Projeto de Algoritmos Baseados em Florestas de Posets para o Problema de Otimização U-curve

Gustavo Estrela

Novembro de 2017

Instituto de Matemática e Estatística
Centro de Toxinas, Resposta-imune e Sinalização Celular (CeTICS)
Laboratório Especial de Ciclo Celular, Instituto Butantan

O problema U-curve

Modelos computacionais

Modelos computacionais são criados para simular sistemas complexos.

Modelos computacionais

Modelos computacionais são criados para simular sistemas complexos.

$$\text{entrada} \longrightarrow \quad \text{sistema} \longrightarrow \qquad \text{saída}$$

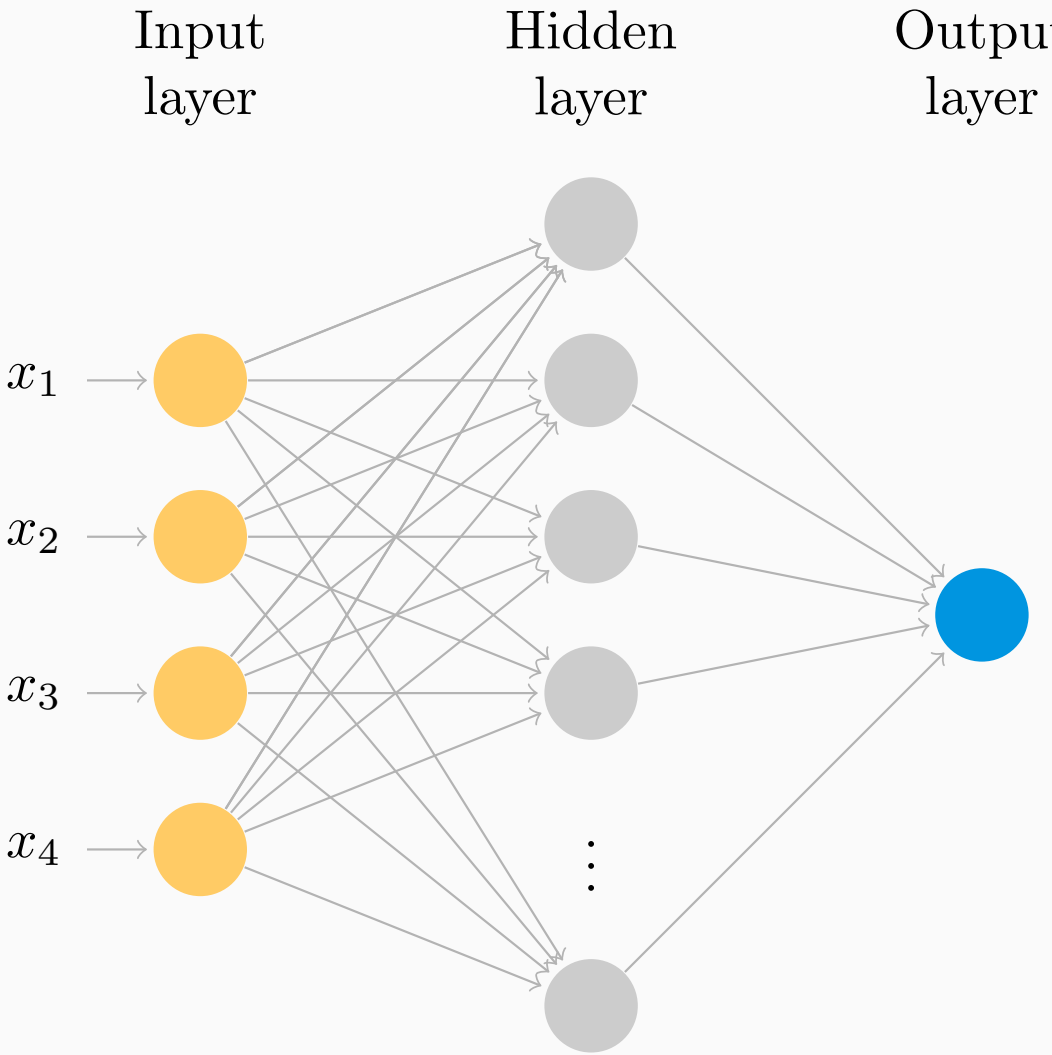
Modelos computacionais

Modelos computacionais são criados para simular sistemas complexos.

$$\text{entrada} \longrightarrow \text{sistema} \longrightarrow \text{saída}$$

$$\text{entrada} \longrightarrow \text{modelo} \longrightarrow \sim \text{saída}$$

Exemplo de modelo computacional



O problema de seleção de características

A seleção de características é uma etapa da seleção de modelos. Ela deve escolher quais são as melhores características para se considerar no modelo.

O problema de seleção de características

A seleção de características é uma etapa da seleção de modelos. Ela deve escolher quais são as melhores características para se considerar no modelo.

Definição

Dado um conjunto S de características e uma função de custo c, ache o subconjunto de $X \in \mathcal{P}(S)$ tal que $c(X)$ é mínimo.

O problema de seleção de características

Podemos representar um conjunto X de características por um vetor de bits que chamamos de vetor característico.

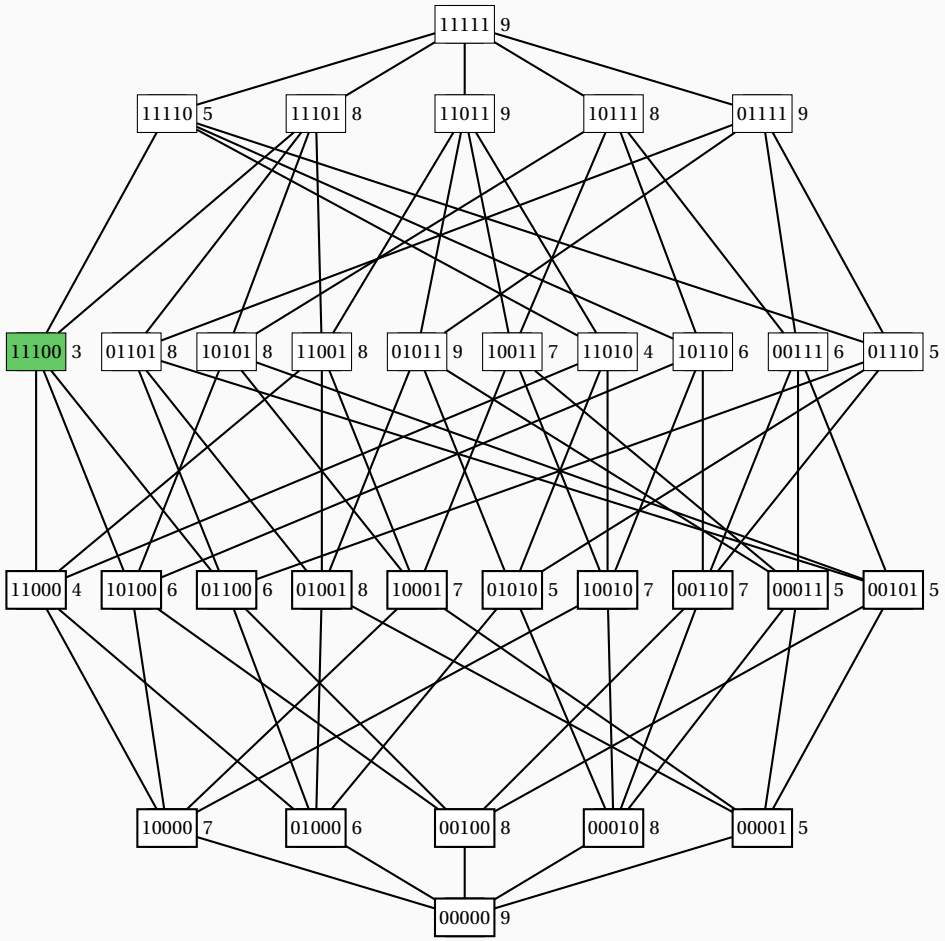
O problema de seleção de características

