Προγραμματισμός με Περιορισμούς για Εφαρμογές της Τεχνητής Νοημοσύνης

- Προβλήματα ικανοποίησης περιορισμών
- Εξαντλητικές μέθοδοι επίλυσης
- Τεχνικές συνέπειας
- Επεκτάσεις
- Συστήματα προγραμματισμού με περιορισμούς
- Εφαρμογές

Προβλήματα Ικανοποίησης Περιορισμών

(Constraint Satisfaction Problems – CSPs)

Ορισμός

Δεδομένων ενός πεπερασμένου συνόλου μεταβλητών $V = \{v_1, v_2, \ldots, v_n\}$ με τα αντίστοιχα διακριτά και πεπερασμένα πεδία (domains) D_{v_i} $(1 \leq i \leq n)$ από τα οποία μπορούν να παίρνουν τιμές, αντίστοιχα, οι μεταβλητές v_i και ενός συνόλου περιορισμών (constraints) $C = \{C_1, C_2, \ldots, C_e\}$, όπου για $1 \leq j \leq e$

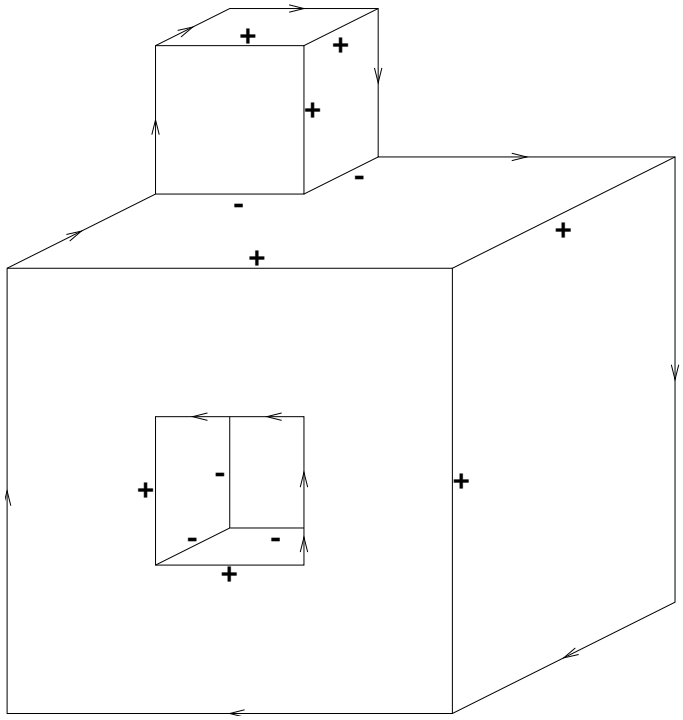
$$\begin{aligned} C_j \ (\equiv C_{v \in S_j}) &= (S_j, T_j) \\ S_j &\subseteq V \\ T_j &\subseteq \times_{v \in S_j} D_v \end{aligned}$$

να βρεθούν όλες (;) οι πλειάδες τιμών για τις n μεταβλητές που ικανοποιούν όλους τους περιορισμούς, δηλαδή το σύνολο

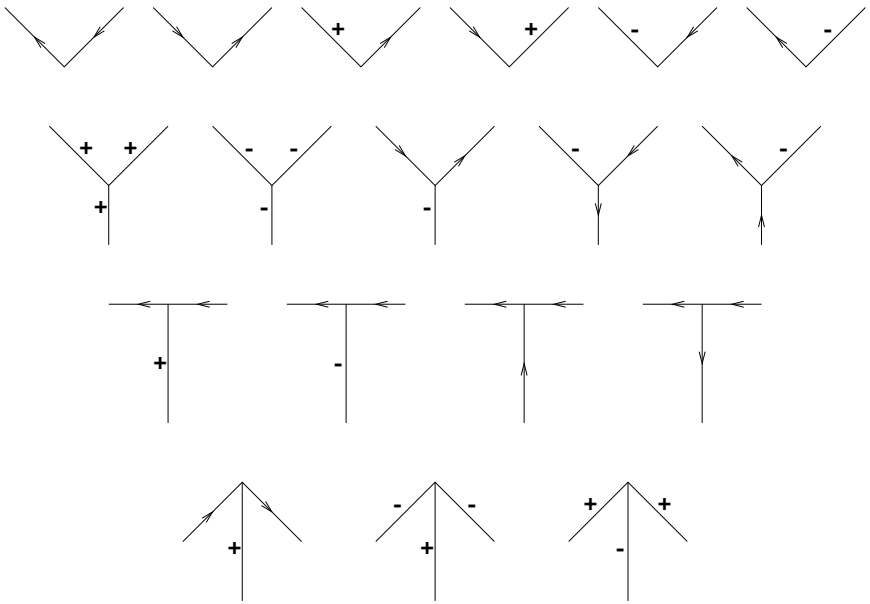
$$T = \{t \mid t \in \times_{v \in V} D_v, \ \ t|_{S_j} \in T_j \ \ \text{για} \ \ 1 \leq j \leq e\}$$

Η αρχή

- Μηχανική Όραση (Machine Vision)
- Αλγόριθμος φιλτραρίσματος (Waltz, 1972) για χαρακτηρισμό ακμών (edge labeling) σε 3-D αντικείμενα απεικονισμένα σε επίπεδο
- Πιθανοί χαρακτηρισμοί ακμών
  - κυρτή (convex)
  - κοίλη (concave)
  - αποκρύπτουσα (occluding)

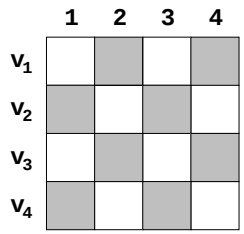


Οι μόνες πιθανές περιπτώσεις



Παραδείγματα CSPs

Το πρόβλημα των n βασιλισσών



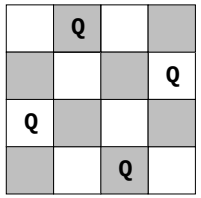
Μεταβλητές: $v_1, v_2, \ldots, v_n$

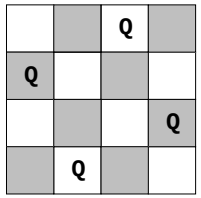
Πεδία: $D_{v_i} = \{1, 2, \ldots, n\}$ για $1 \leq i \leq n$

