Κεφάλαιο 6

Ικανοποίηση Περιορισμών

Τεχνητή Νοημοσύνη - Β' Έκδοση

Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου

Ικανοποίηση Περιορισμών

- Ένα πρόβλημα ικανοποίησης περιορισμών (constraint satisfaction problem αποτελείται από:
  - Ένα σύνολο *n* μεταβλητών $V_1, V_2,...,V_n$,
  - Ένα σύνολο *n* πεδίων τιμών $D_1, ...D_n$, που αντιστοιχούν σε κάθε μεταβλητή έτσι ώστε $V_i \in D_i$, και
  - Ένα σύνολο σχέσεων (περιορισμών) $C_1, C_2,...C_m$ όπου $C_i(V_k,...,V_m)$ μια σχέση μεταξύ των μεταβλητών του προβλήματος.

- Ανάλογα με το πόσες μεταβλητές περιλαμβάνει ένας περιορισμός χαρακτηρίζεται ως:
  - *μοναδιαίος* (*unary*)
  - *δυαδικός* (*binary*)
  - *ανώτερης* τάξης (*higher order*)

Λύση Προβλήματος Περιορισμών

- Λύση αποτελεί μια ανάθεση τιμών στις μεταβλητές του προβλήματος, τέτοια ώστε να ικανοποιούνται οι περιορισμοί, δηλαδή:

Περιορισμοί που αφορούν τα πεδία των μεταβλητών — *Περιορισμοί του προβλήματος*

$$V_1 = d_1, V_2 = d_2, \ldots V_n = d_n \wedge \; d \in D_1 \wedge d_2 \in D_2 \wedge \ldots \wedge d_i \in D_n \qquad \wedge C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

- Τα προβλήματα ικανοποίησης περιορισμών χαρακτηρίζονται από μεγάλο αριθμό μεταβλητών και πιθανών τιμών (πεδίων)
  - Φαινόμενο της συνδυαστικής έκρηξης (combinatorial explosion).
  - Απαιτείται μείωση του χώρου αναζήτησης.

Παράδειγμα Προβλήματος Ικανοποίησης Περιορισμών

- Σειρά με την οποία θα εισαχθούν τα προϊόντα Α, Β, Γ, Δ μέσα σε ένα βιομηχανικό μύλο.
  - Το προϊόν Α πρέπει να εισαχθεί στο μύλο μετά από το Δ, το Γ πριν από το Β, και το Β πριν από το Α.

$V_A \neq V_B$ και $V_A \neq V_\Gamma$ και $V_A \neq V_\Delta$ *έτσι ώστε να μην πάρουν*

$V_B \neq V_\Gamma$ και $V_B \neq V_\Delta$ και $V_\Gamma \neq V_\Delta$ *δύο προϊόντα την ίδια σειρά*

$V_A > V_\Delta$ *το προϊόν Α μετά από το Δ*

$V_\Gamma < V_B$ *το προϊόν Γ πριν από το Β*

$V_B < V_A$ *το προϊόν Β πριν από το Α*

- Τρεις δυνατές λύσεις:

$V_A = 4, V_B = 2, V_\Gamma, = 1, V_\Delta = 3$ Δηλαδή η σειρά είναι: Γ, Β, Δ, Α

$V_A = 4, V_B = 3, V_\Gamma, = 1, V_\Delta = 2$ Δηλαδή η σειρά είναι: Γ, Δ, Β, Α

$V_A = 4, V_B = 3, V_\Gamma, = 2, V_\Delta = 1$ Δηλαδή η σειρά είναι: Δ, Γ, Β, Α

Παραγωγή και Δοκιμή

- Γεννήτρια λύσεων και ελεγκτής.
- Η γεννήτρια λύσεων παράγει λύσεις:

  $V_A = 1, V_B = 1, V_Γ, = 1, V_Δ = 1$

  $V_A = 1, V_B = 1, V_Γ, = 1, V_Δ = 2$

  …

  $V_A = 4, V_B = 4, V_Γ, = 4, V_Δ = 4$
- Ο ελεγκτής ελέγχει τις παραγόμενες λύσεις.
- Αν η γεννήτρια χρησιμοποιεί ως πληροφορία ότι το προϊόν Α παρασκευάζεται πάντα τελευταίο, παράγει μόνο τις λύσεις:

  $V_A = 4, V_B = 1, V_Γ, = 1, V_Δ = 1$

  $V_A = 4, V_B = 2, V_Γ, = 1, V_Δ = 1$

  …

  $V_A = 4, V_B = 4, V_Γ, = 4, V_Δ = 4$
- Μείωση πιθανών λύσεων από $4^4 = 256$ αντί $4^3 = 64$.

Αλγόριθμοι Επιδιόρθωσης

- Στον κλασικό αλγόριθμο παραγωγής και δοκιμής εξετάζεται ένας μεγάλος χώρος αναζήτησης.
- Πρόταση νέων αλγορίθμων οι οποίοι προσπαθούν να βελτιώσουν μια προτεινόμενη λύση σταδιακά.
- Αλγόριθμοι *επιδιόρθωσης* (*repair algorithms*).
  - *Αναρρίχηση λόφου* (*hill-climbing*)
  - *Ευριστικός αλγόριθμος των ελαχίστων συγκρούσεων* (min conflicts heuristic)
- Οι αλγόριθμοι επιδιόρθωσης έχουν εφαρμοστεί με μεγάλη επιτυχία.