Podemos representar um conjunto X de características por um vetor de bits que chamamos de vetor característico.

Por exemplo, se $S = \{s_1, s_2, s_3\}$ e $X = \{s_1, s_3\}$ então o vetor característico de X é 101.

O espaço de busca

Os algoritmos estudados neste trabalho representam o espaço de busca com o reticulado Booleano $(\mathcal{P}(S), \subseteq)$.



O espaço de busca

Chamamos de cadeia uma sequência de conjuntos adjacentes $X_1, X_2, \ldots, X_n$ tal que $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n$.

O espaço de busca

Chamamos de cadeia uma sequência de conjuntos adjacentes $X_1, X_2, \ldots, X_n$ tal que $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n$.

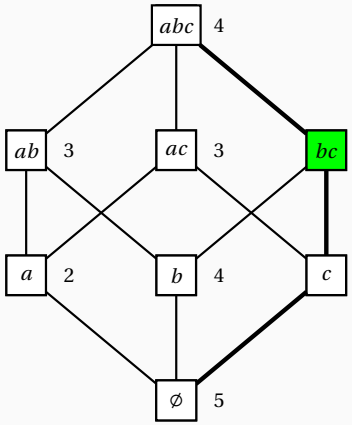
A função de custo

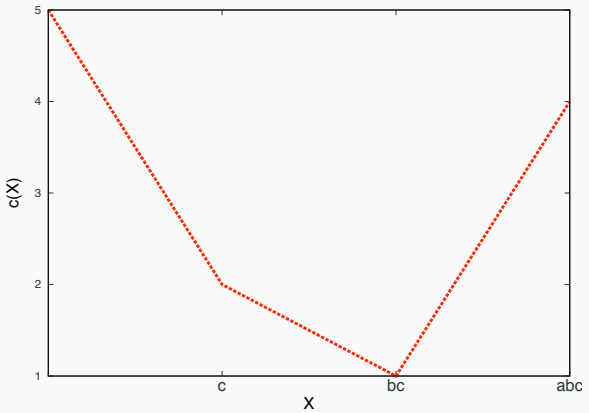
A função de custo c deve refletir a qualidade de um conjunto de características X a ser usado no modelo.

A função de custo

A função de custo c deve refletir a qualidade de um conjunto de características X a ser usado no modelo.

Nestas funções, um fenômeno conhecido em aprendizado de máquina aparece. A função descreve curvas em U nas cadeias do reticulado.





Funções decomponíveis em curvas U

Definição

Uma função de custo c é dita **decomponível em curvas U** *se para toda cadeia maximal $X_1, ..., X_l$, $c(X_j) \leq max\{c(X_i), c(X_k)\}$ sempre que $X_i \subseteq X_j \subseteq X_k$, $i, j, k \in \{1, ..., l\}$.*

O problema U-curve

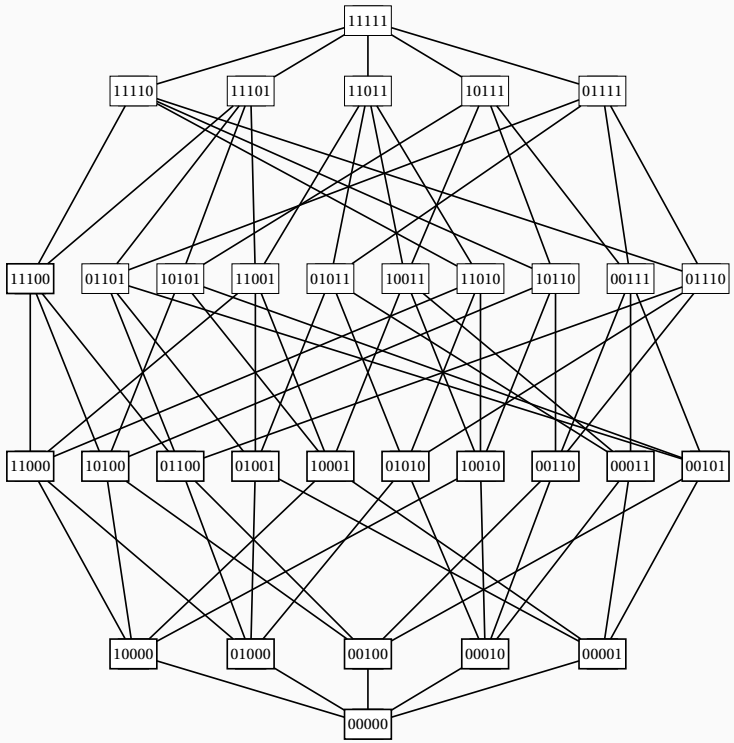
Definição (Problema U-Curve)