Περιορισμοί: C_{v_i, v_j} για $1 \leq i < j \leq n$ οι

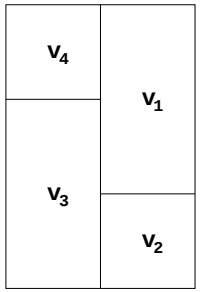
$(v_i \neq v_j) \ \wedge \ (v_i - v_j \neq i - j) \ \wedge \ (v_i - v_j \neq j - i) \quad \rightsquigarrow$ δυαδικό (binary) CSP

Λύσεις:



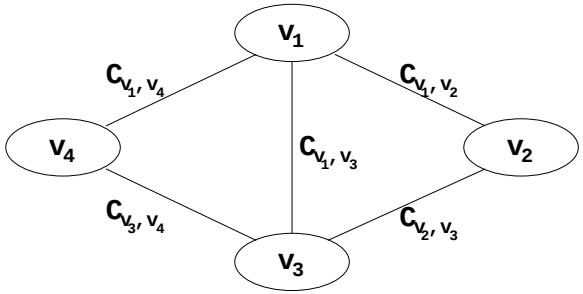


Το πρόβλημα του χρωματισμού χάρτη



$$D_{v_1} = D_{v_2} = D_{v_3} = D_{v_4} = \{\mathbf{r}(\text{ed}), \mathbf{g}(\text{reen}), \mathbf{b}(\text{lue})\}$$

C_{v_i,v_j} : $v_i \neq v_j$ για κάθε ζευγάρι γειτονικών περιοχών i και j



γράφος περιορισμών (constraint graph) για δυαδικούς και μοναδιαίους (unary) περιορισμούς ($C_{v_i,v_i} \equiv C_{v_i}$)

Η προφανής μέθοδος επίλυσης CSPs

Γέννα-και-δοκίμαζε (Generate-and-test – GT)

> Γέννα συστηματικά συνδυασμούς τιμών για τις μεταβλητές και δοκίμαζε αν ικανοποιούν τους περιορισμούς ($O(ed^n)$ χρόνος).

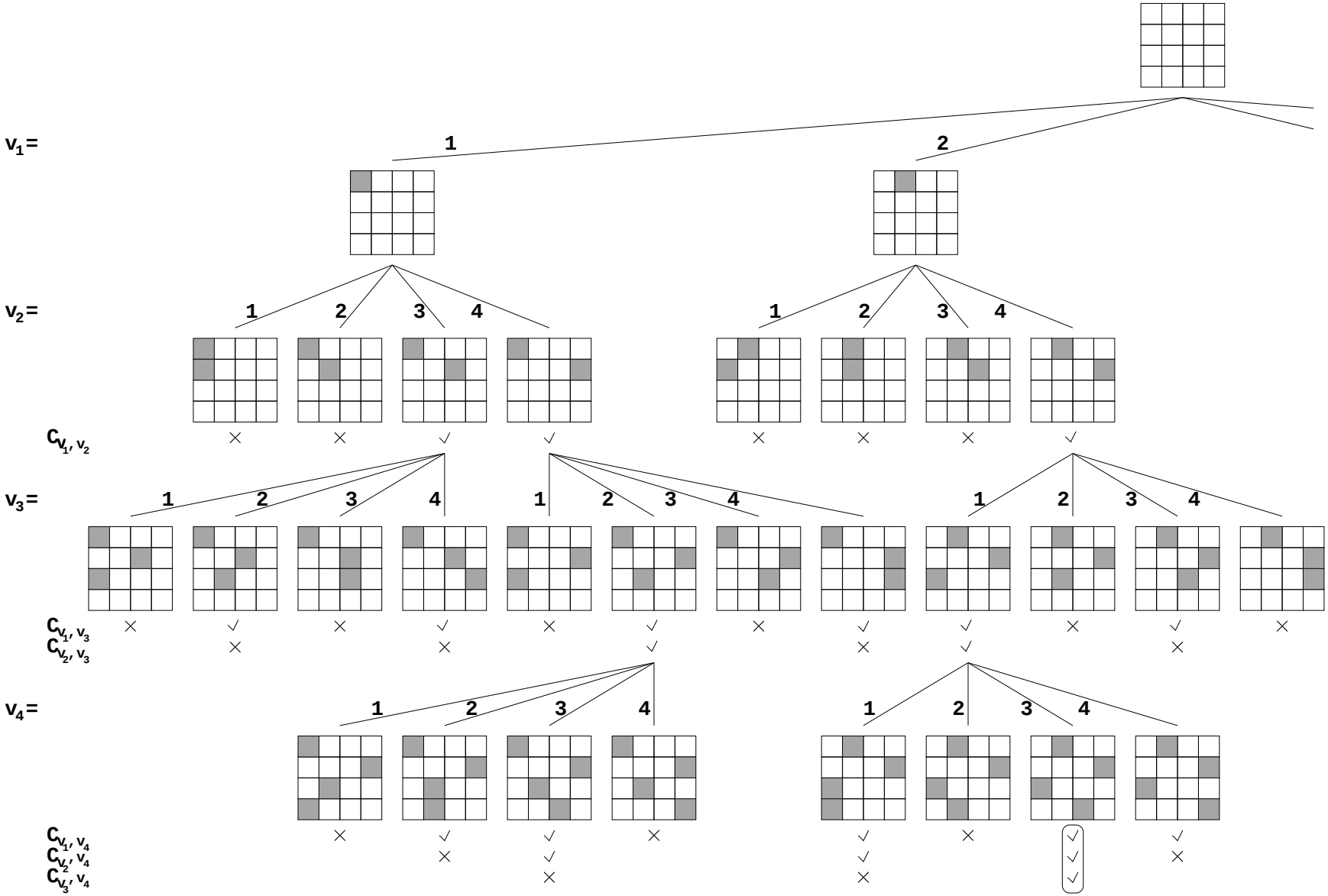
Μία βελτιωμένη μέθοδος

Οπισθοδρόμηση (Backtracking – BT)

