nome: Fábio Pisaruk
NUSP: 3467959

Análise dos métodos de busca

1-) Qual a importância de se evitar o processamento de nós repetidos para BrFS e DFS?

Caso permitíssemos o processamento de um nó que já foi expandido, este geraria, entre outros, o seu nó antecessor(o nó que o gerou) o que levaria o algoritmo a entrar em loop. Em ambas as estratégias de busca teríamos um laço infinito se nós repetidos fossem novamente inseridos na lista de nós a serem expandidos.

2-)Por que UCS garante encontrar a solução de menor custo para o ReguaPuzzle?

Em cada passo do algoritmo, o nó com menor custo é retirado da lista, expandido e, seus sucessores são inseridos na fila. Seja P={a,b,c,d} uma solução, ou seja, um caminho do estado *a* ao estado *d* com custo mínimo. Suponha que exista um caminho P1={a,b,e,d} tal que custo(P) < custo(P1). Sendo assim, (1) custo(b->e) + custo(e->d) < custo(b->c) + custo(c->d).
Os nós e e c estarão na fila por serem adjacentes a b.
Se (2)custo(b->c) < custo(b->e) então o nó c será retirado e expandido. No entanto, por (1) temos que o próximo nó a ser retirado não será o d e sim o e. Logo, por (1) (2) o caminho gerado será P1. Chegamos a uma contradição pois P é mínimo.

3-) Na IDS que cuidados devemos tomar ao se evitar nós repetidos?

Devemos evitar repetições de nós que já foram completamente expandidos. Entretanto, se um novo nó gerado já houver sido expandido anteriormente, precisamos verificar se o seu custo é inferior ao já expandido, se for removemos este nó da lista de expandidos.

4-)Por que IDS garante encontrar a solução ótima?

O IDS vai aumentando a profundidade aceita para a solução a cada novo passo. Desta maneira, começa por considerar as soluções por ordem de custo, das menos custosas às mais custosas. Observe que o IDS só retorna a solução de menor custo quando a função custo for igual à profundidade dos nós.

5-)Por que a heurística criada funciona?

Foram desenvolvidas duas heurísticas para o reguaPuzzle:

a-) MinDiff.

Funcao Diff: Dadas duas réguas retornamos a quantidade de posições que contem objetos diferentes. Exemplos:

AA-BB, BB-AA. Diff=2.
BBAA-, AABB-. Diff=4.
MinDiff:

Sejam R uma régua qualquer,
N o numero de blocos da régua.
M1,M2, ... Mn os estados metas existentes,
temos:
MinDiff = Min(Diff(R,Mi)) com i E [1,N]

Mostrar que esta heurística é admissível é bem simples. Ela apenas computa a quantidade mínima de blocos que precisariam ser movidos a fim de se alcançar uma meta.

b-) Distância:

Relaxamos o problema de forma que blocos podem mover mais de N passos por vez.
Ela leva em conta qual a distancia de um bloco ate sua posição de destino mais próxima.

Exemplo: n = A1A2-B1B2
h(n) = 2(A1 -> -) +
2(A2 -> B1) +
1(B1-> -) +
2(B2 -> A2)
Logo h(n) = 5.

