Τεχνητή Νοημοσύνη

Ενότητα 4: Προβλήματα Ικανοποίησης Περιορισμών

Αν. καθηγητής Στεργίου Κωνσταντίνος
kstergiou@uowm.gr

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Άδειες Χρήσης

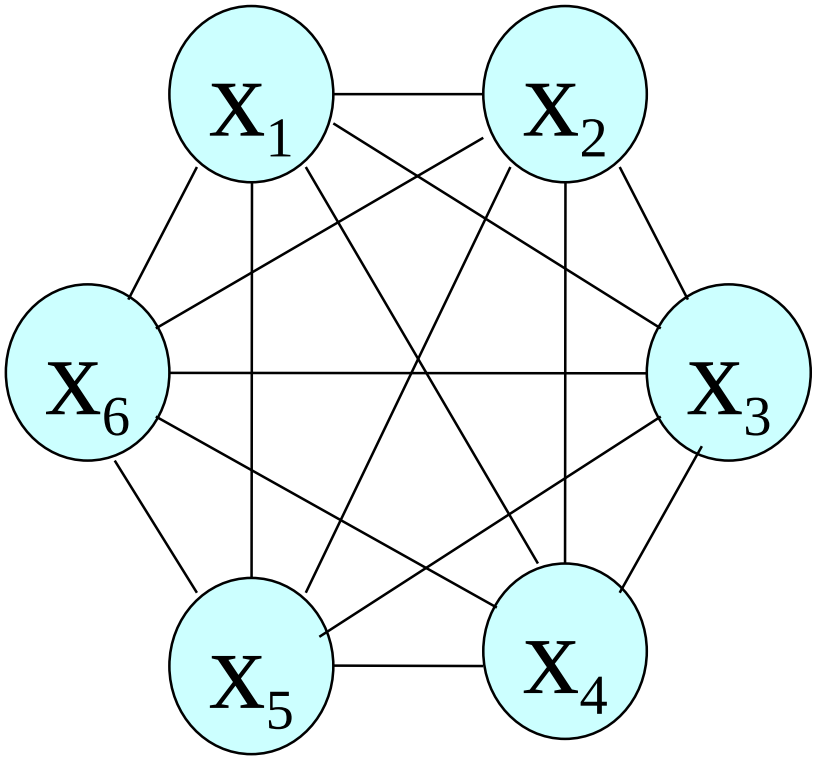
- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Προγραμματισμός με Περιορισμούς (1/2)



Προβλήματα Ικανοποίησης Περιορισμών (1/3)

- Το **Πρόβλημα Ικανοποίησης Περιορισμών** ***(Constraint Satisfaction Problem – CSP)*** είναι μια σημαντική υποκατηγορία της Τεχνητής Νοημοσύνης.
  - Ένα CSP είναι ένα πρόβλημα αναζήτησης με *ιδιαίτερα χαρακτηριστικά* που το διαφοροποιούν από τα γενικά προβλήματα αναζήτησης.
    - ✓ Ο χώρος αναζήτησης είναι πεπερασμένος,
    - ✓ το δέντρο αναζήτησης είναι πεπερασμένο,
    - ✓ το ζητούμενο είναι να αναθέσουμε τιμές σε ένα σύνολο μεταβλητών ώστε να ικανοποιούνται κάποιοι περιορισμοί.
      - Π.χ. n-queens, χρωματισμός γράφων, χρονοπρογραμματισμός, κ.α.
    - ✓ Δε μας ενδιαφέρει το μονοπάτι προς τη λύση.
  - Όπως θα δούμε η βασική τεχνική επίλυσης βασίζεται σε ***αναζήτηση κατά βάθος***!

Περιορισμοί υπάρχουν Παντού!

- Η Αίθουσα Α είναι κατειλημμένη από 16:00 ως 18:00.
- Κίνηση στο Δίκτυο < 100 Gbytes/sec.
- Μισθός < 25k Ευρώ.
- Το τραίνο για Πάτρα πρέπει να φύγει το λιγότερο 20 λεπτά πριν καταφτάσει το τραίνο από Θεσσαλονίκη.
- Δε μπορούμε να τοποθετήσουμε δύο βασίλισσες με τέτοιο τρόπο ώστε να επιτίθεται η μια στην άλλη…

Προγραμματισμός με Περιορισμούς (2/2)

- Η ανάπτυξη τεχνικών και συστημάτων επίλυσης CSPs οδήγησε στη δημιουργία ενός νέου είδους προγραμματισμού.
- Ο **προγραμματισμός με περιορισμούς** ***(constraint programming)*** είναι μια μορφή δηλωτικού προγραμματισμού *(declarative programming)*:
  - Δηλώστε το πρόβλημα *(μεταβλητές και περιορισμούς)*.
  - Η μηχανή επίλυσης θα βρει μια λύση.
- Η αναπαράσταση με περιορισμούς είναι το ενδεδειγμένο μοντέλο σε πολλά δύσκολα συνδυαστικά προβλήματα:
  - **Χρονοπρογραμματισμός.**
  - **Δρομολόγηση οχημάτων.**
  - **Ανάθεση πόρων**...

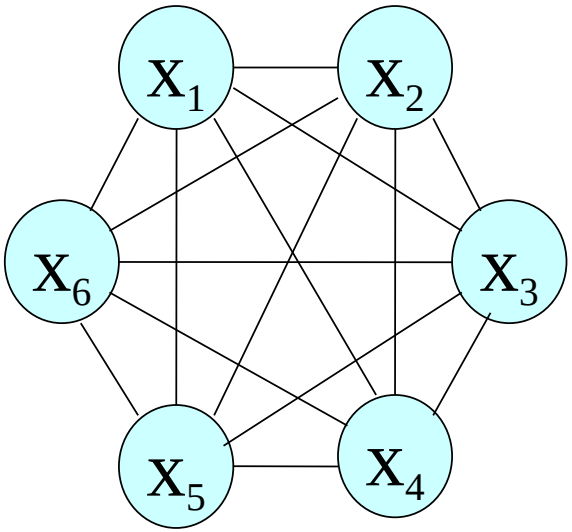
Προβλήματα Ικανοποίησης Περιορισμών (2/3)

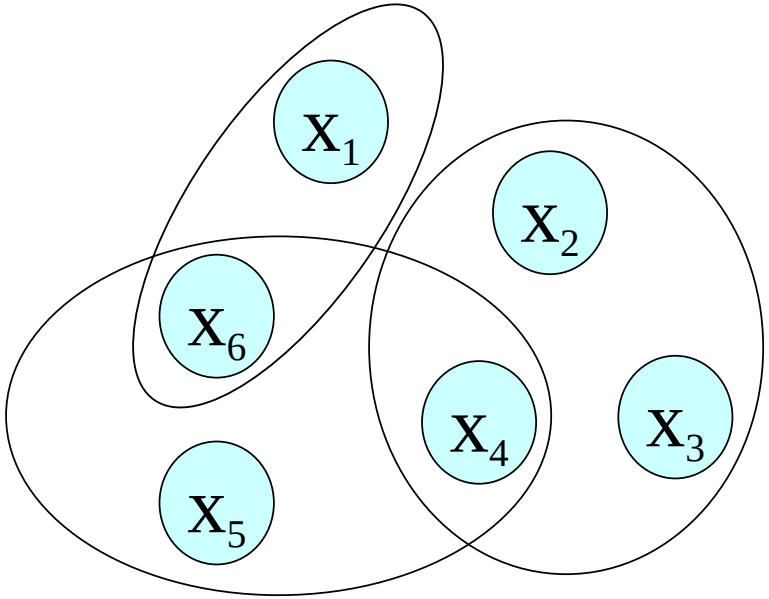
- Ένα **πρόβλημα ικανοποίησης περιορισμών** *(constraint satisfaction problem – CSP)* ορίζεται από:
  - Ένα σύνολο **μεταβλητών** (variables) $X_1,\ldots,X_n$:
    - ✓ Κάθε μεταβλητή X_i έχει ένα **πεδίο ορισμού** (domain) D_i με τις πιθανές της **τιμές** (values).
    - ✓ Συνήθως τα πεδία ορισμού είναι πεπερασμένα.
  - Ένα σύνολο **περιορισμών** (constraints) $C_1,\ldots,C_m$:
    - ✓ Κάθε περιορισμός περιλαμβάνει ένα υποσύνολο των μεταβλητών και προσδιορίζει τους επιτρεπτούς συνδυασμούς τιμών για αυτό το υποσύνολο.
    - ✓ Ένας ***n-αδικός*** (n-ary) περιορισμός C σε ένα σύνολο μεταβλητών $X_1,\ldots,X_k$ είναι ένα υποσύνολο του καρτεσιανού γινομένου D_1 x…x D_k .

Προβλήματα Ικανοποίησης Περιορισμών (3/3)

- **Λύση προβλήματος ικανοποίησης περιορισμών:**
  - Ανάθεση μιας τιμής σε κάθε μεταβλητή έτσι ώστε να μην παραβιάζεται κανένας περιορισμός.
- **Στόχοι:**
  - Εύρεση μιας λύσης.
  - Εύρεση όλων των λύσεων.
  - Εύρεση λύσης που μεγιστοποιεί (ελαχιστοποιεί) κάποια ποσότητα.
  - Εύρεση μιας προσεγγιστικής λύσης.
- **Δυαδικά** (binary) προβλήματα:
  - περιορισμοί ανάμεσα σε το πολύ δύο μεταβλητές.
- **Μη-δυαδικά** (non-binary) ή ***n*-αδικά** (n-ary) προβλήματα:
  - περιορισμοί ανάμεσα σε οσεσδήποτε μεταβλητές.

Γράφοι και Υπεργράφοι Περιορισμών (Constraint Graphs & Hypergraphs)

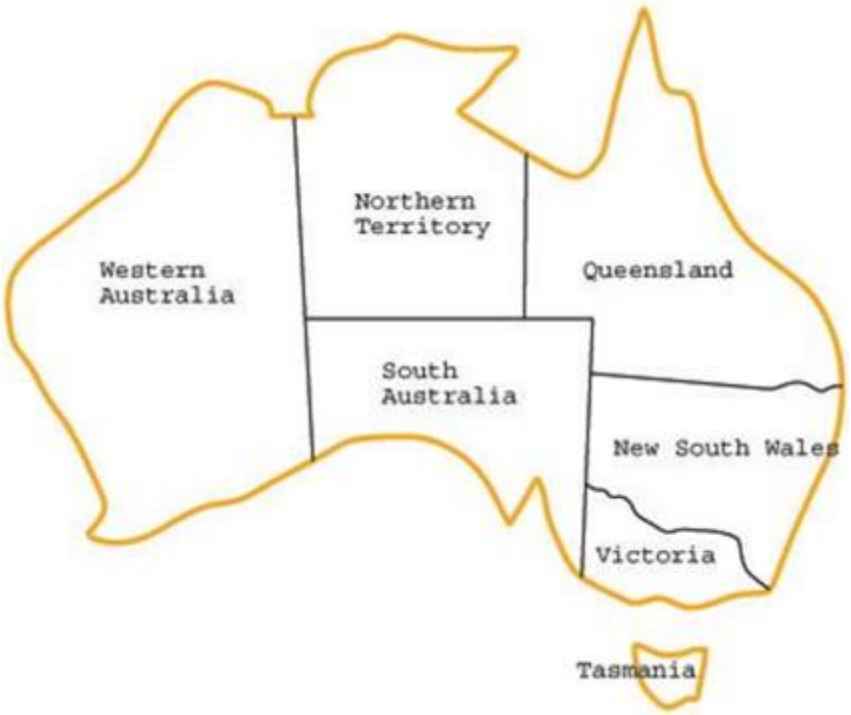




μεταβλητές – κόμβοι	μεταβλητές – κόμβοι
δυαδικοί περιορισμοί – ακμές	n-αδικοί περιορισμοί – υπερακμές

Απλό Παράδειγμα – Χρωματισμός Χαρτών (1/2)

- Θέλουμε να χρωματίσουμε κάθε περιοχή στο χάρτη με διαφορετικό χρώμα.
- Έχουμε τρία χρώματα red, green, blue

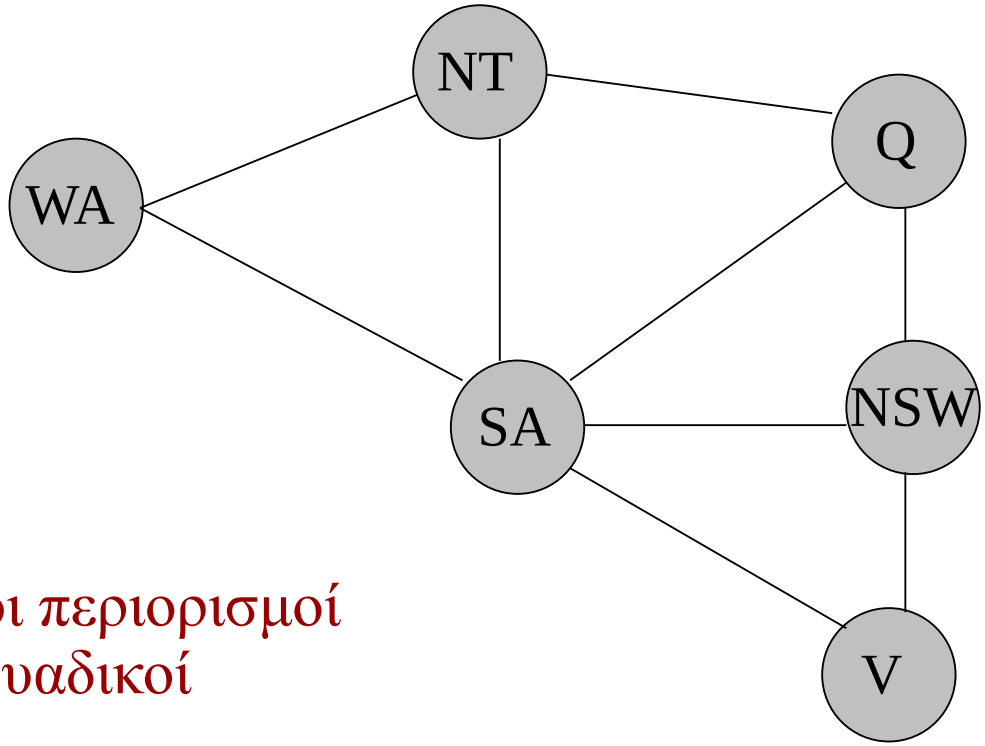


Απλό Παράδειγμα – Χρωματισμός Χαρτών (2/2)

