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.

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

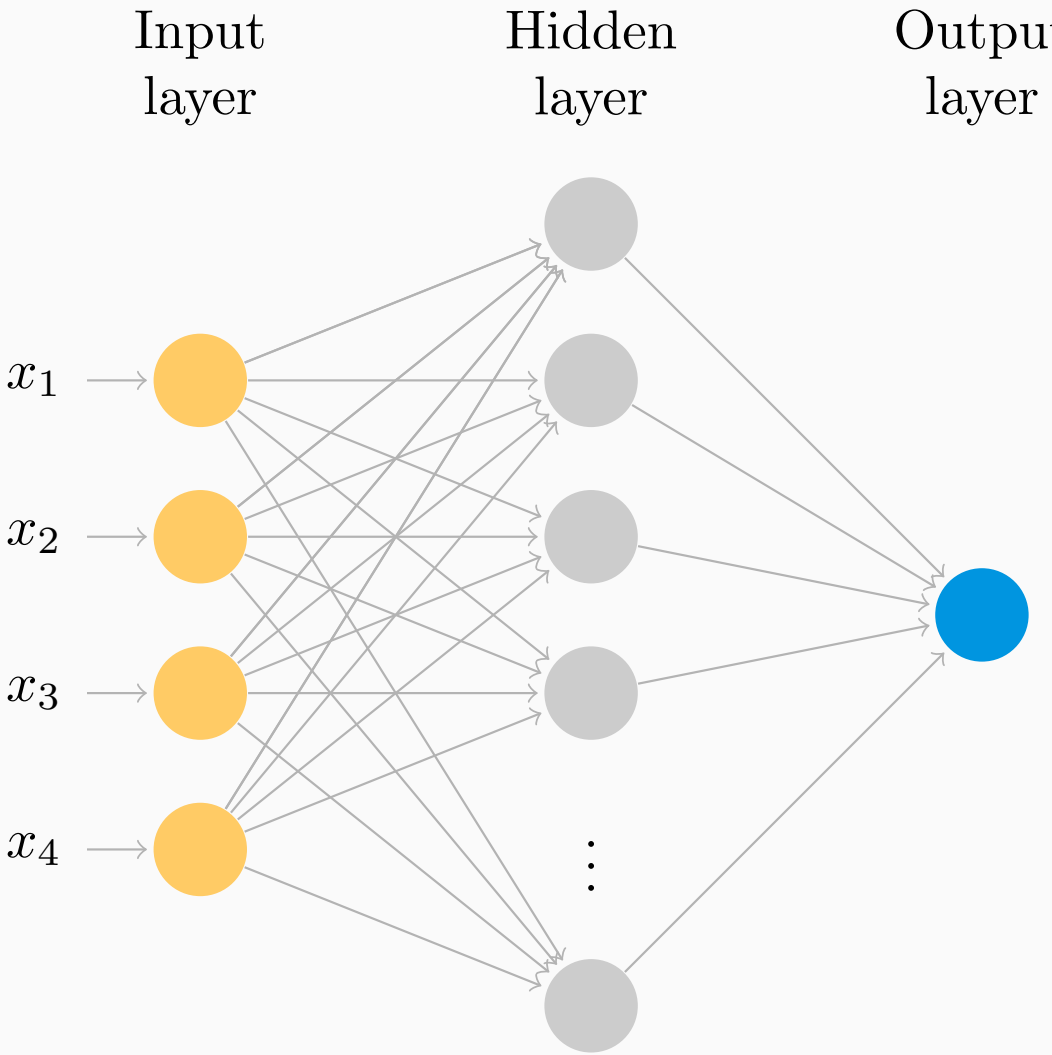
Modelos computacionais

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.

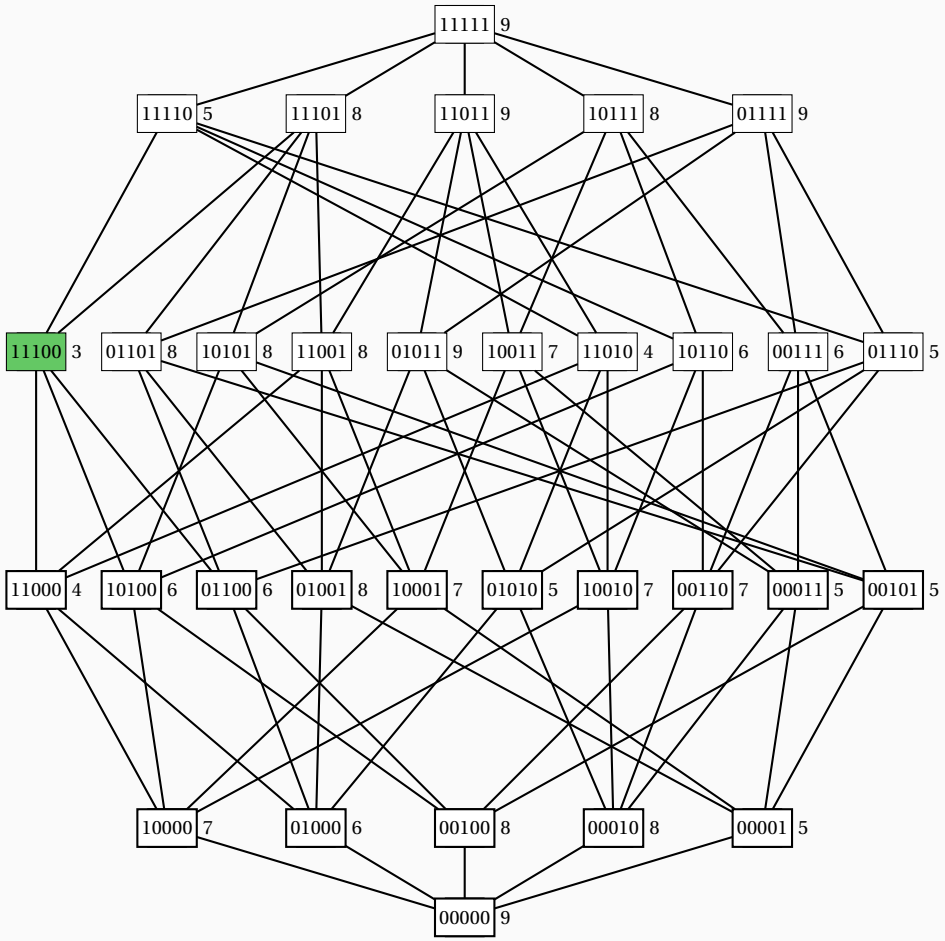
O problema de seleção de características

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 $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n$.

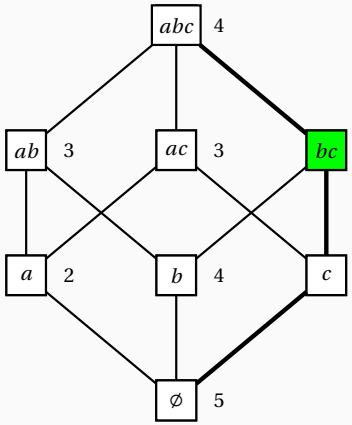
A função de custo

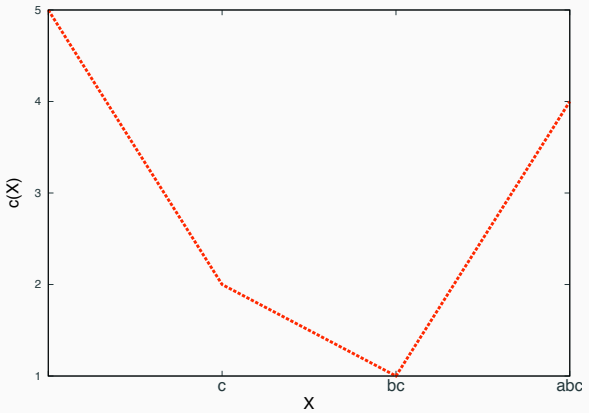
A função de custo c deve refletir a qualidade de um conjunto de características X a ser usado no modelo.

