Problemi di soddisfacimento di vincoli

Capitolo 5

– presentazione basata sui lucidi di S. Russell –

Argomenti

- ♢ Problemi di soddisfacimento di vincoli (CSP)
- ♢ Ricerca con Backtracking per CSP
- ♢ Struttura dei problemi
- ♢ Ricerca locale per CSP

Constraint satisfaction problems (CSPs)

Problemi di ricerca standard:

lo stato è una "black box" — una qualunque struttura dati che supporta il test di goal, la funzione di valutazione, e la funzione successore

CSP:

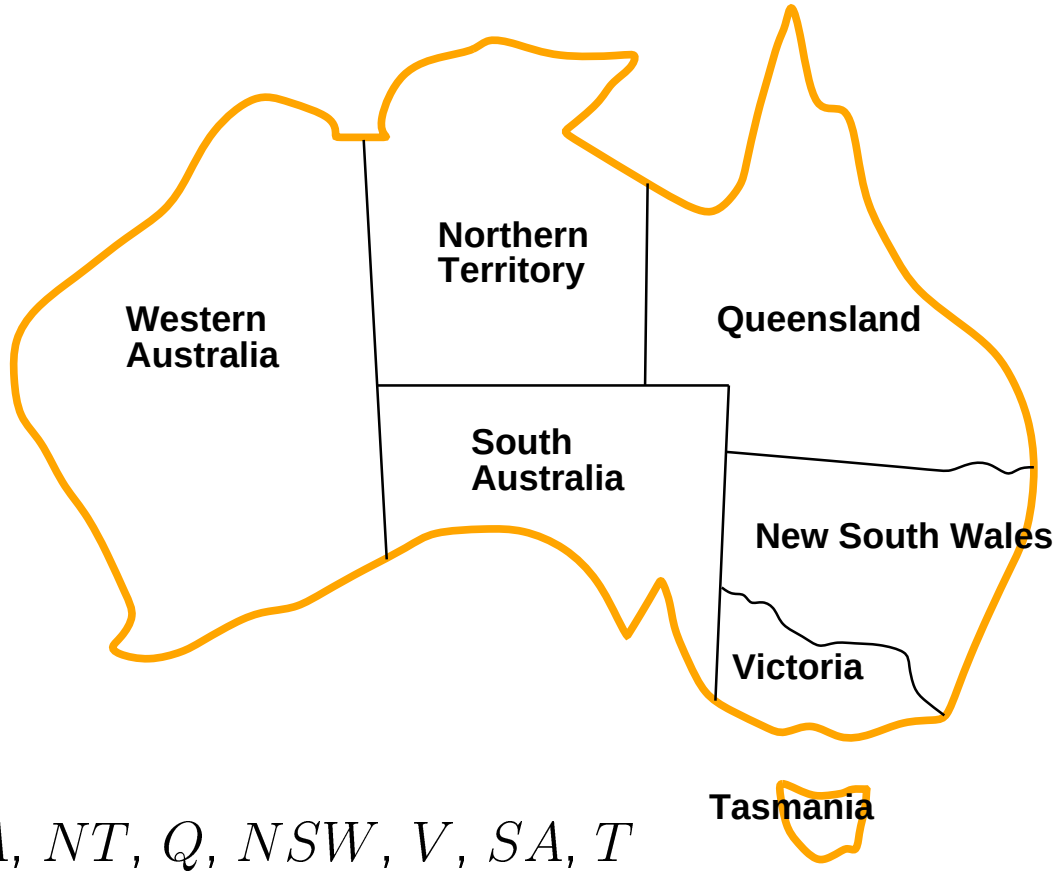
lo stato è definito da *variabili* X_i con *valori* dal *dominio* D_i

il test di goal è un insieme di *vincoli* che specificano combinazioni ammissibili di valori per sottoinsiemi di variabili

Costituiscono un esempio di un *linguaggio di rappresentazione formale*

Permettono di definire algoritmi *general-purpose* più potenti degli algoritmi di ricerca standard