Επίσημος Ορισμός:

- **Μεταβλητές:**
  - WA, NT, SA, Q, NSW, V, T
- **Πεδίο Ορισμού** (ίδιο για όλες τις μεταβλητές):
  - {red, green, blue}
- **Περιορισμοί:**
  - *C*(WA,NT) = {(red,green), (red,blue), (green,red), (green,blue), (blue, red), (blue,green)}
  - *C*(WA,SA) = …

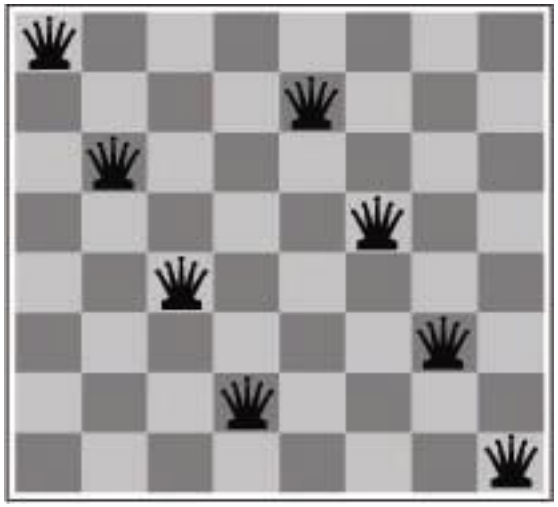
Γράφος Περιορισμών

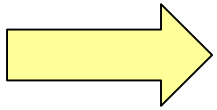


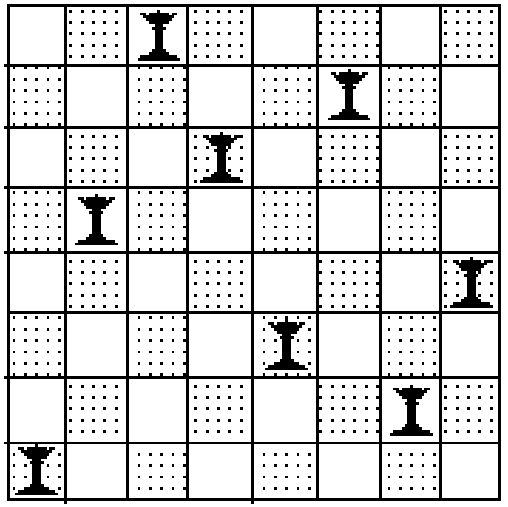
Δύο ασύνδετα τμήματα

Όλοι οι περιορισμοί είναι δυαδικοί

Απλό Παράδειγμα – 8 Queens problem (1/3)







Θέλουμε να τοποθετήσουμε 8 βασίλισσες στη σκακιέρα ώστε καμία να μην επιτίθεται σε άλλη

Απλό Παράδειγμα – 8 Queens problem (2/3)

Επίσημος Ορισμός:

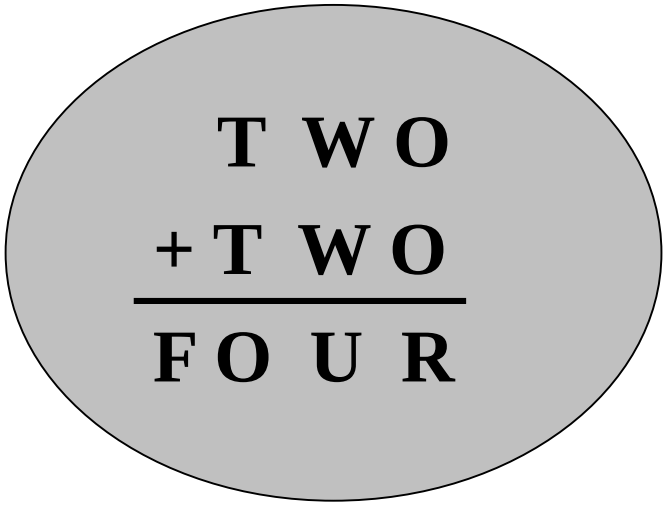
- **Μεταβλητές**
  - Η τιμή κάθε μεταβλητής X_i (i=1,…,8) αντιπροσωπεύει τη στήλη στην οποία βρίσκεται η i-th βασίλισσα στην i-th σειρά.

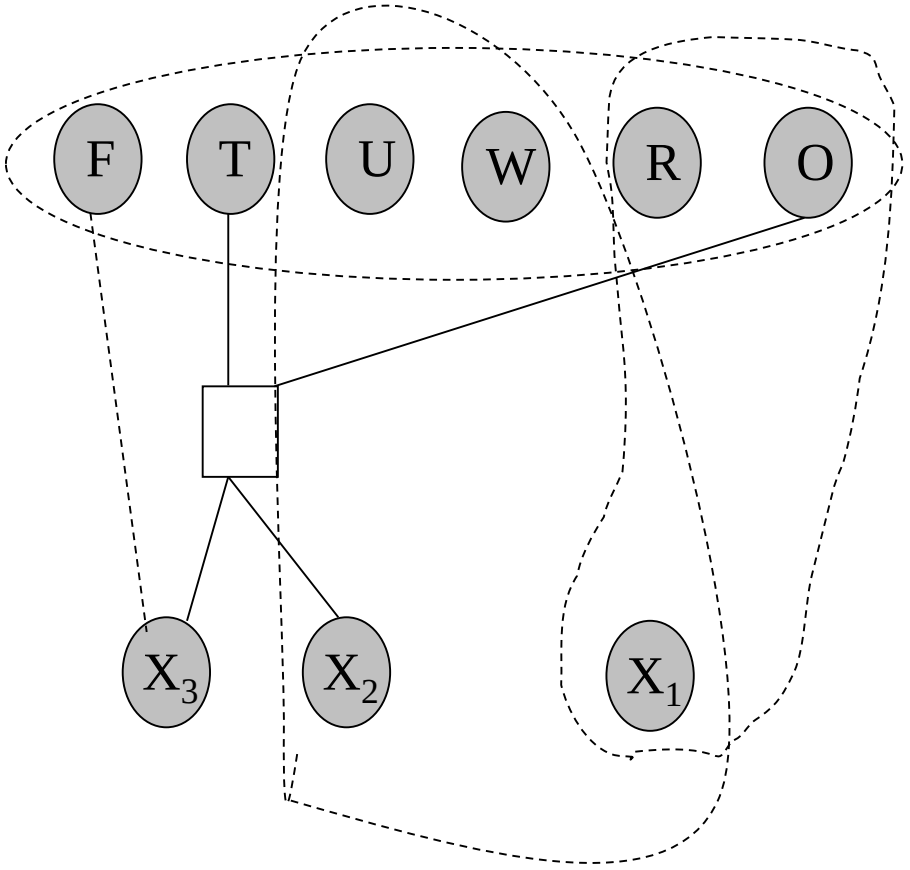
- **Πεδίο Ορισμού**
  - Αν οι στήλες αντιπροσωπεύονται με νούμερα από το 1 ως το 8 τότε το πεδίο ορισμού κάθε μεταβλητής X_i είναι D_i = {1,2,…,8}.

Απλό Παράδειγμα – 8 Queens problem (3/3)

- **Περιορισμοί:**
  - Υπάρχει ένας δυαδικός περιορισμός $C(X_i, X_j)$ για κάθε ζευγάρι μεταβλητών. Αυτοί οι περιορισμοί μπορούν να οριστούν ως εξής:
    - ✓ Για όλες τις μεταβλητές X_i και X_j , $X_i \neq X_j$.
    - ✓ Για όλες τις μεταβλητές X_i και X_j , με i>j, αν $X_i = a$ και $X_j = b$ τότε $i - j \neq a - b$ και $i - j \neq b - a$.

Απλό Παράδειγμα – Κρυπταριθμητική (1/2)

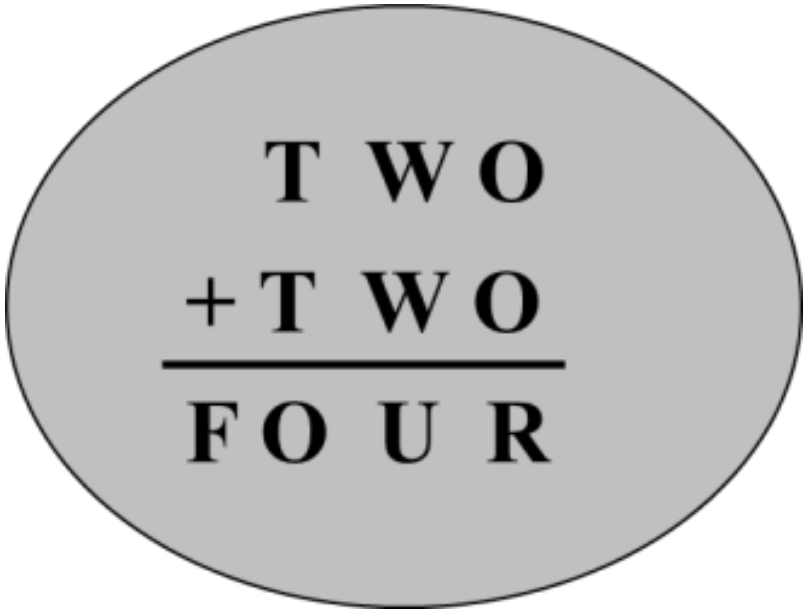




Απλό Παράδειγμα – Κρυπταριθμητική (2/2)

Επίσημος Ορισμός:

- **Μεταβλητές και Πεδία Ορισμού:**
  - $F, T, U, W, R, O \in \{0,1,2,3,4,5,6,7,8,9\}$.
  - $X_1, X_2, X_3 \in \{0,1\}$.
- **Περιορισμοί:**
  - *alldifferent*(F, T, U, W, R, O)
  - $O + O = R + 10\,X_1$
  - $X_1 + W + W = U + 10\,X_2$
  - $X_2 + T + T = O + 10\,X_3$
  - $X_3 = F$



Προβλήματα Περιορισμών

- **Χρονική Συλλογιστική** *(Temporal Reasoning).*
- **Timetabling.**
- **Scheduling.**
  - job-shop, aircrew.
- **Δρομολόγηση Οχημάτων** *(Vehicle Routing).*
- **Κατανομή Πόρων** *(Resource allocation).*
- **Σχεδιασμός Ενεργειών** *(Planning).*
- **Κατανομή Συχνοτήτων** *(Frequency Assignment).*
- **Χωρική Συλλογιστική** *(Spatial Reasoning).*
- **Ακέραιος, γραμμικός και μη-γραμμικός προγραμματισμός** *(integer, linear and non-linear programming).*

CSP Τεχνολογία: Πρακτική & Επιτυχημένη

- Η τεχνολογία ικανοποίησης περιορισμών είναι ένα από τα πιο επιτυχημένα παραδείγματα πρακτικής χρήσης ΤΝ.
- Υπάρχουν πολλές εταιρίες που κατασκευάζουν και εμπορεύονται συστήματα επίλυσης CSPs.
  - ILOG.
  - Cosytec.
  - Parc Technologies.
  - Sictus.
  - …

Περιορισμοί και Βάσεις Δεδομένων

Υπάρχουν στενές συνδέσεις μεταξύ CSPs and σχεσιακής θεωρίας βάσεων δεδομένων.

Constraint terminology	Database terminology
CSP	Database
Variable	Attribute
Domain	Attribute domain
Constraint	Table
Constraint scope	Table schema
Constraint tuples	Table instance
Set of solutions	Join of all tables

Περιορισμοί και Βάσεις Δεδομένων – Παράδειγμα (1/2)

- Consider the following CSP:
  - A set of variables $X = \{x_0,\ldots,x_9\}$.
  - All variables have the domain $D = \{0,1,2\}$.
  - There are constraints with the following allowed tuples:
    - ✓ $c_1 = \{x_0,x_1,x_3\} – \{(0,0,0), (0,1,0), (1,0,1), (1,1,1), (0,1,2)\}$.
    - ✓ $c_2 = \{x_1,x_2,x_3\} – \{(0,0,0), (0,0,1), (1,1,0), (1,0,1), (0,1,2)\}$.
    - ✓ $c_3 = \{x_1,x_4\} – \{(0,0), (1,1)\}$.
    - ✓ $c_4 = \{x_3,x_6\} – \{(0,0), (1,1), (1,0), (2,0)\}$.
    - ✓ $c_5 = \{x_4,x_5,x_6\} – \{(0,0,0), (0,0,1), (1,1,1), (1,0,2)\}$.
    - ✓ $c_6 = \{x_4,x_7\} – \{(0,1), (1,0)\}$.
    - ✓ $c_7 = \{x_5,x_8\} – \{(0,1), (1,0), (1,1)\}$.
    - ✓ $c_8 = \{x_6,x_9\} – \{(0,0), (1,1)\}$.