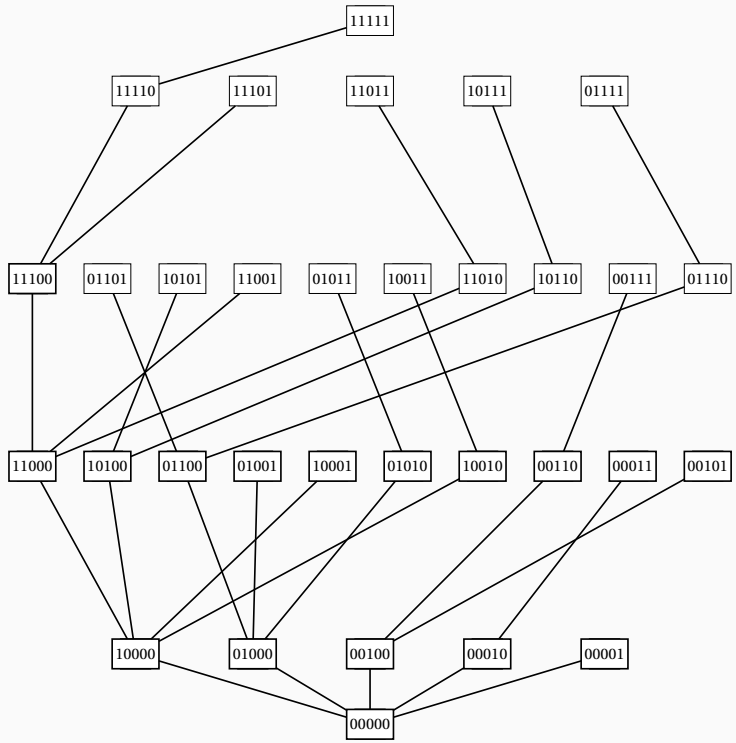
Dados um conjunto finito e não-vazio S e uma função de custo c decomponível em curvas U, encontrar um subconjunto $X \in \mathcal{P}(S)$ tal que $c(X)$ é mínimo.

Algoritmos baseados em florestas

O algoritmo `U-Curve-Branch-and-Bound`

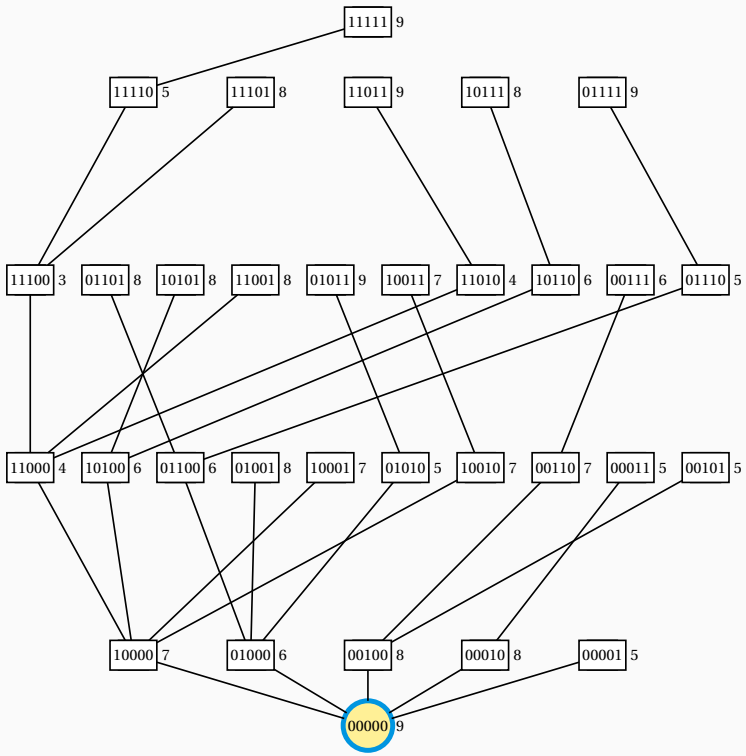
O algoritmo `U-Curve-Branch-and-Bound` (UBB) organiza o espaço de busca em uma árvore.





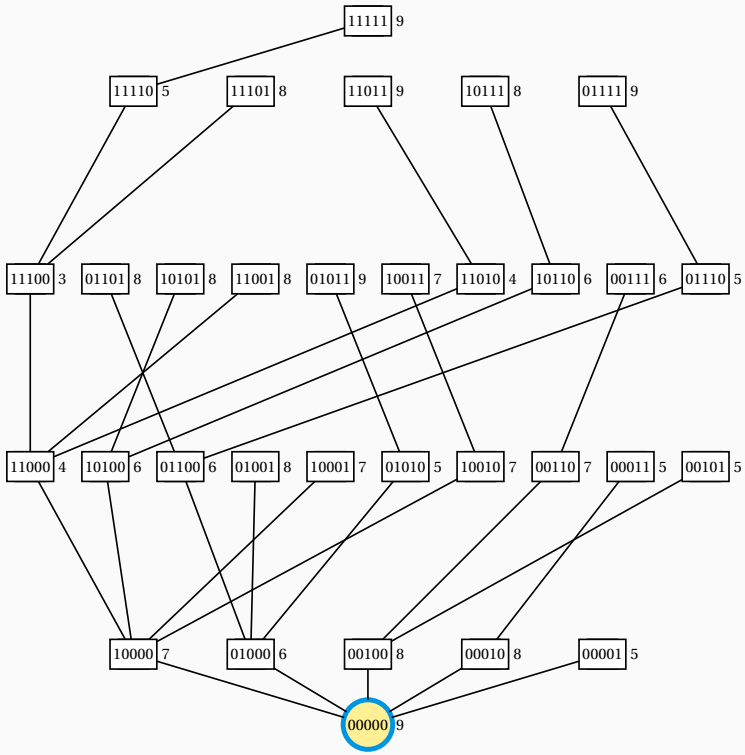
O algoritmo `U-Curve-Branch-and-Bound`

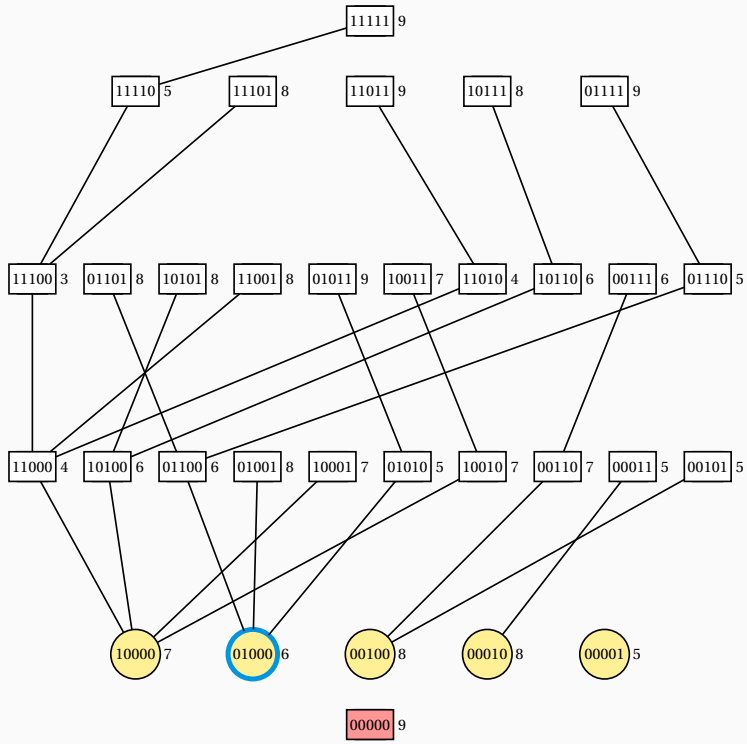
Este algoritmo busca o mínimo global ramificando na árvore como em uma busca em profundidade e faz podas sempre que o custo aumenta.



