Altra proprietà di A*

A* non è solo ottimo:

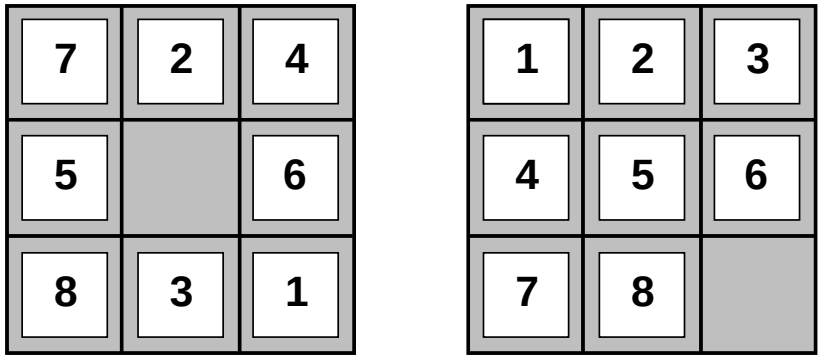
non esiste altro algoritmo ottimo che sicuramente espande meno nodi di A* (se non per risolvere "patte" sui nodi con f uguale al valore ottimo)

Questo è vero perché se così non fosse, allora tale algoritmo rischierebbe di non esplorare nodi ottimi, e quindi non sarebbe un algoritmo ottimo.

Euristiche ammissibili

Per esempio, consideriamo 8-puzzle:

$h_1(n) =$ numero di tasselli in posizione errata
$h_2(n) =$ distanza di Manhattan totale
(cioè numero di "quadrati" dalla posizione desiderata per ogni tassello)

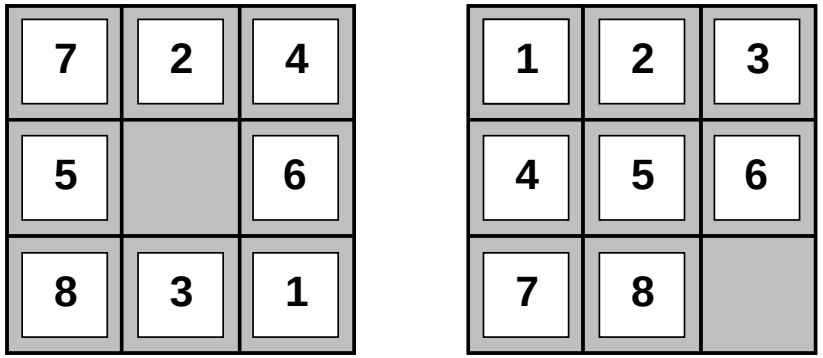


$h_1(S) =$??
$h_2(S) =$??

Euristiche ammissibili

Per esempio, consideriamo 8-puzzle:

$h_1(n)$ = numero di tasselli in posizione errata
$h_2(n)$ = distanza di Manhattan totale
(cioè numero di "quadrati" dalla posizione desiderata per ogni tassello)



$h_1(S) =$?? 6
$h_2(S) =$?? 4+0+3+3+1+0+2+1 = 14

Dominanza

Se $h_2(n) \geq h_1(n)$ per tutti gli n (entrambe ammissibili)
allora h_2 *domina* h_1 ed è migliore per la ricerca

Tipici costi di ricerca:

$d = 14$ IDS = 3,473,941 nodi
A$^*(h_1) = 539$ nodi
A$^*(h_2) = 113$ nodi
$d = 24$ IDS $\approx$ 54,000,000,000 nodi
A$^*(h_1) = 39{,}135$ nodi
A$^*(h_2) = 1{,}641$ nodi

Problemi rilassati

Euristiche ammissibili possono essere derivate dal costo
esatto di una soluzione di una versione *rilassata* del problema

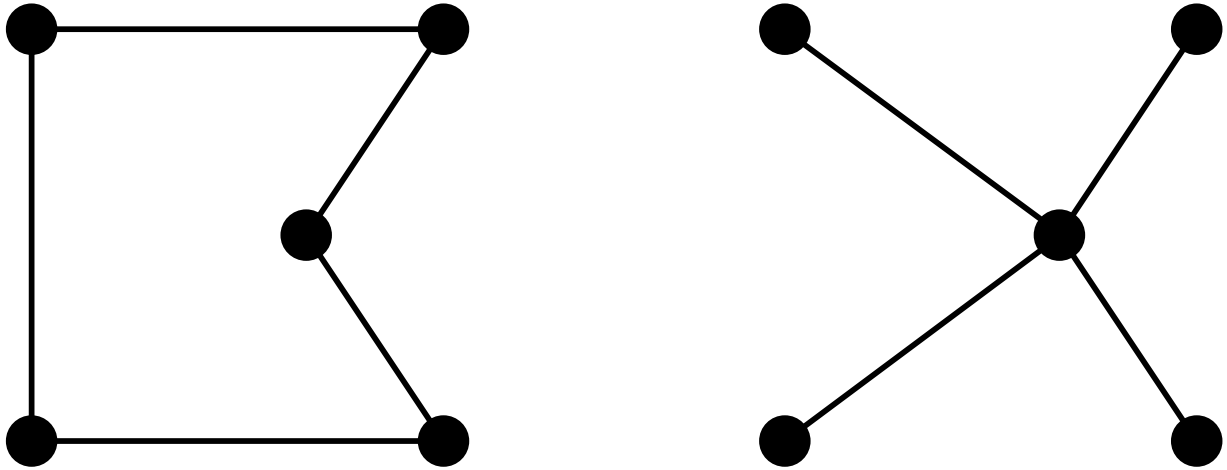
Se le regole di 8-puzzle sono rilassate così che un tassello può muoversi *ovunque*, allora $h_1(n)$ corrisponde alla soluzione sul cammino più breve

