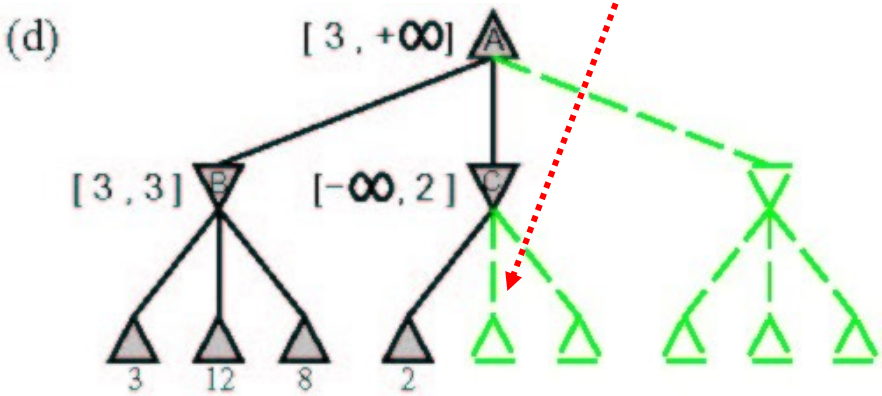




Για κόμβους MIN, θα έχω συχνά άνω φράγμα (≤ β).

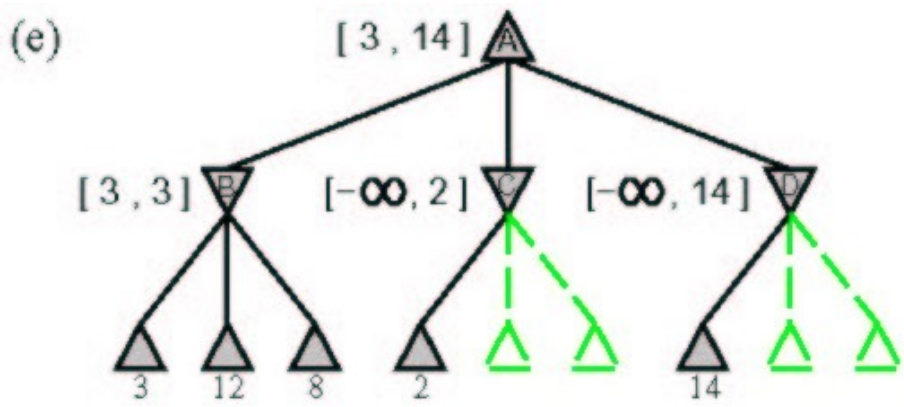
Κλαδιά που δε χρειάζεται να εξερευνήσω (**πριόνισμα**).

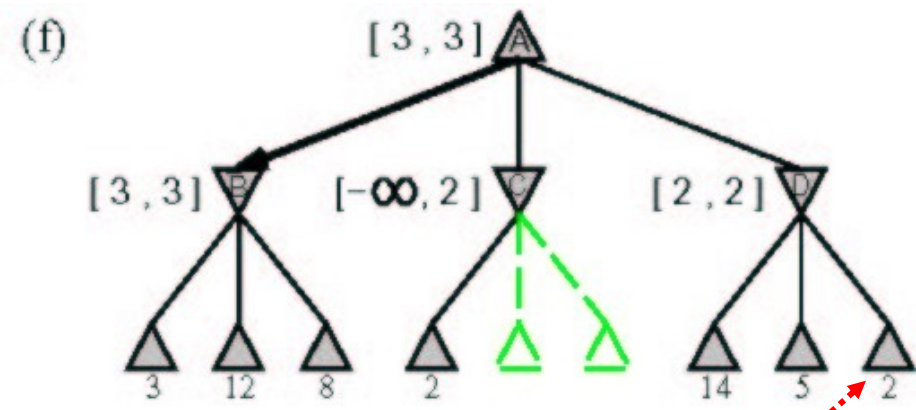




Για κόμβους MAX, θα έχω συχνά κάτω φράγμα (≥ α).