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

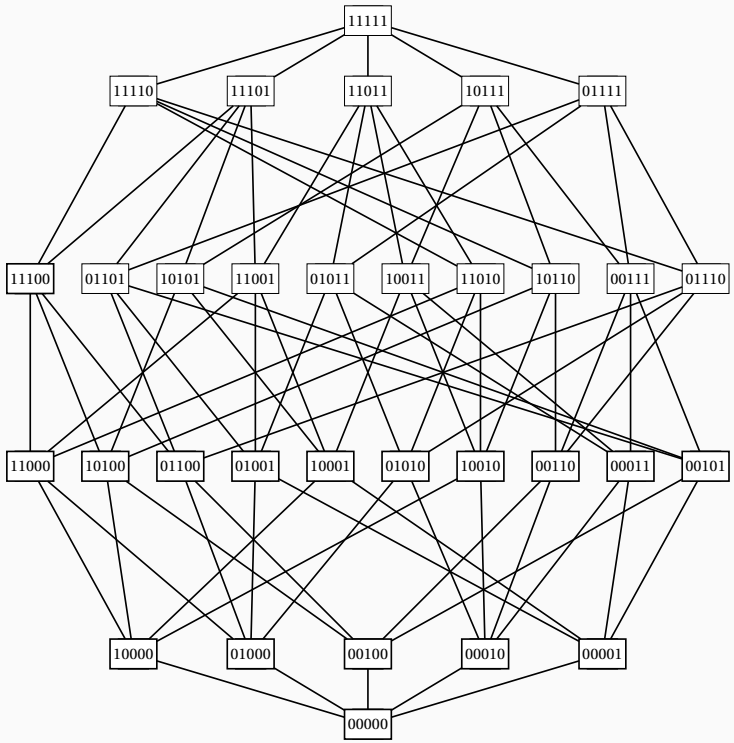
Definição (Problema U-Curve)

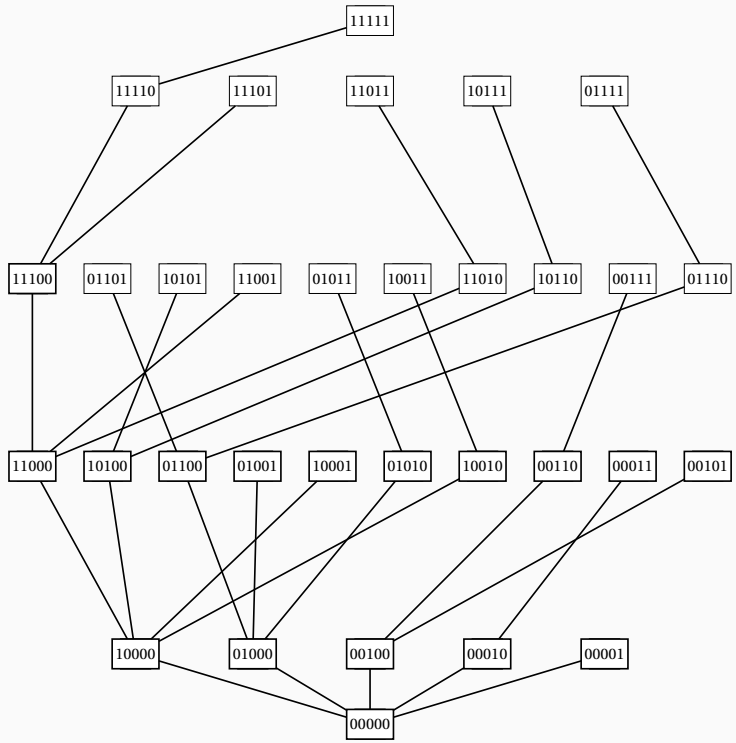
Dados um conjunto finito e não-vazio S e uma função de custo c decomponível em curvas U, encontrar um subconjunto $X \in \mathcal{P}(S)$ tal que $c(X)$ é mínimo.

Algoritmos baseados em florestas

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

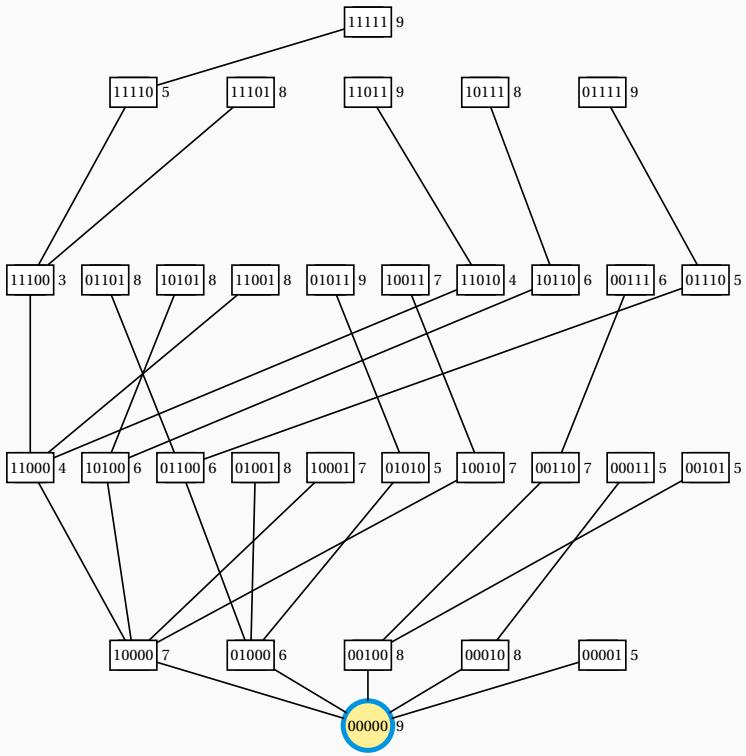
O algoritmo `U-Curve-Branch-and-Bound` (UBB) organiza o espaço de busca em uma árvore.





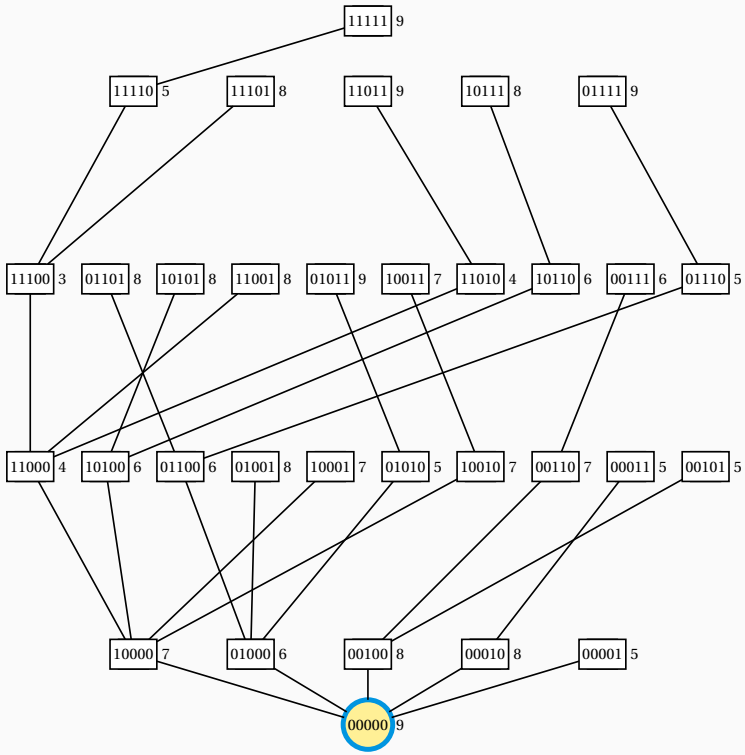
O algoritmo `U-Curve-Branch-and-Bound`

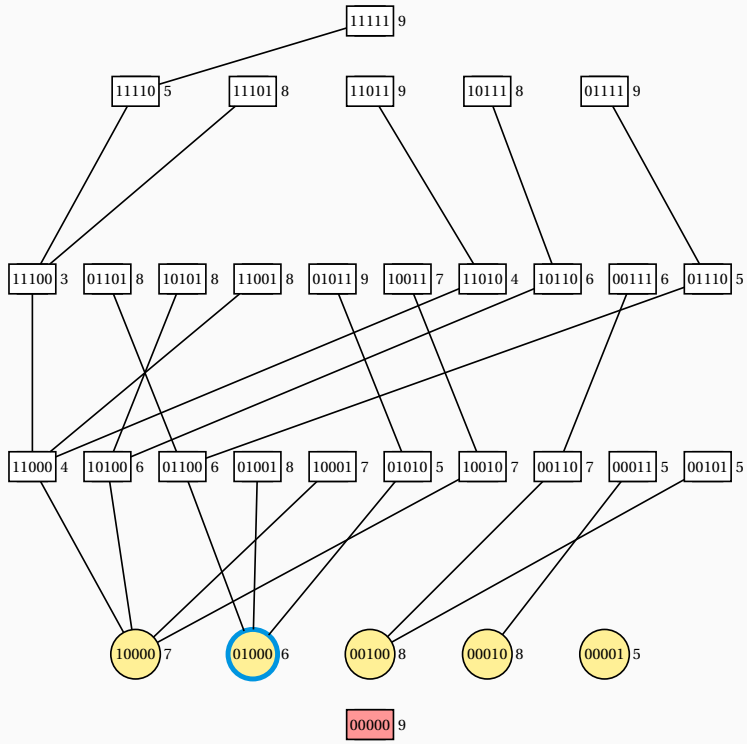
Este algoritmo busca o mínimo global ramificando na árvore como em uma busca em profundidade e faz podas sempre que o custo aumenta.