Esempio: Colorazione di una mappa



Variabili WA, NT, Q, NSW, V, SA, T

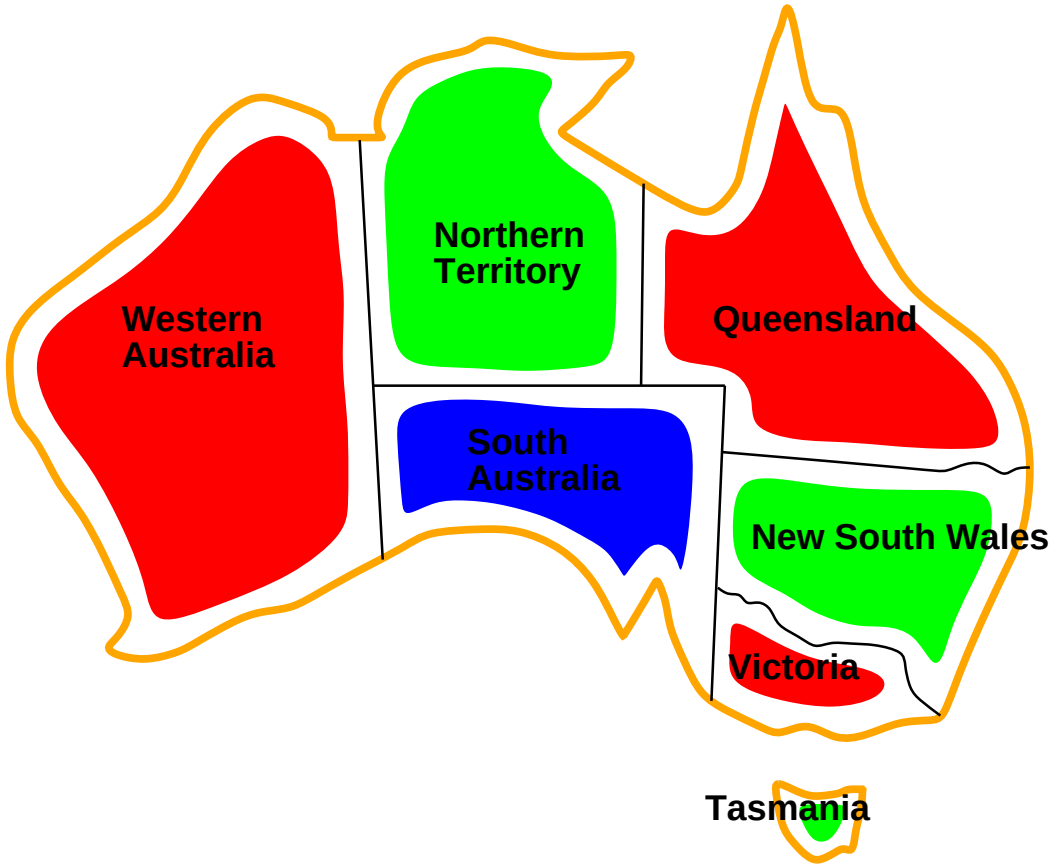
Domini $D_i = \{rosso, verde, blu\}$

Vincoli: regioni adiacenti devono avere colori differenti

ad esempio, $WA \neq NT$ (se il linguaggio lo permette), o

$(WA, NT) \in \{(rosso, verde), (rosso, blu), (verde, rosso), (verde, blu), \ldots\}$

Esempio: Colorazione di una mappa

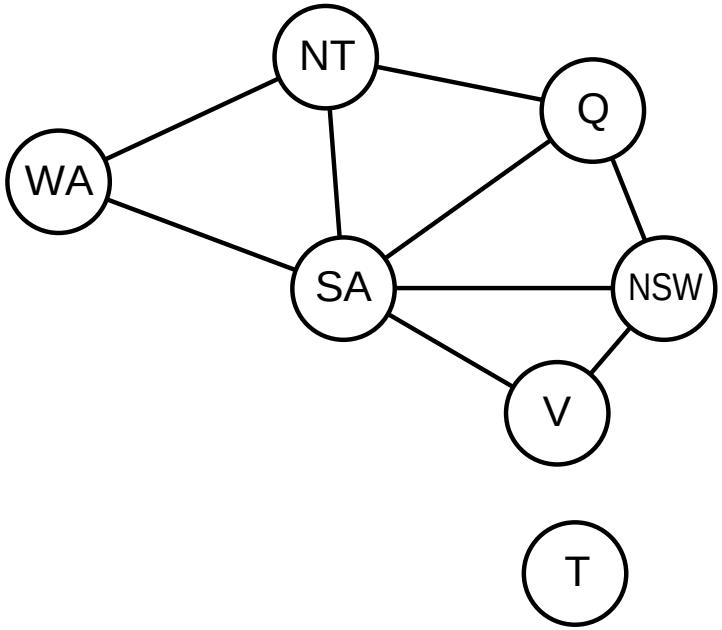


Le soluzioni sono assegnamenti che soddisfano tutti i vincoli, ad esempio,
$\{WA = rosso, NT = verde, Q = rosso, NSW = verde, V = rosso, SA = blu, T = verde\}$

Grafo dei vincoli

CSP Binario: ogni vincolo si riferisce al più a due variabili

Grafo dei vincoli: i nodi sono variabili, gli archi rappresentano i vincoli



Gli algoritmi CSP general-purpose usano la struttura del grafo per velocizzare la ricerca
Ad esempio, Tasmania rappresenta un sottoproblema indipendente!

Varietà di CSP

Variabili discrete

domini finiti; dimensione $d \Rightarrow O(d^n)$ assegnamenti completi

- ♢ p.e., CSP Booleani, include soddisfacibilità Booleana (NP-completo)

domini infiniti (integi, stringhe, etc.)

- ♢ p.e., job scheduling, le variabili sono i giorni di inizio/fine per ogni lavoro (job)
- ♢ necessitano di un linguaggio di vincoli, p.e., $StartJob_1 + 5 \leq StartJob_3$
- ♢ vincoli lineari risolvibili, vincoli nonlineari non decidibili

Variabili continue

- ♢ p.e., tempi di inizio/fine per le osservazioni del Telescopio Hubble
- ♢ vincoli lineari risolvibili in tempo polinomiale tramite metodi di programmazione lineare