Πριόνισμα του δέντρου



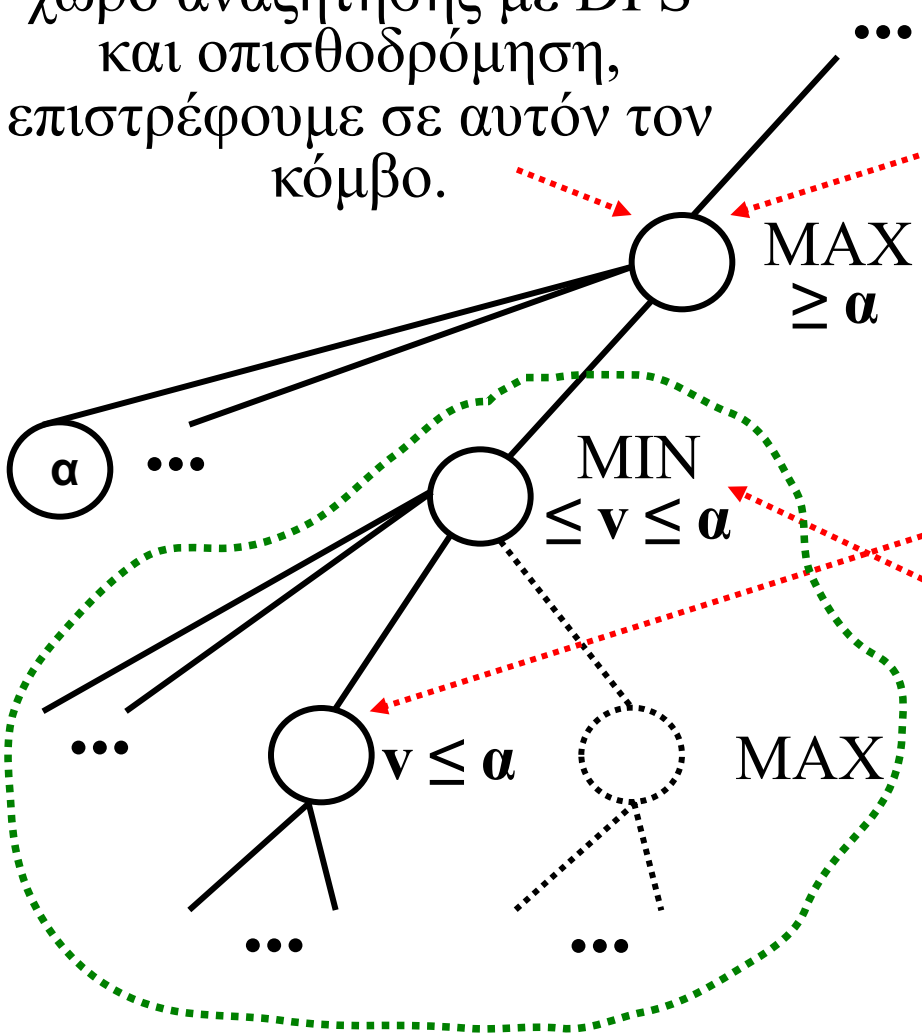


Αν είχα εξερευνήσει πρώτα το δεξί παιδί (όφελος 2), θα μπορούσα να είχα πριονίσει τα κλαδιά προς τα άλλα δύο παιδιά (οφέλη 14 και 5), όπως πριν.

Αξίζει να προσπαθώ να εξερευνώ **πρώτα τα παιδιά που φαίνονται προτιμότερα** (βάσει ευρετικών) **για τον παίκτη που παίζει στον κόμβο-πατέρα** (π.χ. στο σκάκι, να εξερευνώ πρώτα κινήσεις που αιχμαλωτίζουν σημαντικά κομμάτια του αντιπάλου).

Πριόνισμα με α (καθώς παράγουμε παιδιά κόμβου MIN)

Καθώς εξερευνούμε το χώρο αναζήτησης με DFS και οπισθοδρόμηση, επιστρέφουμε σε αυτόν τον κόμβο.

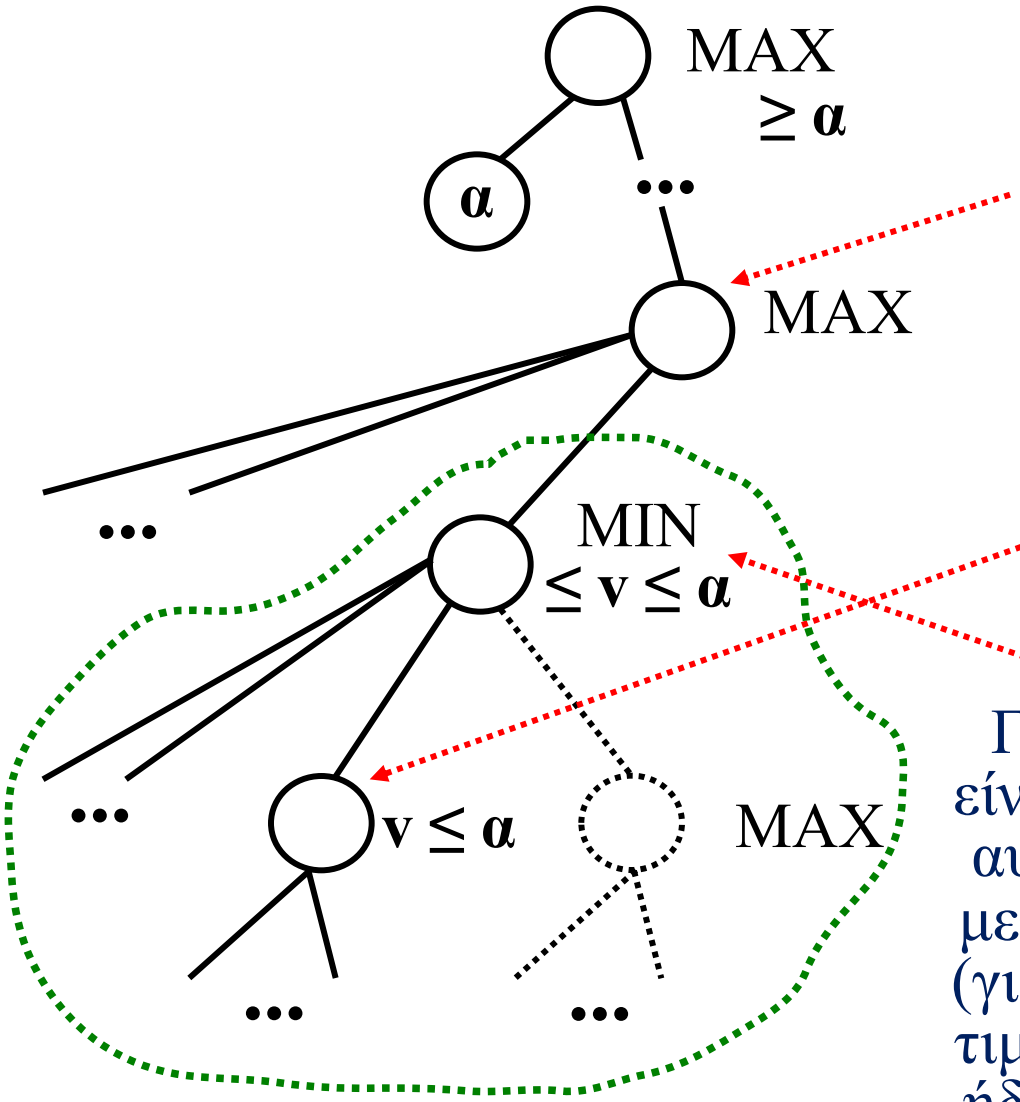


Από τα αριστερότερα υποδέντρα του που έχουμε εξερευνήσει ξέρουμε ότι η τιμή αυτού του κόμβου θα είναι ≥ **α**. (Έχουμε ήδη βρει παιδί του με τιμή α.)

Συνεχίζοντας την εξερεύνηση φτάνουμε εδώ. Αν η τιμή αυτού του κόμβου είναι v ≤ α, μπορούμε να πριονίσουμε τα υπόλοιπα αδέλφια του και όλο το υποδέντρο του MIN.

Γιατί η τιμή του κόμβου MIN θα είναι ≤ v ≤ α. Ο από πάνω κόμβος MAX μπορεί να αγνοήσει το υποδέντρο αυτού του κόμβου MIN, αφού έχει στη διάθεσή του άλλη επιλογή που οδηγεί σε μεγαλύτερη ή ίση τιμή α.