> Κατασκεύαζε συστηματικά μία λύση δίνοντας διαδοχικά τιμές σε μεταβλητές και έλεγχε την ισχύ κάθε περιορισμού αμέσως μόλις αυτό μπορεί να γίνει. Αν δεν ικανοποιείται ένας περιορισμός, οπισθοδρόμησε στην τελευταία χρονολογικά ανάθεση τιμής που είχες κάνει σε μεταβλητή και δώσε της άλλη τιμή. Αν εξαντληθούν οι τιμές μίας μεταβλητής, οπισθοδρόμησε στην ανάθεση της προηγούμενης μεταβλητής.

Καλύτερη μέθοδος από την GT, αφού δεν επεκτείνει μερικές αναθέσεις που είναι ασυνεπείς, αλλά επίσης εκθετική.

Εφαρμογή της BT στο πρόβλημα των 4 βασιλισσών



Οι τρεις λόγοι συντριβής (thrasing) της BT

- Ασυνέπεια κόμβου (node inconsistency)

  Ύπαρξη τιμών στα D_{v_i} που δεν επαληθεύουν τους μοναδιαίους περιορισμούς στις μεταβλητές v_i.

- Ασυνέπεια κατευθυνόμενης ακμής (arc inconsistency)

  Αν οι μεταβλητές αποτιμώνται με τη σειρά v_1, v_2, ..., v_n, η τιμή $v_i = a$ μπορεί να είναι ασύμβατη με όλες τις πιθανές τιμές της μεταβλητής v_j $(j > i)$.

- Ασυνέπεια μονοπατιού (path inconsistency)

  Αν οι μεταβλητές αποτιμώνται με τη σειρά v_1, v_2, ..., v_n, και οι τιμές $v_i = a$ και $v_j = b$ είναι συμβατές, μπορεί να μην υπάρχει δυνατή τιμή για τη μεταβλητή v_k που να είναι συμβατή και με την $v_i = a$ και την $v_j = b$ $(k > j > i)$.

Συνέπεια κόμβων

Ορισμός

Ένα δυαδικό CSP είναι *συνεπές-ως-προς-κόμβους* (node-consistent) όταν για κάθε μεταβλητή του v_i για όλες τις τιμές x στο πεδίο της D_{v_i} ισχύει ότι $(x) \in C_{v_i}$.

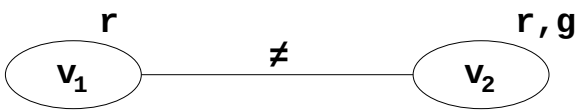
```
Procedure NC
Begin
    For each v_i in V
        For each x in D_{v_i}
            If (x) ∉ C_{v_i} Then
                D_{v_i} ← D_{v_i} − {x}
            End If
        End For
    End For
End
```

Συνέπεια κατευθυνόμενων ακμών

Ορισμός

Ένα δυαδικό CSP είναι *συνεπές-ως-προς-κατευθυνόμενες-ακμές* (arc-consistent) όταν κάθε κατευθυνόμενη ακμή $v_i \rightarrow v_j$ στο γράφο των περιορισμών του CSP είναι συνεπής, δηλαδή όταν για κάθε τιμή $x \in D_{v_i}$ τέτοια ώστε $(x) \in C_{v_i}$ υπάρχει τιμή $y \in D_{v_j}$ τέτοια ώστε $(y) \in C_{v_j}$ και $(x, y) \in C_{v_i,v_j}$.

Αναφερόμαστε σε συνέπεια-ως-προς-κατευθυνόμενες-ακμές γιατί η συνέπεια της $v_i \rightarrow v_j$ δεν επιβάλλει τη συνέπεια της $v_j \rightarrow v_i$.



Η ακμή $v_1 \rightarrow v_2$ είναι συνεπής αλλά όχι και η $v_2 \rightarrow v_1$

Μία διαδικασία επιβολής συνέπειας στην ακμή $v_i \rightarrow v_j$ με τη διαγραφή εκείνων των x από το D_{v_i} για τα οποία δεν υπάρχει $y \in D_{v_j}$ τέτοιο ώστε $(x, y) \in C_{v_i,v_j}$ είναι:

Procedure **Revise**$(v_i \rightarrow v_j)$

```
Begin
    Deleted ← False
    For each x ∈ D_{v_i}
        If ∄y ∈ D_{v_j} : (x, y) ∈ C_{v_i,v_j} Then
            D_{v_i} ← D_{v_i} − {x}
            Deleted ← True
        End If
    End For
    Return(Deleted)
End
```

Ένα CSP μπορεί να γίνει συνεπές-ως-προς-κατευθυνόμενες-ακμές με έναν αλγόριθμο συνέπειας που εφαρμόζει κατάλληλα τη διαδικασία **Revise**.

<u>Procedure **AC-1**</u>
Begin
  NC
  $Q \leftarrow \{(v_i, v_j) \mid C_{v_i,v_j} \in C,\ i \neq j\} \cup \{(v_j, v_i) \mid C_{v_i,v_j} \in C,\ i \neq j\}$
  Repeat
    $Changed \leftarrow$ False
    For each $v_i \rightarrow v_j \in Q$
      $Changed \leftarrow$ **Revise**$(v_i \rightarrow v_j)$ OR $Changed$
    End For
  Until NOT $Changed$
End

$O(end^3)$ χρόνος $\qquad$ $O(nd)$ χώρος

Mackworth (1977)

Η επιβολή συνέπειας-ως-προς-κατευθυνόμενες-ακμές μπορεί να γίνει πιο αποδοτικά αν αναθεωρούμε μόνο τις ακμές που χρειάζεται.