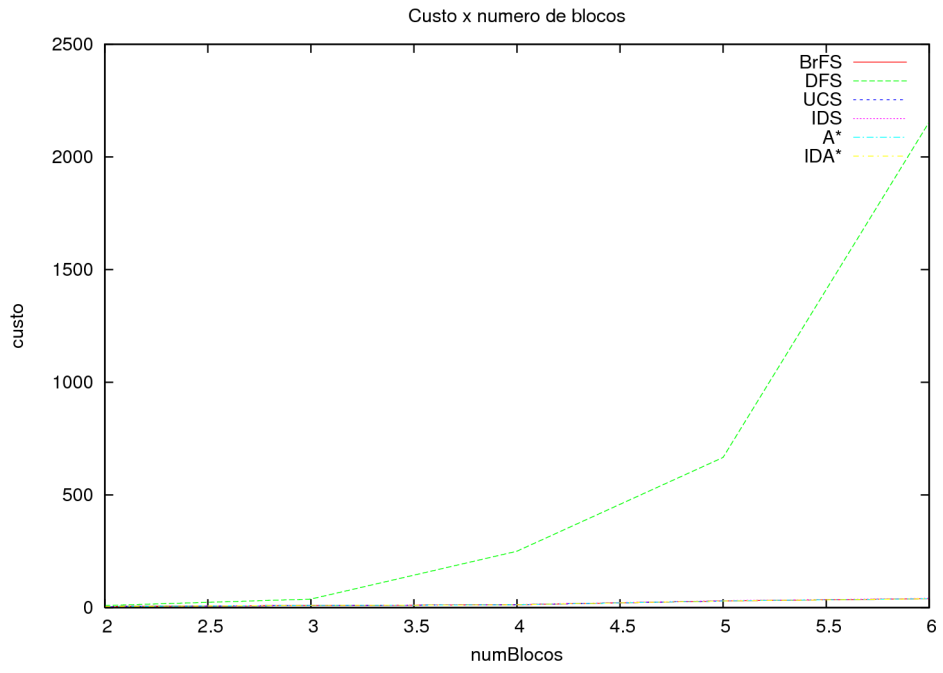
Por que a heurística é admissível?

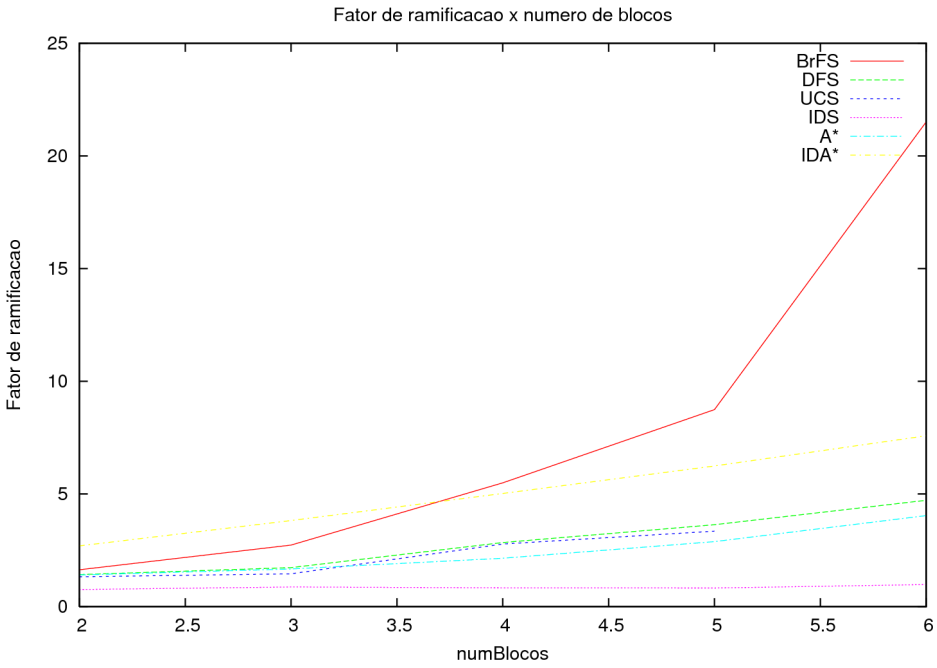
1-)h(estado meta) = 0. Se estamos num estado meta não existem blocos azuis antes da posição N.