Πριόνισμα με α (γενικότερο)



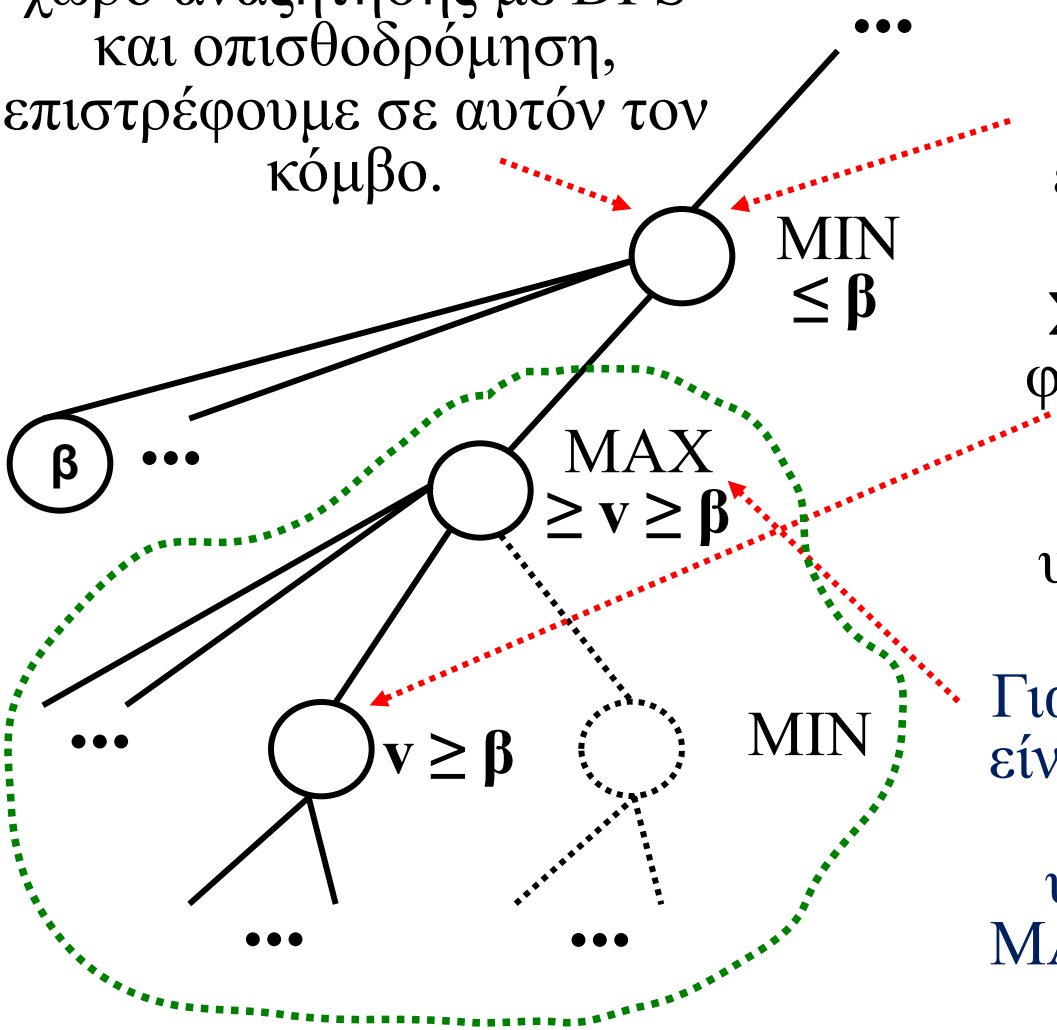
Γενικότερα, το α μπορεί να είναι το καλύτερο (μέγιστο) κάτω φράγμα (≥ **α**) που έχουμε βρει για αυτόν τον κόμβο ή οποιονδήποτε **πρόγονό** του.

Αν η τιμή αυτού του κόμβου είναι v ≤ α, μπορούμε πάλι να πριονίσουμε τα υπόλοιπα αδέλφια του και όλο το υποδέντρο του MIN.

Γιατί η τιμή του κόμβου MIN θα είναι ≤ v ≤ α. Δηλαδή το υποδέντρο αυτού του κόμβου MIN μπορεί να μεταδώσει στον πάνω κόμβο MAX (για τον οποίο έχουμε το φράγμα α) τιμή ≤ α, ενώ εκείνος ο κόμβος έχει ήδη στη διάθεσή του άλλη επιλογή που του δίνει τιμή ≥ α.

Πριόνισμα με β (καθώς παράγουμε παιδιά κόμβου MAX)

Καθώς εξερευνούμε το χώρο αναζήτησης με DFS και οπισθοδρόμηση, επιστρέφουμε σε αυτόν τον κόμβο.