Αναρρίχηση Λόφου (Hill-Climbing)

- Αλγόριθμος επιδιόρθωσης HC:

1. **Ανέθεσε στις μεταβλητές τυχαίες τιμές από τα πεδία τιμών τους.**
2. **Αν οι τιμές των μεταβλητών δεν παραβιάζουν τους περιορισμούς του προβλήματος τότε επέστρεψε τις τιμές αυτές ως λύση.**
3. **Εξέτασε για κάθε μεταβλητή όλες τις δυνατές τιμές που μπορεί να πάρει.**
   i. **Αν κάποια από τις τιμές που εξετάστηκαν ελαχιστοποιεί το πλήθος των περιορισμών που παραβιάζονται, ανέθεσε την τιμή της στην αντίστοιχη μεταβλητή και επέστρεψε στο βήμα 2.**
   ii. **Αν δε υπάρχει τιμή που να ελαχιστοποιεί το πλήθος των περιορισμών, τότε επέστρεψε στο βήμα 1 (τοπικό ελάχιστο – ο αλγόριθμος ξεκινά από μια νέα τυχαία ανάθεση τιμών).**

- Εξετάζει ένα μεγάλο πλήθος "γειτονικών" καταστάσεων.
- Μπορεί να "πέσει" σε τοπικό ελάχιστο.

Ευριστικός αλγόριθμος των ελαχίστων συγκρούσεων

Min conflicts heuristic

- Ο αλγόριθμος ξεκινά από μια τυχαία ανάθεση τιμών και μεταβάλλει την τιμή μιας μεταβλητής έτσι ώστε η νέα τιμή να παραβιάζει λιγότερους περιορισμούς.

```
1. Ανέθεσε στις μεταβλητές τυχαίες τιμές από τα πεδία τιμών τους.
2. Αν οι τιμές των μεταβλητών δεν παραβιάζουν τους περιορισμούς του
   προβλήματος τότε επέστρεψε τις τιμές αυτές ως λύση.
3. Εξέτασε για μια τυχαία μεταβλητή όλες τις δυνατές τιμές που μπορεί να
   πάρει.
   i. Αν κάποια από τις τιμές για τη μεταβλητή που εξετάστηκαν μειώνει
      το πλήθος των περιορισμών που παραβιάζονται, ανέθεσε την τιμή της
      στη μεταβλητή.
   ii. Αν δεν υπάρχει τιμή που να μειώνει το πλήθος των περιορισμών που
      παραβιάζονται, τότε επέλεξε μια τιμή που να διατηρεί τον ίδιο
      αριθμό περιορισμών.
   iii. Αν δεν υπάρχει ούτε τέτοια τιμή, τότε άφησε την τιμή της
        εξεταζόμενης μεταβλητής.
4. Επέστρεψε στο βήμα 2.
```

- Ο αλγόριθμος δεν εξετάζει μεγάλο πλήθος γειτονικών κόμβων.
- Διατηρεί το πρόβλημα του εγκλωβισμού σε κάποιο τοπικό ελάχιστο.

Κλασικοί Αλγόριθμοι Αναζήτησης

- Οι κλασικοί αλγόριθμοι αναζήτησης είναι δυνατό να χρησιμοποιηθούν και για την επίλυση των προβλημάτων ικανοποίησης περιορισμών.
- Αναπαράσταση
  - Μια *κατάσταση* αποτελείται από τις μεταβλητές του προβλήματος.
  - Υπάρχει ένας μόνο *τελεστής*, ο οποίος αντιστοιχεί στην ανάθεση μιας τιμής σε μια *μη-δεσμευμένη μεταβλητή* (μεταβλητή στην οποία δεν έχει ανατεθεί τιμή).
  - *Αρχική κατάσταση*: όλες οι μεταβλητές είναι μη-δεσμευμένες.
  - *Τελική κατάσταση*: ελέγχεται αν έχει γίνει ανάθεση τιμών σε όλες τις μεταβλητές, καθώς επίσης και αν ικανοποιούνται όλοι οι περιορισμοί του προβλήματος.
- Επίλυση με:
  - Κλασικούς αλγορίθμους αναζήτησης.
  - Κλασικούς ευριστικούς αλγορίθμους αναζήτησης

Ευριστικοί Αλγόριθμοι στο Πρόβλημα Ικανοποίησης Περιορισμών

- Εφαρμόζονται οι κλασικοί ευριστικοί αλγόριθμοι αναζήτησης
- Αλγόριθμος αναζήτησης πρώτα στο καλύτερο (Best First)
  - *Αρχή της συντομότερης αποτυχίας* (*first fail principle*).
  - Σε περίπτωση ισοπαλίας επιλέγεται η μεταβλητή που συμμετέχει σε περισσότερους περιορισμούς (*most constraint principle*).
- Βελτίωση έναντι των τυφλών αλγορίθμων αναζήτησης.
- Δεν επιλύουν σε ικανοποιητικό χρόνο προβλήματα μεγάλου μεγέθους.
  - Μη-εκμετάλλευση της πληροφορίας των περιορισμών. (*a posteriori* έλεγχος)
- A posteriori έλεγχος + συνδυαστική έκρηξη = εξαιρετικά χρονοβόρα επίλυση προβλημάτων.