Se le regole di 8-puzzle sono rilassate così che un tassello può muoversi *ad ogni "quadrato" adiacente*, allora $h_2(n)$ corrisponde alla soluzione sul cammino più breve

Punto chiave: il costo della soluzione ottima di un problema rilassato
non è più grande del costo della soluzione ottima del problema originario

Problemi rilassati

Esempi noti: Problema del commesso viaggiatore (TSP)
Trovare il percorso più corto che visiti tutte le città esattamente una volta



Albero di copertura minimo (Minimum spanning tree) può essere calcolato in $O(n^2)$ e fornisce un limite inferiore al percorso (aperto) più breve

Alcune considerazioni finali su A*

A* ha occupazione esponenziale in spazio, e quindi rischia di non essere applicabile a molti problemi interessanti

Alcune considerazioni finali su A*

A* ha occupazione esponenziale in spazio, e quindi rischia di non essere applicabile a molti problemi interessanti

Come si può ridurre l'occupazione di memoria??

Alcune considerazioni finali su A*

A* ha occupazione esponenziale in spazio, e quindi rischia di non essere applicabile a molti problemi interessanti

Come si può ridurre l'occupazione di memoria??

La soluzione più semplice è quella di adattare l'idea dell'iterative deepening al contesto delle euristiche

Come??

Alcune considerazioni finali su A*

A* ha occupazione esponenziale in spazio, e quindi rischia di non essere applicabile a molti problemi interessanti

Come si può ridurre l'occupazione di memoria??

La soluzione più semplice è quella di adattare l'idea dell'iterative deepening al contesto delle euristiche

Come??

Algoritmo Iterative Deepening A* (IDA*):

come A*, però:

- non si inseriscono nella coda nodi con valore di f maggiori del valore di $cutoff$
- il valore di $cutoff$ alla iterazione successiva si pone uguale al minimo valore di f dei nodi non inseriti nella coda

Alcune considerazioni finali su A*

A* ha occupazione esponenziale in spazio, e quindi rischia di non essere applicabile a molti problemi interessanti

Come si può ridurre l'occupazione di memoria??

La soluzione più semplice è quella di adattare l'idea dell'iterative deepening al contesto delle euristiche

Come??

Esistono però soluzioni migliori, come RBFS, MA*, SMA*

Ricerca Best First Ricorsiva (RBFS)

La Ricerca Best First Ricorsiva è un algoritmo ricorsivo che imita una ricerca in profondità, utilizzando spazio lineare

In particolare:

invece di seguire indefinitamente il cammino corrente, tiene traccia dell'f-valore del miglior cammino alternativo che parte da uno degli avi

se il nodo corrente supera l'f-valore alternativo, la ricorsione torna indiatro al cammino alternativo

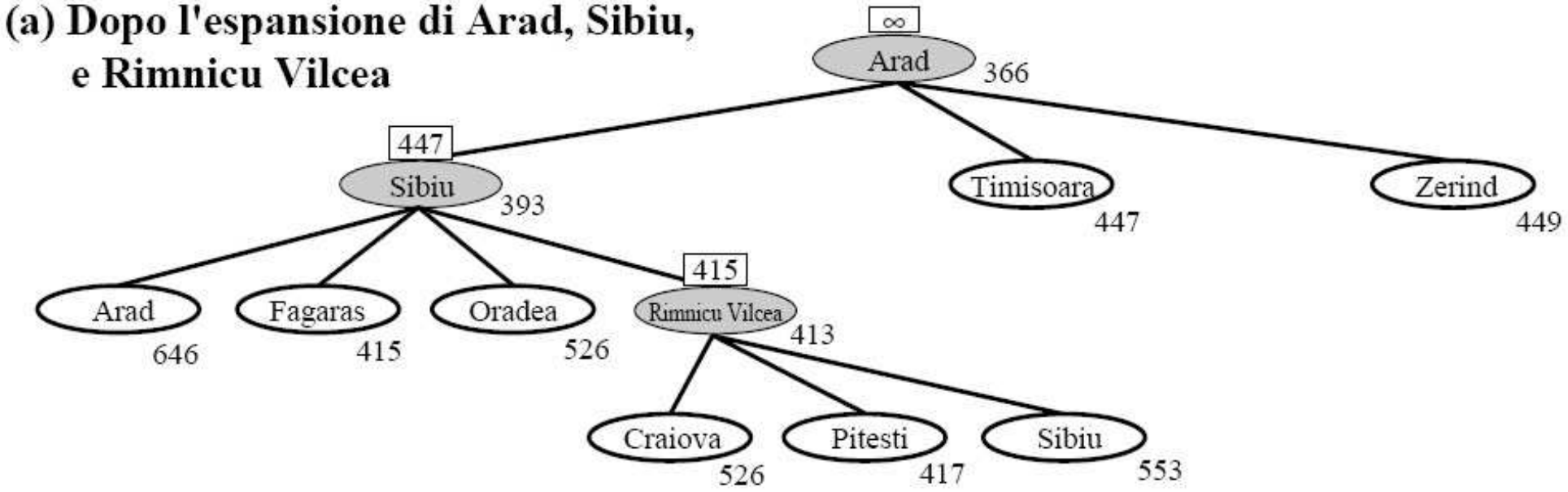
durante il ritorno, si sostituisce l'f-valore di ogni nodo lungo il cammino con il miglior f-valore dei suoi nodi figli