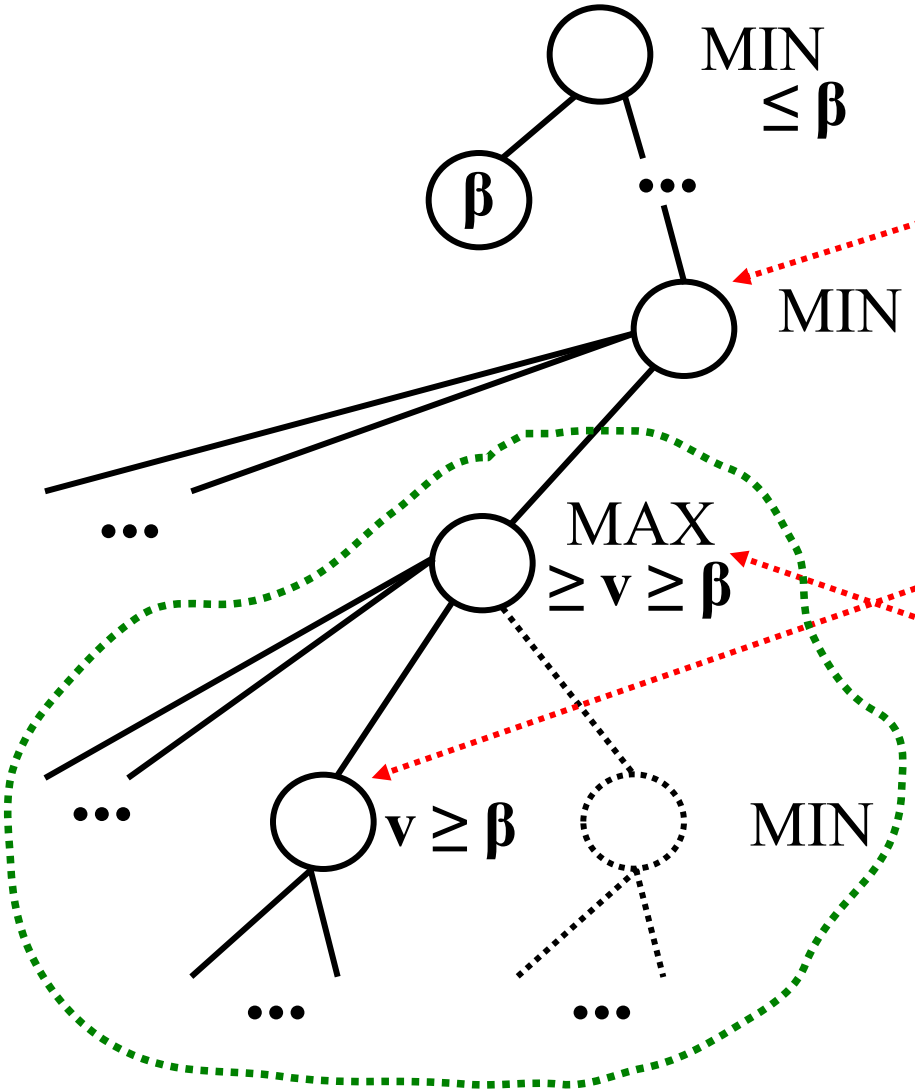


Από τα αριστερότερα υποδέντρα του που έχουμε εξερευνήσει ξέρουμε ότι η τιμή αυτού του κόμβου θα είναι ≤ **β**. (Έχουμε ήδη βρει παιδί του με τιμή β.)

Συνεχίζοντας την εξερεύνηση φτάνουμε εδώ. Αν η τιμή αυτού του κόμβου είναι v ≥ β, μπορούμε να πριονίσουμε τα υπόλοιπα αδέλφια του και όλο το υποδέντρο του MAX.

Γιατί η τιμή του κόμβου MAX θα είναι ≥ v ≥ β. Ο από πάνω κόμβος MIN μπορεί να αγνοήσει το υποδέντρο αυτού του κόμβου MAX, αφού έχει στη διάθεσή του άλλη επιλογή που οδηγεί σε μικρότερη ή ίση τιμή β.

Πριόνισμα με β (γενικότερο)



Γενικότερα, το β μπορεί να είναι το καλύτερο (ελάχιστο) πάνω φράγμα (**≤ β**) που έχουμε βρει για αυτόν τον κόμβο ή οποιονδήποτε **πρόγονό** του.

Αν η τιμή αυτού του κόμβου είναι v ≥ β, μπορούμε πάλι να πριονίσουμε τα υπόλοιπα αδέλφια του και όλο το υποδέντρο του MAX.

Γιατί η τιμή του κόμβου MAX θα είναι ≥ v ≥ β. Δηλαδή το υποδέντρο αυτού του κόμβου MAX μπορεί να μεταδώσει στον πάνω κόμβο MIN (για τον οποίο έχουμε το φράγμα β) τιμή ≥ β, ενώ εκείνος ο κόμβος έχει ήδη στη διάθεσή του άλλη επιλογή που του δίνει τιμή ≤ β.

Αλγόριθμος α-β

- Όπως ο Mimimax, αλλά καθώς παράγουμε τα παιδιά κάθε κόμβου:
  - Καθώς παράγουμε τα **παιδιά ενός κόμβου MIN** (διαφάνεια 14), πριονίζουμε χρησιμοποιώντας **το καλύτερο φράγμα α** των MAX προγόνων του.
  - Καθώς παράγουμε τα **παιδιά ενός κόμβου MAX** (διαφάνεια 16), πριονίζουμε χρησιμοποιώντας το **καλύτερο φράγμα β** των MIN προγόνων του.
- Για ευκολία σε κάθε κόμβο **αποθηκεύουμε τα καλύτερα φράγματα α και β:**
  - **Σε κάθε κόμβο MIN** (δεν έχει δικό του α): αποθηκεύουμε το καλύτερο φράγμα α των MAX προγόνων του και το καλύτερο από τα β φράγματα του ίδιου του κόμβου και των MIN προγόνων του.
  - **Σε κάθε κόμβο MAX** (δεν έχει δικό του β): αποθηκεύουμε το καλύτερο φράγμα β των MIN προγόνων του και το καλύτερο από τα α φράγματα του ίδιου του κόμβου και των MAX προγόνων του.

Αλγόριθμος α-β

```
function alpha-beta(state)
  v ← max-value(state, -άπειρο, +άπειρο) // Ο υπολογιστής είναι ο Max.
  return κίνηση που οδηγεί στην s ∈ succ(state) με τιμή v

function max-value(state, α, β) // Τιμή Minimax κόμβου MAX. Διαφάνεια 16.
  if terminal-test(state) then return utility(state)
  v ← −άπειρο
  for s ∈ succ(state)
    v ← max(v, min-value(s, α, β)) // Τα α,β του κόμβου MAX όπου βρισκόμαστε.
    if v ≥ β then return v  // Πρόνισμα με β. Δεν έχει σημασία τι επιστρέφουμε.
    α ← max(α, v)  // Ενημερώνουμε το α του κόμβου MAX όπου βρισκόμαστε.
  return v

function min-value(state, α, β) // Τιμή Minimax κόμβου MIN. Διαφάνεια 14.
  if terminal-test(state) then return utility(state)
  v ← +άπειρο
  for s ∈ succ(state)
    v ← min(v, max-value(s, α, β)) // Τα α,β του κόμβου MIN όπου βρισκόμαστε.
    if v ≤ α then return v  // Πρόνισμα με α. Δεν έχει σημασία τι επιστρέφουμε.
    β ← min(β, v)  // Ενημερώνουμε το β του κόμβου MIN όπου βρισκόμαστε.
  return v
```

Περιορισμός βάθους και ευρετικές

- Ακόμα **και με πριόνισμα α-β**, συνήθως **δεν μπορούμε** να κατασκευάσουμε το **πλήρες δέντρο αναζήτησης**.
- Στην πράξη κατασκευάζουμε το δέντρο αναζήτησης μέχρι ένα **μέγιστο βάθος *l***.
  - DFS με περιορισμό βάθους.
- Εφαρμόζουμε στους κόμβους **μέγιστου βάθους** μια **ευρετική συνάρτηση** αξιολόγησης.
  - Η ευρετική προσπαθεί να εκτιμήσει το όφελος που θα προκύψει αν βρεθούμε σε καθέναν από αυτούς τους κόμβους.
  - Δεν μπορούμε να εφαρμόσουμε τη συνάρτηση οφέλους, γιατί εφαρμόζεται μόνο σε τελικές καταστάσεις.