<u>Procedure **AC-3**</u>
Begin
  NC
  $Q \leftarrow \{(v_i, v_j) \mid C_{v_i,v_j} \in C,\ i \neq j\} \cup \{(v_j, v_i) \mid C_{v_i,v_j} \in C,\ i \neq j\}$
  While $Q \neq \{\ \}$
    Delete any element $v_i \rightarrow v_j$ from Q
    If **Revise**$(v_i \rightarrow v_j)$ Then
      $Q \leftarrow Q \cup \{v_k \rightarrow v_i \mid C_{v_k,v_i} \in C,\ k \neq i,\ k \neq j\}$
    End If
  End While
End

$O(ed^3)$ χρόνος $O(nd)$ χώρος
Mackworth (1977)

Ο αλγόριθμος AC-2 είναι αυτός του Waltz και είναι ειδική περίπτωση του AC-3.

Η αλληλουχία των AC-n αλγορίθμων συνεχίστηκε με τους:

AC-4 (Mohr, Henderson, 1986): Στο χρόνο επιτυγχάνει το βέλτιστο $O(ed^2)$ και στο χώρο θέλει $O(ed^2)$

AC-5 (van Hentenryck, Deville, Teng, 1992): Είναι γενικός αλγόριθμος συνέπειας όπου για κάθε περιορισμό μπορούμε να έχουμε ιδιαίτερη διαδικασία φιλτραρίσματος τιμών, π.χ. συνέπεια-ορίων (bound-consistency)

AC-6 (Bessière, 1994): Βελτιώνει τις απαιτήσεις χώρου του AC-4 σε $O(ed)$

AC-7 (Bessière, Freuder, Régin, 1995): Είναι βέλτιστος στο πλήθος των περιορισμών που ελέγχει

Συνέπεια μονοπατιών

Ορισμός

Ένα δυαδικό CSP είναι *συνεπές-ως-προς-μονοπάτια* (path-consistent) όταν κάθε μονοπάτι $(v_{i_0}, v_{i_1}, \ldots, v_{i_m})$ στο γράφο περιορισμών του CSP είναι συνεπές, δηλαδή όταν για κάθε ζευγάρι τιμών $x \in D_{v_{i_0}}$ και $y \in D_{v_{i_m}}$ με $(x) \in C_{v_{i_0}}$, $(y) \in C_{v_{i_m}}$ και $(x, y) \in C_{v_{i_0}, v_{i_m}}$ υπάρχουν τιμές $z_1 \in D_{v_{i_1}}, \ldots, z_{m-1} \in D_{v_{i_{m-1}}}$ τέτοιες ώστε $(z_1) \in C_{v_{i_1}}, \ldots, (z_{m-1}) \in C_{v_{i_{m-1}}}$ και $(x, z_1) \in C_{v_{i_0}, v_{i_1}}, (z_1, z_2) \in C_{v_{i_1}, v_{i_2}}, \ldots, (z_{m-1}, y) \in C_{v_{i_{m-1}}, v_{i_m}}$.

Θεώρημα

Ένα CSP είναι συνεπές-ως-προς-μονοπάτια αν και μόνο αν κάθε μονοπάτι μήκους 2 ($m = 2$) στο γράφο των περιορισμών του CSP είναι συνεπές (Montanari, 1974).

Υπάρχουν αλγόριθμοι επιβολής συνέπειας μονοπατιών, όπως PC-1, PC-2, PC-3, PC-4.

Επίλυση των CSPs

Ερώτηση: Είναι δυνατόν η επιβολή κάποιας μορφής συνέπειας στο γράφο περιορισμών ενός CSP να οδηγήσει στην επίλυσή του; Μπορούμε να βρίσκουμε λύσεις του προβλήματος χωρίς να χρειασθεί να οπισθοδρομούμε;

Απάντηση: ΟΧΙ (στη γενική περίπτωση)

Οι διέξοδοι:

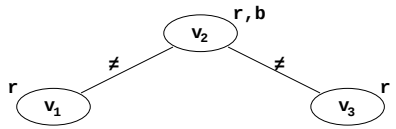
- Βρίσκουμε κλάσεις προβλημάτων που επιβολή κάποιας μορφής συνέπειας οδηγεί σε επίλυση χωρίς οπισθοδρόμηση

- Εμπεδώνουμε τεχνικές συνέπειας στη διαδικασία οπισθοδρόμησης BT για να κλαδέψουμε το χώρο αναζήτησης και να βρούμε "ευκολότερα" τις λύσεις

k-συνέπεια και ισχυρή k-συνέπεια

Ορισμός

Ένα CSP είναι *1-συνεπές* όταν όλες οι τιμές στο πεδίο κάθε μεταβλητής ικανοποιούν τους μοναδιαίους περιορισμούς στη μεταβλητή. Ένα CSP είναι *k-συνεπές* όταν οποιοσδήποτε συνδυασμός τιμών για $(k-1)$ μεταβλητές, που ικανοποιεί τους εμπλεκόμενους περιορισμούς, μπορεί να επεκταθεί με μία τιμή για οποιαδήποτε πρόσθετη μεταβλητή, έτσι ώστε επίσης να ικανοποιεί τους εμπλεκόμενους περιορισμούς.



3-συνεπές αλλά όχι 2-συνεπές CSP

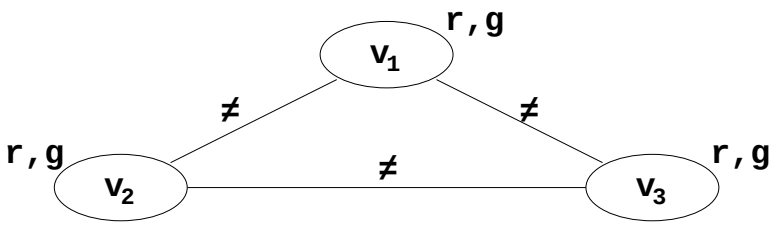
Ορισμός

Ένα CSP είναι *ισχυρά k-συνεπές* όταν είναι j-συνεπές για κάθε j με $1 \leq j \leq k$.