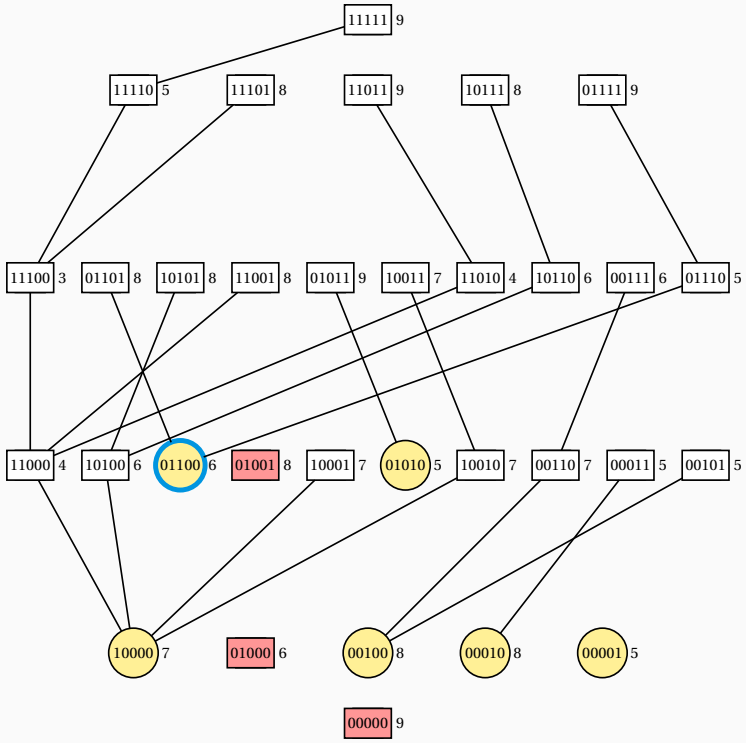
O algoritmo `U-Curve-Branch-and-Bound`

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

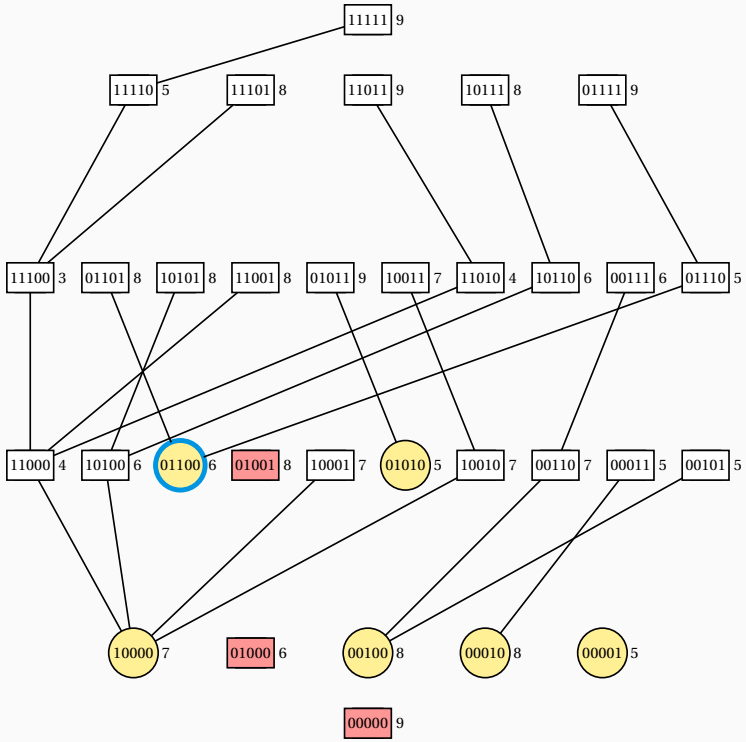


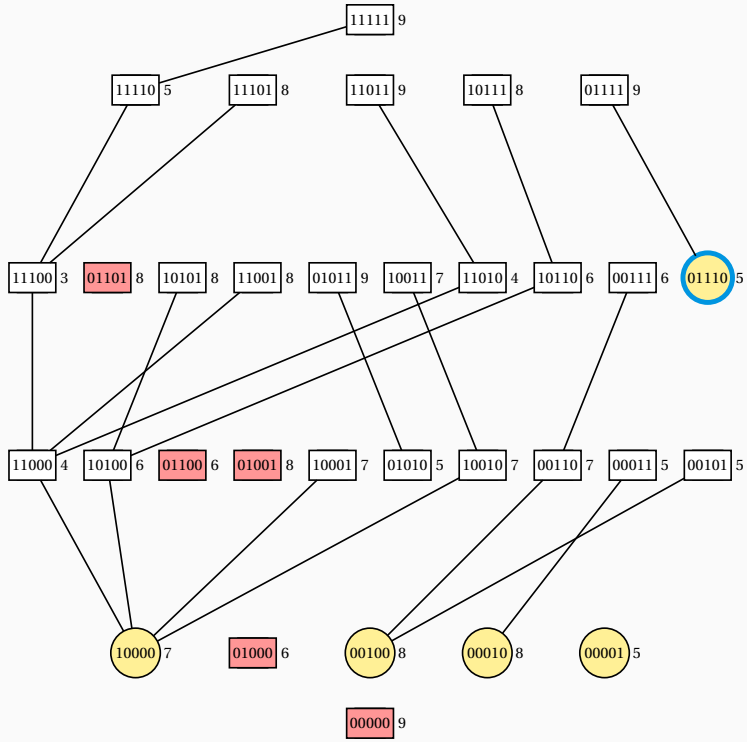


O algoritmo U-Curve-Branch-and-Bound

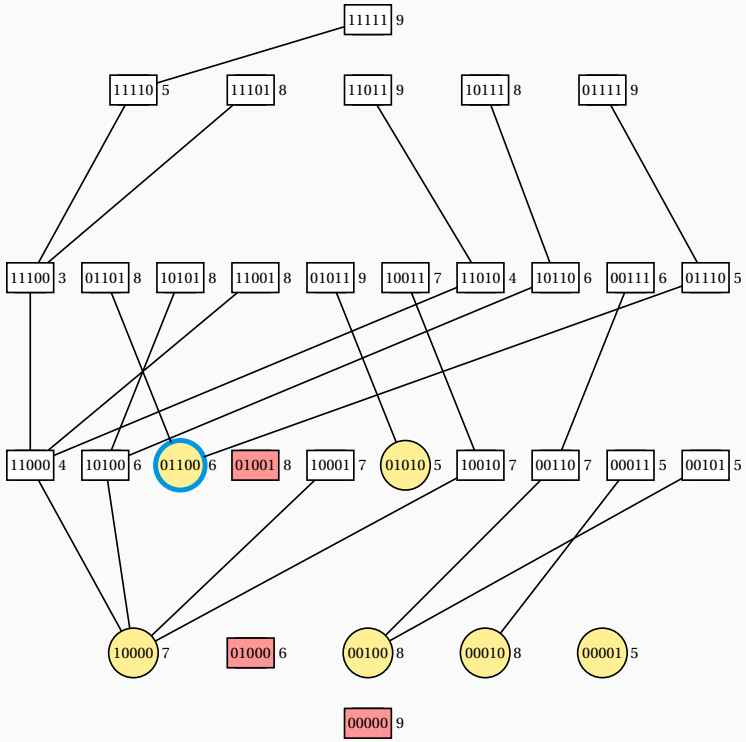


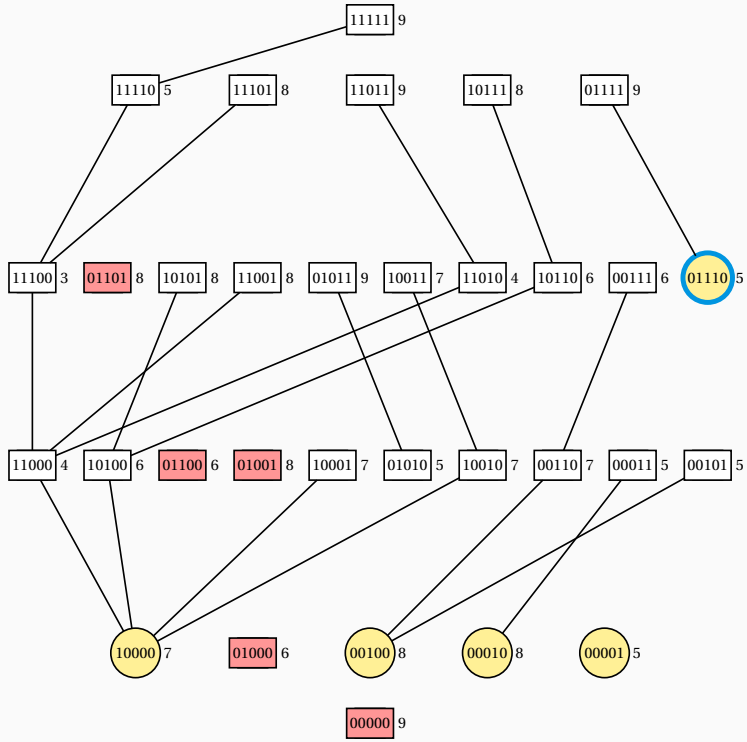
O algoritmo U-Curve-Branch-and-Bound





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.