Ευρετικές συναρτήσεις αξιολόγησης

- Στο σκάκι:
  - **Συνολική αξία κομματιών** κάθε αντιπάλου: π.χ. βασιλιάς: 10, άλογο: 5, πιόνι: 1.
  - **Θέσεις** κομματιών: π.χ. κάθε κομμάτι στα 4 κεντρικά τετράγωνα παίρνει επιπλέον 2 πόντους.
  - **Απειλές**: π.χ. για κάθε απειλή κομματιού αντιπάλου 3 επιπλέον πόντοι και για απειλή προς βασιλιά 20 πόντοι.
- Συνήθως χρησιμοποιούμε **συνδυασμό ευρετικών**:
  - Π.χ. **γραμμικό συνδυασμό**: $w_1 \cdot h_1(n) + \ldots + w_k \cdot h_k(n)$.
  - Μπορούμε να χρησιμοποιήσουμε τεχνικές **μηχανικής μάθησης** για να μάθουμε τα βάρη.
  - Παραδείγματα εκπαίδευσης: οι καταστάσεις που έχουμε συναντήσει στο παρελθόν (π.χ. παίζοντας με τον εαυτό μας). Ιδιότητες: οι ευρετικές. Επιθυμητή απόκριση: η πιθανότητα να κερδίσουμε αν βρεθούμε στη συγκεκριμένη κατάσταση. *(Θα δούμε αργότερα άσκηση μελέτης/εργασία.)*

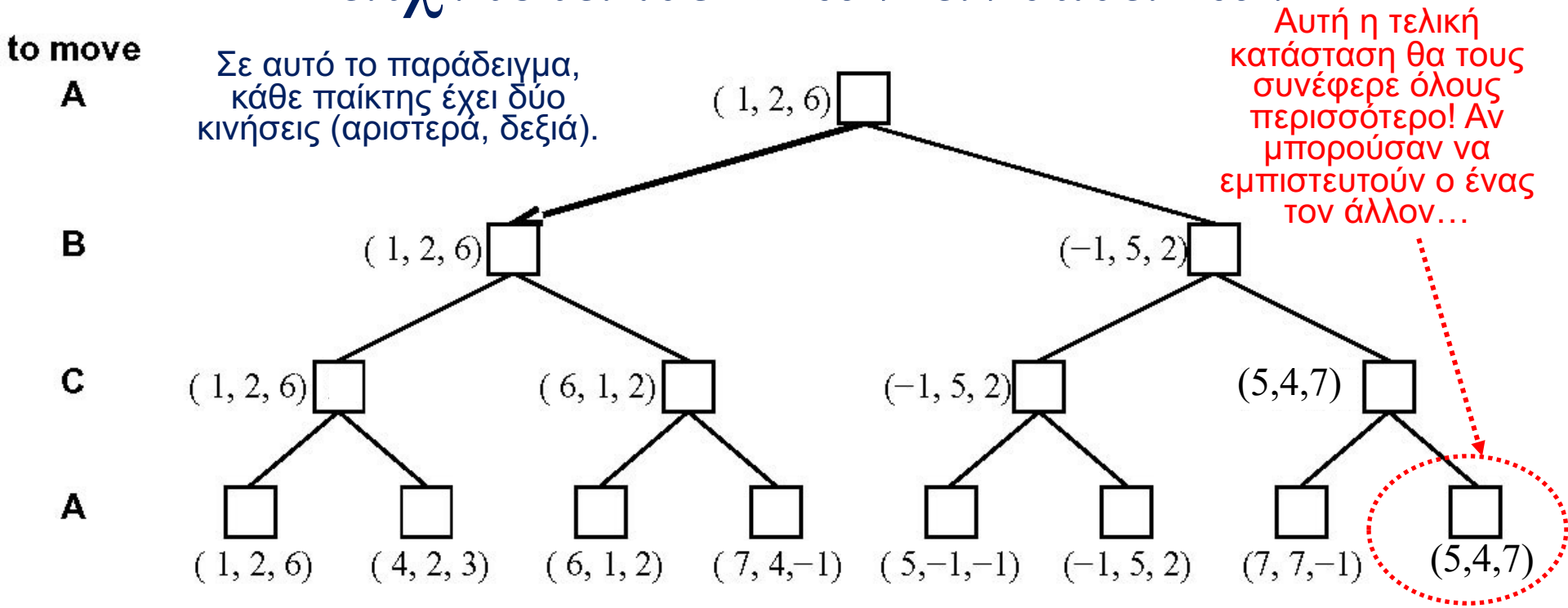
Τιμή Minimax με ευρετικές

- **minimax-value(n)** =
  - h(n), αν n κόμβος μέγιστου βάθους,
  - $\max_{s \in succ(n)}$minimax-value(s), αν στον n παίζει ο Max,
  - $\min_{s \in succ(n)}$minimax-value(s), αν στον n παίζει ο Min.
- Αποτελεί **εκτίμηση** του οφέλους στο οποίο θα οδηγηθούμε αν φτάσουμε στον κόμβο n.
  - Αν επιλέξουν στη συνέχεια και οι δύο τις κινήσεις τους βάσει των τιμών Minimax που υπολογίσαμε.
  - Στην πράξη, όμως, ο **αντίπαλος** (ακόμα κι αν είναι το ίδιο πρόγραμμα, με την ίδια ευρετική) θα ψάξει μέχρι ένα **επίπεδο πιο κάτω** όταν έρθει η σειρά του, άρα θα αποφασίσει βάσει άλλου δέντρου και μπορεί να μην κάνει την κίνηση που αναμέναμε.