συνέπεια-ως-προς-κόμβους	$\equiv$ 1-συνέπεια
συνέπεια-ως-προς-κατευθυνόμενες-ακμές	$\equiv$ 2-συνέπεια
συνέπεια-ως-προς-μονοπάτια	$\equiv$ 3-συνέπεια (για δυαδικά CSPs)

Σε ένα CSP με n μεταβλητές, η επιβολή ισχυρής n-συνέπειας εγγυάται ότι μπορούν να βρεθούν οι λύσεις του προβλήματος χωρίς οπισθοδρόμηση. Αλγόριθμοι για την επιβολή αυτής της συνέπειας υπάρχουν, αλλά είναι εκθετικοί.

Η επιβολή ισχυρής k-συνέπειας σε ένα CSP με n μεταβλητές, όπου $k < n$, κατ΄ αρχήν, δεν εγγυάται τίποτα.



CSP ισχυρά 2-συνεπές αλλά χωρίς λύση

Ορισμός: Ένας γράφος περιορισμών λέγεται διατεταγμένος όταν έχει τεθεί στους κόμβους του μία σειρά.

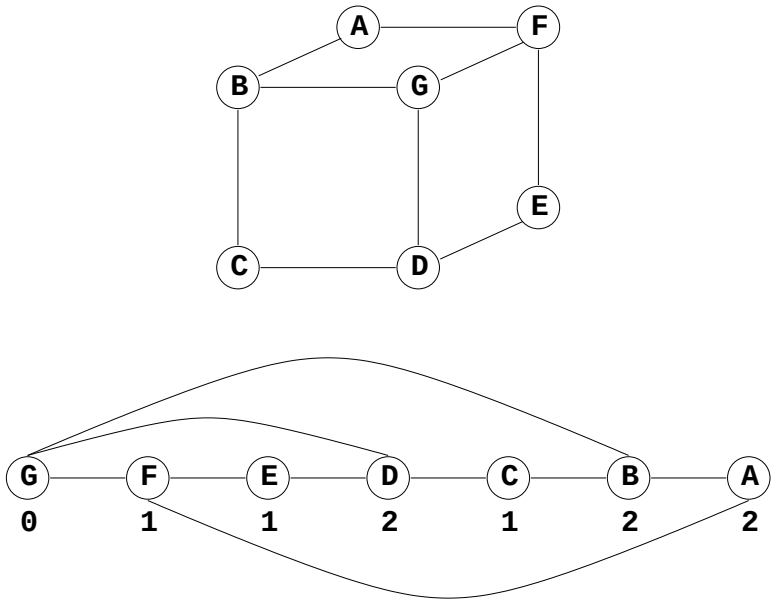
Ορισμός: Το πλάτος ενός κόμβου σε ένα διατεταγμένο γράφο περιορισμών είναι ο αριθμός των ακμών που υπάρχουν από τον κόμβο αυτό σε προηγουμένους του.

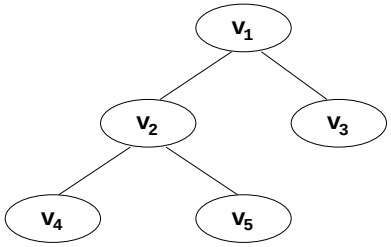
Ορισμός: Το πλάτος ενός διατεταγμένου γράφου περιορισμών είναι το μέγιστο πλάτος των κόμβων του.

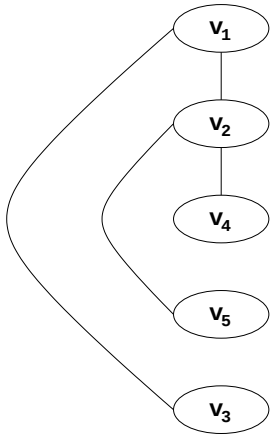
Ορισμός: Το πλάτος ενός γράφου περιορισμών είναι το ελάχιστο πλάτος των διατεταγμένων γράφων που προκύπτουν από αυτόν.

Θεώρημα: Αν ένα CSP είναι ισχυρά k-συνεπές και το πλάτος του γράφου περιορισμών του είναι w, με $k > w$, τότε υπάρχει σειρά αποτίμησης των κόμβων-μεταβλητών του τέτοια ώστε να μην απαιτείται οπισθοδρόμηση για να βρεθεί λύση (backtrack free) (Freuder, 1982).

Το θεώρημα του Freuder έχει πρακτική χρησιμότητα για $k = 2$ και $w = 1$, δηλαδή όταν επιβάλλουμε ισχυρή 2-συνέπεια, και έχουμε "γρήγορους" τρόπους να το κάνουμε, σε ένα CSP με γράφο περιορισμών πλάτους 1, δηλαδή δέντρο.







Μερική συνέπεια και οπισθοδρόμηση

Με τη σταδιακή εισαγωγή τεχνικών μερικής συνέπειας σε διαδικασίες αναζήτησης λύσεων ενός CSP προκύπτουν οι μέθοδοι εμπρόσθιου ελέγχου (forward checking) και πρόβλεψης (lookahead).