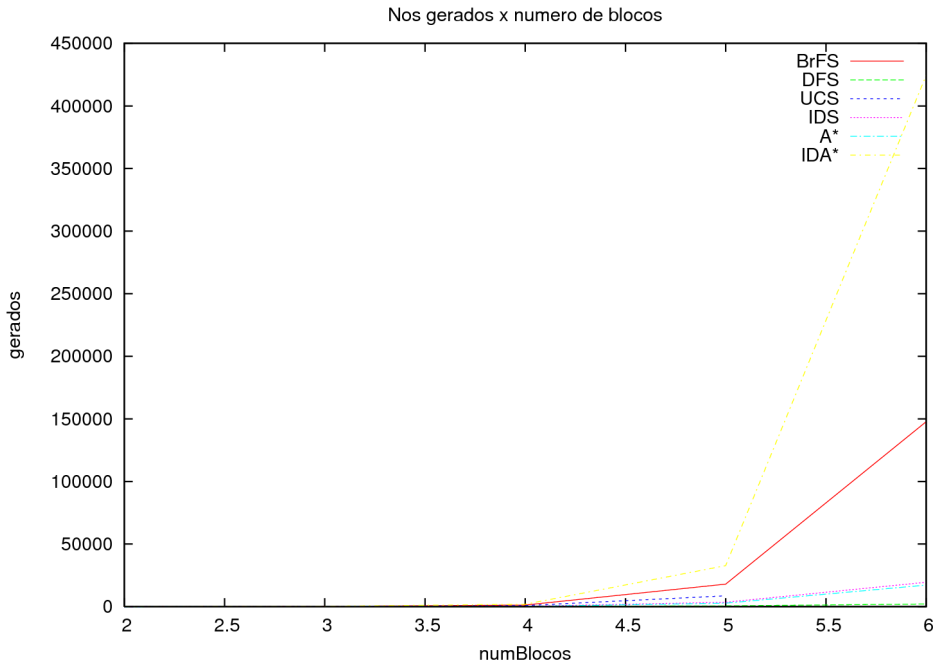
2-)Vamos mostrar que h(n) <= h*(n) para todo estado n.
Seja o estado n um estado não meta
n = {....A.....B....-....A....B.....}
Queremos mostrar que h(n)<=h*(n).
Como n não é meta, existem blocos azuis nas posições 0 a N e blocos brancos nas posições N a 2 * N. Um estado meta é caracterizado por possuir os blocos brancos nas posições 0 a N e os blocos azuis nas posições N a 2*N. A função h(n) calcula o deslocamento mínimo que cada bloco deve fazer para se chegar a um estado meta considerando o problema relaxado. Desta maneira fica claro que a heurística não pode calcular um valor superior ao custo mínimo do problema.