Αλγόριθμοι Ελέγχου Συνέπειας

- Εκμετάλλευση πληροφορίας που υπάρχει στους περιορισμούς για μείωση του χώρου αναζήτησης.
- A priori έλεγχος συνέπειας τιμών.
- Βασική ιδέα των αλγορίθμων της κατηγορίας:

Απαλοιφή από τα αρχικά πεδία των μεταβλητών εκείνων των τιμών οι οποίες δεν μπορούν να συμμετέχουν στην τελική λύση.

- *Έλεγχος συνέπειας* (*consistency check*)
  - Διαγραφή από το πεδίο κάθε μεταβλητής εκείνων των τιμών οι οποίες είναι ασυνεπείς ως προς κάποιο περιορισμό.
  - Αναφέρονται και ως αλγόριθμοι διήθησης τιμών (filtering algorithms).

Παράδειγμα Απαλοιφής τιμών από τα πεδία των Μεταβλητών (1/2)

- Πρόβλημα σειράς παρασκευής των προϊόντων για ένα βιομηχανικό μύλο.
- Οι περιορισμοί είναι:

$V_A \neq V_B$	(C1)	$V_B \neq V_Γ$	(C4)	$V_A > V_Δ$	(C7)
$V_A \neq V_Γ$	(C2)	$V_B \neq V_Δ$	(C5)	$V_Γ < V_B$	(C8)
$V_A \neq V_Δ$	(C3)	$V_Γ \neq V_Δ$	(C6)	$V_B < V_A$	(C9)

- Τα πεδία τιμών των μεταβλητών:

  $V_A \in \{1,2,3,4\}$

  $V_B \in \{1,2,3,4\}$

  $V_Γ \in \{1,2,3,4\}$

  $V_Δ \in \{1,2,3,4\}$

- Λόγω C9 ($V_B < V_A$):

  $V_A \in \{2,3,4\}$

  $V_B \in \{1,2,3\}$

  $V_Γ \in \{1,2,3,4\}$

  $V_Δ \in \{1,2,3,4\}$

Παράδειγμα Απαλοιφής τιμών από τα πεδία των Μεταβλητών (2/2)

- Λόγω $V_Γ < V_B$ (C8):

  $V_A \in \{2,3,4\}$ $V_B \in \{2,3\}$

  $V_Γ \in \{1,2\}$ $V_Δ \in \{1,2,3,4\}$

- Λόγω $V_A > V_Δ$ (C7):

  $V_A \in \{2,3,4\}$ $V_B \in \{2,3\}$

  $V_Γ \in \{1,2\}$ $V_Δ \in \{1,2,3\}$

- Λόγω του $V_B < V_A$ (C9):

  $V_A \in \{3,4\}$

  $V_B \in \{2,3\}$

  $V_Γ \in \{1,2\}$

  $V_Δ \in \{1,2,3\}$

- Οι πιθανοί συνδυασμοί γίνονται $2\times2\times2\times3=24$, από 256 που υπήρχαν αρχικά.

Γράφος Περιορισμών

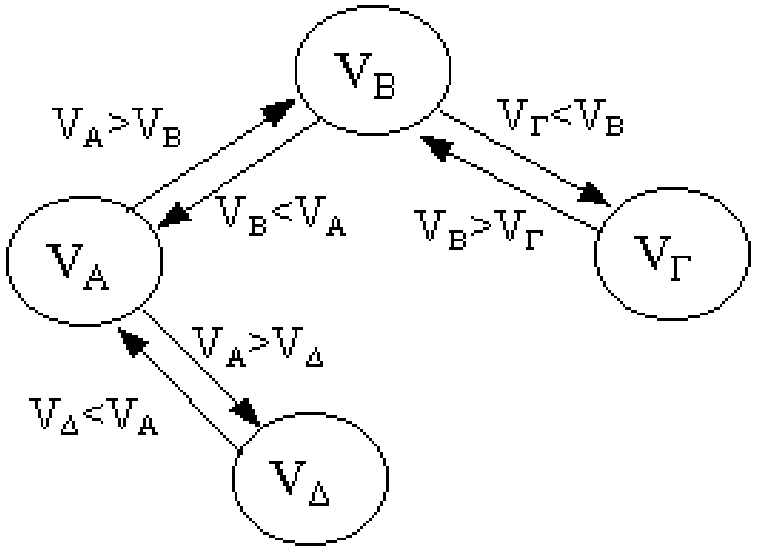
- Το πρόβλημα αναπαρίσταται ως γράφος (*γράφος περιορισμών* - *constraint graph*).
  - τα *τόξα* (*arcs*) αναπαριστούν περιορισμούς
  - οι *κόμβοι* (*nodes*) που αναπαριστούν τις μεταβλητές.