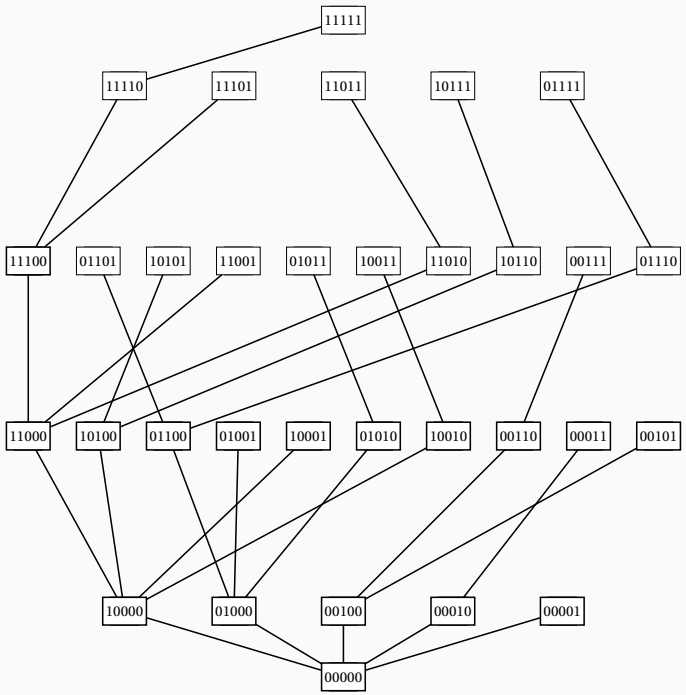
O algoritmo `Poset-Forest-Search`

Solução: percorrer o espaço de busca em duas direções.

O algoritmo `Poset-Forest-Search`

Solução: percorrer o espaço de busca em duas direções.

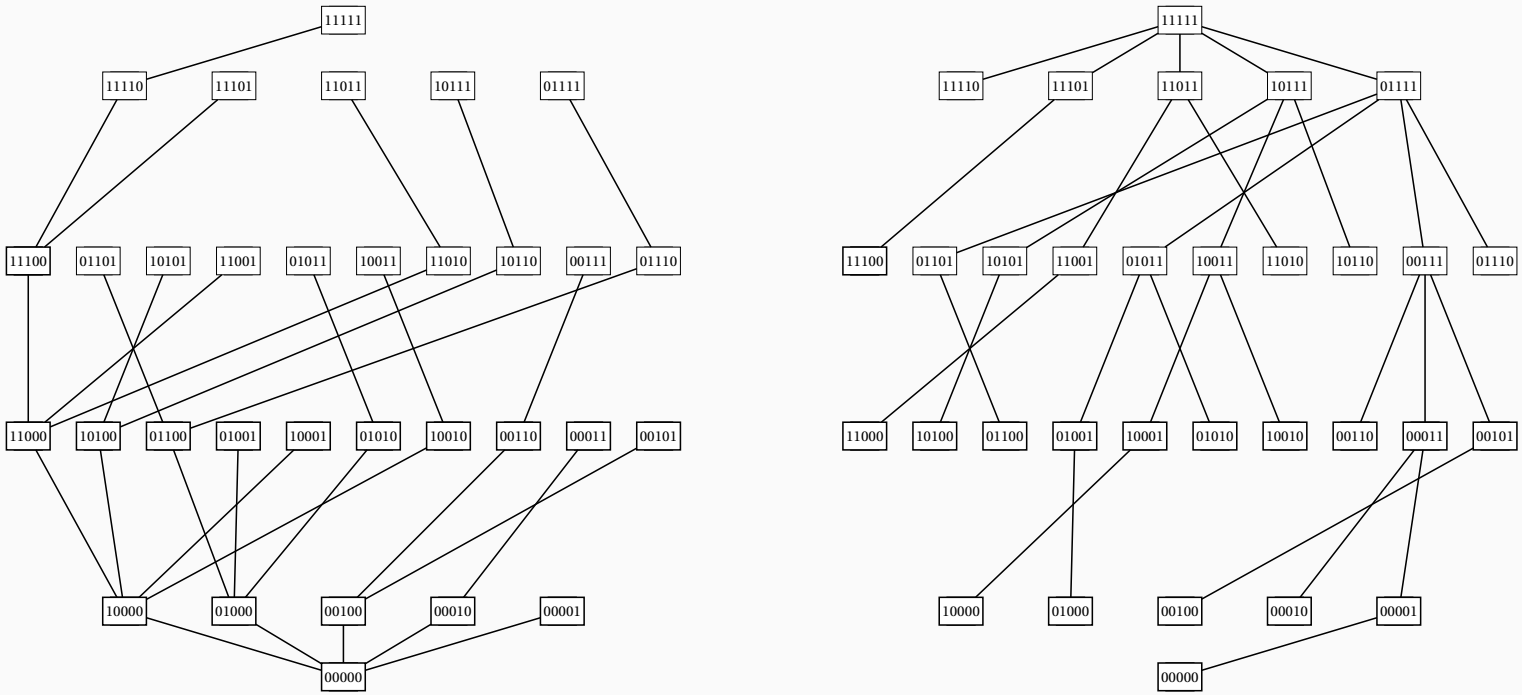
O algoritmo `Poset-Fores-Search` (`PFS`) pode fazer isso porque decompõe o espaço em duas árvores.



O algoritmo `Poset-Forest-Search`

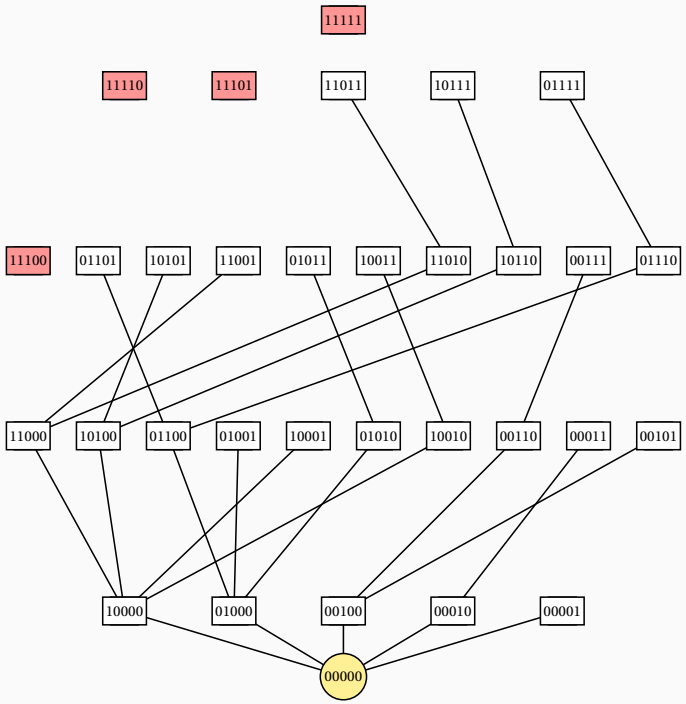
Solução: percorrer o espaço de busca em duas direções.

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



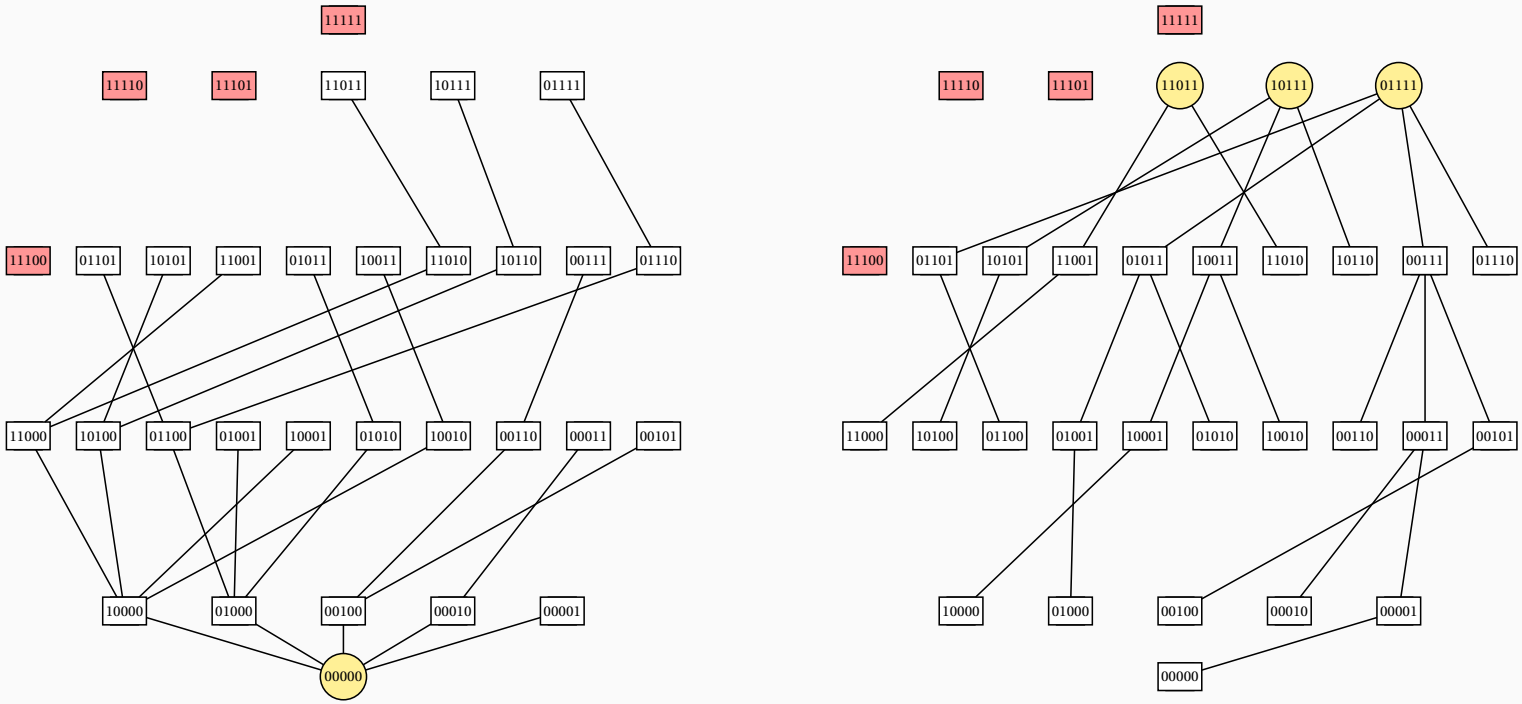
O algoritmo `Poset-Forest-Search`

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
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.