Varietà di vincoli

Vincoli unari coinvolgono variabili singole,
p.e., $SA \neq verde$

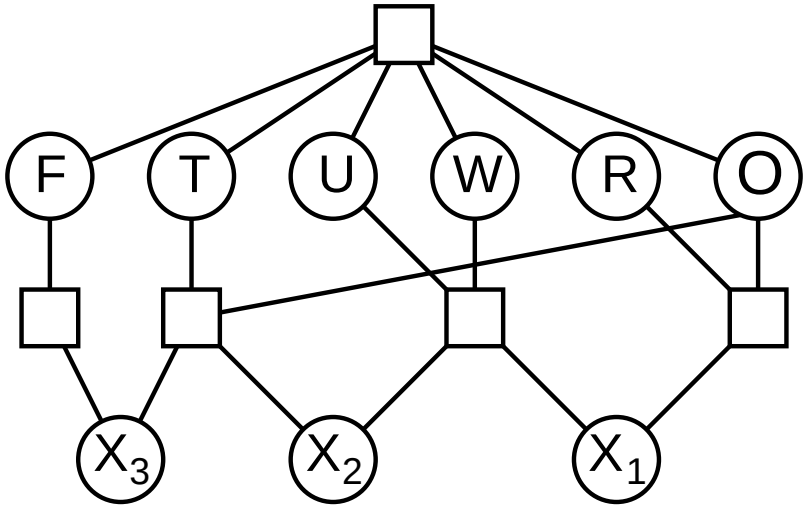
Vincoli binari coinvolgono coppie di variabili,
p.e., $SA \neq WA$

Vincoli di ordine superiore coinvolgono 3 o più variabili,
p.e., vincoli di cripto-aritmetica (vedi Sudoku)

Preferenze (vincoli soft), p.e., $rosso$ è meglio di $verde$
spesso rappresentabili tramite un assegnamento di costo ad ogni variabile
$\rightarrow$ problemi di ottimizzazione vincolata

Esempio: Cripto-aritmetica

$$\begin{array}{cccc} & \mathsf{T} & \mathsf{W} & \mathsf{O} \\ + & \mathsf{T} & \mathsf{W} & \mathsf{O} \\ \hline \mathsf{F} & \mathsf{O} & \mathsf{U} & \mathsf{R} \end{array}$$



Variabili: $F\ T\ U\ W\ R\ O\ X_1\ X_2\ X_3$

Domini: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Vincoli:

alldiff(F, T, U, W, R, O)

$O + O = R + 10 \cdot X_1$, etc.

Esempi di problemi reali di CSP

Problemi di assegnamento
    p.e., chi insegna quale lezione

Problemi di orario
    p.e., quando e dove è tenuta quale lezione ?

Configurazione di hardware

Fogli di calcolo

Logistica

Schedulazione di attività industriali

Notare che molti problemi reali coinvolgono variabili a valori reali

Formulazione di ricerca standard (incrementale)

Partiamo con l'approccio più semplice, che poi andremo a migliorare

Gli stati sono definiti dai valori assegnati fino ad ora

♢ Stato iniziale: l'assegnamento vuoto, $\{\}$

♢ Funzione successore: assegna un valore ad una variabile non assegnata
che è compatibile con l'assegnamento corrente
$\Rightarrow$ fallisce se non esiste un assegnamento legale (non risolvibile!)

♢ Test di goal: l'assegnamento corrente è completo

1) Valido per tutti i CSP!
2) Ogni soluzione appare a profondità n con n variabili
$\Rightarrow$ usare la ricerca depth-first
3) Il cammino è irrilevante, quindi si può usare una formulazione a stato completo
4) $b = (n - \ell)d$ a profondità ℓ, quindi $n!d^n$ foglie!!!!

Ricerca con Backtracking

Gli assegnamenti di variabili sono commutativi, cioè,

$[WA = rosso$ poi $NT = verde]$ uguale a $[NT = verde$ poi $WA = rosso]$

Bisogna considerare solo assegnamenti ad una singola variabile ad ogni nodo

$\Rightarrow\ b = d$ e ci sono d^n foglie

La ricerca depth-first per CSP con assegnamento di singole variabili è chiamata ricerca con backtracking

La ricerca con backtracking costituisce l'algoritmo non informato di base per i CSP

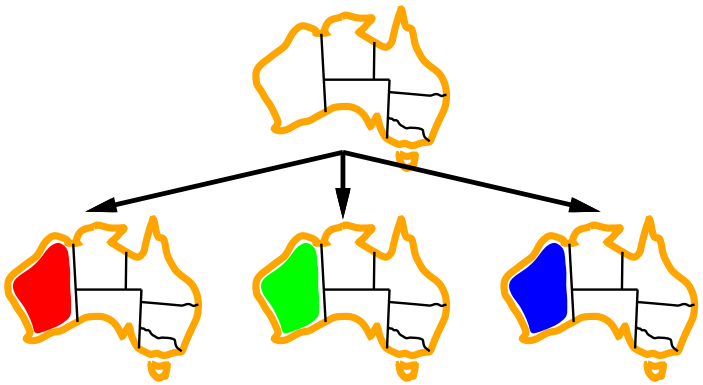
Può risolvere il problema delle n-regine con $n \approx 25$