Περιορισμοί:

$V_A > V_\Delta$

$V_\Gamma < V_B$

$V_B < V_A$



Αλγόριθμοι συνέπειας/διήθησης τιμών

- *Βαθμός συνέπειας* (*degree of consistency*).
  - Πόσες ασυνεπείς τιμές αφαιρούν από τα πεδία
  - Βαθμός συνέπειας είναι αντιστρόφως ανάλογος με τον απαιτούμενο χρόνο εκτέλεσης.
- *Αλγόριθμος συνέπειας κόμβου* (*Node Consistency*).
  - Μοναδιαίοι περιορισμοί.
- *Αλγόριθμοι συνέπειας τόξου* (*Arc Consistency*-AC),
  - Δυαδικοί περιορισμοί
  - Διάφοροι αλγόριθμοι συνέπειας τόξου, όπως οι AC-3, AC-4, AC-5, AC-6, κλπ,
  - Η δυσκολία που παρουσιάζουν οι αλγόριθμοι της κατηγορίας:
    - Διαγραφή μιας τιμής οδηγεί σε αλλαγές στα πεδία άλλων μεταβλητών.
    - Μετά από κάθε διαγραφή ασυνεπούς τιμής πρέπει να επανεξεταστούν τα πεδία των "άμεσα" συνδεδεμένων μεταβλητών
- *Αλγόριθμοι συνέπειας μονοπατιού (path consistency algorithms).*
  - Υψηλό υπολογιστικό κόστος.
- Υπάρχουν και άλλες κατηγορίες αλγορίθμων συνέπειας,
- Διάδοση περιορισμών (constraint propagation).

Ο Αλγόριθμος AC3

- Ο απλούστερος αλγόριθμος συνέπειας τόξου είναι ο AC3.
- Έστω οι μεταβλητές V_1, V_2,..V_n με τιμές d_1, d_2,…,d_n από τα πεδία τιμών των μεταβλητών D_1, D_2,…,D_n ($d_1 \in D_1$, $d_2 \in D_2$,… $d_n \in D_n$) και ένα σύνολο περιορισμών $C(V_i,V_j)$ για τις μεταβλητές αυτές, οι οποίοι αναπαριστώνται ως τόξα (V_i,V_j).
- Για συντομία, κάθε τόξο (V_i,V_j) αναφέρεται ως (i,j).

Επανέλαβε τα ακόλουθα βήματα μέχρι το Q να γίνει κενό:

1. **Επέλεξε ένα τόξο (i,j) και διέγραψε το από το Q**
2. **Για κάθε τιμή d_i του πεδίου της μεταβλητής V_i έλεγξε αν υπάρχει τουλάχιστον μία τιμή d_j του πεδίου της μεταβλητής V_j τέτοια ώστε να ικανοποιεί το περιορισμό $C(V_i,V_j)$ που αντιστοιχεί στο τοξο (i, j).**
3. **Αν δεν υπάρχει τέτοια τιμή d_j τότε αφαίρεσε την τιμή d_i από το πεδίο τιμών της V_i. Αν το πεδίο τιμών της V_i είναι κενό τότε τερμάτισε με αποτυχία.**
4. **Αν έχει μεταβληθεί το πεδίο τιμών της V_i τότε πρόσθεσε στο σύνολο Q όλα τα τόξα (k,i), που αντιστοιχούν στους περιορισμούς $C(V_k,V_i)$, για $k \neq i$.**

Χαρακτηριστικά του αλγορίθμου AC-3

- Προϋποθέτει δυαδικούς περιορισμούς.
  - Μετασχηματισμός σε πρόβλημα δυαδικών περιορισμών (*binarization*).
- Μη-πληρότητα
  - Στο προηγούμενο παράδειγμα στο πεδίο τιμών της μεταβλητής V_A παρέμεινε η τιμή 3:

    VA ∈ {3,4}

    VB ∈ {2,3}

    VΓ ∈ {1,2}

    VΔ ∈ {1,2,3}
  - Οι αλγόριθμοι συνέπειας τόξου δεν απαλείφουν *όλες* τις ασυνεπείς τιμές.
- Επίλυση προβλήματος περιορισμών = αλγόριθμοι ελέγχου συνέπειας τόξου σε συνδυασμό με αλγόριθμο αναζήτησης.

Εξασφάλιση Λύσης Χωρίς Αναζήτηση

- Είναι δυνατό να βρεθεί λύση με χρήση μόνο αλγορίθμων ελέγχου συνέπειας;

K-συνέπεια

Ένας γράφος περιορισμών είναι K-συνεπής (K-consistent) εάν για κάθε K-1 μεταβλητές που ικανοποιούν τους περιορισμούς υπάρχει μια μεταβλητή K με τέτοιο πεδίο ώστε να ικανοποιούνται ταυτόχρονα όλοι τους οι περιορισμοί που συνδέουν τις K μεταβλητές.

Ένας γράφος είναι ισχυρά K-συνεπής (strongly K- consistent) εάν για κάθε L ≤ K, είναι L-συνεπής.