Άλλες βελτιώσεις

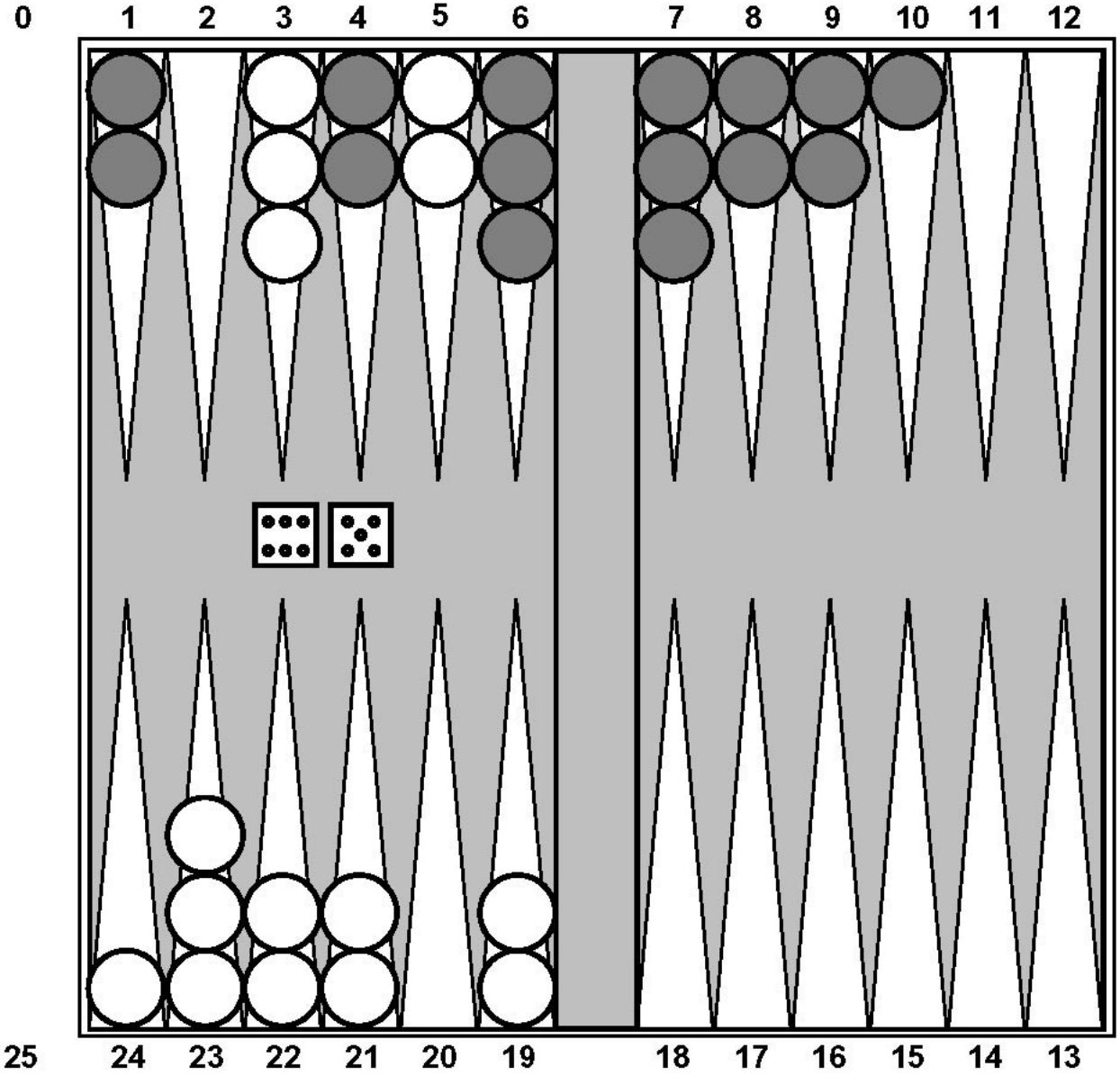
- **Κλειστό σύνολο** σε **κάθε επίπεδο**.
  - Διαφορετικοί συνδυασμοί κινήσεων μπορεί να οδηγούν στις ίδιες καταστάσεις. Αν έχουμε ήδη συναντήσει την **ίδια κατάσταση στο ίδιο βάθος**, γνωρίζουμε ήδη την τιμή της, δεν χρειάζεται να ξανα-υπολογίσουμε το υπο-δέντρο της.
  - Αλλά σε άλλο βάθος, η τιμή της ίδιας κατάστασης μπορεί να είναι διαφορετική όταν ψάχνουμε ως περιορισμένο βάθος.
  - **Υπερβαίνουμε** το μέγιστο βάθος αν μια κατάσταση φαίνεται (βάσει ευρετικών) κρίσιμη ή σταματάμε σε **μικρότερο βάθος** αν δε φαίνεται ενδιαφέρουσα.
- Χρήση «**βίβλων**» κλασικών **ανοιγμάτων/τελειωμάτων**.
  - Κλασικές **ακολουθίες αρχικών κινήσεων**, προϋπολογισμένες **βέλτιστες κινήσεις** για **λίγα εναπομείναντα** κομμάτια.

Παιχνίδια πολλών αντιπάλων



- Η **συνάρτηση οφέλους** επιστρέφει διάνυσμα.
  - Ο 1ο αριθμός είναι το όφελος για τον Α, ο 2ος το όφελος για τον Β κ.ό.κ.
  - Εδώ έχουμε παίγνιο μη μηδενικού αθροίσματος.
- Η **τιμή Minimax** κάθε κόμβου είναι το διάνυσμα του παιδιού με το μεγαλύτερο όφελος για αυτόν που παίζει.