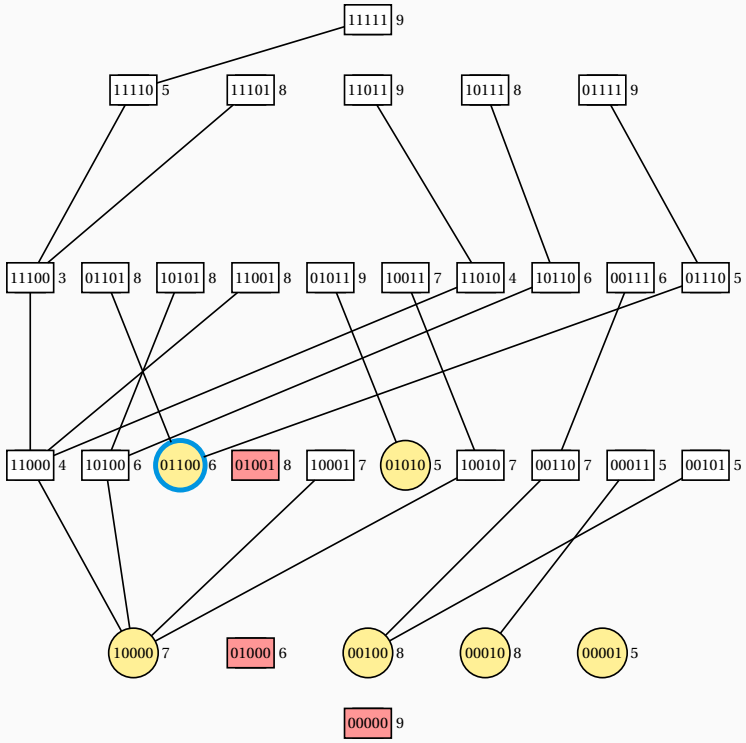
O algoritmo `U-Curve-Branch-and-Bound`

Este algoritmo busca o mínimo global ramificando na árvore como em uma busca em profundidade e faz podas sempre que o custo aumenta.