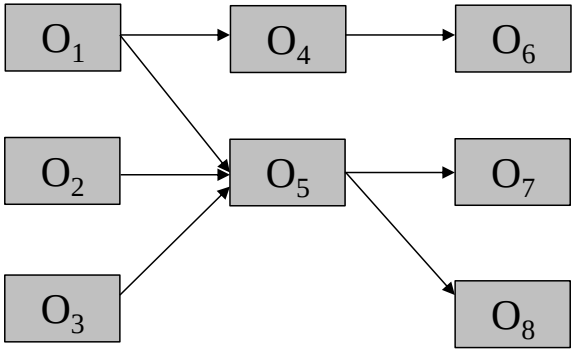
Περιορισμοί και Βάσεις Δεδομένων – Παράδειγμα (2/2)

The constraints as a relational database

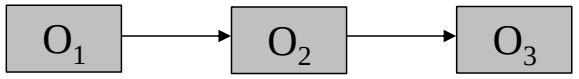
c_1			c_2			c_3		c_4		c_5			c_6		c_7		c_8	
x_0	x_1	x_3	x_1	x_2	x_3	x_1	x_4	x_3	x_6	x_4	x_5	x_6	x_4	x_7	x_5	x_8	x_6	x_9
0	0	0	0	0	0	0	0	0	0									
0	1	0	0	0	1	1	1	1	0				e	t	c.			
1	0	1	1	1	0			1	1									
1	1	1	1	0	1			2	0									
0	1	2	0	1	2													

Σύνθετο Παράδειγμα - Job-shop scheduling problem (JSSP) (1/2)

- Ένα JSSP απαιτεί το χρονοπρογραμματισμό ενός συνόλου διεργασιών $J=\{j_1, \ldots, j_n\}$ με χρήση ενός συνόλου φυσικών πόρων $RES=\{R_1, \ldots, R_m\}$.
  - Κάθε διεργασία j αποτελείται από ένα σύνολο λειτουργιών $O=\{O_1, \ldots, O_n\}$ που πρέπει να προγραμματιστούν με βάση ένα πλάνο που προσδιορίζει μια μερική διάταξη ανάμεσα σε αυτές τις λειτουργίες (π.χ. O_i BEFORE O_j).



Διεργασία 1



Διεργασία 2

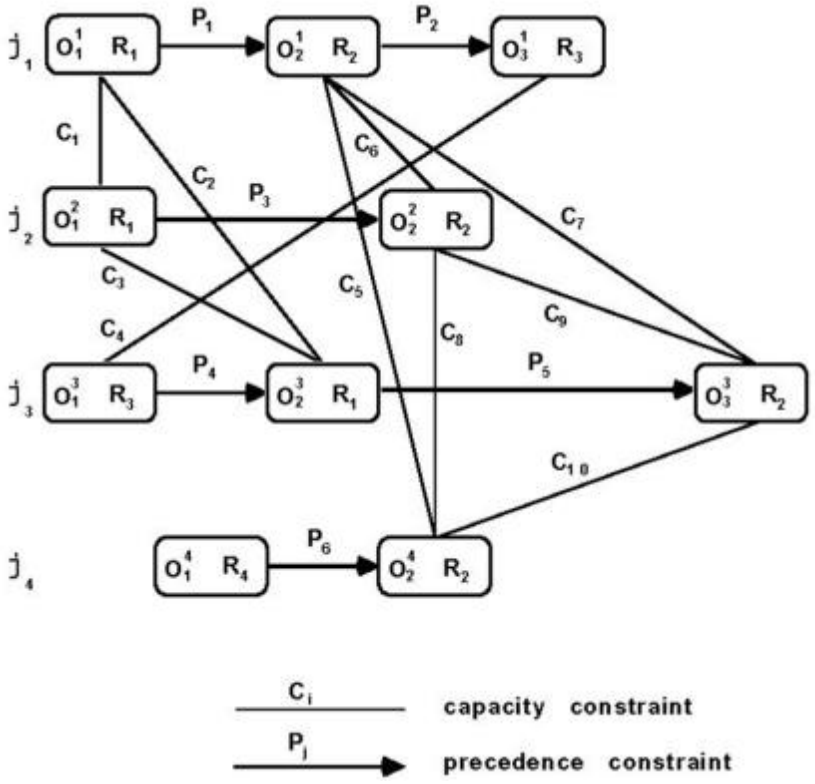
Σύνθετο Παράδειγμα - Job-shop scheduling problem (JSSP) (2/2)

- Κάθε διεργασία ***j*** έχει μια χρονική στιγμή εκκίνησης $\boldsymbol{rd_j}$ και μια χρονική στιγμή προθεσμίας ολοκλήρωσης $\boldsymbol{dd_j}$ ανάμεσα στις οποίες όλες οι λειτουργίες της πρέπει να έχουν ολοκληρωθεί.
- Κάθε λειτουργία $\boldsymbol{O_i}$ έχει συγκεκριμένη διάρκεια $\boldsymbol{du_i}$ και μια στιγμή εκκίνησης $\boldsymbol{st_i}$ της οποίας την τιμή πρέπει να προσδιορίσουμε.
- Το πεδίο ορισμού των πιθανών στιγμών εκκίνησης για κάθε λειτουργία αρχικά περιορίζεται από τη χρονική στιγμή εκκίνησης και τη χρονική στιγμή προθεσμίας ολοκλήρωσης της διεργασίας στην οποία ανήκει η λειτουργία.
- Για να εκτελεστεί επιτυχώς, κάθε λειτουργία $\boldsymbol{O_i}$ απαιτεί $\boldsymbol{p_i}$ διαφορετικούς πόρους (π.χ. μηχανές) $\boldsymbol{R_{ij}}$ ($1 \leq \boldsymbol{j} \leq \boldsymbol{p_i}$).

Το JSSP ως CSP (1/2)

- **Μεταβλητές:**
  - Ένα σύνολο μεταβλητών για κάθε λειτουργία, O_i, που αποτελείται από:
    - ✓ τη **χρονική στιγμή εκκίνησης**, st_i,
    - ✓ τις **απαιτήσεις πόρων**, R_{ij}.
- **Περιορισμοί:**
  - **Περιορισμοί διάταξης** που ορίζονται από το πλάνο κάθε διεργασίας μεταφρασμένοι σε γραμμικές ανισότητες του τύπου: $st_i + du_i \leq st_j$
    (δηλ. O_i BEFORE O_j).
  - **Περιορισμοί χωρητικότητας** που περιορίζουν τη χρήση καθενός πόρου σε μια λειτουργία κάθε χρονική στιγμή μεταφρασμένοι σε διαζευκτικούς περιορισμούς του τύπου:
    $(\forall p \forall q\ R_{ip} = R_{jq}) \ st_i + du_i \leq st_j \vee st_j + du_j \leq st_i$.
    - ✓ Αυτοί οι περιορισμοί απλά εκφράζουν το ότι αν δύο λειτουργίες O_i και O_j χρησιμοποιούν τον ίδιο πόρο τότε δε μπορούν να επικαλύπτονται χρονικά.

Το JSSP ως CSP (2/2)



Ένα job shop πρόβλημα με 4 διεργασίες.

Κάθε κόμβος αντιστοιχεί σε μια λειτουργία που απαιτεί έναν πόρο.

Κάθε λειτουργία απαιτεί τη χρήση μόνο ενός συγκεκριμένου πόρου.

Οι στιγμές εκκίνησης των λειτουργιών είναι οι μόνες μεταβλητές.

Σύνθετο Παράδειγμα – Car Sequencing (1/2)

- Στην παραγωγή αυτοκινήτων, τα αυτοκίνητα τοποθετούνται σε conveyor belts οι οποίες μετακινούνται σε διάφορες περιοχές στο χώρο εργασίας.
  - Μια γραμμή παραγωγής συνήθως πρέπει να παράγει αυτοκίνητα διαφορετικών μοντέλων. Το πλήθος των αυτοκινήτων που απαιτούνται για κάθε μοντέλο ονομάζεται **απαίτηση παραγωγής**.
  - Κάθε περιοχή εργασίας περιορίζεται από έναν **Περιορισμό Χωρητικότητας**.
- **Μεταβλητές** – οι θέσεις στην εφοδιαστική ταινία που θα καταληφθούν από αυτοκίνητα *(π.χ. Αν υπάρχουν n αυτοκίνητα, έχουμε n μεταβλητές).*
- **Πεδία τιμών** – το σύνολο των μοντέλων,
  π.χ. «από μοντέλο A ως D».
- **Στόχος** – να δοθεί μια τιμή *(ένα μοντέλο)* σε κάθε μεταβλητή *(θέση στη conveyor belt),* έτσι ώστε να ικανοποιούνται οι απαιτήσεις παραγωγής και οι περιορισμοί χωρητικότητας.

Σύνθετο Παράδειγμα – Car Sequencing (2/2)

Production Requirements:

Options (✓ = required, × = not):	Model A	Model B	Model C	Model D	
Sunroof	×	✓	✓	×	
Radio cassette	✓	×	✓	✓	
Air-conditioning	✓	✓	×	✓	
Anti-rust treatment	×	✓	✓	✓	
Power brakes	✓	×	✓	×	
Number of cars required:	30	30	20	40	Total: 120

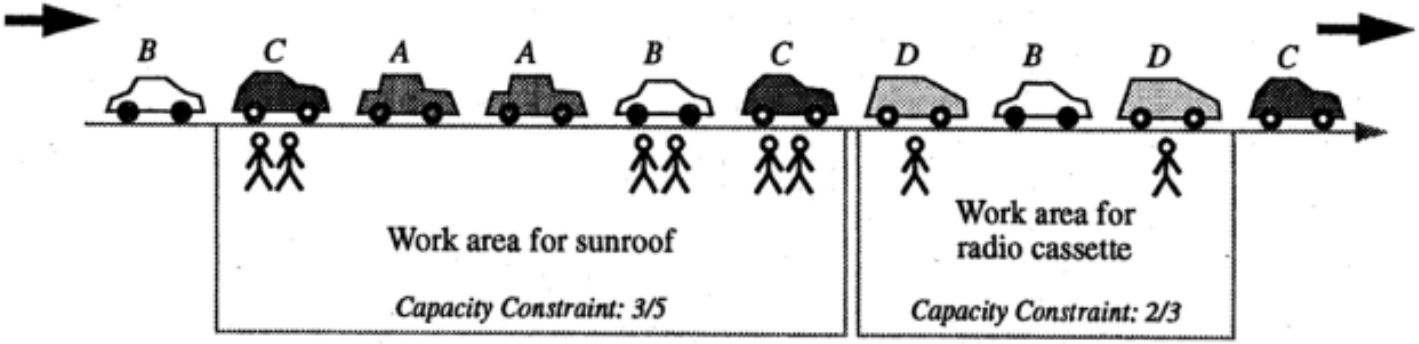


Figure 1.2 Example of a car sequencing problem

Επίλυση Προβλημάτων Περιορισμών

- Υποθέτοντας ότι έχουμε αναπαραστήσει ένα πρόβλημα ως CSP:
  - πως μπορούμε να βρούμε μια λύση (αν υπάρχει);
  - πως μπορούμε να βρούμε όλες τις λύσεις;
  - πως μπορούμε να παράγουμε καινούργια γνώση;
    - ✓ π.χ. καινούργιους περιορισμούς.
- Τεχνικές που μπορούν να χρησιμοποιηθούν:
  - **"Παρήγαγε κι εξέτασε"** *(generate and test).*
  - **Αλγόριθμοι οπισθοδρόμησης** *(backtracking search algorithms)*.
  - **Αλγόριθμοι τοπικής αναζήτησης** *(local search algorithms).*
  - **Αλγόριθμοι διάδοσης περιορισμών** *(constraint propagation algorithms).*

Generate & Test

- Η πιο γενική μέθοδος
- **Αλγόριθμος:**

  δώσε από μια τιμή σε όλες τις μεταβλητές

  έλεγξε αν είναι λύση.

Μειονεκτήματα:

Τυφλή γεννήτρια αναθέσεων τιμών

Αργή ανακάλυψη ασυνεπειών

Βελτιώσεις:

έξυπνη γεννήτρια,
→ τοπική αναζήτηση,

έλεγχος περιορισμών κατά τη διάρκεια της ανάθεσης ,
→ αναζήτηση οπισθοδρόμησης.

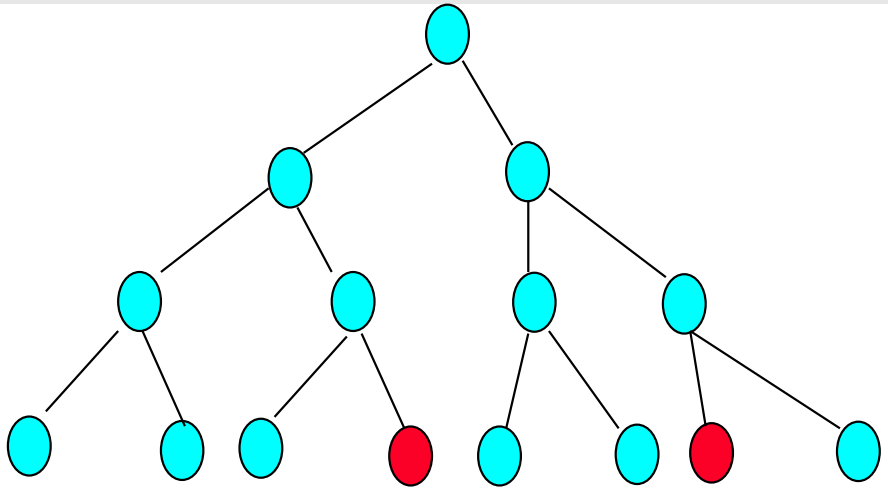
Αλγόριθμοι Αναζήτησης για CSPs (1/3)

Γενικός Αλγόριθμος Αναζήτησης για CSPs:

- **Αρχική Κατάσταση:**
  - Δεν έχει ανατεθεί τιμή σε καμία μεταβλητή.
- **Ενέργειες:**
  - Δώσε σε μια μεταβλητή X_i (που δεν έχει τιμή) μια τιμή από το D_i.
- **Τεστ Στόχου:**
  - Έχει γίνει ανάθεση τιμών σε όλες τις μεταβλητές και όλοι οι περιορισμοί ικανοποιούνται.
- Η σειρά εκτέλεσης των ενεργειών δεν έχει σημασία
  - Μπορούμε να το εκμεταλλευτούμε αυτό!