Παιχνίδια με παράγοντα τύχης

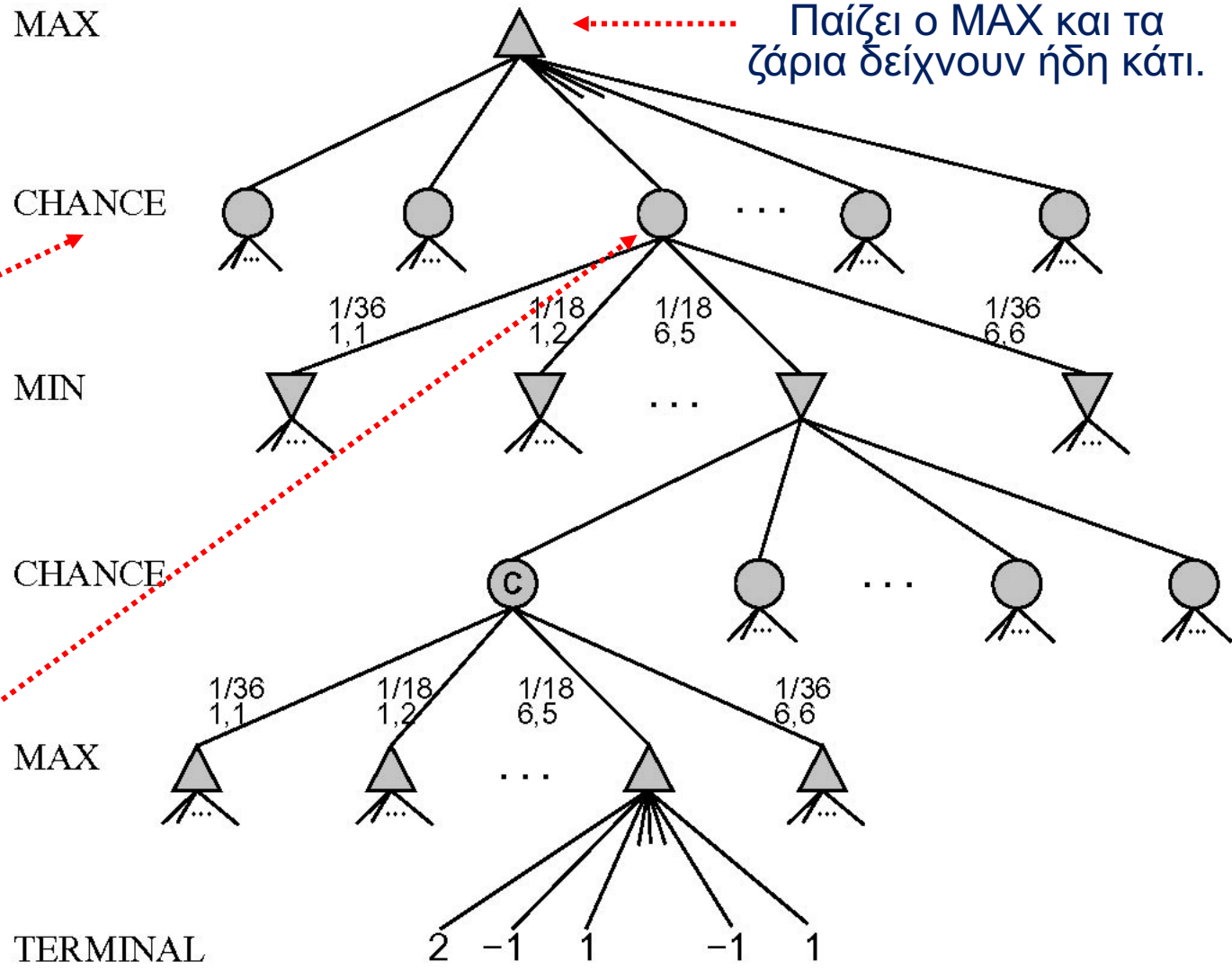


Δέντρο Minimax για τάβλι

Παίζει ο MAX και τα ζάρια δείχνουν ήδη κάτι.

Αφού παίξει ο MAX, παίζει ένας τρίτος «παίκτης» που αλλάζει μόνο τι δείχνουν τα ζάρια.

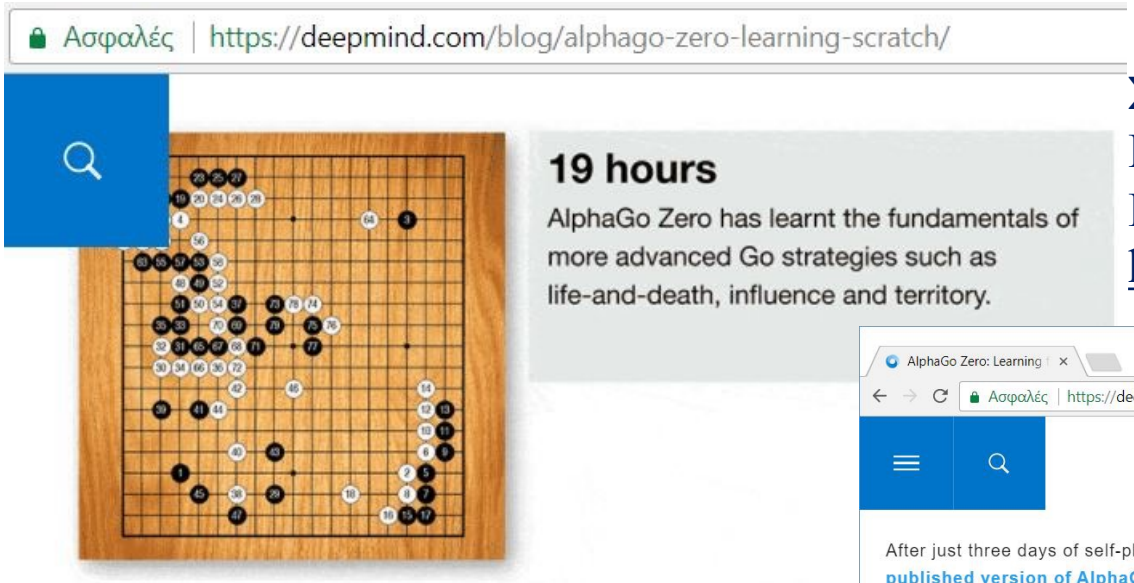
Το αποτέλεσμα της κίνησης του τρίτου «παίκτη» είναι τυχαίο.