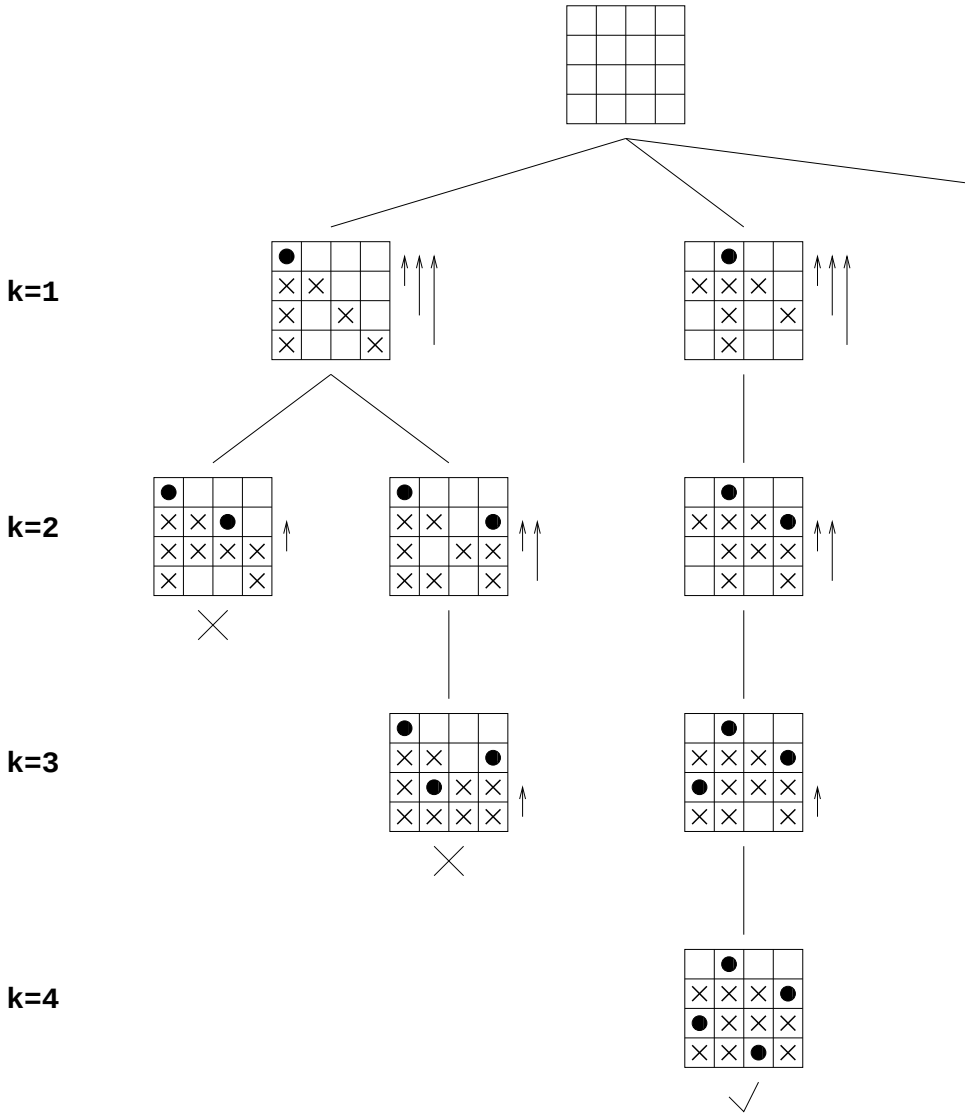
Γέννα-και-δοκίμαζε	(GT)
Οπισθοδρόμηση	(BT = GT + AC$\frac{1}{5}$)
Εμπρόσθιος έλεγχος	(FC = GT + AC$\frac{1}{4}$)
Μερική πρόβλεψη	(PL = FC + AC$\frac{1}{3}$)
Πλήρης πρόβλεψη	(FL = FC + AC$\frac{1}{2}$)
Πραγματικά πλήρης πρόβλεψη	(RFL = FC + AC)

Μετά την απόδοση τιμής στην k-οστή μεταβλητή από σύνολο n μεταβλητών:

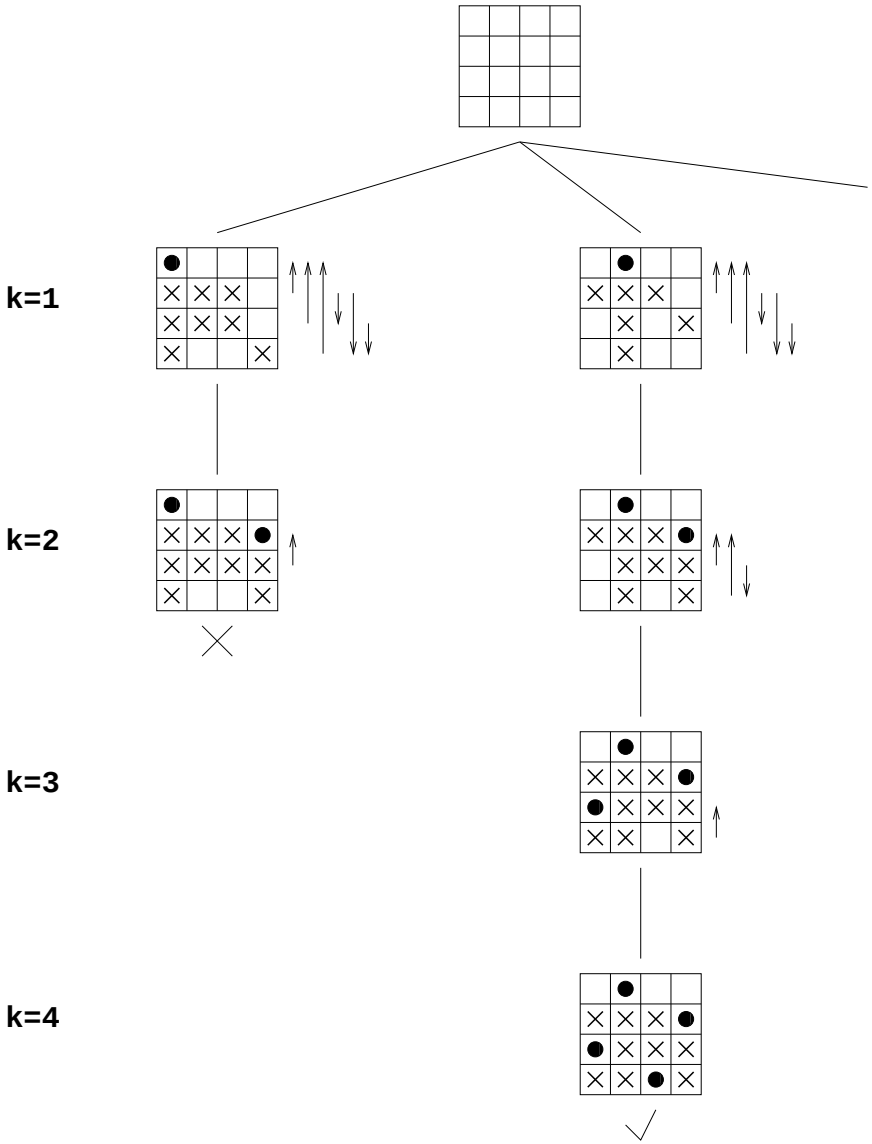
AC$\frac{1}{5}$: **Revise**(k, p)	για κάθε p με $1 \leq p < k$
AC$\frac{1}{4}$: **Revise**(f, k)	για κάθε f με $k < f \leq n$
AC$\frac{1}{3}$: **Revise**(f_1, f_2)	για κάθε f_1, f_2 με $k < f_1 < f_2 \leq n$
AC$\frac{1}{2}$: **Revise**(f_1, f_2)	για κάθε f_1, f_2 με $k < f_1 \neq f_2 \leq n$