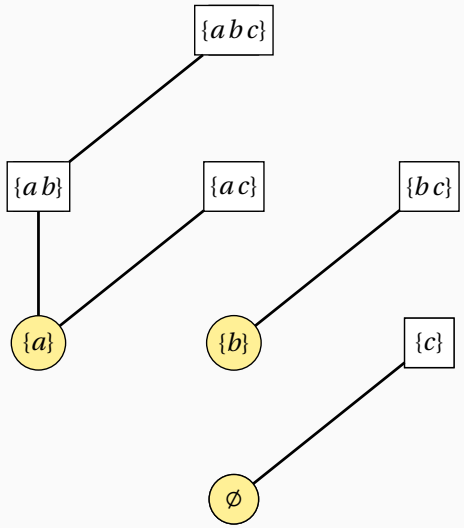
Instância		Tempo de execução médio (s)		Número médio de cálculos de custo	
$\|S\|$	$2^{\|S\|}$	PFS	PFS_LEFTMOST	PFS	PFS_LEFTMOST
15	32768	**0.196 ± 0.085**	0.672 ± 0.274	**13780.3 ± 6049.9**	17071.6 ± 7005.1
16	65536	**0.348 ± 0.189**	1.271 ± 0.661	**24106.5 ± 13159.9**	30055.6 ± 15363.6
17	131072	**0.785 ± 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

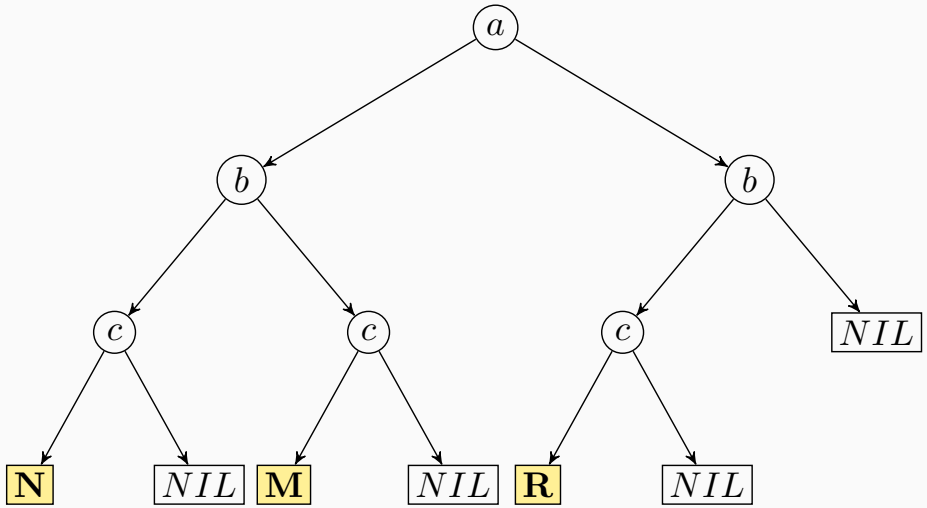
Melhoramentos na estrutura de armazenamento de raízes

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`
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**
21	2097152	**11.416 ± 8.296**	21.818 ± 16.269	616936.2 ± 436491.2	**613526.2 ± 438580.0**
22	4194304	**19.950 ± 17.799**	42.112 ± 45.109	960842.2 ± 785137.2	**959905.4 ± 783257.3**
23	8388608	**42.792 ± 35.622**	87.262 ± 90.579	**2053472.4 ± 1690882.1**	2060184.5 ± 1682011.0

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
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
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`

Ideia do `Parallel-U-Curve-Search` (PUCS)

Um algoritmo de fácil paralelização e pouco entrelace de linhas de processamento,

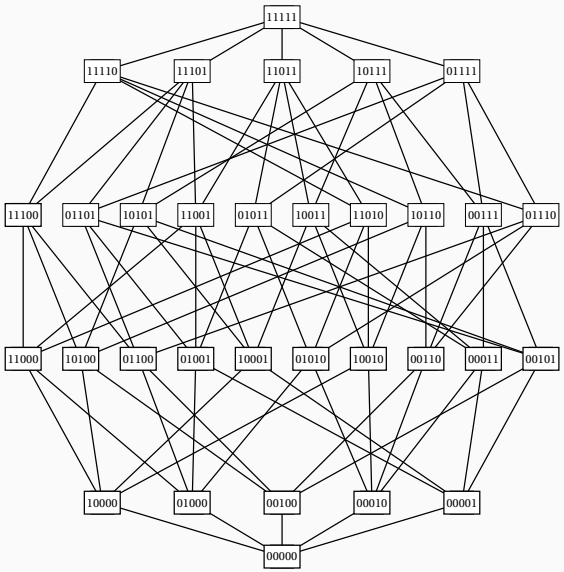
Ideia do `Parallel-U-Curve-Search` (PUCS)

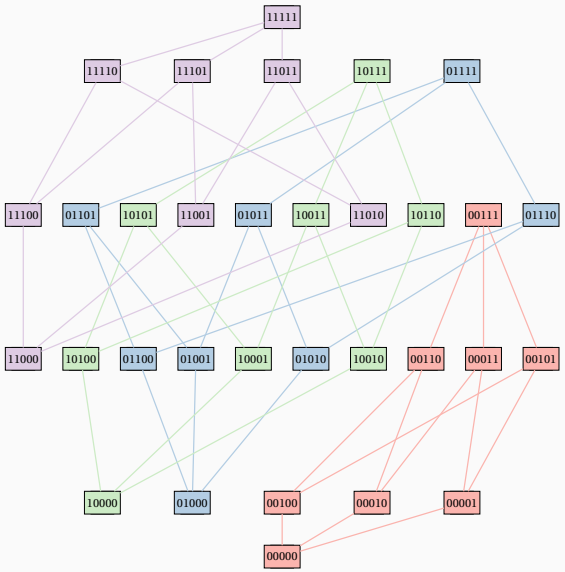
Um algoritmo de fácil paralelização e pouco entrelace de linhas de processamento, e que também distribua o trabalho em partes de tamanho parecido.

Ideia do `Parallel-U-Curve-Search` (PUCS)

Um algoritmo de fácil paralelização e pouco entrelace de linhas de processamento, e que também distribua o trabalho em partes de tamanho parecido.

Fazemos isso ao definir um particionamento do espaço.





Particionamento do espaço de busca

Para particionar o espaço, escolhemos um conjunto arbitrário de variáveis S'.

Particionamento do espaço de busca

Para particionar o espaço, escolhemos um conjunto arbitrário de variáveis S'.