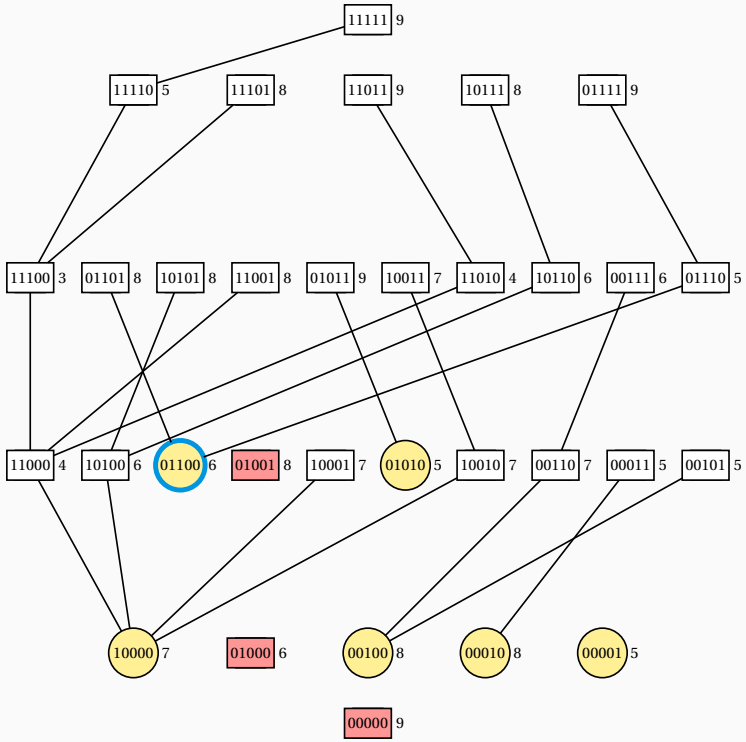


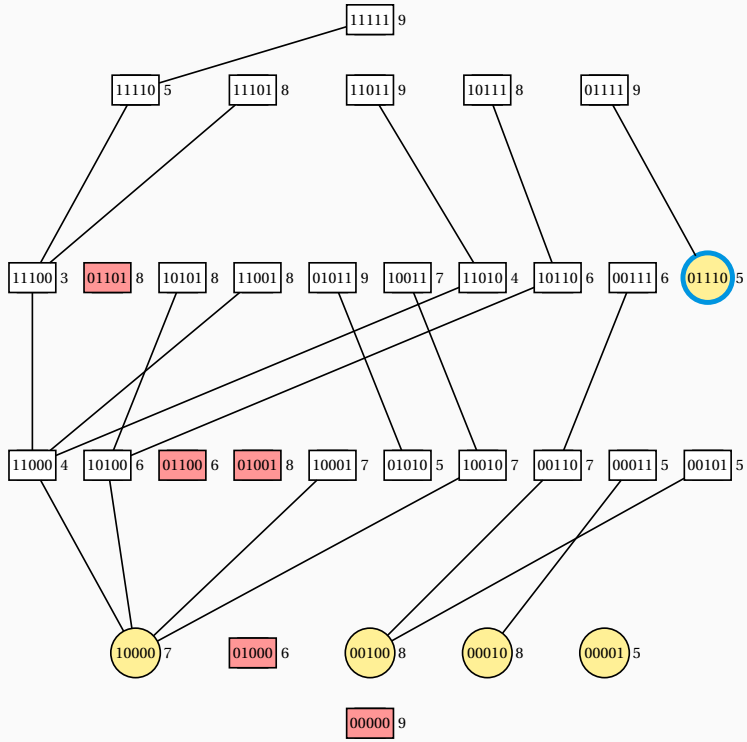


O algoritmo U-Curve-Branch-and-Bound

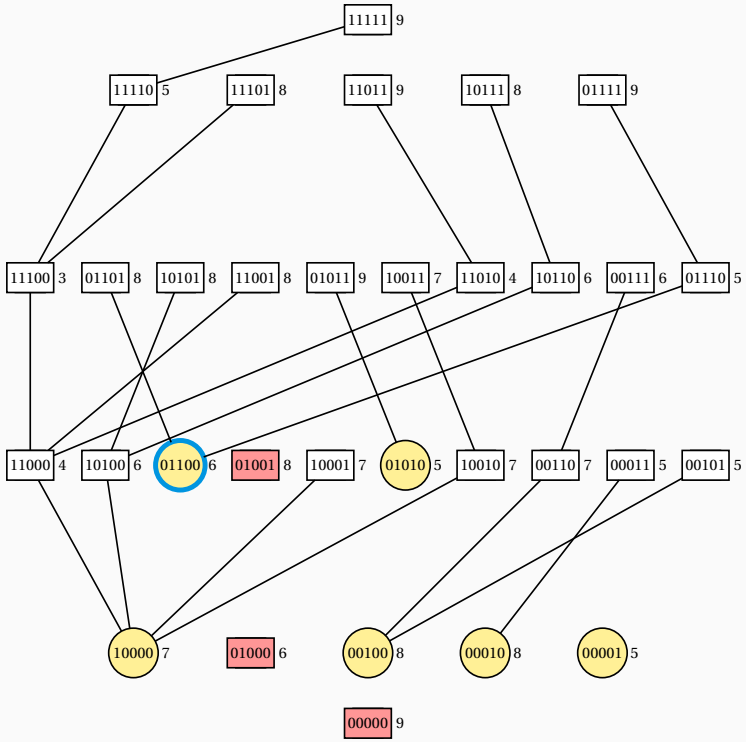


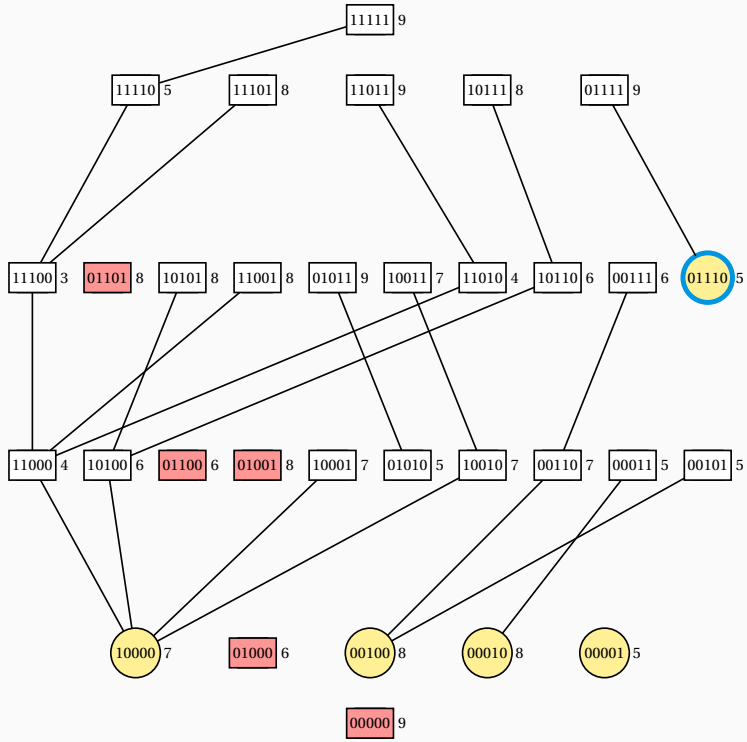
O algoritmo U-Curve-Branch-and-Bound





O algoritmo `U-Curve-Branch-and-Bound`





Note que se a condição de poda nunca é verdadeira, então o espaço de busca inteiro é percorrido.

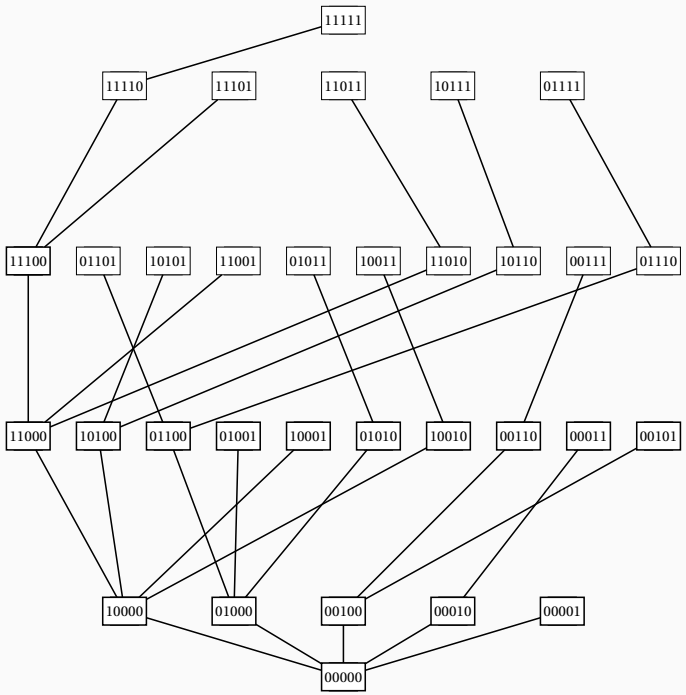
O algoritmo `Poset-Forest-Search`

Solução: percorrer o espaço de busca em duas direções.

O algoritmo `Poset-Forest-Search`

Solução: percorrer o espaço de busca em duas direções.

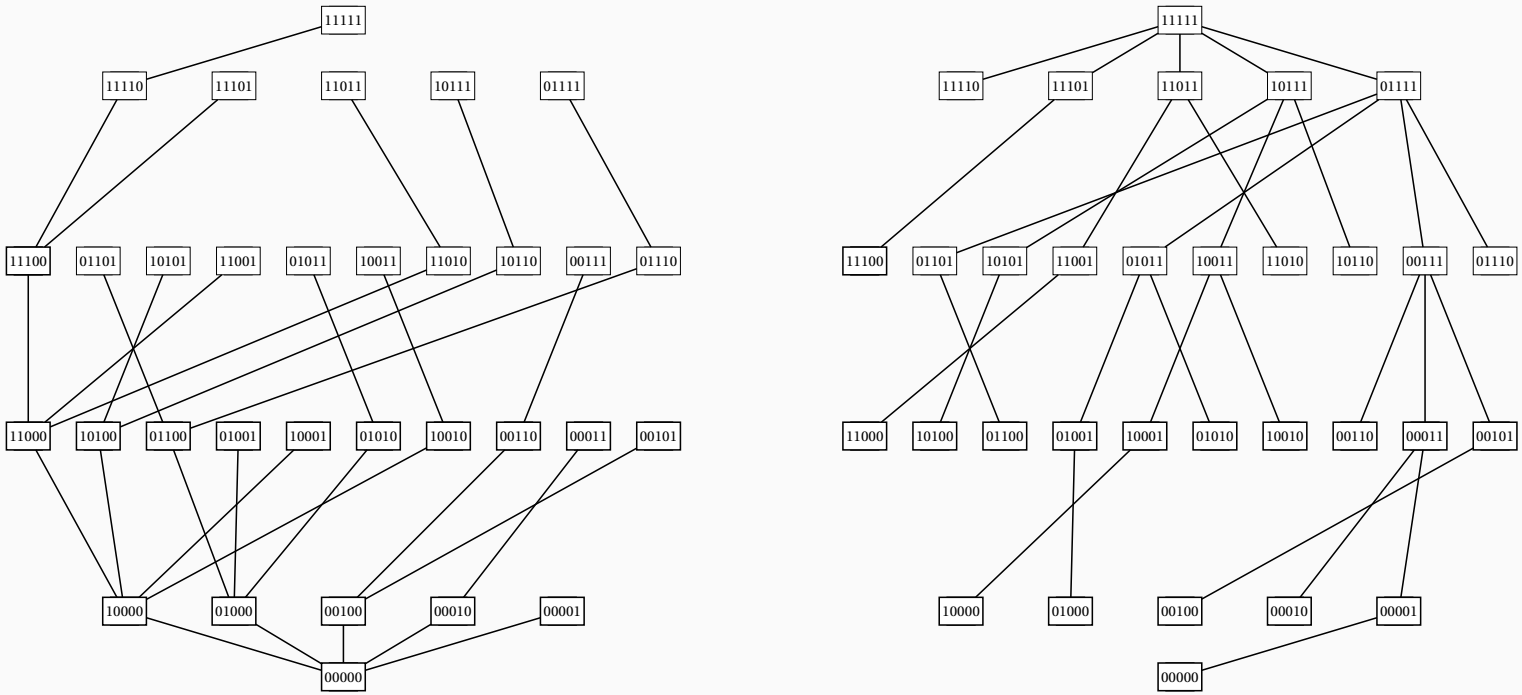
O algoritmo `Poset-Fores-Search` (`PFS`) pode fazer isso porque decompõe o espaço em duas árvores.