Αν το πεδίο κάποιας μεταβλητής γίνει κενό $\Longrightarrow$ οπισθοδρόμηση

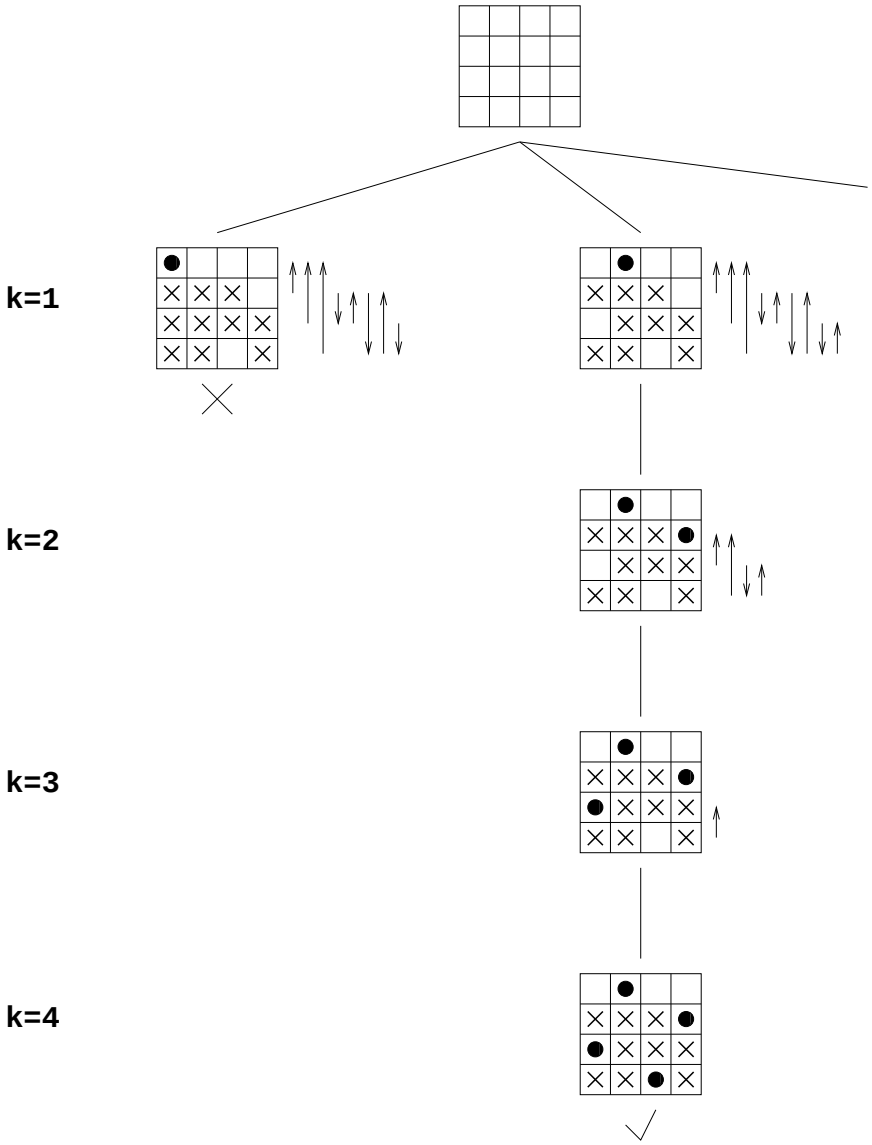
Εφαρμογή της FC στο πρόβλημα των 4 βασιλισσών



Εφαρμογή της PL στο πρόβλημα των 4 βασιλισσών



Εφαρμογή της FL στο πρόβλημα των 4 βασιλισσών



Σειρά μεταβλητών και σειρά τιμών

- Η σειρά (στατική ή δυναμική) με την οποία αποτιμώνται οι μεταβλητές σε μία (επηυξημένη) μέθοδο οπισθοδρόμησης για την επίλυση ενός CSP μπορεί και να είναι καίρια. Πιθανά ευριστικά:
  - Πρώτα οι μεταβλητές με τα μικρότερα πεδία
  - Πρώτα οι μεταβλητές που συμμετέχουν σε περισσότερους περιορισμούς
- Για δεδομένη μεταβλητή προς αποτίμηση, η επιλογή τιμής, από τις πιθανές, για να αποδοθεί σ΄ αυτήν μπορεί επίσης να είναι κρίσιμη. Πιθανό ευριστικό:
  - Πρώτα οι τιμές με τις λιγότερες συγκρούσεις με τις πιθανές τιμές των μη αποτιμημένων μεταβλητών

Βελτιστοποίηση (optimization) και CSPs

- Εύρεση λύσης σε ένα CSP που βελτιστοποιεί (ελαχιστοποιεί ή μεγιστοποιεί) μία αντικειμενική συνάρτηση (objective function)