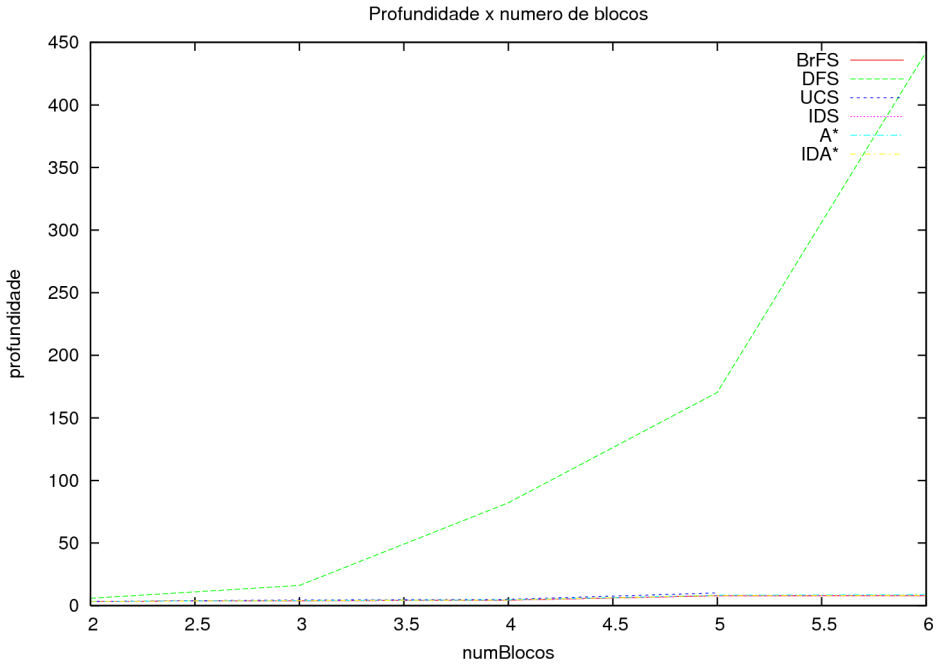
Gráficos

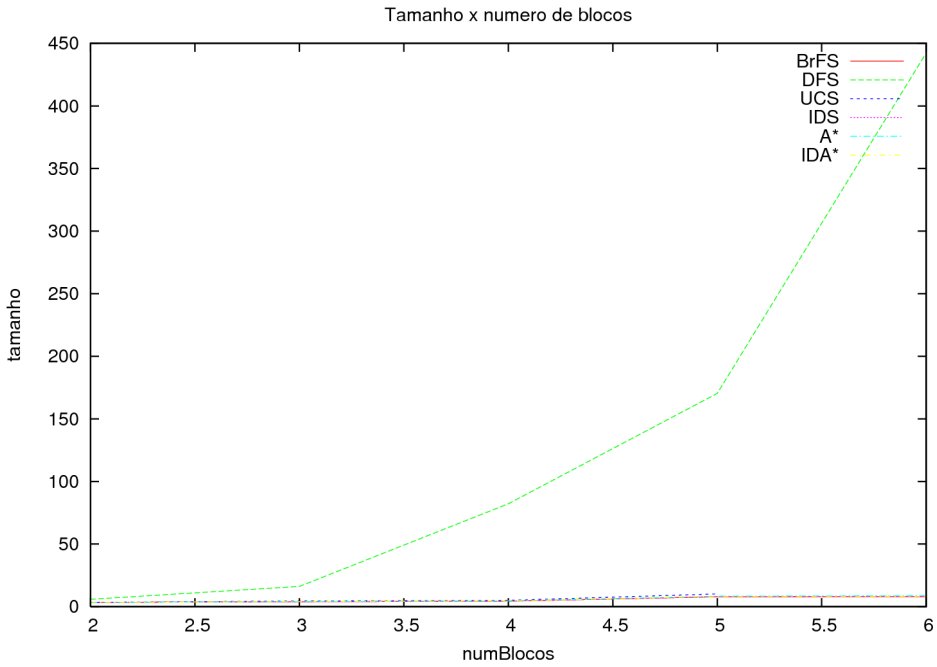
Para a construção dos gráficos foram geradas réguas aleatórias, medidos os tempos para cada item em questão e calculada a média. Geramos 128 réguas de 2 blocos, 64 de 3, 32 de 4, 16 de 5 e 8 de 6.