O algoritmo `Poset-Forest-Search`

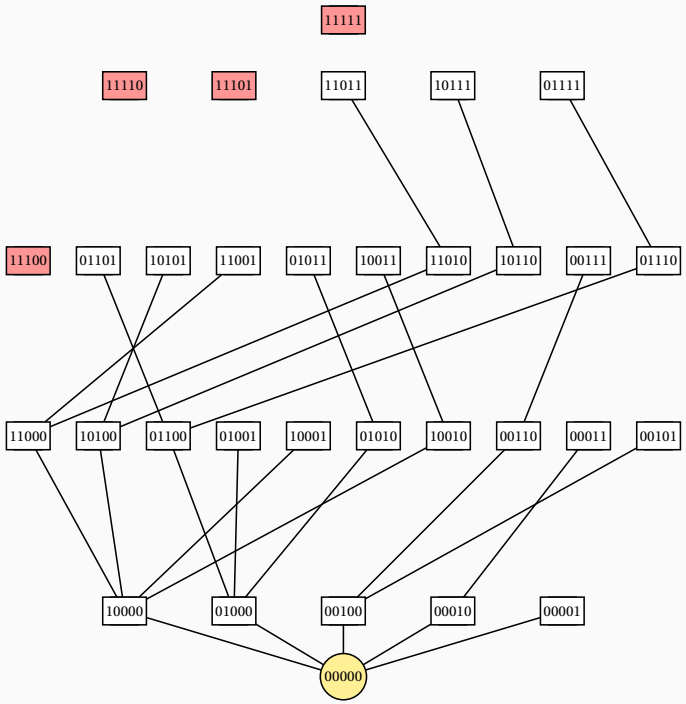
Solução: percorrer o espaço de busca em duas direções.

O algoritmo `Poset-Fores-Search` (`PFS`) pode fazer isso porque decompõe o espaço em duas árvores.



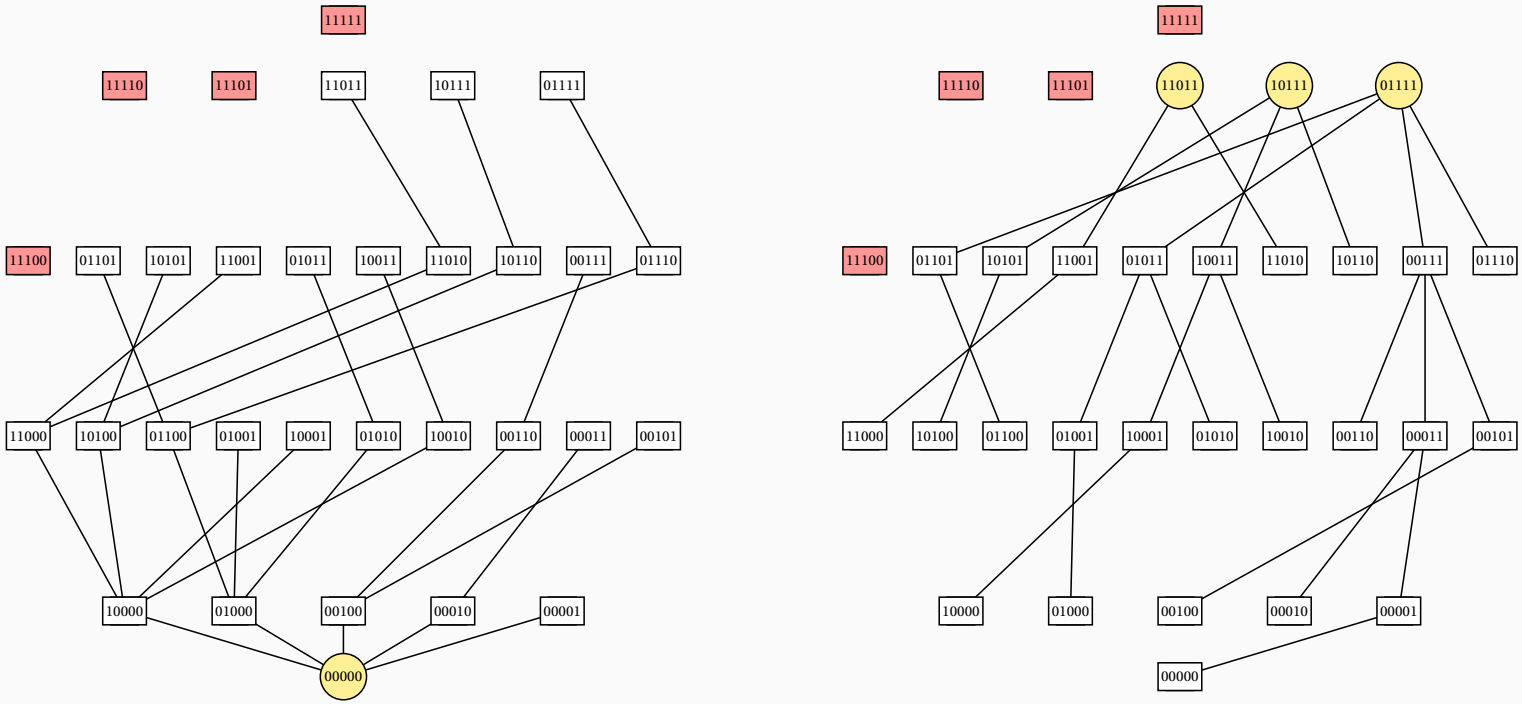
O algoritmo `Poset-Forest-Search`

Problema: agora é necessário manter as duas árvores equivalentes, ou seja, representando o mesmo espaço de busca.