Agora, definimos a relação de equivalência para os conjuntos de características:

$$X \sim Y \iff (X \cap S') = (Y \cap S')$$

Estrutura recursiva do problema

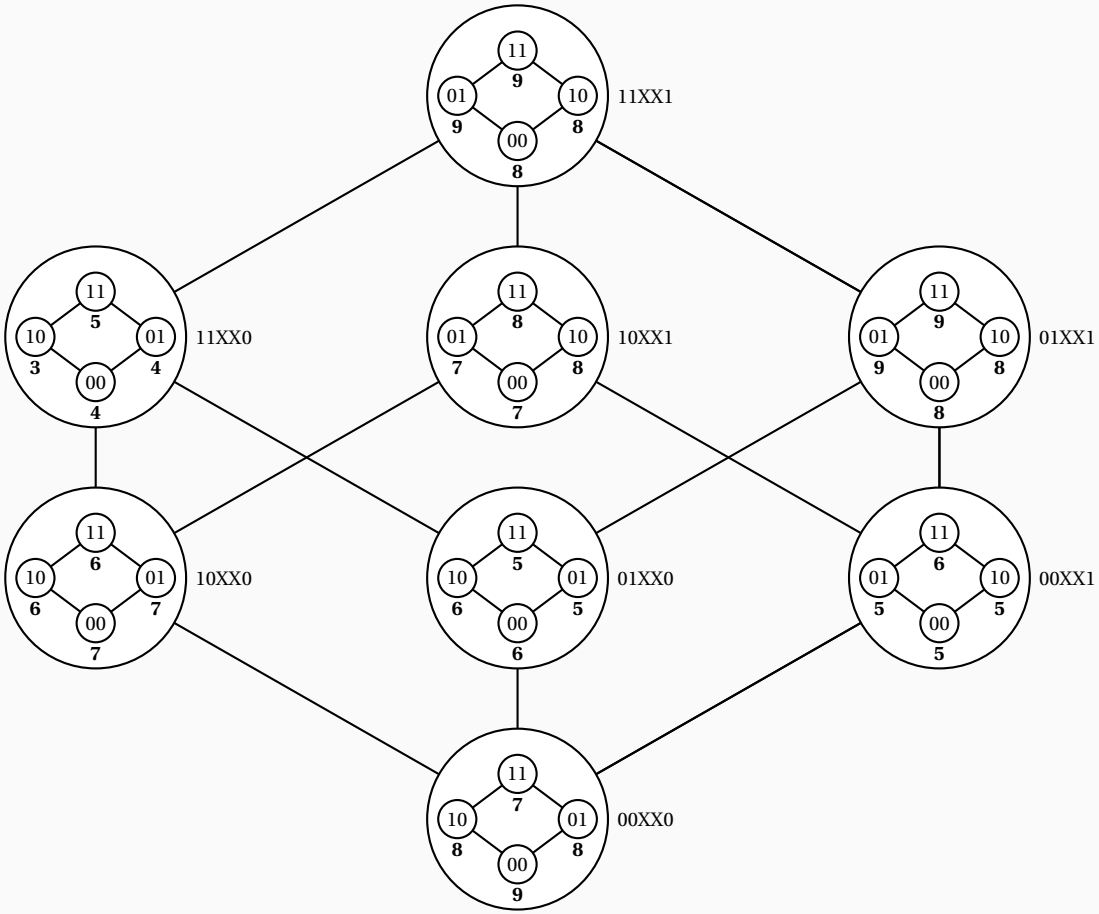
Se representamos cada parte por um subconjunto de características de S', então temos um reticulado Booleano de partes. Chamamos este reticulado de reticulado externo.

Estrutura recursiva do problema

Se representamos cada parte por um subconjunto de características de S', então temos um reticulado Booleano de partes. Chamamos este reticulado de reticulado externo.

Se representamos, cada nó de uma parte por um subconjunto de características de $S - S'$, então temos um reticulado Booleano de nós de uma parte. Chamamos este reticulado de reticulado interno.

Estrutura recursiva do problema



Dinâmica

Podemos resumir a dinâmica deste algoritmo nos passos:

- passeio aleatório no reticulado externo, com podas;
- solução de partes não podadas;
- união de respostas das partes.

Pontas de um reticulado

Se X é um conjunto maximal em um reticulado Booleano, isto é, $Y \supseteq X \iff X = Y$, então X é ponta de cima do reticulado.

Pontas de um reticulado

Se X é um conjunto maximal em um reticulado Booleano, isto é, $Y \supseteq X \iff X = Y$, então X é ponta de cima do reticulado.

Se X é conjunto minimal, isto é, $Y \subseteq X \iff X = Y$, então X é ponta de baixo do reticulado.

Condições de poda

Sejam P e Q duas partes adjacentes, sendo que $Q \subseteq P$.

Condições de poda

Sejam P e Q duas partes adjacentes, sendo que $Q \subseteq P$.

Definição (Condição de poda inferior do `PUCS`)
Se o custo da ponta de cima de Q é maior do que a ponta de cima de P, então nenhuma das partes abaixo de Q contém o mínimo global.

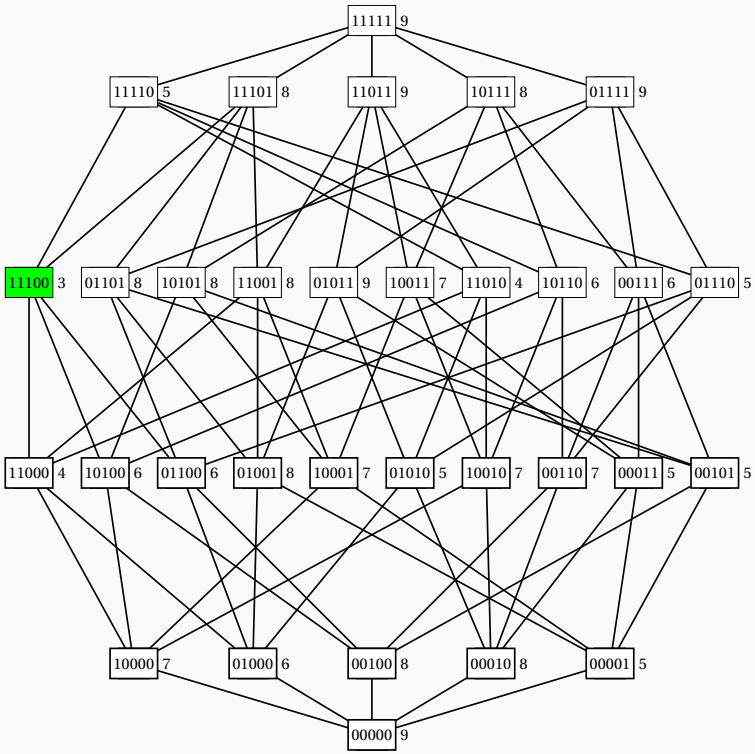
Condições de poda

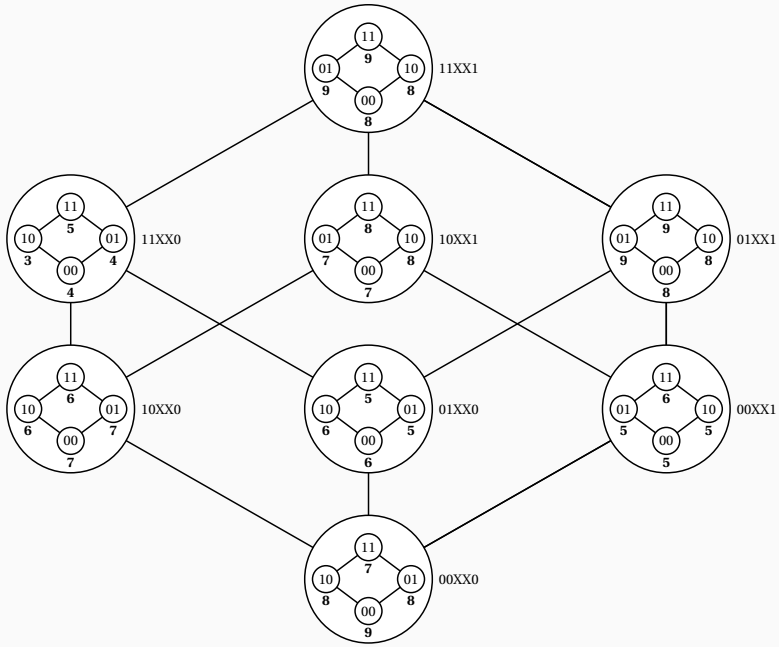
Sejam P e Q duas partes adjacentes, sendo que $Q \subseteq P$.

Definição (Condição de poda inferior do `PUCS`)
Se o custo da ponta de cima de Q é maior do que a ponta de cima de P, então nenhuma das partes abaixo de Q contém o mínimo global.

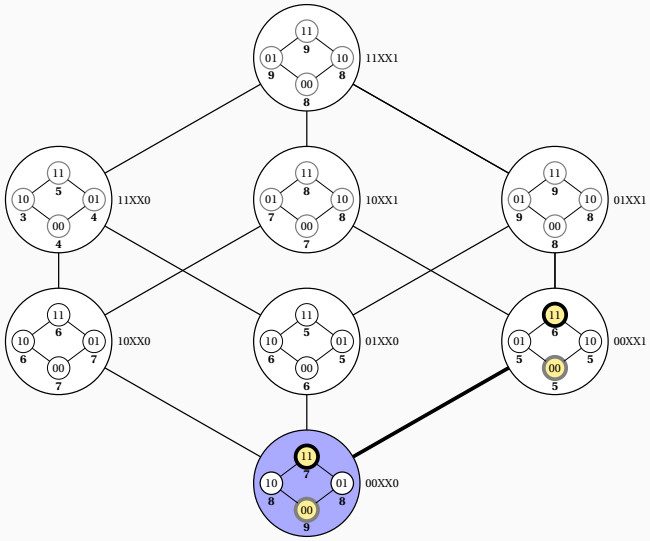
Definição (Condição de poda superior do `PUCS`)
Se o custo da ponta de baixo de P é maior do que a ponta de baixo de P, então nenhuma das partes acima de Q contém o mínimo global.