- Ο αλγόριθμος συνέπειας κόμβου εξασφαλίζει ότι ο γράφος είναι ισχυρά 1-συνεπής.
- Οι αλγόριθμοι συνέπειας τόξου εξασφαλίζουν ισχυρή 2-συνεπεία.
- Προφανώς σε ένα γράφο με N κόμβους, εάν εξασφαλισθεί ότι ο γράφος είναι ισχυρά N-συνεπής:
  - Λύση χωρίς αναζήτηση.
  - Υψηλό υπολογιστικό κόστος εφαρμογής για K>2.

Συνδυάζοντας Αναζήτηση και Αλγορίθμους Διήθησης Τιμών

- Συνδυασμός αλγορίθμων διήθησης και αναζήτησης καθώς:
  - Αλγόριθμοι συνέπειας: μη-πλήρεις αλλά αποδοτικοί.
  - Κλασικοί αλγόριθμοι αναζήτησης: πλήρεις/μη-αποδοτικοί.
- Βασική ιδέα των αλγορίθμων της κατηγορίας:

Μείωση του χώρου αναζήτησης με την χρήση ενός αλγορίθμου συνέπειας πριν από κάθε βήμα ανάθεσης τιμών (a priori pruning).

- Έτσι η λύση μπορεί να βρεθεί σε σημαντικά μικρότερο χρόνο με κάποιον κλασικό αλγόριθμο αναζήτησης.
- Υπάρχουν τρεις βασικοί τρόποι.
  - Διαφέρουν στο βαθμό ελέγχου των πεδίων των μεταβλητών σε κάθε βήμα.
  - Πριν την εκκίνηση της διαδικασίας αναζήτησης εφαρμόζεται ένας αλγόριθμος συνέπειας.

Συνδυασμοί Διήθησης και Αναζήτησης Λύσης

- Ο *προοπτικός έλεγχος* (*forward checking*)
  - Απαλείφει τιμές από τα πεδία των μη-δεσμευμένων μεταβλητών που συνδέονται άμεσα με περιορισμούς με την μεταβλητή στην οποία μόλις ανατέθηκε τιμή.
  - Παραμένει μεγάλος αριθμός ασυνεπών τιμών στα πεδία,
  - Χαμηλό υπολογιστικό κόστος κάθε βήματος.

- Ο αλγόριθμος *έγκαιρης μερικής εξέτασης* (*Partial Look Ahead*)
  - Κατευθυντική συνέπεια (directional consistency) σε κάθε βήμα.
  - Παραμένουν στα πεδία των μεταβλητών μη συνεπείς τιμές.

- Ο αλγόριθμος *έγκαιρης πλήρους εξέτασης* (*Full Look Ahead*) ή *διατήρησης συνέπειας τόξου* (Maintaining Arc Consistency - MAC).
  - Πλήρης συνέπεια τόξου σε κάθε βήμα.
  - Αφαιρεί το μεγαλύτερο αριθμό ασυνεπών τιμών από τους τρεις.
  - Υψηλό υπολογιστικό κόστος