Ο Χώρος Αναζήτησης των CSPs

- Το μέγεθος του χώρου αναζήτησης είναι πεπερασμένο.
- Το βάθος του δέντρου αναζήτησης είναι συγκεκριμένο:
  - Ίσο με το πλήθος των μεταβλητών.
- Οι λύσεις βρίσκονται **πάντα** στα φύλλα του δέντρου αναζήτησης.



Φύλλα

Αλγόριθμοι Αναζήτησης για CSPs (2/3)

- Ποιος αλγόριθμος αναζήτησης φαίνεται να ταιριάζει σε CSPs;
- **Breadth-First Search;**
  - Όχι! Η BFS δε θα είναι αποτελεσματική γιατί οι λύσεις βρίσκονται στα φύλλα.
- **Depth-First Search;**
  - Καλύτερη από την BFS. Αλλά συχνά θα σπαταλάει χρόνο αναζητώντας ενώ έχουν ήδη παραβιαστεί περιορισμοί.
- **Hill Climbing;**
  - Ναι όταν μας ενδιαφέρει κυρίως η ταχύτητα.

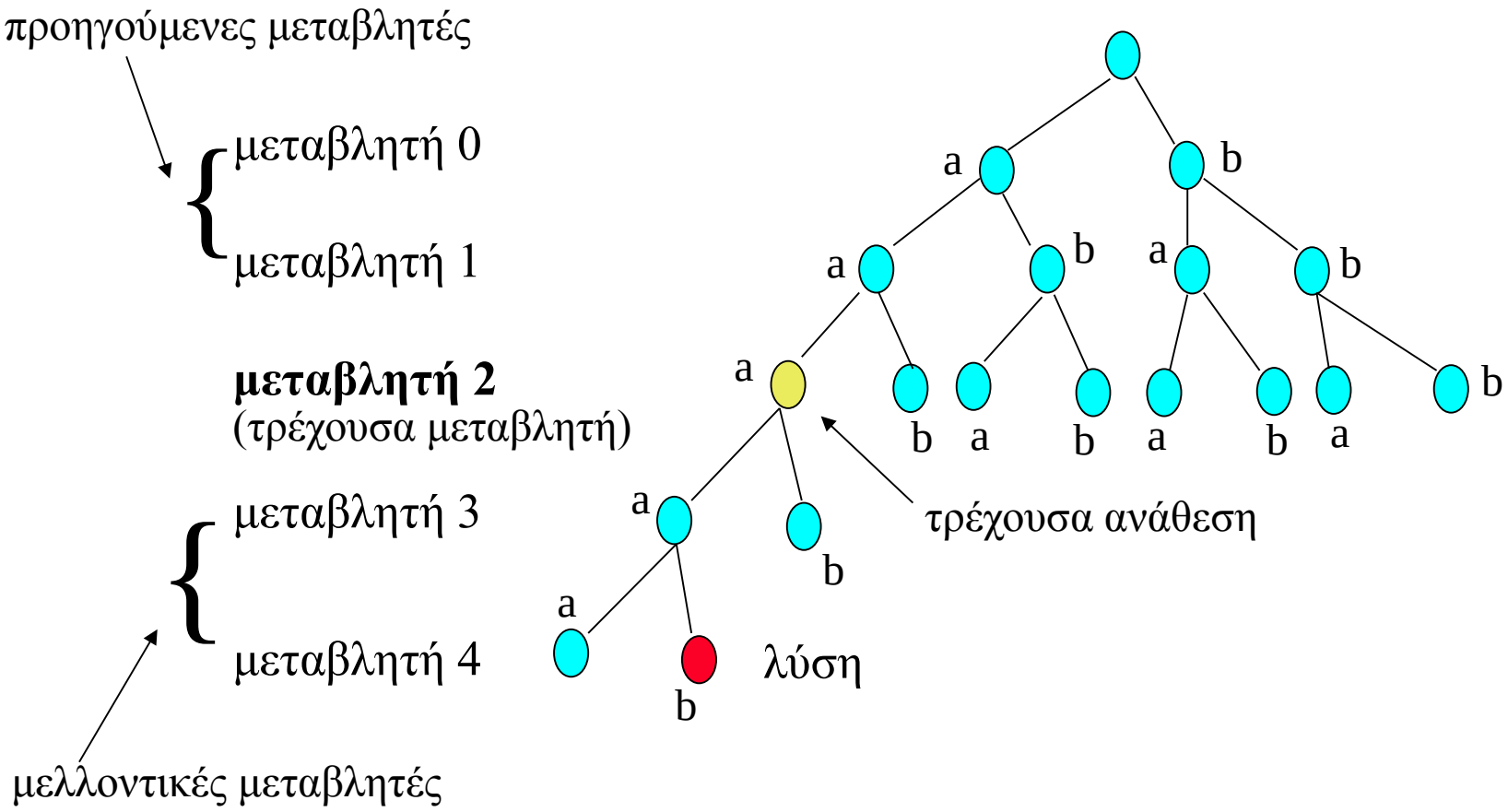
Αλγόριθμοι Αναζήτησης για CSPs (3/3)

- Θα μελετήσουμε παραλλαγές του DFS ειδικά για CSPs.
  - Οι αλγόριθμοι αυτοί βασίζονται στην ιδέα της **αναζήτησης με οπισθοδρόμηση** *(backtracking search).*
    - **Simple or Chronological Backtracking (BT).**
    - **Backjumping (BJ).**
    - **Forward Checking (FC).**
    - Maintaining Arc Consistency (MAC).
- Επίσης μια παραλλαγή του hill climbing
  - **Min-conflicts**

Chronological Backtracking (BT) (1/4)

- Η βασική ιδέα σε όλους τους αλγόριθμους οπισθοδρόμησης είναι να ξεκινάμε με μια **μερική λύση** *(δηλ. μερική ανάθεση μεταβλητών)* και να συνεχίζουμε τις αναθέσεις μέχρι να φτάσουμε σε **πλήρη λύση.**
- Ο BT ακολουθεί αυτή την τεχνική:
  - Αν φτάσει σε **αδιέξοδο** *(dead end)* οπισθοδρομεί στην αμέσως προηγούμενη επιλογή του.
    - Αδιέξοδο έχουμε όταν δε μπορούμε να κάνουμε ανάθεση τιμής σε μεταβλητή χωρίς να παραβιαστεί κάποιος περιορισμός,
  - και δοκιμάζει μια άλλη τιμή για τη συγκεκριμένη μεταβλητή.

Chronological Backtracking (BT) (2/4)

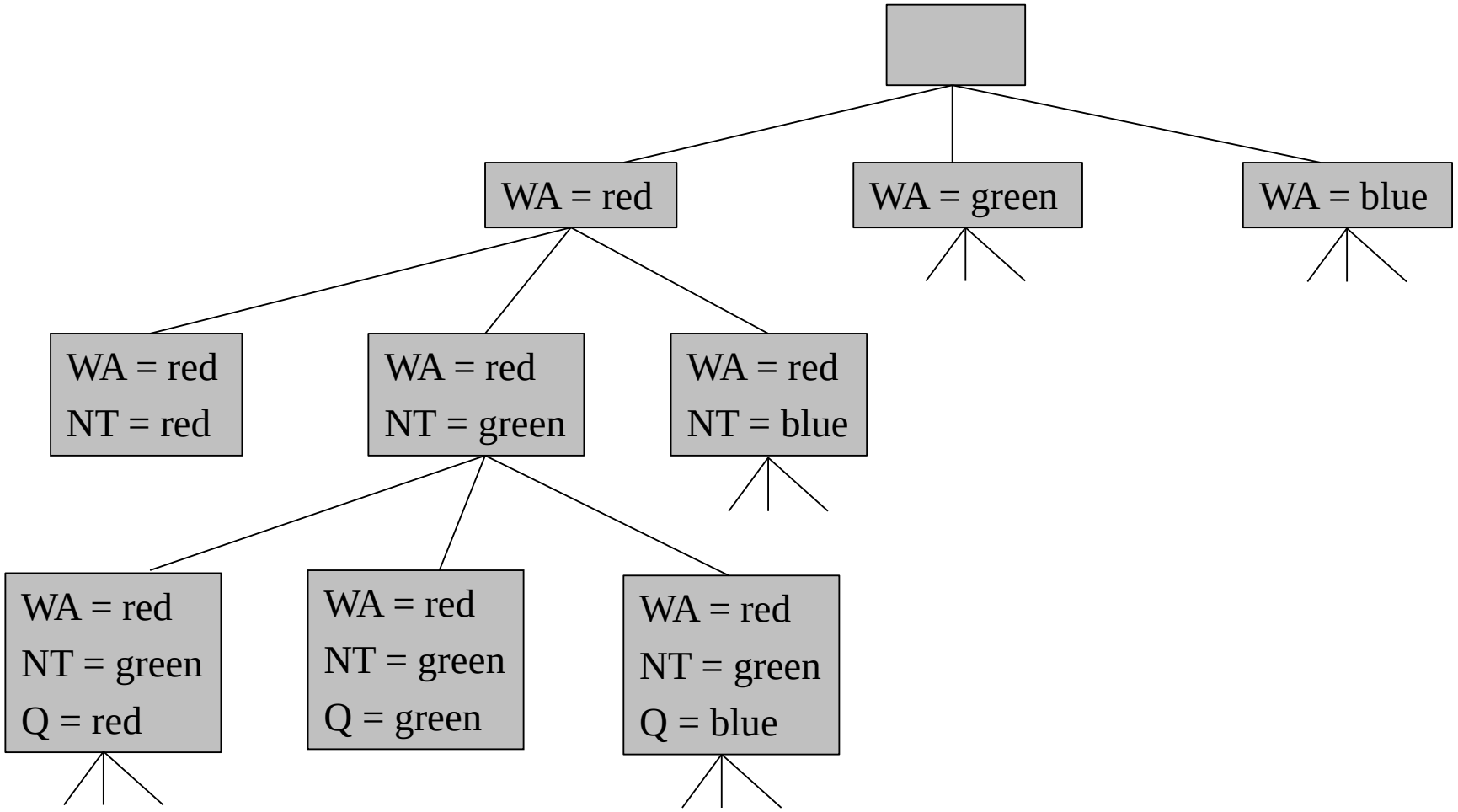


Chronological Backtracking (BT) (3/4)

procedure CHRONOLOGICAL_BACKTRACKING (*vars*,*doms*,*cons*)
    solution ← BT (*vars*,Ø,*doms*,*cons*)

function BT (*unlabelled,compound_label,doms,cons*)
returns a solution or NIL
    if *unlabelled* = Ø **then return** *compound_label*
    else pick a variable *x* from *unlabelled*
        repeat
            pick a value *v* from D_x; delete *v* from D_x
            if *compound_label* + *{(x,v)}* violates no constraints **then**
                result ← BT(*unlabelled* - {*x*}, *compound_label* + {(*x*,*v*)}, *doms*,*cons*)
                if *result* ≠ NIL **then return** *result*
            end
        until D_x = Ø
        return NIL
    end

Chronological Backtracking *(σε δράση)*



Chronological Backtracking (BT) (4/4)

Αξιολόγηση:

- **Πλήρης;**
  - Ναι.
- **Χρόνος:** $O(\delta^n e)$
  - ‘Οπου δ το μέγιστο μέγεθος πεδίου ορισμού, n το πλήθος των μεταβλητών, και e το πλήθος των περιορισμών.
- **Χώρος:** $O(n\delta)$
  - Ο χώρος που απαιτείται για την αποθήκευση των πεδίων ορισμού όλων των μεταβλητών.
- Οι πολυπλοκότητες ισχύουν υποθέτοντας ότι οι περιορισμοί αποθηκεύονται σε σταθερή ποσότητα χώρου και οι έλεγχοι περιορισμών γίνονται σε σταθερό χρόνο.

GT & BT – Παράδειγμα 1

- Πρόβλημα:

  X::{1,2}, Y::{1,2}, Z::{1,2}

  $X = Y$, $X \neq Z$, $Y > Z$

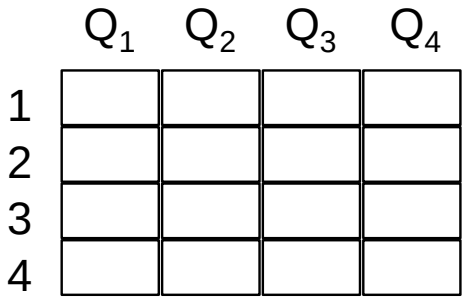
generate & test

X	Y	Z	test
1	1	1	fail
1	1	2	fail
1	2	1	fail
1	2	2	fail
2	1	1	fail
2	1	2	fail
2	2	1	passed

backtracking

X	Y	Z	test
1	1	1	fail
		2	fail
	2		fail
2	1		fail
	2	1	passed

GT & BT 4-queen πρόβλημα

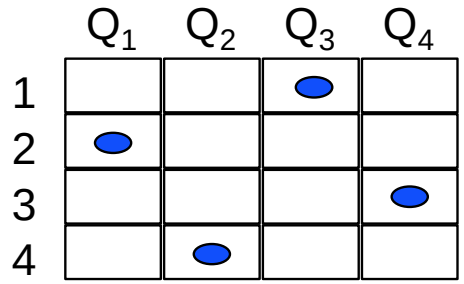


Place 4 queens so that no two queens are in attack.