Ricerca con Backtracking

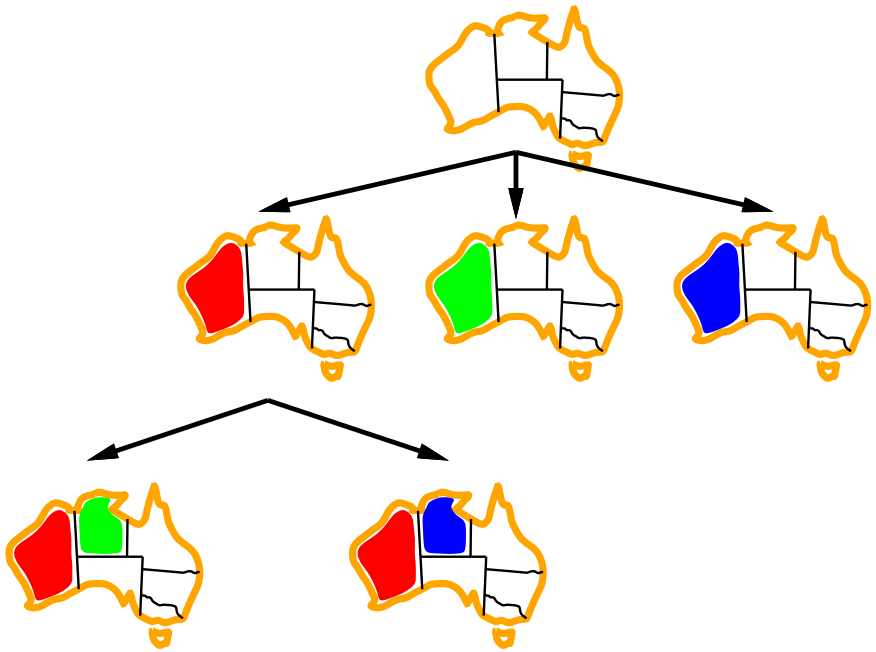
```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING([ ], csp)

function RECURSIVE-BACKTRACKING(assigned, csp) returns solution/failure
    if assigned is complete then return assigned
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assigned, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assigned, csp) do
        if value is consistent with assigned according to CONSTRAINTS[csp] then
            result ← RECURSIVE-BACKTRACKING([var = value|assigned], csp)
            if result ≠ failure then return result
    end
    return failure
```