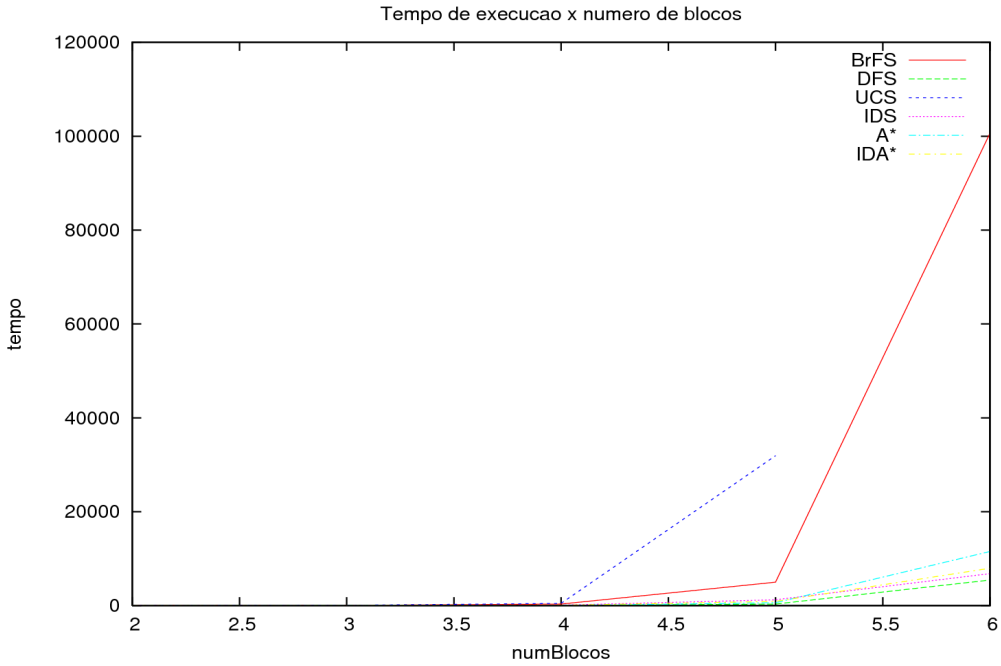


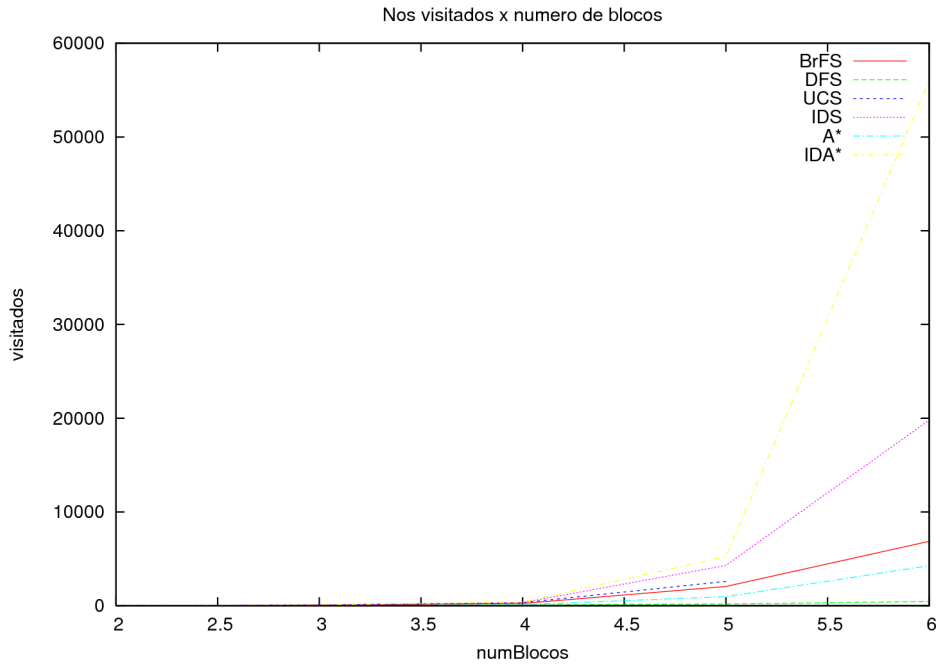

Fator de ramificacao x numero de blocos
Fator de ramificacao
numBlocos
BrFS
DFS
UCS
IDS
A*
IDA*
0
5
10
15
20
25
2
2.5
3
3.5
4
4.5
5
5.5
6