Q_i: line number of queen in column i, for $1 \le i \le 4$

$Q_1, Q_2, Q_3, Q_4 \in \{1, 2, 3, 4\}$
$Q_1 \neq Q_2$, $Q_1 \neq Q_3$, $Q_1 \neq Q_4$,
$Q_2 \neq Q_3$, $Q_2 \neq Q_4$, $Q_3 \neq Q_4$,
$Q_1 \neq Q_2-1$, $Q_1 \neq Q_2+1$, $Q_1 \neq Q_3-2$, $Q_1 \neq Q_3+2$,
$Q_1 \neq Q_4-3$, $Q_1 \neq Q_4+3$,
$Q_2 \neq Q_3-1$, $Q_2 \neq Q_3+1$, $Q_2 \neq Q_4-2$, $Q_2 \neq Q_4+2$,
$Q_3 \neq Q_4-1$, $Q_3 \neq Q_4+1$

4-queen πρόβλημα

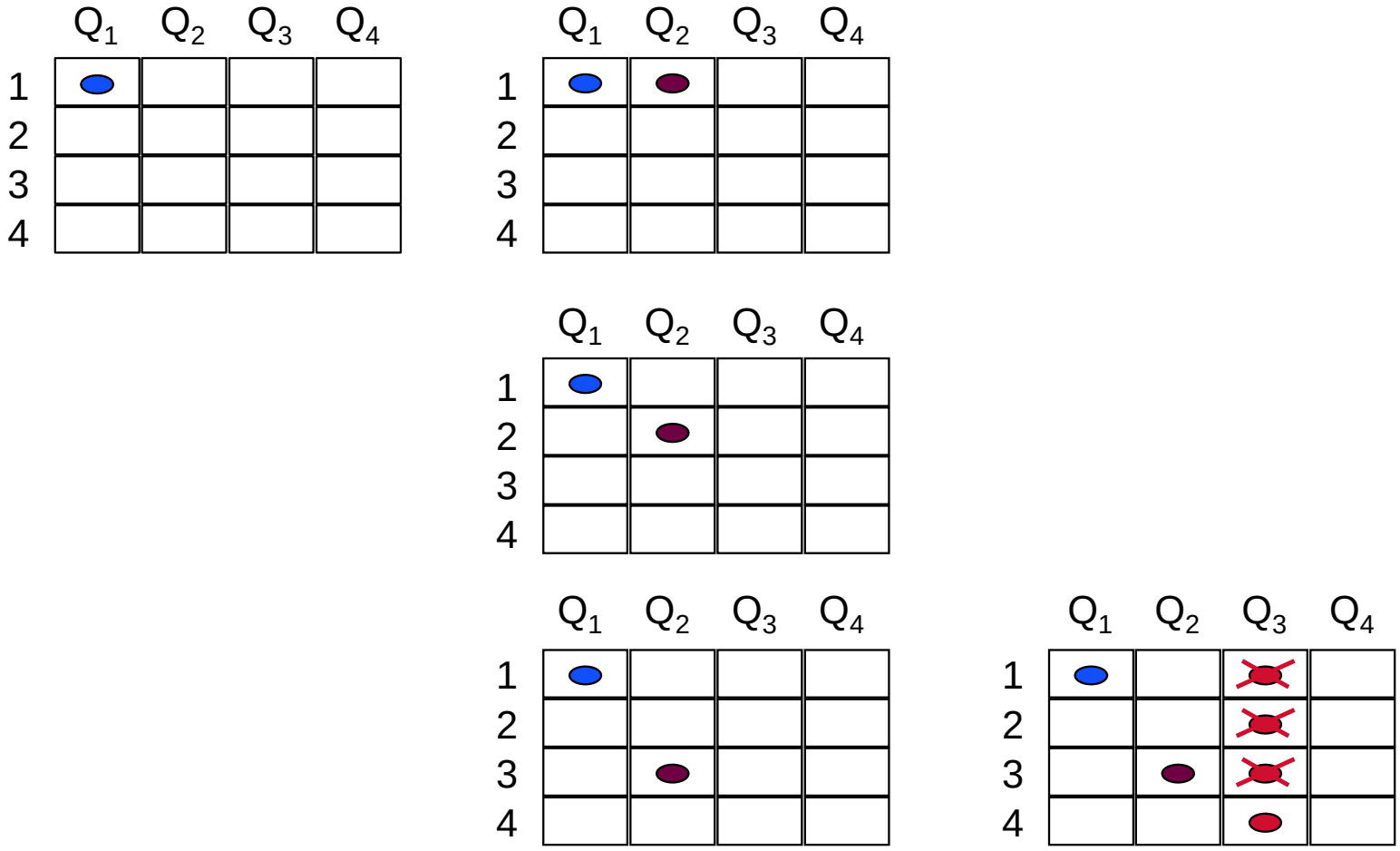


Υπάρχουν συνολικά 256 δυνατές αναθέσεις τιμών

Ο GT αλγόριθμος θα παράγει

	64	αναθέσεις με $Q_1=1$;
+	48	αναθέσεις με $Q_1=2$, $1 \le Q_2 \le 3$;
+	3	αναθέσεις με $Q_1=2$, $Q_2=4$, $Q_3=1$;
=	115	αναθέσεις μέχρι να βρει την πρώτη λύση

4-queen πρόβλημα, BT αλγόριθμος

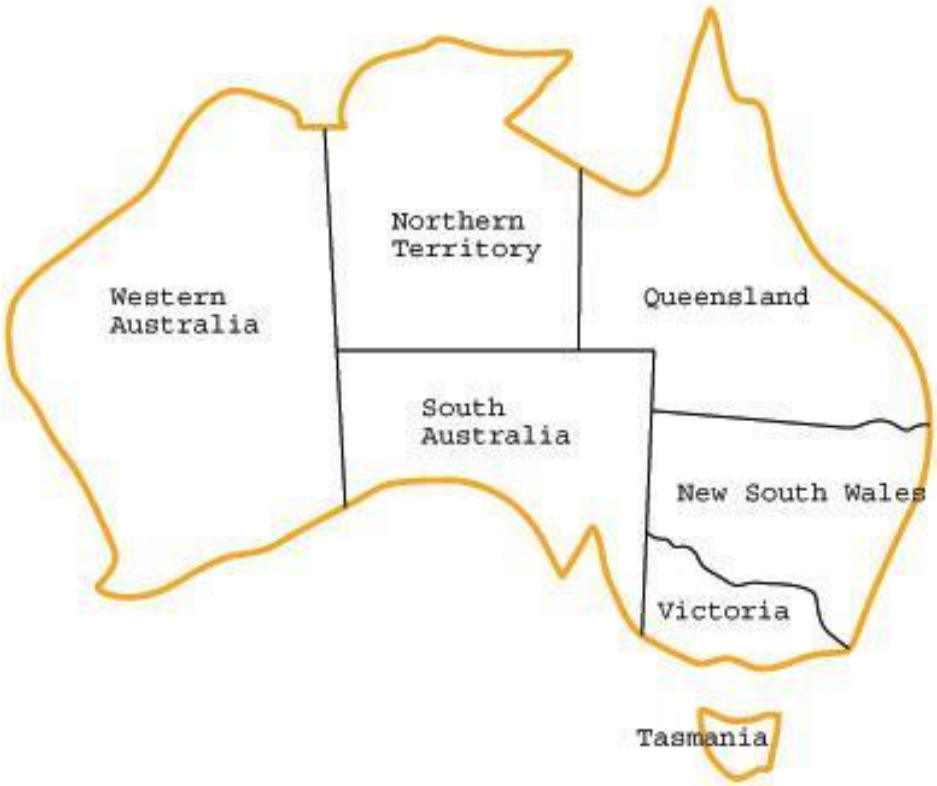


Έξυπνο Backtracking

- Ο BT υποφέρει από το φαινόμενο του **thrashing:**
  - Επισκέπτεται ξανά και ξανά παρόμοιες περιοχές του δέντρου αναζήτησης επειδή έχει πολύ τοπική εικόνα του προβλήματος.
- Ένας τρόπος απαλοιφής του προβλήματος είναι οι αλγόριθμοι **έξυπνης οπισθοδρόμησης** *(intelligent backtracking).*
  - BJ, CBJ, DB, Graph-based BJ, Learning.
- Ο **backjumping** (**BJ**) διαφέρει από τον BT στο εξής:
  - Όταν φτάνει σε αδιέξοδο, ο BJ δεν οπισθοδρομεί στην αμέσως προηγούμενη μεταβλητή. Αλλά οπισθοδρομεί στην πιο βαθιά μεταβλητή στο δέντρο αναζήτησης που είναι σε σύγκρουση με την τρέχουσα μεταβλητή.

BJ vs. BT (1/3)

- We want to color each area in the map with a different color



- We have three colors
  red, green, blue

BJ vs. BT (2/3)

- Ας σκεφτούμε τι κάνει ο BT στο πρόβλημα χρωματισμού χάρτη:
  - Υποθέτουμε ότι η ανάθεση μεταβλητών γίνεται με τη σειρά *Q*, *NSW*, *V*, *T*, *SA*, *WA*, *NT.*
  - Ας υποθέσουμε ότι έχουμε φτάσει στη μερική ανάθεση τιμών

    Q = *red*, *NSW* = *green*, *V* = *blue*, *T* = *red.*
  - Όταν δοκιμάσουμε να δώσουμε τιμή στην επόμενη μεταβλητή *SA*, βλέπουμε ότι όλες οι τιμές παραβιάζουν κάποιον περιορισμό:
    - Αδιέξοδο!
  - Ο BT μας λέει τώρα να οπισθοδρομήσουμε και να δοκιμάσουμε μια νέα τιμή για την *T*!
    - Not a good idea!

BJ vs. BT (3/3)

- Ο BJ έχει μια πιο έξυπνη προσέγγιση στην οπισθοδρόμηση:
  - Μας λέει να γυρίσουμε πίσω σε μια από τις μεταβλητές που είναι υπεύθυνες για το αδιέξοδο.
  - Το σύνολο αυτό μεταβλητών ονομάζεται **σύνολο συγκρούσεων** *(conflict set).*
  - Το conflict set για την SA είναι {Q, NSW, V}
  - Ο BJ οπισθοδρομεί στην **πιο βαθιά** μεταβλητή του conflict set της μεταβλητής όπου συναντήσαμε αδιέξοδο:
    - ✓ πιο βαθιά = αυτή που επισκεφτήκαμε πιο πρόσφατα.
- Ο **BJ** ανήκει στην οικογένεια αλγορίθμων οπισθοδρόμησης που ονομάζονται **lookback** αλγόριθμοι.
  - **CBJ**, **DB**, **Graph-based BJ**, **Learning.**

Forward Checking (1/4)

- Ο **Forward Checking** (FC) ανήκει στην οικογένεια αλγορίθμων οπισθοδρόμησης που ονομάζονται **lookahead** αλγόριθμοι.
  - Η βασική ιδέα του lookahead είναι ότι μόλις ανατεθεί μια τιμή σε μια μεταβλητή το μέγεθος του προβλήματος ελαττώνεται μέσω της **διάδοσης περιορισμών** *(constraint propagation).*
    - ✓ Η διάδοση περιορισμών ορίζεται διαφορετικά για κάθε lookahead αλγόριθμο.
- Ο FC κάνει το εξής:
  - Μόλις μια μεταβλητή *x* πάρει μια τιμή *v*, για κάθε μελλοντική μεταβλητή *y* που εμφανίζεται σε περιορισμό μαζί με τη *x* αφαιρούνται από το D_y όλες οι τιμές που δεν είναι συμβατές με τη *v*.

Forward Checking (2/4)

- Αν το πεδίο ορισμού κάποιας μεταβλητής μείνει άδειο τότε απορρίπτεται η τιμή *v* για τη *x* και δοκιμάζουμε την επόμενη.
- Η λειτουργία του FC έχει ως αποτέλεσμα να ισχύει το εξής σε κάθε βήμα της αναζήτησης:
  - Όλες οι τιμές κάθε μελλοντικής μεταβλητής (future variable) είναι συμβατές με όλες τις τιμές που έχουν ανατεθεί σε προηγούμενες μεταβλητές *(past variables).*
- Ο FC διατηρεί μια περιορισμένη μορφή ***συνέπειας τόξου*** (**arc consistency).**

Forward Checking (3/4)

```
procedure FORWARD_CHECKING (vars,doms,cons)
    solution ← FC (vars,Ø,doms,cons)

function FC (unlabelled,compound_label,doms,cons)
returns a solution or NIL
    if unlabelled = Ø then return compound_label
    else  pick a variable x from unlabelled
          repeat
              pick a value v from D_x; delete v from D_x
              doms' ← UPDATE(unlabelled-{x},doms,cons,compound_label +
    {(x,v)})                    if no domain in doms' is empty then
                        result ← FC(unlabelled - {x}, compound_label +
                                                        {(x,v)}, doms',cons)
                        if result ≠ NIL then return result
          end
          until D_x = Ø
          return NIL
    end
```

Forward Checking (4/4)