Esempio di backtracking

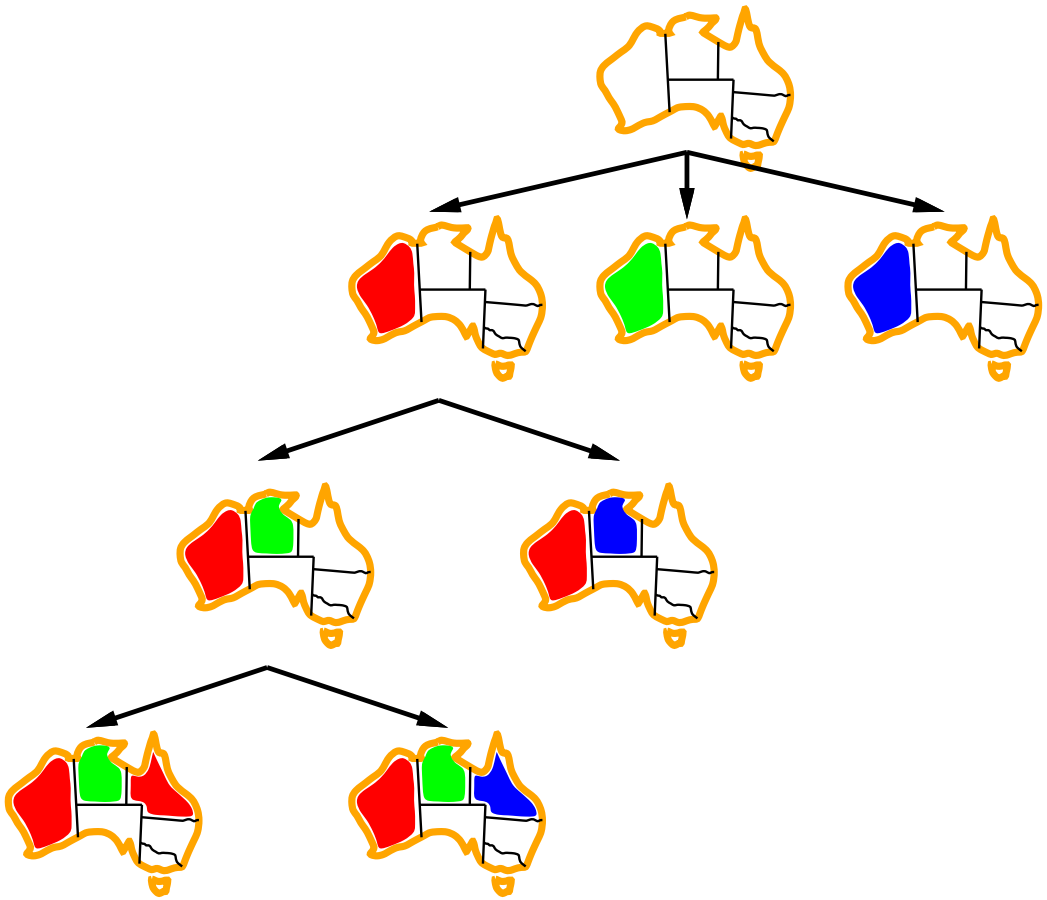
Esempio di backtracking



Esempio di backtracking



Esempio di backtracking



Miglioramento della efficienza del backtracking

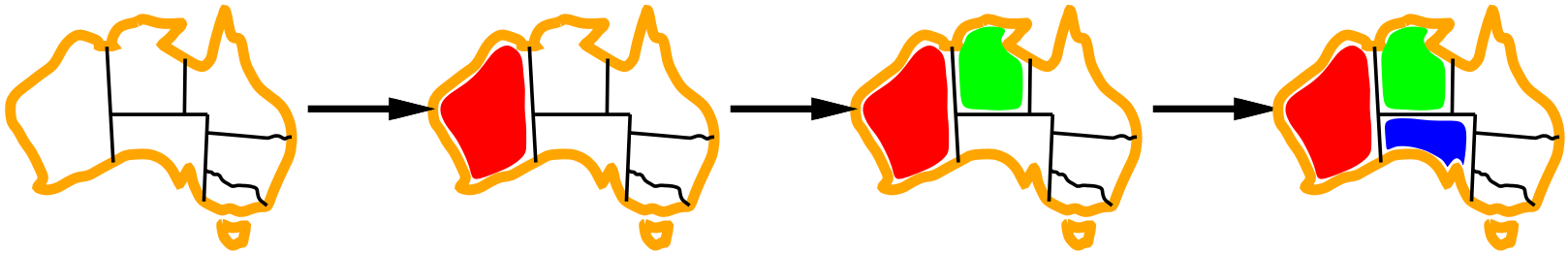
Metodi *general-purpose* possono dare enormi guadagni in velocità:

1. Quale variabile si deve assegnare al prossimo passo ?
2. In quale ordine bisogna testare i valori che può assumere ?
3. È possibile rilevare precocemente fallimenti inevitabili ?
4. È possibile sfruttare la struttura del problema ?

Variabile più vincolata

Variabile più vincolata:

scegliere la variabile con il numero minore di valori legali

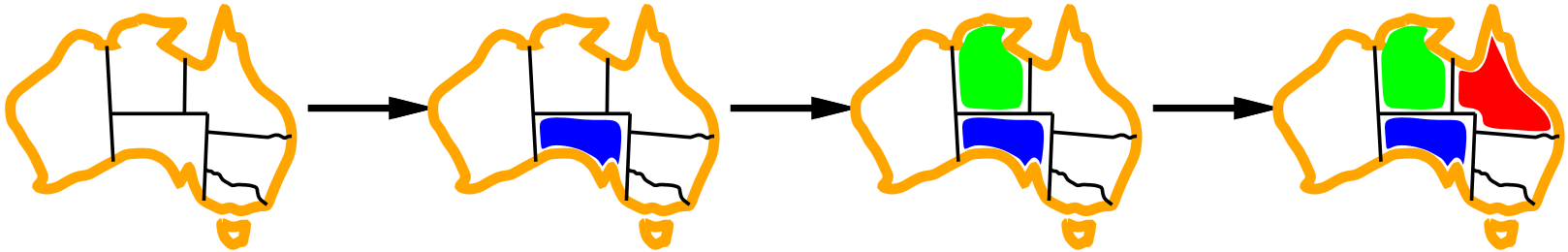


Variabile più vincolata

Tie-breaker tra le variabili più vincolate

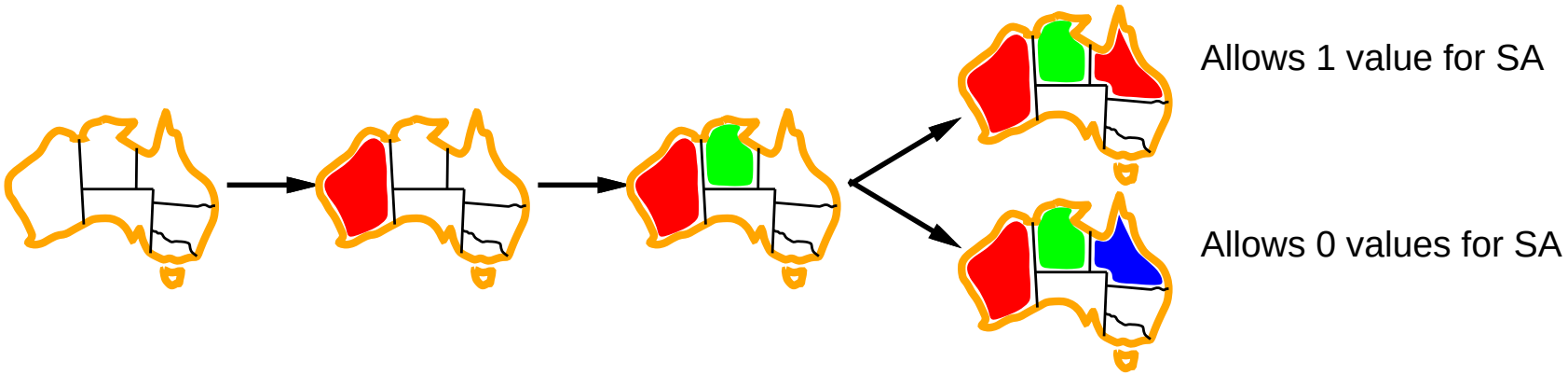
Variabile più vincolata:

scegliere quella che ha più vincoli con le variabili non assegnate



Valore meno vincolante

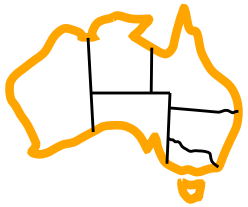
Data una variabile, scegliere il valore meno vincolante:
quello che esclude meno valori nelle rimenenti variabili (non assegnate)