- Μέθοδος *διακλάδωσε-και-φράξε* (branch-and-bound)
  - Προέλευση: Επιχειρησιακή έρευνα
  - Διαδικασία BT + (μερικές) τεχνικές συνέπειας + φράξιμο της αντικειμενικής συνάρτησης μετά την εύρεση κάθε λύσης που ικανοποιεί τους περιορισμούς

- Μέθοδος A^*
  - Προέλευση: Τεχνητή νοημοσύνη
  - Διάσχιση ενός χώρου αναζήτησης με εξερεύνηση σε κάθε βήμα του κόμβου που "υπόσχεται περισσότερο" ότι οδηγεί στη βέλτιστη λύση

- Μεγάλη ποικιλία εφαρμογών στον πραγματικό κόσμο

Υπερ-περιορισμένα (over-constrained) προβλήματα

- CSPs με μεγάλο πλήθος περιορισμών που τα καθιστά ανέφικτα (infeasible)
- Πιθανές αντιμετωπίσεις
  - Μερικά (partial) CSPs
  - CSPs με ιεραρχίες (hierarchies)
  - Ασαφή (fuzzy) CSPs
  - Πιθανοτικά (probabilistic) CSPs
  - CSPs με βάρη (weights)
- Η γενίκευση: CSPs βασισμένα σε ημιδακτυλίους (semiring-based)

Τεχνικές τοπικής αναζήτησης (local search)

- Εφαρμόσιμες σε
  - Κλασικά CSPs
  - CSPs με βελιστοποίηση
  - Υπερ-περιορισμένα CSPs
- Εναλλακτικές μέθοδοι
  - Αναρρίχηση λόφου (hill climbing)
  - Αναζήτηση με απαγορεύσεις (tabu search)
  - Προσομοιωμένη ανόπτηση (simulated annealing)
  - Γενετικοί αλγόριθμοι (genetic algorithms)

Συστήματα για προγραμματισμό με περιορισμούς

- Επεκτάσεις γλωσσών λογικού προγραμματισμού
  - CHIP, ECL^{i}PS^e
  - CLP($\Re$)
  - Prolog IV
- Αυτόνομα συστήματα και γλώσσες
  - ALICE
  - Oz, Mozart
- Βιβλιοθήκες
  - ILOG Solver (C++)

Βιβλιοθήκη πεπερασμένων πεδίων της ECL^{i}PS^e

- Μεταβλητές πεδίων

```
X :: 10..20
X :: [5, 7, 9, 20]
[X, Y] :: [10..15, 18, 20..22]
```

- Γραμμικοί όροι

```
2 * X + 3 * Y - Z + 7
5 * (3 + (4 - 6) * Y - X * 3)
```

- Αριθμητικοί περιορισμοί (##, #=, #<, #<=, #>, #>=)

```
X ## Y
2 * X - 3 * Y #> 3 * Z + 2
```

- Λογικοί περιορισμοί (#\+, #/\, #\/, #=>, #<=>)

```
X ## Y #\/ X #> Z + 2
X #= Y - 2 #=> Y ## Z + W
```

- Μετα-περιορισμοί

```
B isd X ## Y
```

- Συμβολικοί περιορισμοί

```
element(I, [10, 20, 30, 40], X)
outof(X, [3, 5, 8, 9])
alldistinct([X, Y, Z, W])
```

- Επιλογή μεταβλητών πεδίων

```
deleteff(V, [X, Y, Z, W], R)
```

- Γέννηση τιμών για μεταβλητές πεδίων

```
indomain(X)
labeling([X, Y, Z])
```

- Ελαχιστοποίηση γραμμικών όρων

```
min_max(labeling([X, Y]), [X + Y, 2 * X - Y])
minimize(labeling([X, Y, Z]), 3 * X - Y + 2 * Z, 20, 1000, 80, 600)
```

- Δυνατότητα ορισμού νέων περιορισμών

Πρόβλημα των n βασιλισσών σε ECL^{i}PS^e

```
:- use_module(library(fd)).

queens(N, Solution) :- length(Solution, N), Solution :: 1..N,
                       constrain(Solution), generate(Solution).

constrain([]).
constrain([X|Xs]) :- noattack(X, Xs, 1), constrain(Xs).

noattack(_, [], _).
noattack(X, [Y|Ys], M) :- X ## Y, X ## Y-M, X ## Y+M,
                          M1 is M+1,
                          noattack(X, Ys, M1).

generate([]).
generate([X|Xs]) :- indomain(X), generate(Xs).
```

Πρόβλημα των n βασιλισσών σε ILOG Solver

```
#include <ilsolver/ilcint.h>
int main(int argc, char** argv){
  IlcManager m(IlcNoEdit);
  IlcInt nqueen = (argc > 1) ? atoi(argv[1]) : 8;
  IlcIntVarArray q(m, nqueen, 1, nqueen), q1(m, nqueen), q2(m, nqueen);
  for (IlcInt i = 0; i < nqueen; i++) {
     q1[i] = q[i]+i;
     q2[i] = q[i]-i; }
  m.add(IlcAllDiff(q));
  m.add(IlcAllDiff(q1));
  m.add(IlcAllDiff(q2));
  m.add(IlcGenerate(q, IlcChooseMinSizeInt));
  while (m.nextSolution()) {
     for (i = 0; i < nqueen; i++)
        m.out() << q[i].getValue() << " ";
     m.out() << endl; }
  m.end();
  return 0; }
```

Εφαρμογές

- Αυτόματη κατασκευή ωρολογίων προγραμμάτων (automated timetabling)
- Χρονοδρομολόγηση προσωπικού (personnel scheduling)
- Ανάθεση στόλου (fleet assignment)
- Πρόβλεψη δομής πρωτεϊνών (protein structure prediction)
- Δημιουργία τουριστικών περιηγήσεων (tour generation)
- Προγραμματισμός παραγωγής (production planning)
- Διανομή αγαθών (goods distribution)
- ...