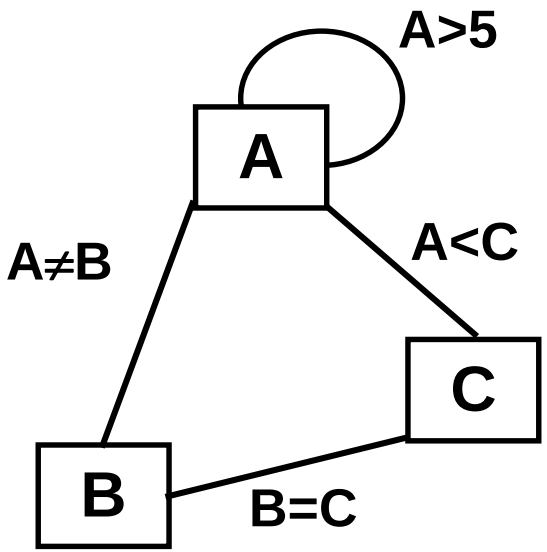
function UPDATE (*unlab_vars,doms,cons,compound_label*)
returns an updated set of domains
    for each variable *y* in *unlab_vars* **do**
        for each value *v* in D_y' **do**
            if (*y*,*v*) is incompatible with *compound_label* with respect
        to the constraints between *y* and the variables of *compound_label*
            then $D_y' \leftarrow D_y' - \{v\}$
        end
    end
    return *doms*'

Ο FC σε λειτουργία

	WA	NT	Q	NSW	V	SA	T
Αρχικά πεδία τιμών	R G B	R G B	R G B	R G B	R G B	R G B	R G B
Μετά από WA=R	R	G B	R G B	R G B	R G B	G B	R G B
Μετά από Q=G	R	B	G	R B	R G B	B	R G B
Μετά από V=B	R	B	G	R	B		R G B

Consistency Techniques *(Τεχνικές Συνέπειας)*

- Διαγράφουν μη-συνεπείς τιμές από τα πεδία ορισμού των μεταβλητών.
  - Μπορούν να εφαρμοστούν πριν ή κατά τη διάρκεια της αναζήτησης.
- Για δυαδικά CSPs:
  - **Συνέπεια κόμβου** *(node consistency - NC).*
  - **Συνέπεια τόξου** *(arc consistency - AC).*
  - **Συνέπεια μονοπατιού** *(path consistency - PC).*
  - **k-συνέπεια** *(k-consistency).*



Συνέπεια Κόμβου *(Node Consistency)*

- Μια μεταβλητή *X* είναι **node consistent** αν κάθε τιμή *a* της *X* είναι συνεπής με κάθε μοναδιαίο (unary) περιορισμό στη *X.*
- Π.χ. D(X) = {0,1,2,3,4}

  $X > 1$

  $X \neq 4$

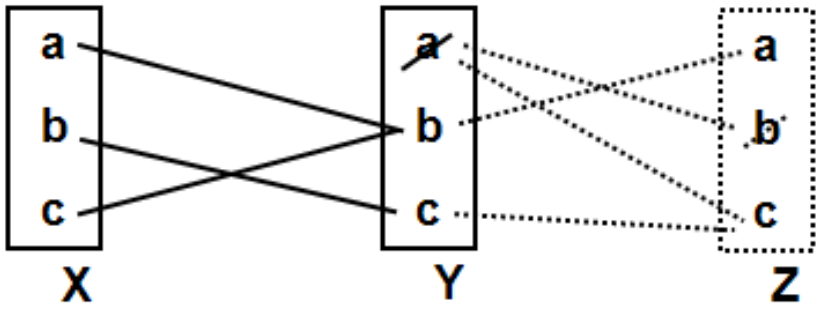
Η εφαρμογή του node consistency θα δώσει D(X)= {2,3}.

Αυτή η τεχνική μπορεί να εφαρμοστεί μια για πάντα πριν ξεκινήσει η αναζήτηση (ως βήμα προ-επεξεργασίας).

Συνέπεια Τόξου (Arc Consistency)

Μια μεταβλητή *X* είναι **arc consistent** αν για κάθε άλλη μεταβλητή *Y* ισχύει το εξής: Για κάθε τιμή *a* της *X* υπάρχει τουλάχιστον μια τιμή *b* της *Y* τέτοια ώστε η *a* και *b* να είναι συμβατές.

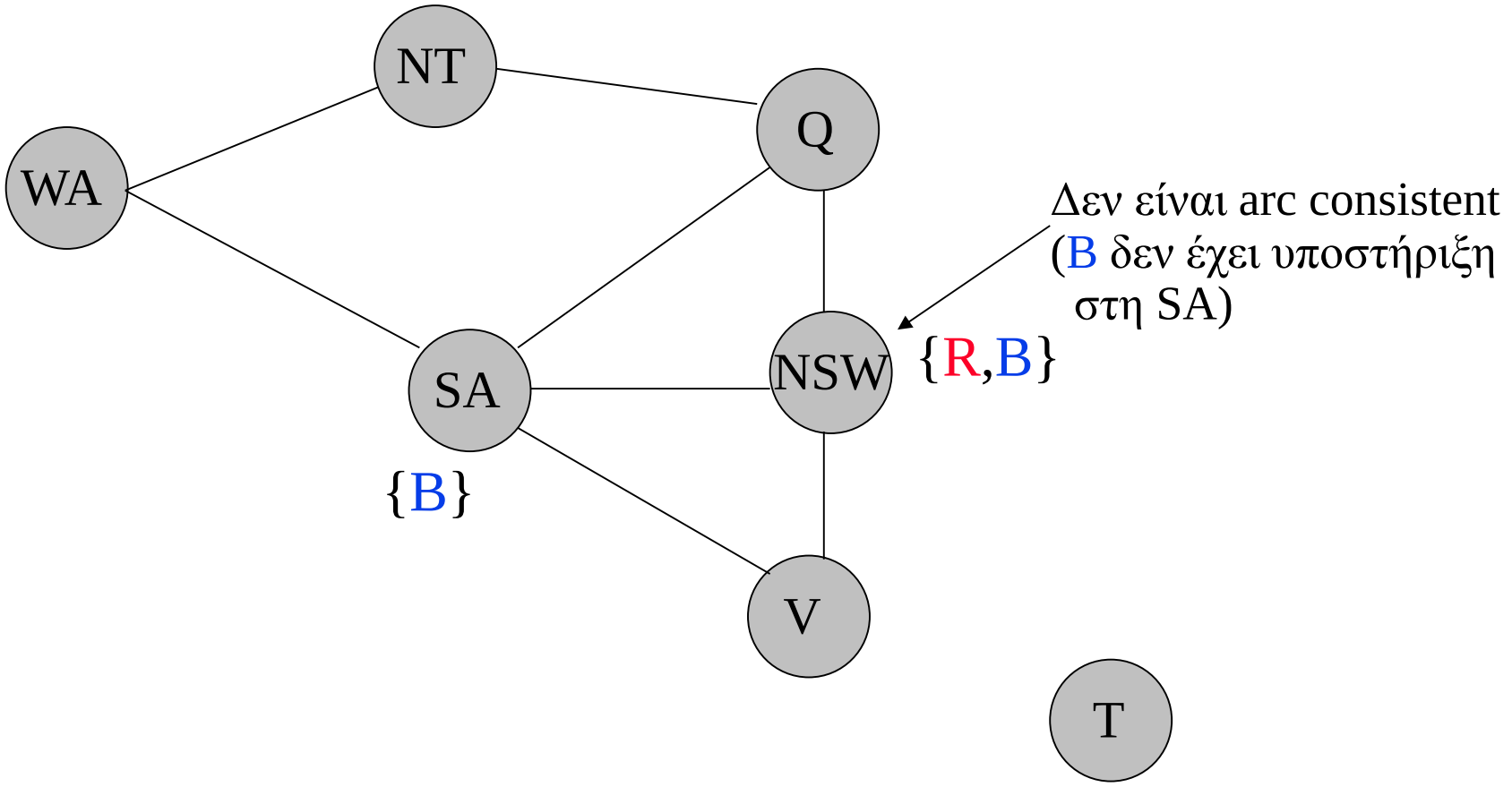
- Τότε λέμε ότι η *a* **υποστηρίζει** (supports) την *b.*
- Ένας αλγόριθμος που εφαρμόζει arc consistency σβήνει τιμές από το πεδίο ορισμού μιας μεταβλητής όταν αυτές δεν υποστηρίζονται από καμία τιμή σε μια άλλη μεταβλητή.



Είναι η πιο ευρέως διαδεδομένη τεχνική συνέπειας (απομακρύνει αρκετές τιμές με χαμηλό κόστος).

Επεξεργάζεται έναν-έναν τους δυαδικούς περιορισμού

Arc Consistency - Παράδειγμα

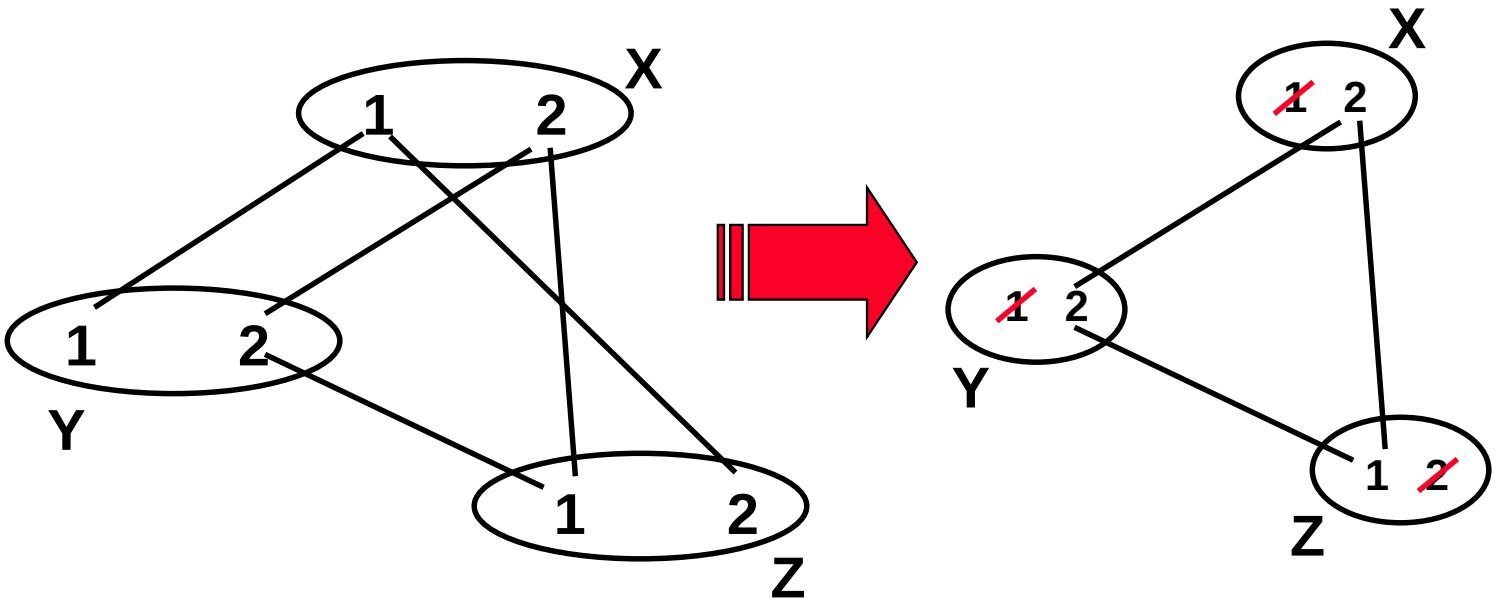


AC – Παράδειγμα (1/2)

- **Πρόβλημα:**

  X::{1,2}, Y::{1,2}, Z::{1,2}

  X = Y, X ≠ Z, Y > Z



AC – Παράδειγμα (2/2)

- **Πρόβλημα:**

  X::{1,2}, Y::{1,2}, Z::{1,2}

  $X = Y, X \neq Z, Y > Z$

X	Y	Z	action	result
1			labelling	
	{1}	{}	AC propagation	fail
2			labelling	
	{2}	{1}	AC propagation	solution

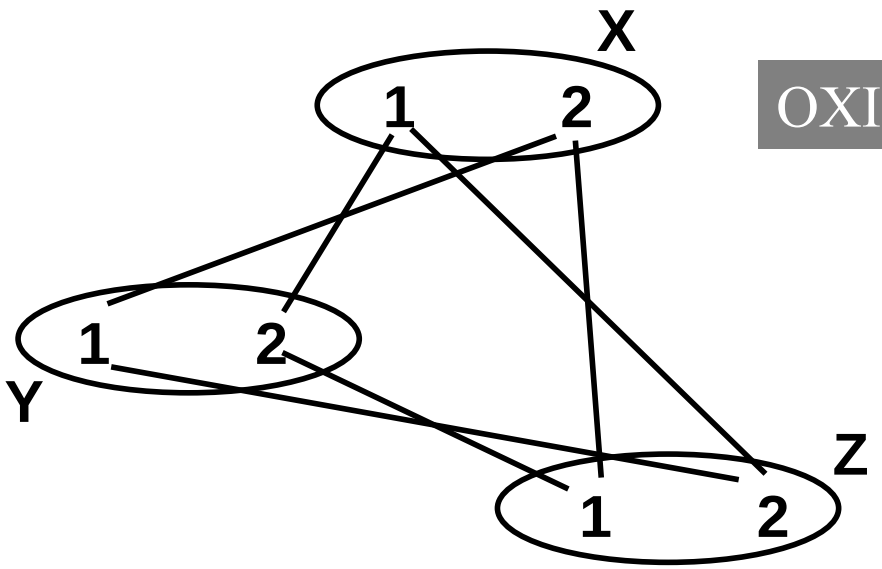
generate & test - 7 steps
backtracking - 5 steps
AC propagation - 2 steps

Είναι το AC αρκετό για πληρότητα;

- **Πρόβλημα:**

  X::{1,2}, Y::{1,2}, Z::{1,2}

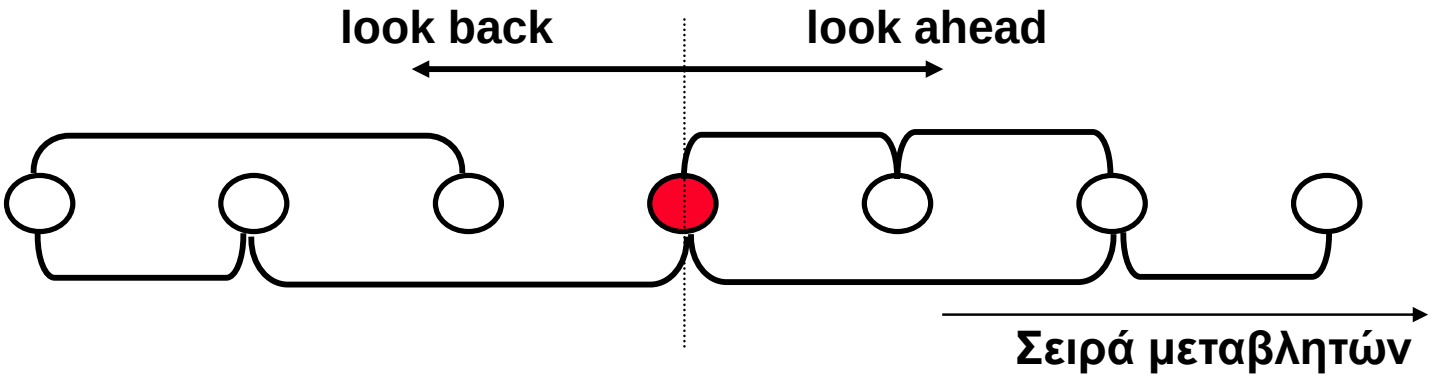
  $X \neq Y, X \neq Z, Y \neq Z$



ΟΧΙ!