Usando queste due euristiche è possibile risolvere il problema con 1000 regine

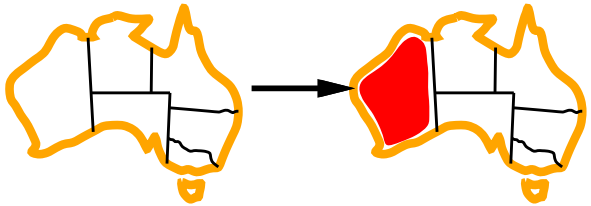
Forward checking

Idea: Tenere traccia dei rimanenti valori legali per variabili non assegnate
Terminare la ricerca quando c'è qualche variabile senza valori legali



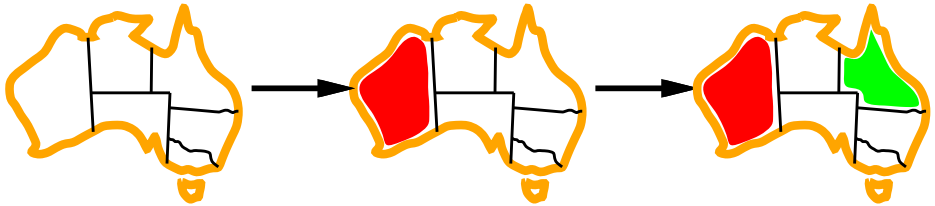
Forward checking

Idea: Tenere traccia dei rimanenti valori legali per variabili non assegnate
Terminare la ricerca quando c'è qualche variabile senza valori legali



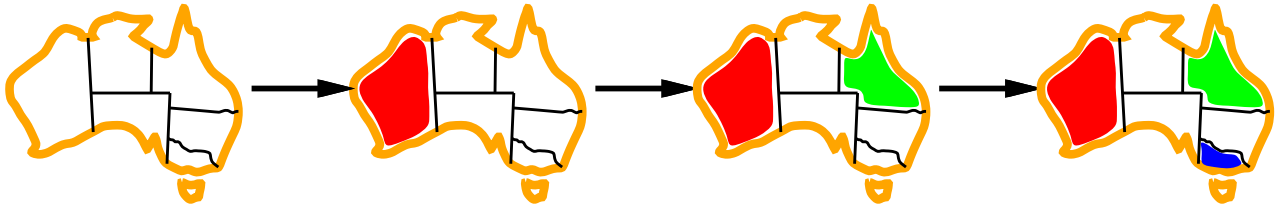
Forward checking

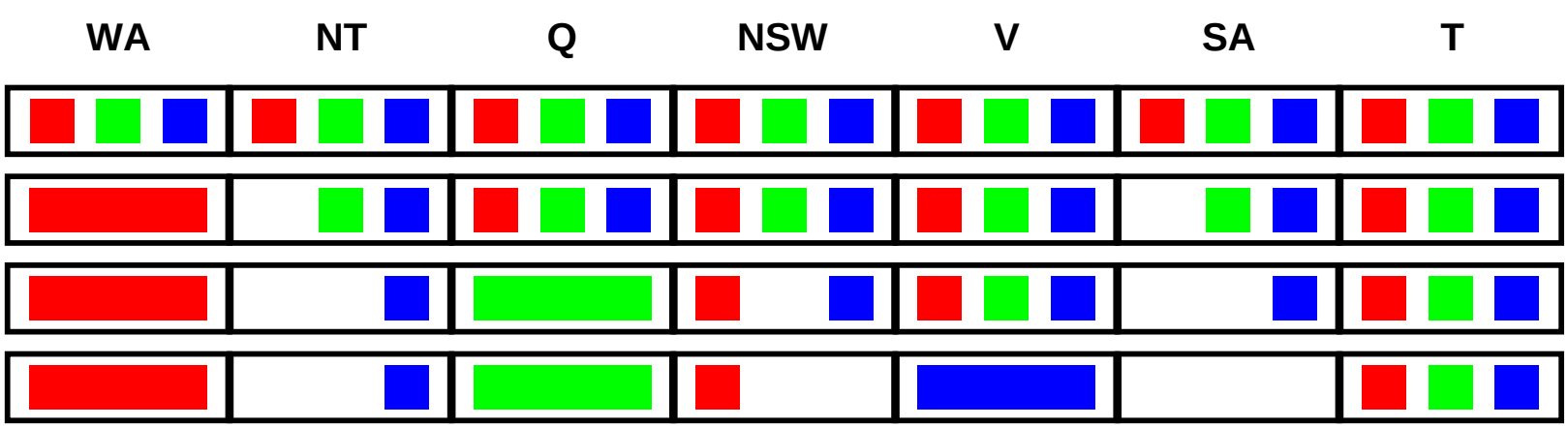
Idea: Tenere traccia dei rimanenti valori legali per variabili non assegnate
Terminare la ricerca quando c'è qualche variabile senza valori legali