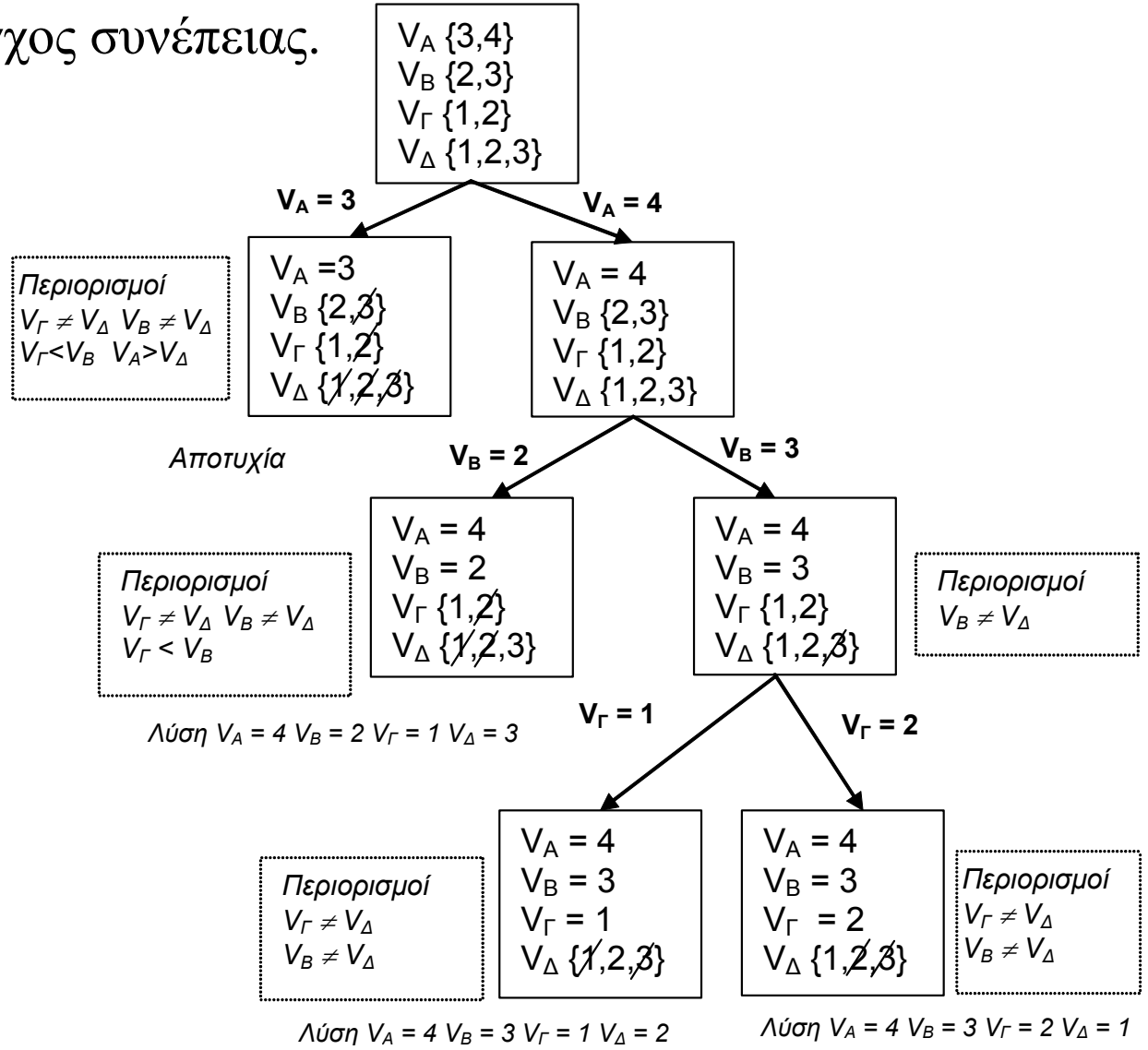
Αλγόριθμος Διατήρησης Συνέπειας Τόξου

- Ο ολοκληρωμένος αλγόριθμος διατήρησης συνέπειας τόξου για την επίλυση προβλημάτων περιορισμών είναι:

1. **Για κάθε περιορισμό αφαίρεσε από τα πεδία τιμών των μεταβλητών τις τιμές εκείνες που δεν μπορούν να συμμετέχουν στην τελική λύση με ένα αλγόριθμο ελέγχου συνέπειας.**
2. **Στο μειωμένο χώρο αναζήτησης που προκύπτει από το προηγούμενο βήμα εφάρμοσε έναν κλασικό αλγόριθμο αναζήτησης για να βρεθεί η λύση. Σε κάθε βήμα (ανάθεση τιμής) αυτής της αναζήτησης εφάρμοσε ξανά τον αλγόριθμο ελέγχου συνέπειας έτσι ώστε να αφαιρεθούν τυχόν τιμές από τα πεδία των μεταβλητών οι οποίες δεν μπορούν να συμμετέχουν στην λύση.**

Πρόβλημα Βιομηχανικού Μύλου

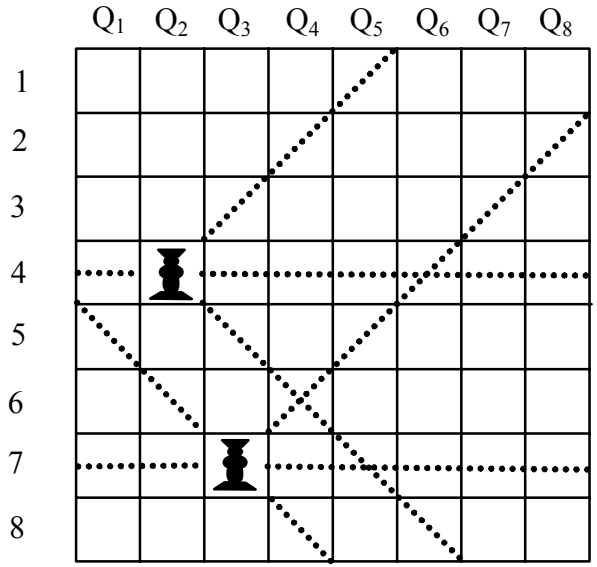
- Μετά την εφαρμογή των περιορισμών απομένουν οι ακόλουθες τιμές στα πεδία των μεταβλητών: $V_A \in \{3,4\}, V_B \in \{2,3\}, V_\Gamma \in \{1,2\}, V_\Delta \in \{1,2,3\}$
- Σε κάθε βήμα εφαρμόζεται έλεγχος συνέπειας.
- Δένδρο αναζήτησης.