Ricerca Best First Ricorsiva

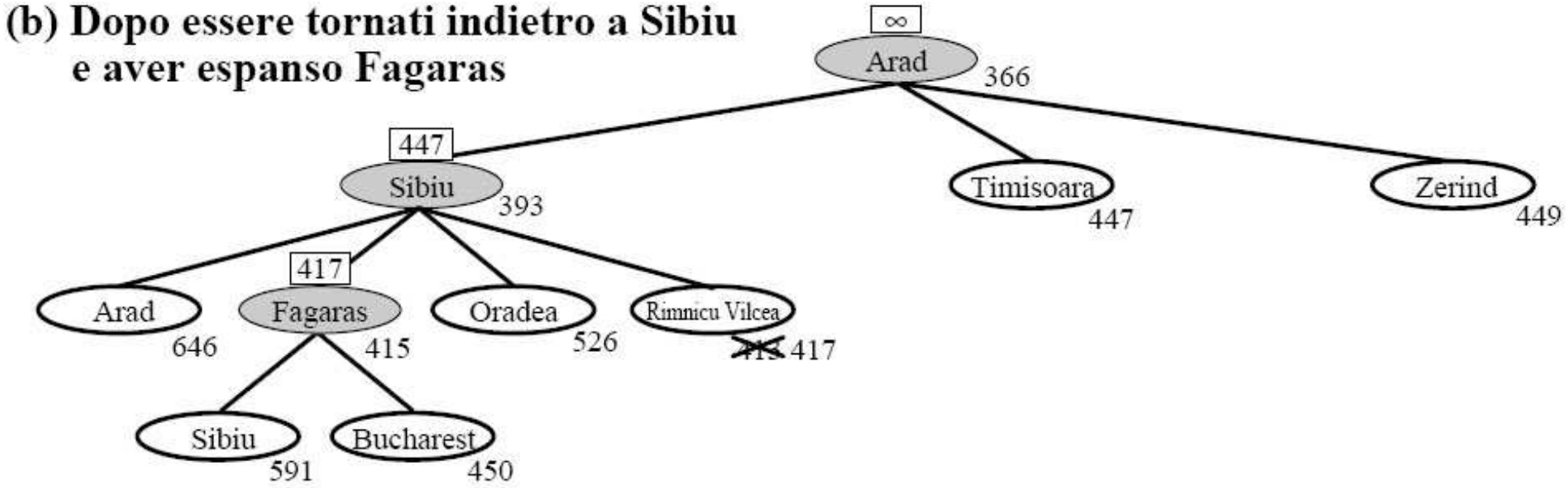
function RICERCA-BEST-FIRST-RICORSIVA(*problema*) **returns** una soluzione, o fallimento
    RBFS(*problema*, CREA-NODO(STATO-INIZIALE[*problema*]), ∞)

function RBFS(*problema*, *nodo*, *f-limite*) **returns** una soluzione, o fallimento e un nuovo limite all' f-costo
    if TEST-OBIETTIVO[*problema*](STATO[*nodo*]) **then return** SOLUZIONE(*nodo*)
    successori $\leftarrow$ ESPANDI(*nodo*, *problema*)
    if *successori* è vuoto **then return** *fallimento*, ∞
    for each s **in** *successori* **do**
        f[s] $\leftarrow \max(g(s) + h(s), f[nodo])$
    repeat
        best $\leftarrow$ il nodo con f-valore minimo in *successori*
        if $f[best] > f$-*limite* **then return** *fallimento*, $f[best]$
        alternativa $\leftarrow$ il secondo f-valore più basso in tutti i *successori*
        risultato, $f[best] \leftarrow$ RBFS(*problema*, *best*, $\min(f$-*limite*,*alternativa*))
        if *risultato* $\neq$ *fallimento* **then return** *risultato*

Esempio esecuzione



Esempio esecuzione



Esempio esecuzione