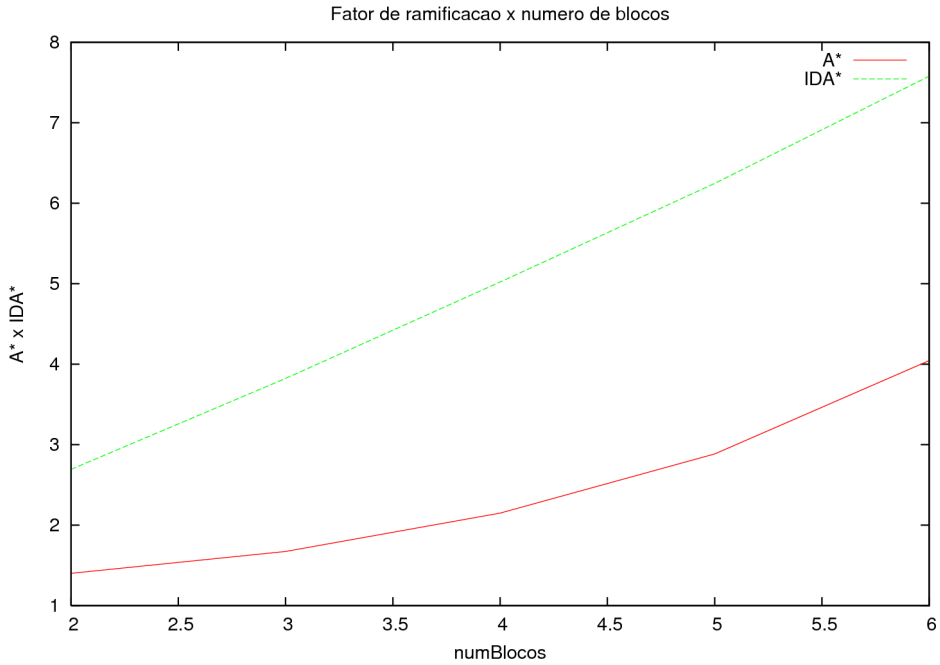

Nos gerados x numero de blocos
gerados
numBlocos
BrFS
DFS
UCS
IDS
A*
IDA*
0
50000
100000
150000
200000
250000
300000
350000
400000
450000
2
2.5
3
3.5
4
4.5
5
5.5
6

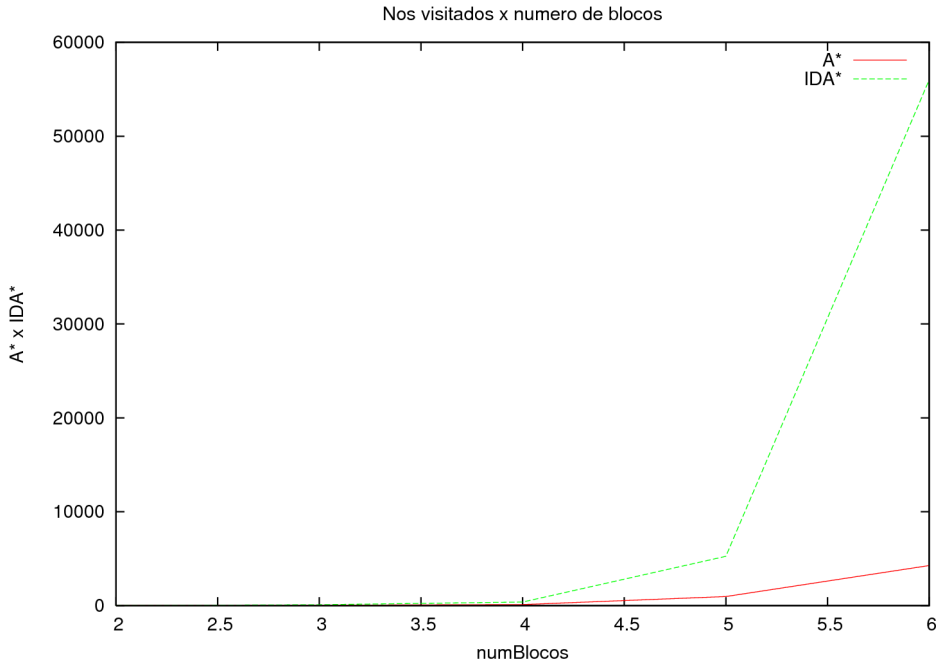

Profundidade x numero de blocos
profundidade
numBlocos
BrFS
DFS
UCS
IDS
A*
IDA*
0
50
100
150
200
250
300
350
400
450
2
2.5
3
3.5
4
4.5
5
5.5
6


Tamanho x numero de blocos
tamanho
numBlocos
BrFS
DFS
UCS
IDS
A*
IDA*
0
50
100
150
200
250
300
350
400
450
2
2.5
3
3.5
4
4.5
5
5.5
6


Tempo de execucao x numero de blocos
tempo
numBlocos
BrFS
DFS
UCS
IDS
A*
IDA*
0
20000
40000
60000
80000
100000
120000
2
2.5
3
3.5
4
4.5
5
5.5
6


Nos visitados x numero de blocos
visitados
numBlocos
BrFS
DFS
UCS
IDS
A*
IDA*
0
10000
20000
30000
40000
50000
60000
2
2.5
3
3.5
4
4.5
5
5.5
6


Fator de ramificacao x numero de blocos
A*
IDA*
A* x IDA*
numBlocos
1
2
3
4
5
6
7
8
2
2.5
3
3.5
4
4.5
5
5.5
6


Nos visitados x numero de blocos
A*
IDA*
A* x IDA*
numBlocos
0
10000
20000
30000
40000
50000
60000
2
2.5
3
3.5
4
4.5
5
5.5
6