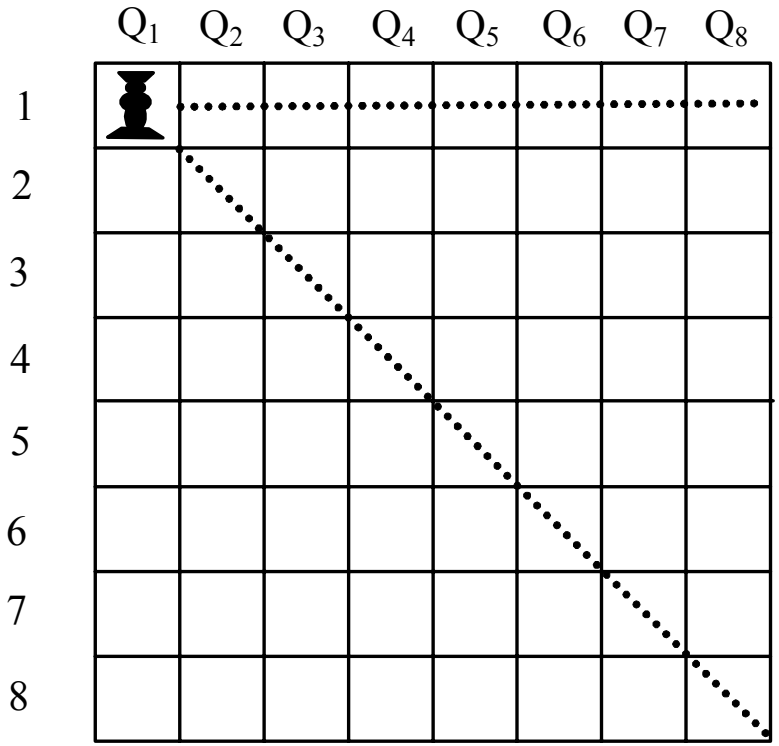
Το Πρόβλημα των Ν-Βασιλισσών (1/3)

- Κλασικό παράδειγμα προβλήματος περιορισμών.
  - Το πρόβλημα απαιτεί να τοποθετηθούν 8 βασίλισσες σε μια σκακιέρα 8_x8 χωρίς να απειλούν η μια την άλλη.
  - Το πρόβλημα ορίζεται και για περισσότερες των 8 βασιλισσών
  - Η δυσκολία στην επίλυσή του αυξάνει εκθετικά.
  - Χρησιμοποιείται για την μέτρηση της απόδοσης αλγορίθμων ικανοποίησης περιορισμών.

- Συνθήκη μη απειλής μεταξύ των βασιλισσών:
  - Όλες οι βασίλισσες πρέπει να είναι σε διαφορετική γραμμή:
    - $\forall$ i, j: $Q_j \neq Q_i$.
  - Ισχύουν οι περιορισμοί:
    - $Q_j \neq Q_{j+n} + n$ για $n>1$ και $n+j \leq 8$
    - $Q_j \neq Q_{j+n} - n$ για $n>1$ και $n+j \leq 8$
  - Σχηματική αναπαράσταση περιορισμών με δύο βασίλισσες στην σκακιέρα.

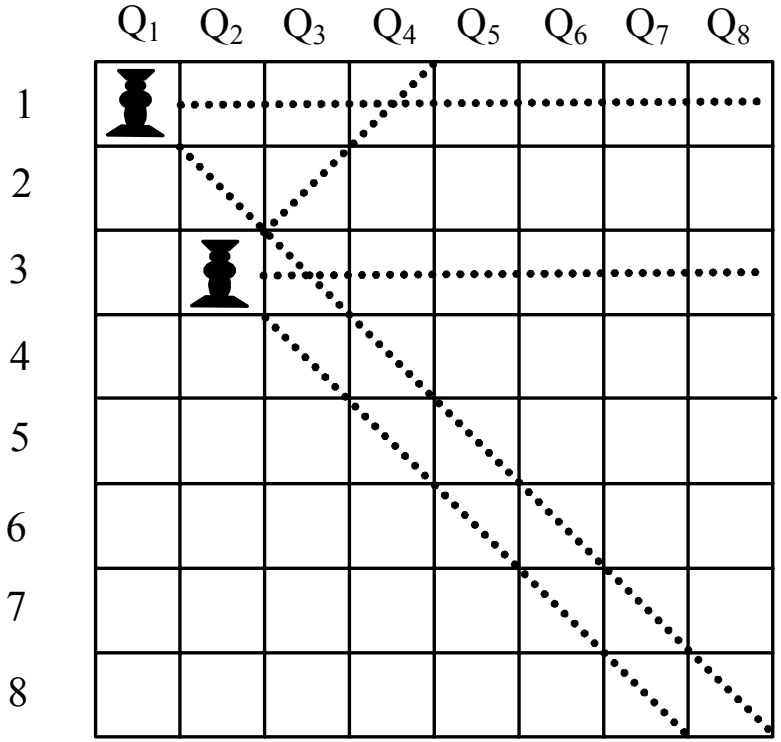


Το Πρόβλημα των N-Βασιλισσών (2/3)