Forward checking

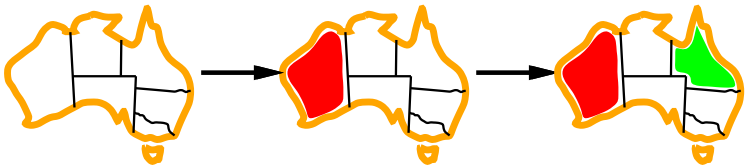
Idea: Tenere traccia dei rimanenti valori legali per variabili non assegnate
Terminare la ricerca quando c'è qualche variabile senza valori legali

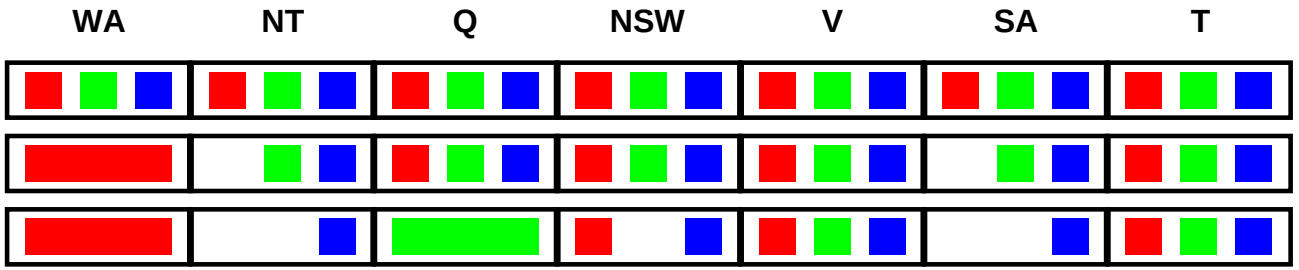




Propagazione dei vincoli

Forward checking propaga informazione da variabili assegnate a variabili non assegnate, ma non fornisce il rilevamento precoce per tutti i fallimenti:

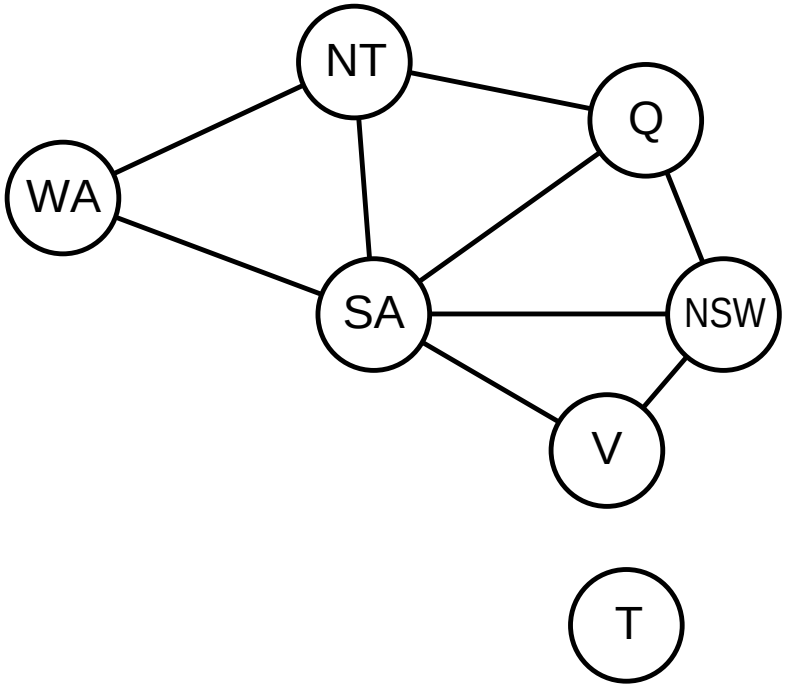




NT e SA non possono essere tutti e due blu!

Constraint propagation impone ripetutamente vincoli locali
vari algoritmi (consistenza d'arco, consistenza di nodo, ...) che NON vedremo

Struttura dei Problemi



Tasmania e continente sono sottoproblemi indipendenti

Identificabili come componenti connesse del grafo dei vincoli

Struttura dei Problemi

Supponiamo che ogni sottoproblema abbia c variabili su un totale di n

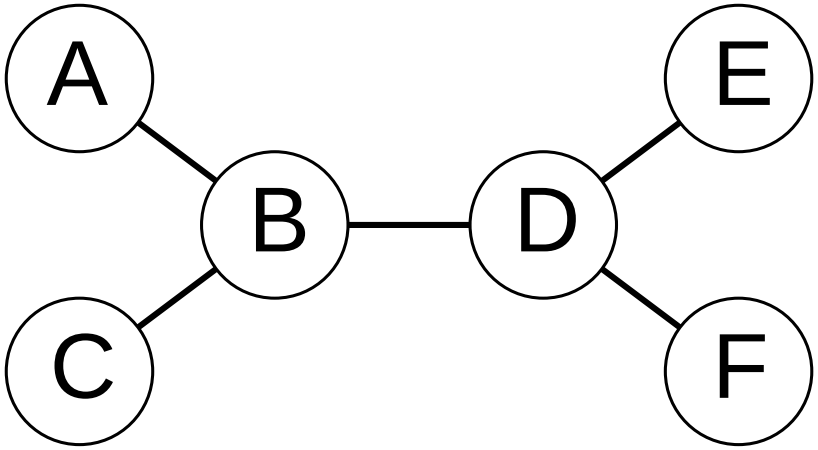
La soluzione di caso pessimo costa $n/c \cdot d^c$, *lineare* in n