Διάδοση Περιορισμών

- Απλή συστηματική αναζήτηση → μη αποδοτική.
- Απλή εφαρμογή τεχνικών συνέπειας → μη πλήρης.
- Συνδυασμός αναζήτησης *(οπισθοδρόμησης)* με τεχνικές συνέπειας.
- **Μέθοδοι:**
  - look back *(αντίδραση σε συγκρούσεις).*
  - look ahead *(αποτροπή συγκρούσεων).*



Arc Consistency

- Arc consistency εφαρμόζουμε:
  - Ως προ-επεξεργαστικό (**preprocessing**) βήμα πριν ξεκινήσει η αναζήτηση.
    - ✓μειώνεται το μέγεθος του δέντρου αναζήτησης.
  - Κατά τη διάρκεια της αναζήτησης μετά από κάθε ανάθεση τιμής σε μεταβλητή.
    - ✓constraint propagation → γρήγορη ανακάλυψη αδιεξόδων.
  - Ο αλγόριθμος αναζήτησης που εφαρμόζει arc consistency μετά από κάθε ανάθεση τιμής σε μεταβλητή ονομάζεται **MAC** (***maintaining arc consistency***).

MAC

procedure Maintaining Arc Consistency (*vars*,*doms*,*cons*)
    solution ← MAC (*vars*,Ø,*doms*,*cons*)

function MAC (*unlabelled,compound_label,doms,cons*)
returns a solution or NIL
    if *unlabelled* = Ø **then return** *compound_label*
    else pick a variable *x* from *unlabelled*
        repeat
            pick a value *v* from D_x; delete *v* from D_x
            doms' ← AC(*unlabelled-{x},doms,cons,compound_label* + {(*x,v*)})
            if no domain in *doms'* is empty **then**
                result ← MAC(*unlabelled* - {*x*}, *compound_label* + {(*x,v*)}, *doms'*,*cons*)
                if *result* ≠ NIL **then return** *result*
        end
        until D_x = Ø
        return NIL
    end

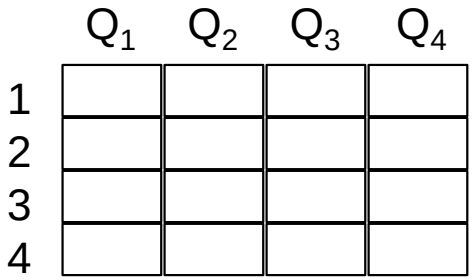
Αλγόριθμοι για Arc Consistency

- Arc consistency μπορεί να εφαρμοστεί με $O(ed^2)$ χρονική πολυπλοκότητα:
  - AC-4, AC-6, AC-7, **AC-2001.**
  - AC-3: μη-βέλτιστος χρονικά, αλλά απλός AC αλγόριθμος.
- Οι AC-3 και AC-2001 χρησιμοποιούν:
  - μια ουρά όπου μπαίνουν οι μεταβλητές που ελέγχονται για arc consistency,
  - μια ρουτίνα Revise που σβήνει τιμές που δεν υποστηρίζονται.
- Οι AC-4, AC-6, AC-7 χρησιμοποιούν πολύπλοκες δομές δεδομένων.

Ισχυρότερα Επίπεδα Συνέπειας

- Πέρα από το arc consistency υπάρχουν και άλλα επίπεδα συνέπειας:
  - **path consistency.**
  - **singleton arc consistency.**
  - **neighborhood inverse consistency.**
  - …
- Αυτά είναι πιο ισχυρά από arc consistency *(δηλ. σβήνουν πιο πολλές τιμές όταν εφαρμόζονται):*
  - Αλλά είναι και πιο ακριβά *(μεγαλύτερη χρονική πολυπλοκότητα).*

4-queen problem

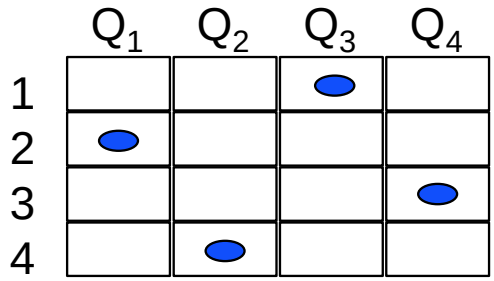


Place 4 queens so that no two queens are in attack.

Q_i: line number of queen in column i, for $1 \le i \le 4$

$Q_1, Q_2, Q_3, Q_4 \in \{1, 2, 3, 4\}$
$Q_1 \neq Q_2$, $Q_1 \neq Q_3$, $Q_1 \neq Q_4$,
$Q_2 \neq Q_3$, $Q_2 \neq Q_4$,
$Q_3 \neq Q_4$,
$Q_1 \neq Q_2-1$, $Q_1 \neq Q_2+1$, $Q_1 \neq Q_3-2$, $Q_1 \neq Q_3+2$,
$Q_1 \neq Q_4-3$, $Q_1 \neq Q_4+3$,
$Q_2 \neq Q_3-1$, $Q_2 \neq Q_3+1$, $Q_2 \neq Q_4-2$, $Q_2 \neq Q_4+2$,
$Q_3 \neq Q_4-1$, $Q_3 \neq Q_4+1$

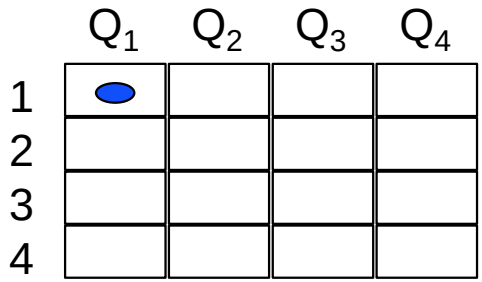
4-queen problem first solution

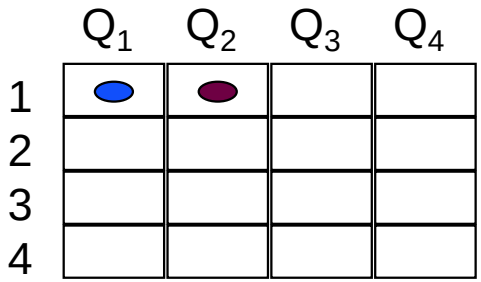


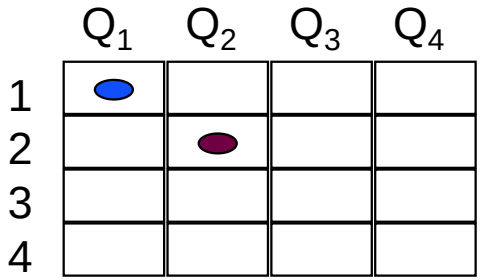
There is a total of 256 valuations

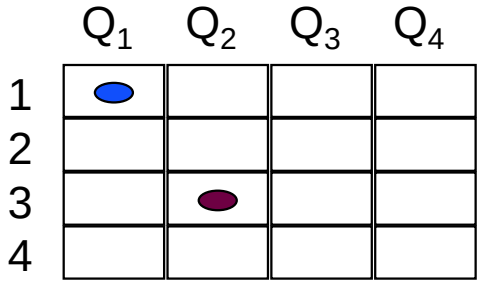
GT algorithm will generate

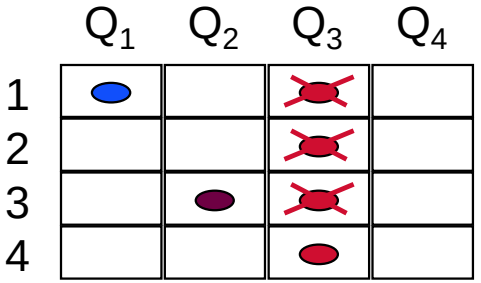
	64	valuations with $Q_1=1$;
+	48	valuations with $Q_1=2$, $1 \le Q_2 \le 3$;
+	3	valuations with $Q_1=2$, $Q_2=4$, $Q_3=1$;
=	115	valuations to find first solution



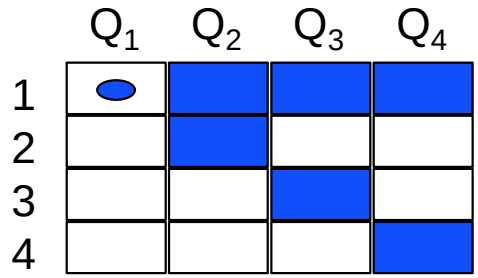


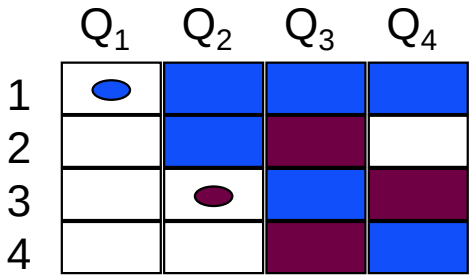


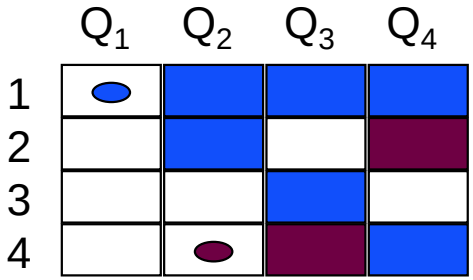


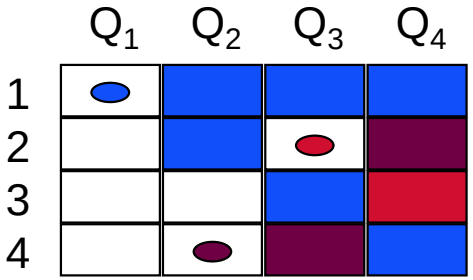


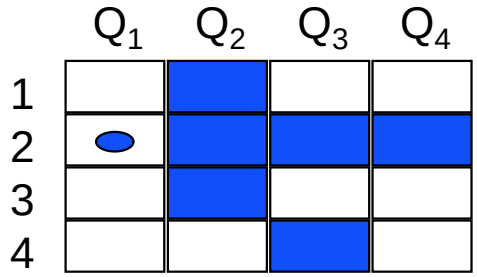
4-queen problem, BT algorithm

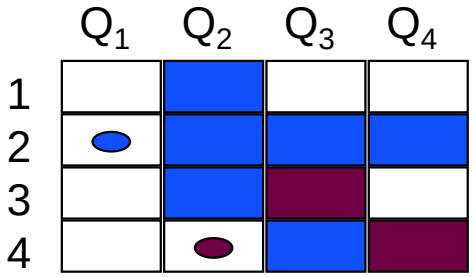


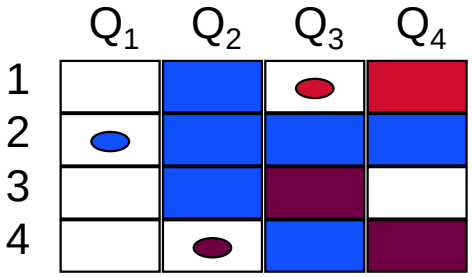






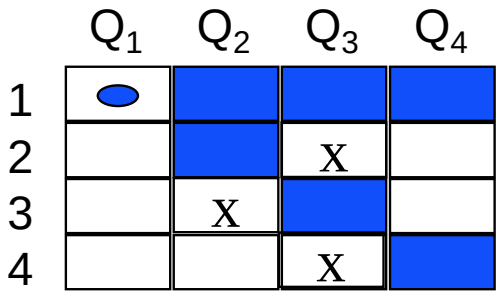




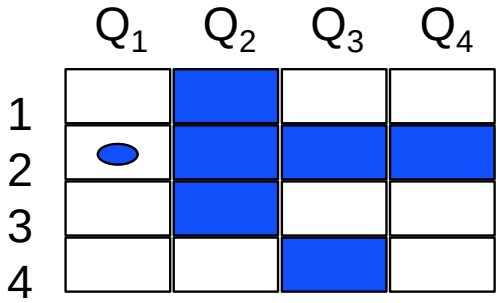


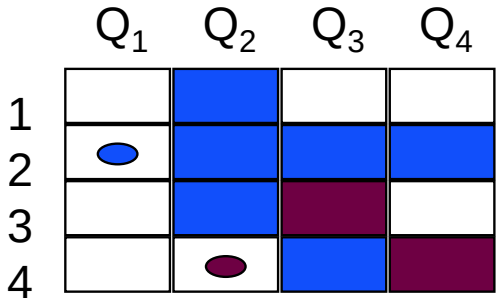
4-queen problem, FC algorithm

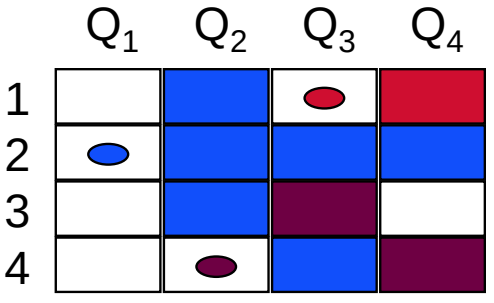
4-queen problem, MAC algorithm



Value 3 of Q_2 is unsupported in Q_3,
Value 4 of Q_3 is unsupported in Q_2,
Value 2 of Q_3 is unsupported in Q_4,