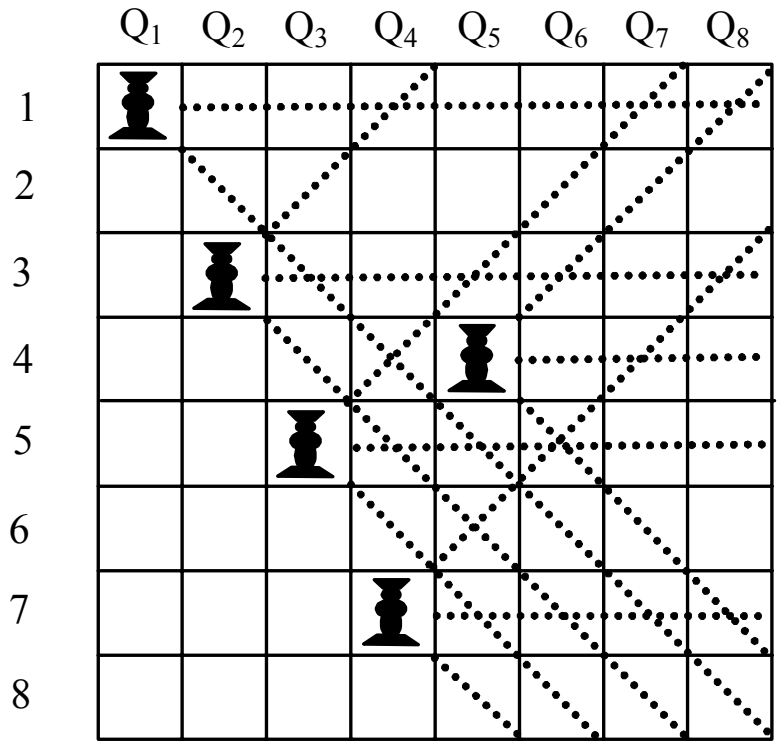


- Ανάθεση τιμής στην πρώτη βασίλισσα

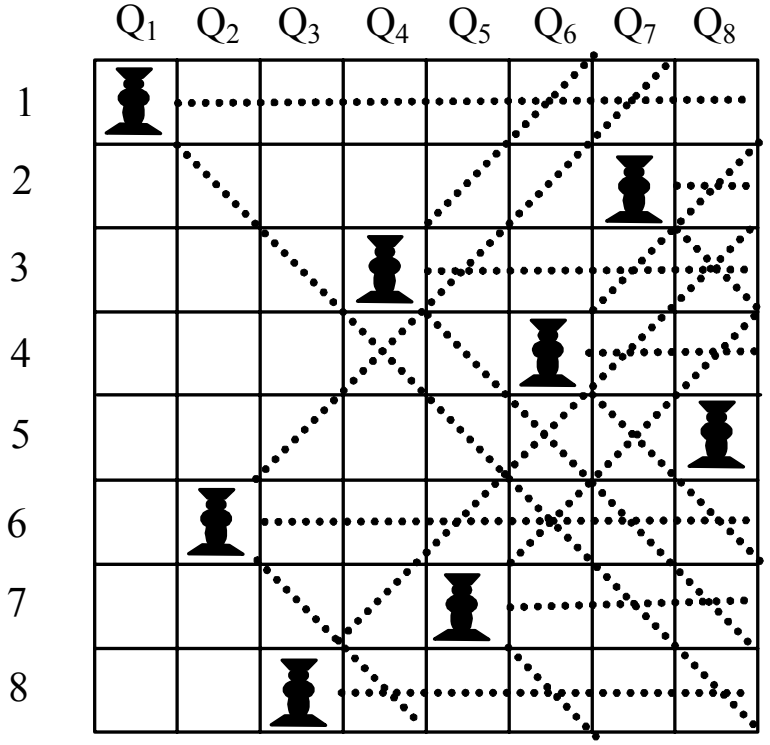
- Ανάθεση τιμών στις δύο πρώτες βασίλισσες



Το Πρόβλημα των N-Βασιλισσών (3/3)



- Ανάθεση τιμών που δεν οδηγεί σε λύση



- Λύση στο πρόβλημα των 8 βασιλισσών

Λογικός Προγραμματισμός με Περιορισμούς

- Δημιουργία μιας νέας "σχολής" προγραμματισμού, του *προγραμματισμού με περιορισμούς* (*constraint programming*).
- *Λογικός Προγραμματισμός με Περιορισμούς* (*Constraint Logic Programming - CLP*),
  - Επέκταση των γλωσσών λογικού προγραμματισμού (π.χ. PROLOG).
- Παράδειγμα τέτοιου συστήματος είναι το CHIP με πλήθος βιομηχανικών εφαρμογών:
  - σύνταξη ωρολογίου προγράμματος για την κατανομή ωρών εργασίας σε νοσοκομείο (GYMNASTE, στο νοσοκομείο BLINGY),
  - σχεδιασμό ενεργειών (planning) για την οργάνωση γραμμών παραγωγής στην αεροπορική βιομηχανία (PLANE στη DASSAULT), κτλ.
- Άλλες ιδιαίτερα διαδεδομένες γλώσσες που υποστηρίζουν προγραμματισμό με περιορισμούς είναι η SICSTUS, ECLiPSE PROLOG, η OZ και η gnu-prolog, κλπ
- Πλέον οι περισσότερες εκδόσεις της γλώσσας PROLOG υποστηρίζουν σε μεγαλύτερο ή μικρότερο βαθμό την νέα αυτή σχολή προγραμματισμού.