p.e., $n = 80$, $d = 2$, $c = 20$

$2^{80} =$ 4 miliardi di anni a 10 milioni di nodi/sec

$4 \cdot 2^{20} =$ 0.4 secondi a 10 milioni di nodi/sec

CSP con struttura ad albero



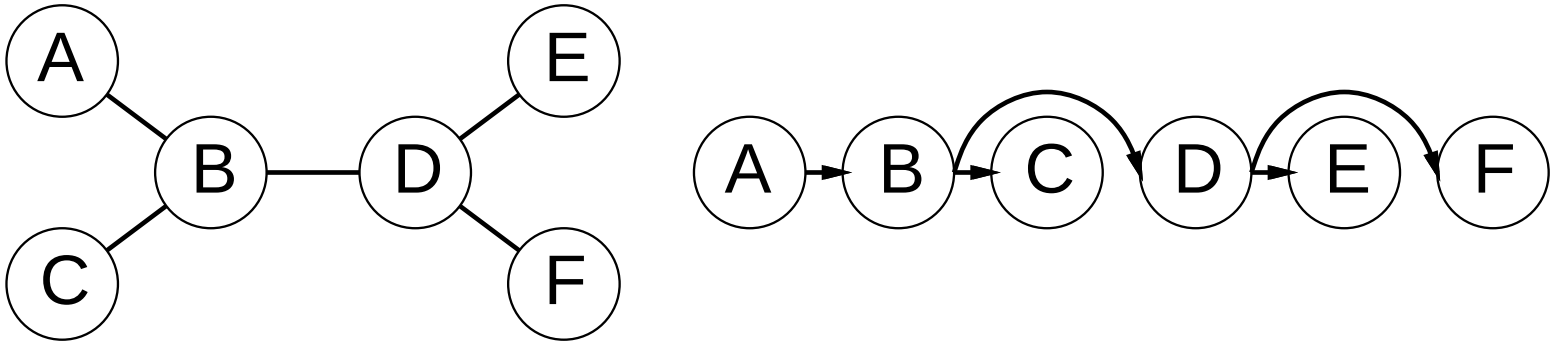
Teorema: se il grafo dei vincoli non ha cicli, il CSP può essere risolto in tempo $O(n\,d^2)$

Da confrontare con il caso generale, dove il caso pessimo è $O(d^n)$

Questa proprietà si applica anche al ragionamento logico e probabilistico:
un esempio importante della relazione fra restrizioni sintattiche
e la complessità del ragionamento

Algoritmi per CSP con struttura ad albero

1. Scegliere una variabile come radice, ordinare le variabili dalla radice alle foglie in modo tale che ogni nodo si preceduto dal suo genitore (ordine topologico)



2. Per j da n a 2, applicare $\textsc{RemoveInconsistent}(Parent(X_j), X_j)$

3. Per j da 1 a n, assegnare X_j consistentemente con $Parent(X_j)$

Algoritmi Iterativi per CSP

Sia Hill-climbing che Simulated Annealing tipicamente lavorano con stati "completi', cioè con tutte le variabili assegnate

Per applicarli ad un problema CSP:
- permettere stati con vincoli non soddisfatti
- gli operatori *riassegnano* valori ad una variabile

Selezione di variabile: selezionare a caso una delle variabili in conflitto

Selezionare il valore tramite l'euristica *minino conflitto (min-conflicts)*:
- scegliere il valore che viola il minor numero di vincoli
- cioè, utilizzare $h(n) =$ numero totale di vincoli violati

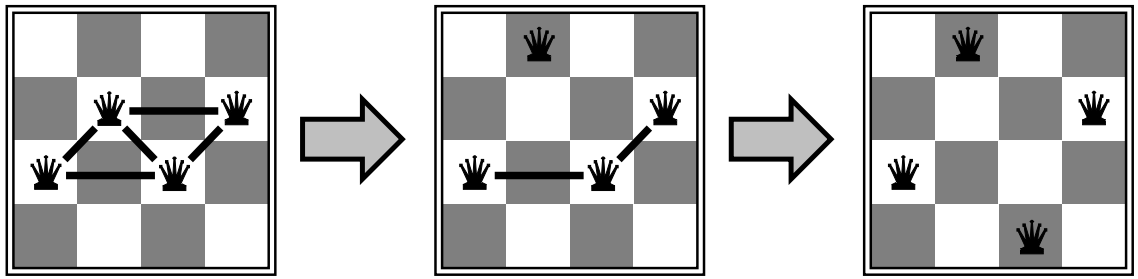
Esempio: 4-regine

Stati: 4 regine in 4 colonne ($4^4 = 256$ stati)

Operatori: muovere una regina sulla rispettiva colonna

Test di goal: assenza di attacchi

Funzione di valutazione: $h(n) =$ numero di attacchi



Prestazioni di min-conflicts

Dato uno stato iniziale random, può risolvere n-regine in tempo quasi costante per n arbitrario con alta probabilità (p.e., $n = 10.000.000$)

Lo stesso sembra essere vero per un qualunque CSP generato a caso tranne che in un intervallo ristretto del rapporto

$$R = \frac{\text{numero di vincoli}}{\text{numero di variabili}}$$