Υβριδικοί Αλγόριθμοι

- Μπορούμε να συνδυάσουμε τις λειτουργίες διαφορετικών αλγορίθμων οπισθοδρόμησης και να πάρουμε **υβριδικούς αλγόριθμους** *(hybrid algorithms).*
- Π.χ. Μπορούμε να συνδυάσουμε τη lookahead λειτουργία του forward checking και τη lookback λειτουργία του BJ:
  - FC-BJ
  - FC-CBJ
  - MAC-BJ
  - MAC-CBJ
  - …

Conflict-based Backjumping (CBJ)

- Το **Conflict-based Backjumping** ανήκει στην κατηγορία των look-back αλγορίθμων που κάνουν έξυπνη οπισθοδρόμηση από αδιέξοδα.
- Σε αντίθεση με το από BJ που κάνει backjumps μόνο από αδιέξοδα σε φύλλα, το CBJ μπορεί να κάνει backjumps και από αδιέξοδα σε εσωτερικούς κόμβους:
  - για κάθε μεταβλητή *x* έχουμε ένα σύνολο συγκρούσεων (conflict set),
  - όταν μια ανάθεση (*x,a*) αποτύχει λόγω παραβίασης περιορισμού με μια προηγούμενη μεταβλητή *y*, η *y* προστίθεται στο conflict set της *x,*
  - αν δεν υπάρχουν άλλες τιμές στο domain της τρέχουσας μεταβλητής *x*, ο CBJ οπισθοδρομεί στην πιο βαθιά μεταβλητή *w* στο conflict set της *x.*
    - ✓ **και το conflict set της *x* προστίθεται στο conflict set της *w,***
    - ✓ μετά μπορεί να γίνει περαιτέρω backjump από την *w.*

FC-CBJ

- **Forward Checking με Conflict-based Backjumping:**
  - Ο FC-CBJ συνδυάζει το look-ahead του FC με το έξυπνο backjumping του CBJ,
  - για κάθε μεταβλητή έχουμε ένα σύνολο συγκρούσεων (conflict set),
  - όταν το forward checking μιας ανάθεσης (*x,a*) έχει ως αποτέλεσμα τη διαγραφή τιμής από το domain μιας μεταβλητής *y*, το *x* προστίθεται στο conflict set του *y,*
  - αν μετά το forward checking μιας ανάθεσης (*x,a*) το domain μιας μεταβλητής *y* μείνει κενό, τότε οι μεταβλητές στο conflict set του *y* προστίθενται στο conflict set του *x.*
    - ✓ γιατί γίνεται αυτό;
  - αν δεν υπάρχουν άλλες τιμές στο domain της τρέχουσας μεταβλητής *x*, ο FC-CBJ οπισθοδρομεί στην πιο βαθιά μεταβλητή *w* στο conflict set του *x,*
    - ✓ και το conflict set της *x* προστίθεται στο conflict set της *w.*

Αξιολόγηση Αλγορίθμων Οπισθοδρόμησης (1/2)

- Πως μπορούμε να συγκρίνουμε αλγόριθμους οπισθοδρόμησης για CSPs;
  - Χρονική / Χωρική Πολυπλοκότητα.
    - ✓όχι και πολύ χρήσιμη. Όλοι είναι εκθετικοί!
  - Χρόνοι cpu.
  - Πλήθος κόμβων που επισκέπτονται στο δέντρο αναζήτησης.
  - Πλήθος ελέγχων περιορισμών *(consistency checks)* που εκτελούν.
  - Πόσες φορές οπισθοδρομούν.

Αξιολόγηση Αλγορίθμων Οπισθοδρόμησης (2/2)

- **Μερικά θεωρητικά αποτελέσματα:**
  - **Κόμβοι στο δέντρο αναζήτησης:**

    $FC\text{-}CBJ \leq FC\text{-}BJ \leq FC \leq BJ \leq BT$

    $CBJ \leq BJ$
  - **Πλήθος ελέγχων περιορισμών:**

    $CBJ \leq BJ \leq BT$

    $FC\text{-}CBJ \leq FC\text{-}BJ \leq FC$
  - **CPU χρόνοι;**

Πάντα χρειάζονται ΠΕΙΡΑΜΑΤΑ!!!

Ευριστικές Μέθοδοι για CSPs (1/2)

- Οι αλγόριθμοι αναζήτησης πρέπει να παίρνουν αποφάσεις:
  1) Ποια θα είναι η επόμενη μεταβλητή;
  2) Ποια τιμή να της δώσω;
  3) Ποιόν περιορισμό να εξετάσω;
- Οι αποφάσεις που παίρνει ο αλγόριθμος κάθε φορά επηρεάζουν δραστικά το μέγεθος του χώρου αναζήτησης *(και την απόδοση του αλγορίθμου).*
  - Ειδικά η (1).
- Ευριστικές μέθοδοι βοηθούν τους αλγόριθμους να παίρνουν σωστές αποφάσεις.

Ευριστικές Μέθοδοι για CSPs (2/2)

- **Ευριστικές μέθοδοι διάταξης μεταβλητών** (variable ordering heuristics).
  - **στατικά heuristics:**
    - MaxDegree, Bandwidth, …
  - **δυναμικά heuristics:**
    - MRV, Brelaz, dom/deg, …
- **Ευριστικές μέθοδοι διάταξης τιμών** (value ordering heuristics).
  - Geelen's promise, least-constraining…
- **Ευριστικές μέθοδοι διάταξης περιορισμών.**
  - ;

Heuristics Δυναμικής Διάταξης Μεταβλητών

- **Ελάχιστες Εναπομείναντες Τιμές (MRV) ή Μικρότερο Πεδίο Ορισμού (SD)**
  - Σε κάθε βήμα της αναζήτησης διάλεξε τη μεταβλητή με το μικρότερο πεδίο ορισμού.
  - Γιατί; (Fail-First Principle).
- **Αν υπάρχουν πολλές;**
  - Διάλεξε μια στην τύχη.
  - Διάλεξε τη μεταβλητή με το μεγαλύτερο βαθμό στον αρχικό γράφο περιορισμών.
  - Διάλεξε τη μεταβλητή με το μεγαλύτερο μελλοντικό βαθμό (δηλ. αυτή που εμπλέκεται στους περισσότερους περιορισμούς με μελλοντικές μεταβλητές).
    - **Brelaz** heuristic.
- Πολλές παραλλαγές έχουν προταθεί:
  - **dom/deg.**

Heuristics Διάταξης Τιμών

- **Min-Conflicts:**
  - Συσχέτισε με κάθε τιμή *a* το συνολικό πλήθος τιμών σε μελλοντικές μεταβλητές που είναι ασύμβατες με την *a.*
  - Διάλεξε την τιμή με το μικρότερο τέτοιο πλήθος.
  - Εναλλακτικά: Διαίρεσε το πλήθος των ασύμβατων τιμών κάθε μελλοντικής μεταβλητής x με το μέγεθος του πεδίου ορισμού της x.
- **Geelen's Promise:**
  - Για κάθε τιμή *a* μέτρησε το συνολικό πλήθος τιμών σε μελλοντικές μεταβλητές που είναι συμβατές με την *a.*
  - Υπολόγισε το γινόμενο αυτών. Αυτό ονομάζεται η **υπόσχεση (promise)** της τιμής *a.*
  - Διάλεξε την τιμή με τη μεγαλύτερη υπόσχεση.

Μέθοδοι Τοπικής Αναζήτησης

- **Τοπική αναζήτηση:**
  - *Hill climbing:*
    γειτονιά = αλλαγή τιμής σε μια μεταβλητή.
  - *min-conflicts:*
    γειτονιά = αλλαγή τιμής σε μια μεταβλητή από αυτές που συμμετέχουν σε παραβίαση περιορισμού.
  - Αποφυγή τοπικών βέλτιστων => heuristics
    - *random-walk.*

      μερικές φορές διάλεξε μια τοπική κίνηση στην τύχη.
    - *tabu search*
      απέφυγε πρόσφατες κινήσεις που έχεις δοκιμάσει και οδήγησαν σε τοπικό βέλτιστο
- **Δεν εγγυώνται πληρότητα.**

Min-Conflicts Αλγόριθμος

- ***Ξεκίνα με μια τυχαία ανάθεση τιμών σε μεταβλητές:***
  - ή μια που φαίνεται να είναι καλή με βάση κάποιο heuristic
  - κάποιοι περιορισμοί θα παραβιάζονται.
- ***Προσπάθησε να επιδιορθώσεις την αρχική ανάθεση:***
  - άλλαξε την ανάθεση τιμής σε μεταβλητή που κάνει να ικανοποιούνται όσο το δυνατόν περισσότεροι περιορισμοί,
  - τοπικά βέλτιστα.
- *Random restarts.*
- *Simulated annealing.*
- *Tabu search.*

Min-Conflicts (version 1)

```
procedure Min_Conflicts(P, maxTries, maxChanges)
    for i :=1 to maxTries do
        A := initial complete assignment of the variables in P
        for j:=1 to maxChanges do
            if A satisfies P then return (A)
            else
                x := randomly chosen variable whose assignment is
    in conflict
                (x,a) := alternative assignment of x which satisfies
                                                the maximum number of
    constraints under the current                       assignment A
                if by making assignment (x,a) you get a cost ≤
    current cost then
                        make the assignment
            endif
        endfor
    endfor
    return ("No solution found")
```

Min-Conflicts (version 2)

```
procedure Min_Conflicts(P, maxTries, maxChanges)
    for i :=1 to maxTries do
            A := initial complete assignment of the variables in P
            for j:=1 to maxChanges do
                    if A satisfies P then return (A)
                    else
                        (x,a) := the alternative assignment of a variable x
    which minimizes                                  the number of constraint
    violations under the current                                assignment A
                        if by making assignment (x,a) you get a cost ≤
    current cost then
                                    make the assignment
                       else break
                    endif
            endfor
    endfor
    return ("No solution found")
```

Min-Conflicts με Random Walk

- Πως μπορούμε να ξεφύγουμε από τοπικά βέλτιστα χωρίς επανεκκινήσεις;
    - (δηλ. με τοπικά βήματα);
- Προσθέτοντας “θόρυβο” στον αλγόριθμο!
- *Random walk (τυχαίος περίπατος):*
    - η γειτονική κατάσταση πάντα επιλέγεται τυχαία
    - μια τέτοια τεχνική είναι σχεδόν αδύνατο να βρει λύση
    - άρα χρειάζεται κάποια καθοδήγηση
- **Το random walk μπορεί να συνδυαστεί με το heuristic που καθοδηγεί την αναζήτηση χρησιμοποιώντας μια πιθανοτική κατανομή:**
    - ***p*** – πιθανότητα χρήσης του random walk.
    - ***(1-p)*** – πιθανότητα χρήσης του heuristic.

Min-Conflicts with Random Walk (version 1)

```
procedure Min_Conflicts(P, maxChanges,p)
    A := initial complete assignment of the variables in P
    for j:=1 to maxChanges do
            if A satisfies P then return (A)
            else
                if probability p verified
                    x := randomly chosen variable whose assignment is in
    conflict
                    (x,a) := randomly chosen alternative assignment of x
                else
                    (x,a) := the alternative assignment of a variable x which
    minimizes                    the number of constraint violations under
    the current                                 assignment A
                make the assignment (x,a)
            endif
    endfor
    return ("No solution found")
```

Min-Conflicts with Random Walk (version 2)

```
procedure Min_Conflicts(P, maxChanges,p)
    A := initial complete assignment of the variables in P
    for j:=1 to maxChanges do
            if A satisfies P then return (A)
            else
                x := randomly chosen variable whose assignment is in
    conflict
                if probability p verified
                     (x,a) := randomly chosen alternative assignment of x
                else
                    (x,a) := the alternative assignment of x which satisfies
                                        the maximum number of constraints under
    the current assignment A
                make the assignment (x,a)
            endif
    endfor
    return ("No solution found")
```

Useful Links

- **On-line guide to Constraint Programming:**
  - kti.ms.mff.cuni.cz/%7Ebartak/constraints/
- **Constraints Archive**:
  - cs.unh.edu/ccc/archive/
- **CSPLib : a problem library for constraints:**
  - 4c.ucc.ie/~tw/csplib/
- **Course on Theory and Practice of Constraint Satisfaction:**
  - cse.unl.edu/~choueiry/CSCE990-05/schedule.htm

Τέλος Ενότητας

Σημείωμα Αναφοράς

- Copyright Πανεπιστήμιο Δυτικής Μακεδονίας, Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, Στεργίου Κωνσταντίνος. «**Τεχνητή Νοημοσύνη**». Έκδοση: 1.0. Κοζάνη 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:

  https: //eclass.uowm.gr/courses/ICTE103/

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

- Τεχνητή Νοημοσύνη, Μια σύγχρονη προσέγγιση, S. Russel, P. Norvig, Εκδόσεις Κλειδάριθμος