O algoritmo `Poset-Forest-Search`

Problema: agora é necessário manter as duas árvores equivalentes, ou seja, representando o mesmo espaço de busca.



O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

- Escolher uma direção de percorrimento

O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

- Escolher uma direção de percorrimento
- Percorrer uma cadeia da floresta escolhida

O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

- Escolher uma direção de percorrimento
- Percorrer uma cadeia da floresta escolhida
- Sempre que a condição de poda for verdadeira:

O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

- Escolher uma direção de percorrimento
- Percorrer uma cadeia da floresta escolhida
- Sempre que a condição de poda for verdadeira:
    - Podar a floresta de percorrimento

O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

- Escolher uma direção de percorrimento
- Percorrer uma cadeia da floresta escolhida
- Sempre que a condição de poda for verdadeira:
  - Podar a floresta de percorrimento
  - Atualizar a floresta dual

O algoritmo `Poset-Forest-Search`

Podemos resumir o funcionamento do `PFS` aos seguintes passos:

- Escolher uma direção de percorrimento
- Percorrer uma cadeia da floresta escolhida
- Sempre que a condição de poda for verdadeira:
  - Podar a floresta de percorrimento
  - Atualizar a floresta dual

Melhoramentos ao

`Poset-Forest-Search`

Melhoramentos na implementação atual do PFS

O algoritmo implementado por Marcelo possui pontos que podiam ser explorados para se ter melhor desempenho computacional.

Mudanças na escolha de raízes

A implementação de Marcelo escolhia arbitrariamente como raiz de percorrimento a primeira quando ordenadas lexicograficamente.

Mudanças na escolha de raízes

A implementação de Marcelo escolhia arbitrariamente como raiz de percorrimento a primeira quando ordenadas lexicograficamente.

Propomos duas estratégias de escolhas:

- escolha aleatória e uniforme entre raízes;

Mudanças na escolha de raízes

A implementação de Marcelo escolhia arbitrariamente como raiz de percorrimento a primeira quando ordenadas lexicograficamente.

Propomos duas estratégias de escolhas:

- escolha aleatória e uniforme entre raízes;
- escolha da raiz com maior sub-árvore.

Resultados da mudança de escolha de raízes

Chamamos a variação do `PFS` que escolhe raízes de maneira aleatória e identicamente provável de `PFS-RAND`.

Resultados da mudança de escolha de raízes

Chamamos a variação do PFS que escolhe raízes de maneira aleatória e identicamente provável de PFS-RAND.

Instância		Tempo de execução médio (s)		Número médio de cálculos de custo	
$\|S\|$	$2^{\|S\|}$	PFS	PFS_RAND	PFS	PFS_RAND
10	1024	0.013 ± 0.003	0.014 ± 0.003	590.8 ± 198.5	599.5 ± 177.5
11	2048	0.019 ± 0.004	0.022 ± 0.007	1114.8 ± 331.3	1090.1 ± 350.3
12	4096	0.029 ± 0.008	0.036 ± 0.013	1848.6 ± 600.8	1835.7 ± 683.0
13	8192	0.060 ± 0.018	0.090 ± 0.039	4314.4 ± 1496.4	4201.1 ± 1580.7
14	16384	0.100 ± 0.041	0.191 ± 0.110	7323.4 ± 3318.9	7333.8 ± 3161.0
15	32768	0.180 ± 0.076	0.453 ± 0.311	12958.1 ± 5654.0	12807.5 ± 5753.7
16	65536	0.406 ± 0.185	1.715 ± 1.400	27573.8 ± 12459.5	27036.9 ± 12687.5
17	131072	0.717 ± 0.397	5.416 ± 5.266	48176.2 ± 26938.3	47852.1 ± 26427.6
18	262144	1.325 ± 0.754	15.890 ± 17.726	84417.9 ± 48587.7	84025.0 ± 48882.4
19	524288	2.771 ± 1.603	69.600 ± 82.342	167659.1 ± 99686.7	164612.1 ± 102018.3

Resultados da mudança de escolha de raízes

Chamamos a variação do `PFS` que escolhe as raízes com maior sub-árvore de `PFS-LEFTMOST`.

Resultados da mudança de escolha de raízes

Chamamos a variação do PFS que escolhe as raízes com maior sub-árvore de PFS-LEFTMOST.

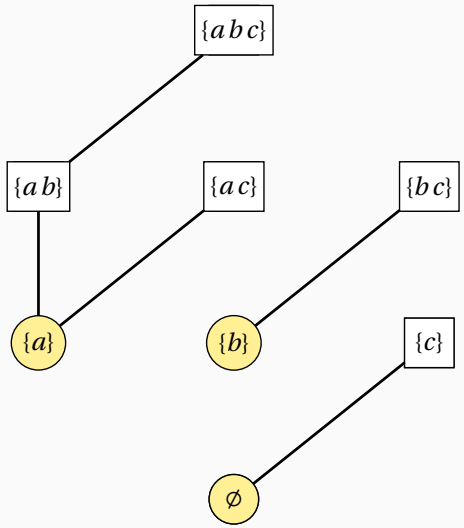
Instância		Tempo de execução médio (s)		Número médio de cálculos de custo	
$\|S\|$	$2^{\|S\|}$	PFS	PFS_LEFTMOST	PFS	PFS_LEFTMOST
10	1024	0.013 ± 0.002	0.023 ± 0.004	606.1 ± 133.5	665.0 ± 165.8
11	2048	0.020 ± 0.004	0.042 ± 0.010	1122.1 ± 351.2	1316.6 ± 382.2
12	4096	0.032 ± 0.008	0.078 ± 0.024	2183.7 ± 733.2	2515.8 ± 871.3
13	8192	0.054 ± 0.017	0.160 ± 0.061	3887.7 ± 1389.9	4716.8 ± 1777.8
14	16384	0.107 ± 0.034	0.345 ± 0.133	7851.2 ± 2793.0	9506.8 ± 3673.9
15	32768	0.196 ± 0.085	0.672 ± 0.274	13780.3 ± 6049.9	17071.6 ± 7005.1
16	65536	0.348 ± 0.189	1.271 ± 0.661	24106.5 ± 13159.9	30055.6 ± 15363.6
17	131072	0.785 ± 0.361	3.137 ± 1.476	52369.0 ± 24751.2	67585.6 ± 30978.4
18	262144	1.445 ± 0.657	6.146 ± 3.032	92095.9 ± 42566.6	120635.7 ± 58039.0
19	524288	3.298 ± 1.883	13.881 ± 7.595	199151.0 ± 112167.8	256078.6 ± 135958.4