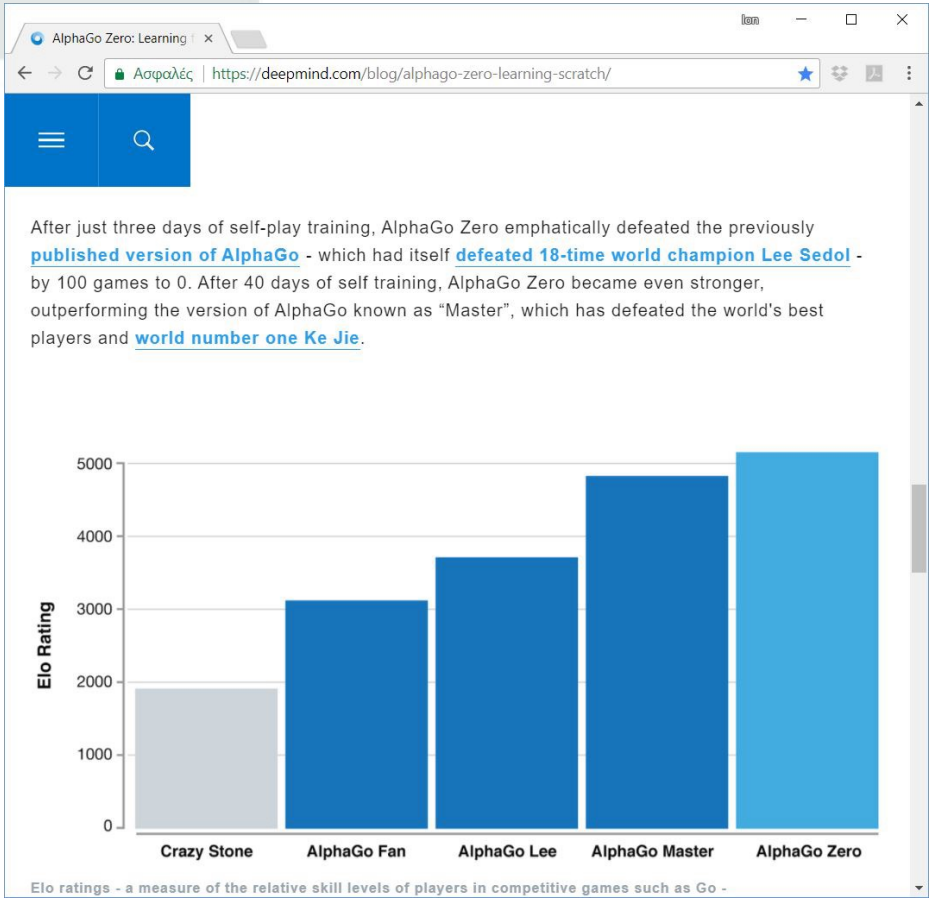
Αναμενόμενη τιμή Minimax

- **exp-minimax-value(n)** =
  - h(n), αν n κόμβος μέγιστου βάθους,
  - $\max_{s \in succ(n)}$exp-minimax-value(s), αν στον n παίζει ο Max,
  - $\min_{s \in succ(n)}$exp-minimax-value(s), αν στον n παίζει ο Min,
  - $\Sigma_{s \in succ(n)} p(s) \cdot$ exp-minimax-value(s), αν n κόμβος τύχης.
- Εκτίμηση του **αναμενόμενου** οφέλους στο οποίο θα οδηγηθούμε αν φτάσουμε στον κόμβο n.
- Χρονική πολυπλοκότητα Minimax με ζάρια: $\mathbf{O(b^m k^m)}$.
  - Όπου k ο αριθμός διακριτών δυνατών ζαριών.
  - Αντί για $O(b^m)$, όταν δεν έχουμε ζάρια.
  - Κάθε κόμβος MIN ή MAX έχει ουσιαστικά τώρα bk MAX ή MIN εγγόνια (παιδιά είναι οι κόμβοι τύχης), αντίστοιχα.
  - Μειώνεται το βάθος μέχρι το οποίο αντέχουμε να ψάξουμε.

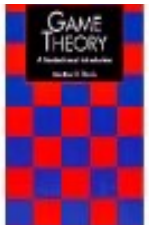
Το AlphaGo της DeepMind



Συνδυάζει **Αναζήτηση Δέντρου Monte Carlo**, **Βαθιά Μάθηση** και **Ενισχυτική Μάθηση**. Βλ. https://en.wikipedia.org/wiki/AlphaGo



Θα **πρέπει να μελετήσετε τις διαφάνειες** αυτής της διάλεξης **και τις ασκήσεις μελέτης** της (βλ. έγγραφα μαθήματος στο e-class) **πριν από την επόμενη διάλεξη και πριν από το αντίστοιχο εργαστήριο**. **Γράψτε μόνοι σας** τις **λύσεις των ασκήσεων μελέτης** (ακόμα και αν δίνονται οι λύσεις). Επίσης, γράψτε μόνοι σας τους **ορισμούς**, **αλγορίθμους**, **παραδείγματα** εκτέλεσης των αλγορίθμων και λοιπούς **υπολογισμούς** των διαφανειών. Ανατρέξτε στη **βιβλιογραφία** για πρόσθετες εξηγήσεις και γνώσεις (ιδιαίτερα όσοι ενδιαφέρεστε ιδιαίτερα για την ΤΝ).

Βιβλιογραφία

- Russel & Norvig (4η έκδοση): κεφάλαιο 5 ως και ενότητα 5.3.2, ενότητα 5.5.
  - Όσοι ενδιαφέρονται μπορούν να διαβάσουν προαιρετικά και τα υπόλοιπα τμήματα του κεφαλαίου 5, ιδιαίτερα την ενότητα 5.4 (Αναζήτηση Δέντρου Monte Carlo).
- Βλαχάβας κ.ά.: κεφάλαιο 5.
- Aggarwal: σχετική ύλη υπάρχει στο κεφάλαιο 3.
- Poole & Mackworth: σχετική ύλη υπάρχει στο κεφάλαιο 14.
- Η (μαθηματική) μελέτη της συμπεριφοράς αντιπάλων είναι το αντικείμενο της Θεωρίας Παιγνίων.
  - Μια ενδιαφέρουσα μη τεχνική εισαγωγή στη Θεωρία Παιγνίων είναι το βιβλίο «Game Theory: A Nontechnical Introduction» του M.D. Davis, Dover Publications, που διαβάζεται πολύ εύκολα.