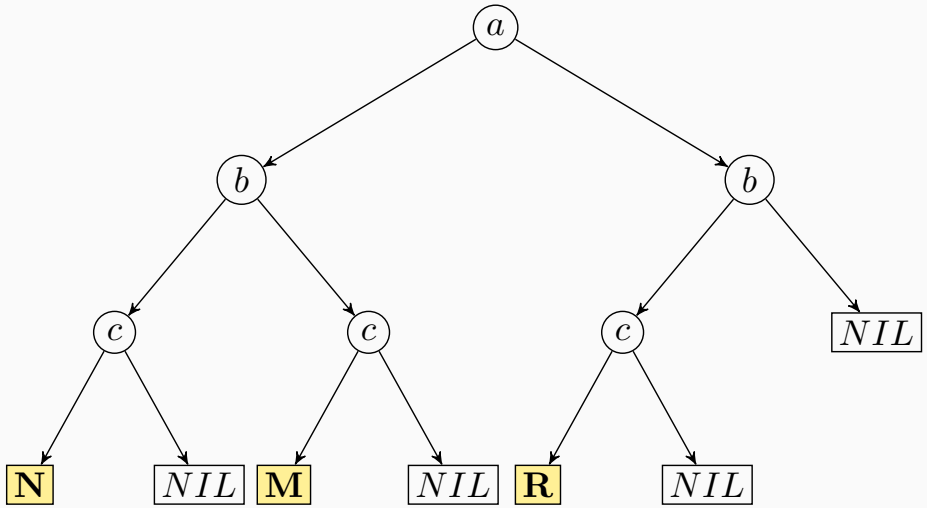
Melhoramentos na estrutura de armazenamento de raízes

Mudamos a implementação de Marcelo para usar diagramas de decisão binários ordenados (OBDDs).

Melhoramentos na estrutura de armazenamento de raízes

Mudamos a implementação de Marcelo para usar diagramas de decisão binários ordenados (OBDDs).





Resultados da mudança de estrutura para armazenamento de raízes

Chamamos de `OPFS` o algoritmo que usa OBDDs para armazenamento de raízes.

Instância		Tempo de execução médio (s)		Número médio de cálculos de custo	
$\|S\|$	$2^{\|S\|}$	`PFS`	`OPFS`	`PFS`	`OPFS`
10	1024	0.013 ± 0.003	0.018 ± 0.003	598.0 ± 192.8	635.5 ± 171.9
11	2048	0.020 ± 0.004	0.029 ± 0.007	1152.1 ± 314.7	1117.9 ± 336.4
12	4096	0.031 ± 0.010	0.049 ± 0.013	2024.1 ± 751.6	2048.2 ± 700.9
13	8192	0.057 ± 0.017	0.097 ± 0.033	3996.3 ± 1431.6	3973.4 ± 1462.6
14	16384	0.094 ± 0.038	0.171 ± 0.063	6634.8 ± 2944.0	6906.5 ± 2786.5
15	32768	0.182 ± 0.079	0.323 ± 0.156	13140.1 ± 6020.6	12711.2 ± 6319.7
16	65536	0.370 ± 0.169	0.660 ± 0.314	25658.2 ± 11606.7	25303.4 ± 12169.5
17	131072	0.819 ± 0.370	1.480 ± 0.665	53344.9 ± 24350.4	53217.2 ± 24154.5
18	262144	1.515 ± 0.905	2.736 ± 1.626	94677.6 ± 54496.3	94079.4 ± 55435.6
19	524288	2.612 ± 1.869	4.818 ± 3.355	156150.5 ± 107369.8	156021.8 ± 107516.8
20	1048576	6.085 ± 3.900	11.550 ± 7.661	344144.1 ± 212627.1	343229.2 ± 212624.4

Paralelização do PFS

Implementamos também uma versão paralela do algoritmo PFS.

Paralelização do PFS

Implementamos também uma versão paralela do algoritmo PFS.

Entretanto, a etapa de atualização da floresta dual é complicada e pode gerar condições de corrida, o que deixou a paralelização complicada.

O algoritmo UBB-PFS

Este algoritmo é uma nova alternativa paralela que é dividida em duas partes:

O algoritmo UBB-PFS

Este algoritmo é uma nova alternativa paralela que é dividida em duas partes:

- Percorrimento sequencial: idêntico ao UBB deve criar sub-árvores no espaço enquanto faz uma ramificação do tipo busca em profundidade.

O algoritmo UBB-PFS

Este algoritmo é uma nova alternativa paralela que é dividida em duas partes:

- Percorrimento sequencial: idêntico ao UBB deve criar sub-árvores no espaço enquanto faz uma ramificação do tipo busca em profundidade.
- Solução em paralelo: cada sub-árvore gerada na etapa de ramificação deve ser resolvida por uma chamada do PFS.

Resultados do UBB-PFS

O UBB-PFS foi mais rápido do que o PFS.

Instância		Tempo de execução médio (s)		
$\|S\|$	$2^{\|S\|}$	UBB	PFS	UBB-PFS
17	131072	0.161 ± 0.122	0.650 ± 0.347	0.326 ± 0.175
18	262144	0.321 ± 0.233	1.482 ± 0.768	0.703 ± 0.380
19	524288	0.620 ± 0.447	2.711 ± 1.562	1.309 ± 0.729
20	1048576	1.312 ± 0.970	5.007 ± 3.302	2.478 ± 1.547
21	2097152	2.494 ± 1.893	11.125 ± 6.749	5.458 ± 3.294
22	4194304	4.589 ± 4.122	19.085 ± 15.147	8.832 ± 6.846
23	8388608	12.228 ± 7.922	40.323 ± 29.649	18.891 ± 12.786
24	16777216	24.273 ± 16.277	113.332 ± 76.688	67.178 ± 46.516

Resultados do UBB-PFS

E computou menos a função custo do que o UBB.

Instância		Número médio de cálculos de custo		
$\|S\|$	$2^{\|S\|}$	UBB	PFS	UBB-PFS
17	131072	73373.3 ± 55994.3	46808.9 ± 24533.5	49348.6 ± 24556.7
18	262144	150035.2 ± 108299.3	103166.6 ± 52464.7	105306.4 ± 53472.0
19	524288	292561.2 ± 210771.2	183125.7 ± 104965.4	189545.7 ± 102145.9
20	1048576	617049.5 ± 450468.2	323097.4 ± 213634.3	340694.2 ± 202389.6
21	2097152	1172641.6 ± 879148.5	691991.3 ± 413262.9	704790.2 ± 407143.8
22	4194304	2099973.2 ± 1863285.8	1133395.1 ± 874492.0	1156564.2 ± 862152.0
23	8388608	5435778.8 ± 3468245.3	2276694.5 ± 1621342.2	2345648.2 ± 1558258.5
24	16777216	10146842.9 ± 6673018.3	5527504.2 ± 3413432.3	5609052.7 ± 3337059.1

O algoritmo

`Parallel-U-Curve-Search`