Simulação de execução

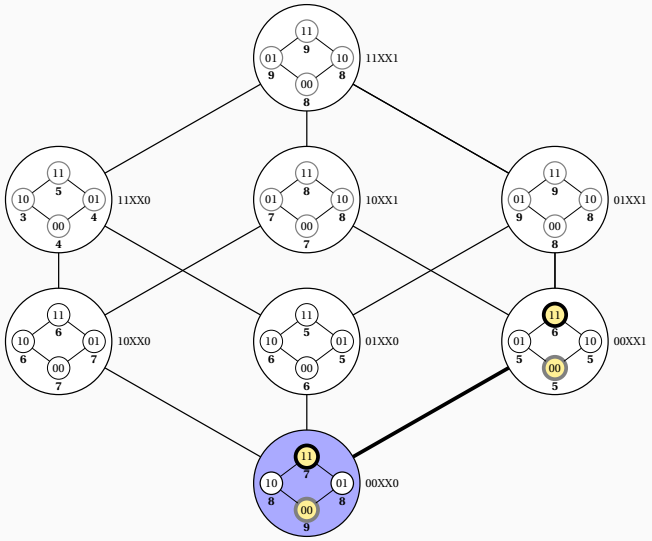


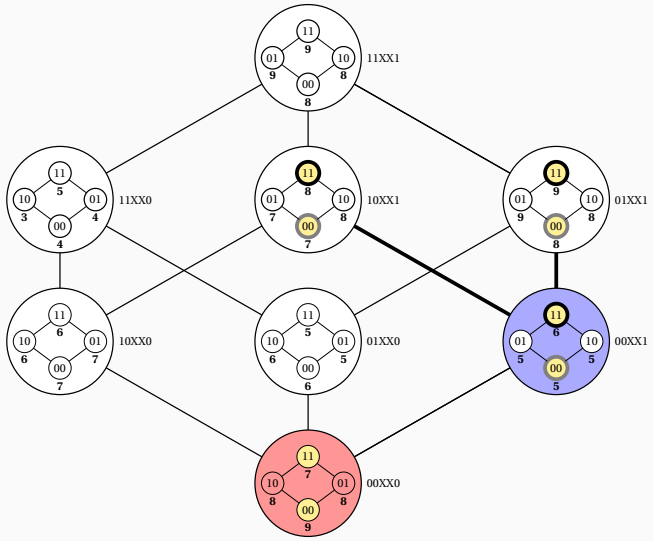


Simulação de execução

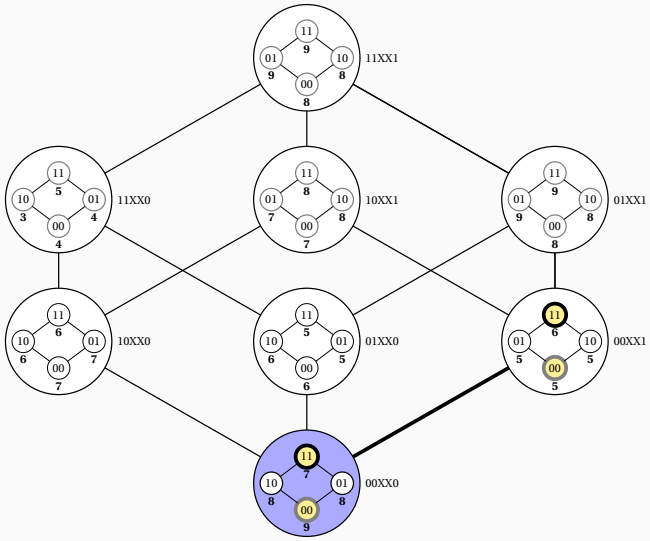


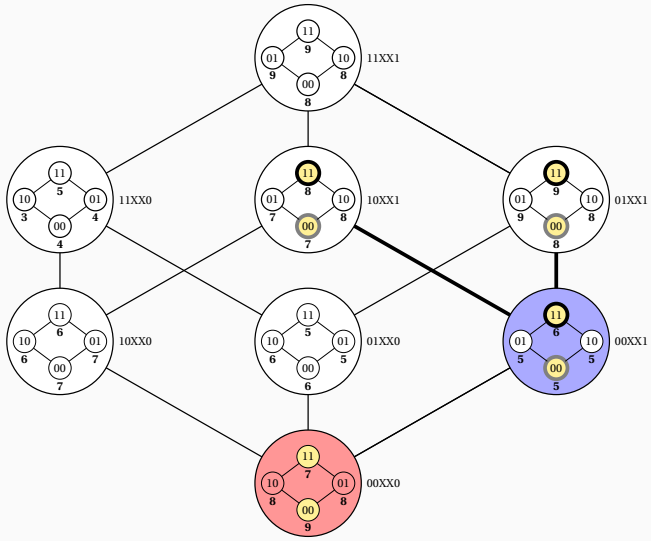
Simulação de execução

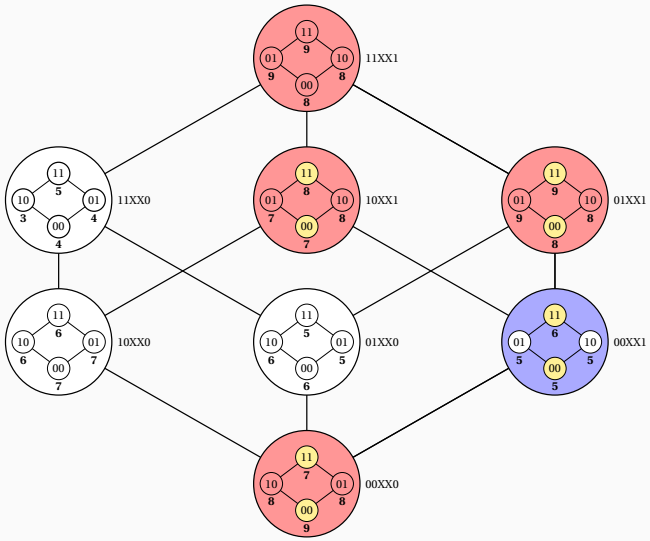




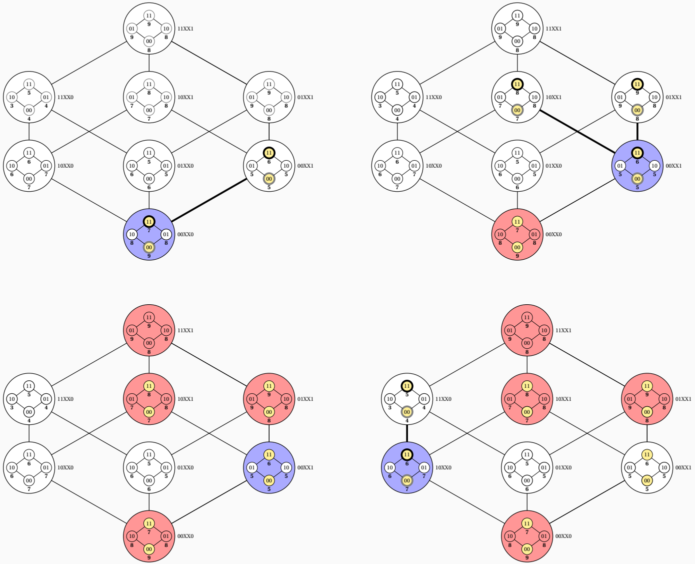
Simulação de execução



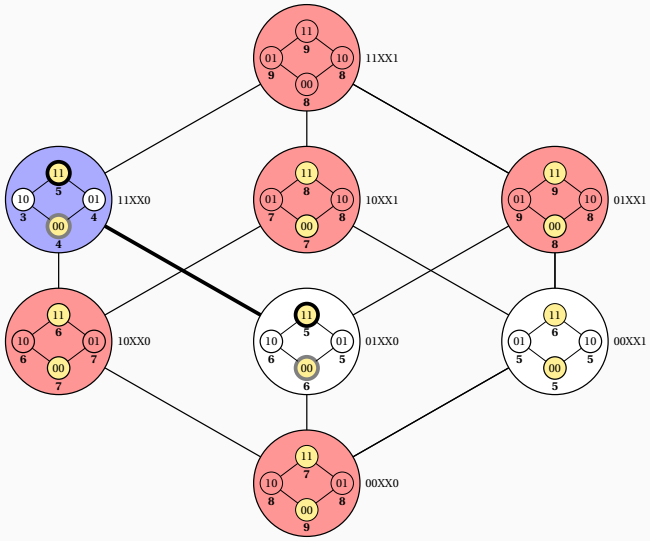




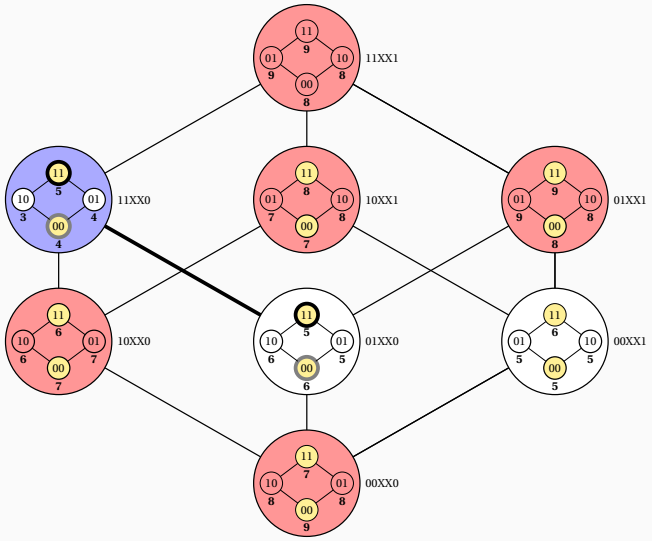
Simulação de execução

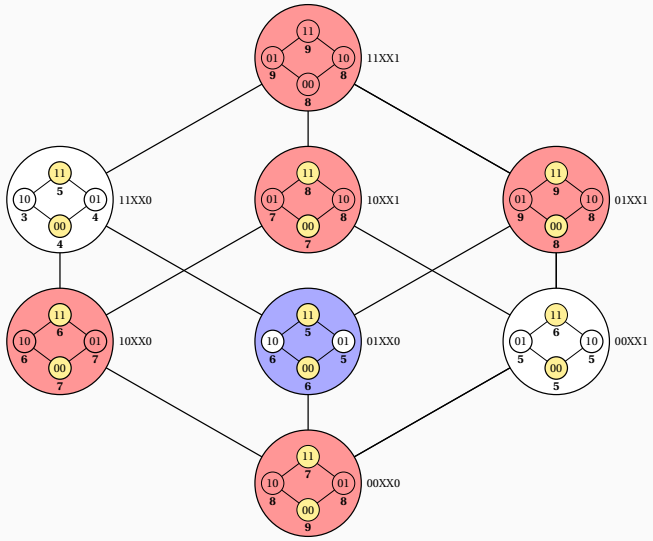


Simulação de execução

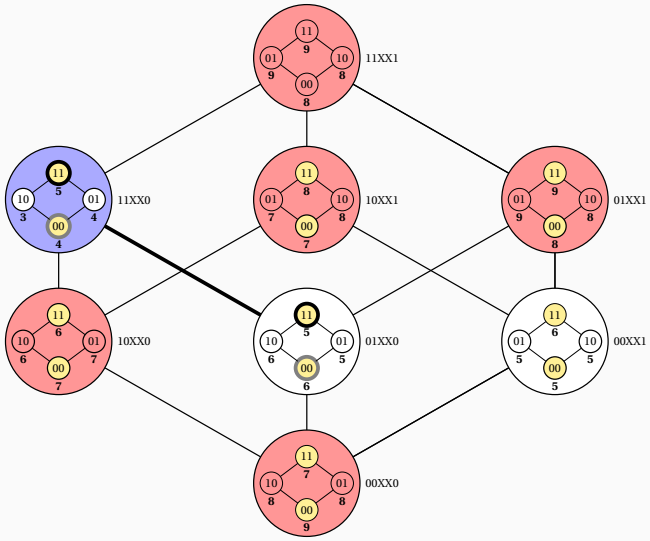


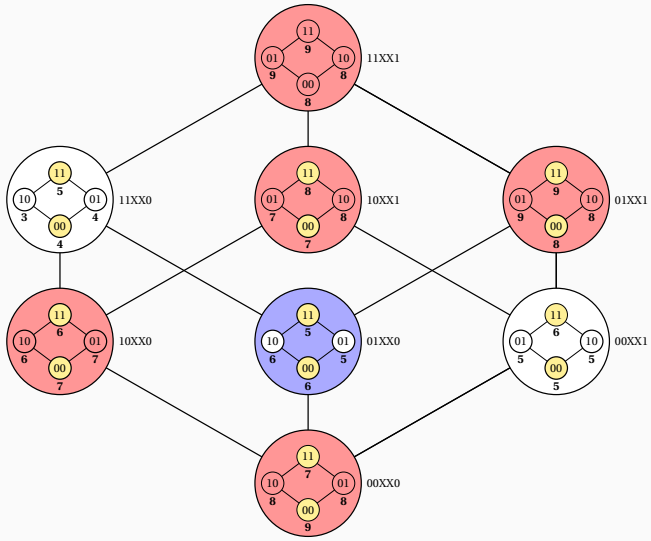
Simulação de execução

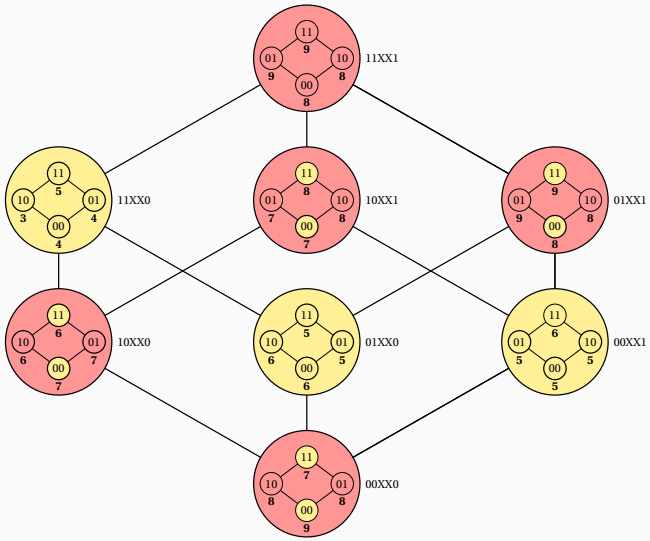




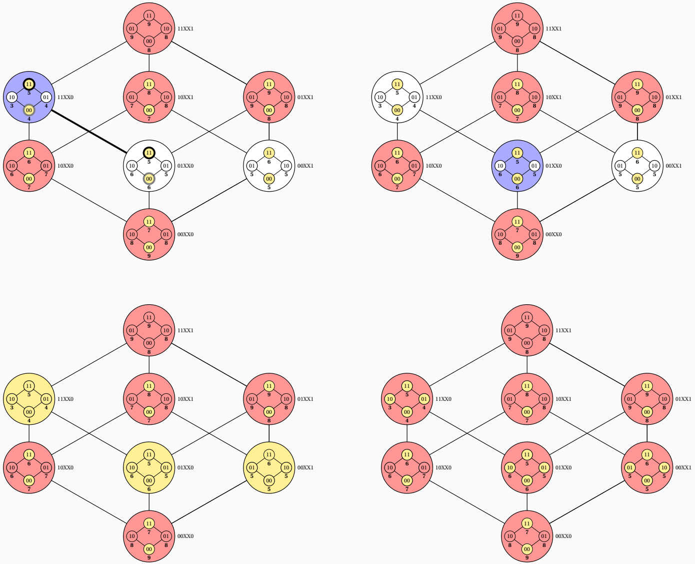
Simulação de execução







Simulação de execução



Parâmetros de funcionamento

O PUCS tem parâmetros que controlam seu funcionamento:

Parâmetros de funcionamento

O PUCS tem parâmetros que controlam seu funcionamento:

- p controla a quantidade de variáveis fixas;

Parâmetros de funcionamento

O PUCS tem parâmetros que controlam seu funcionamento:

- p controla a quantidade de variáveis fixas;
- l controla a quantidade de chamadas recursivas do algoritmo;

Parâmetros de funcionamento

O PUCS tem parâmetros que controlam seu funcionamento:

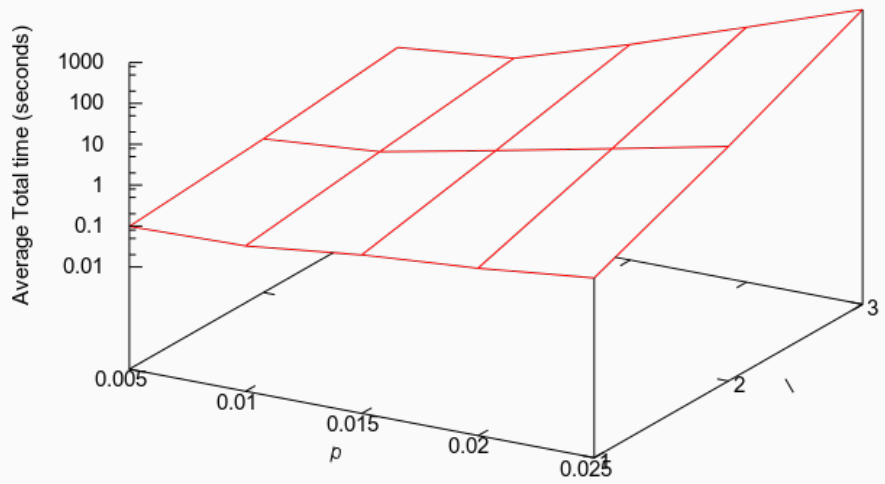
- p controla a quantidade de variáveis fixas;
- l controla a quantidade de chamadas recursivas do algoritmo;
- um algoritmo base que deve resolver as partes.

Parâmetros de funcionamento

Os parâmetros p e l influenciam no tempo de execução do algoritmo.

Parâmetros de funcionamento

Os parâmetros p e l influenciam no tempo de execução do algoritmo.



Quanto maior o valor dos parâmetros, maior o tempo de execução.

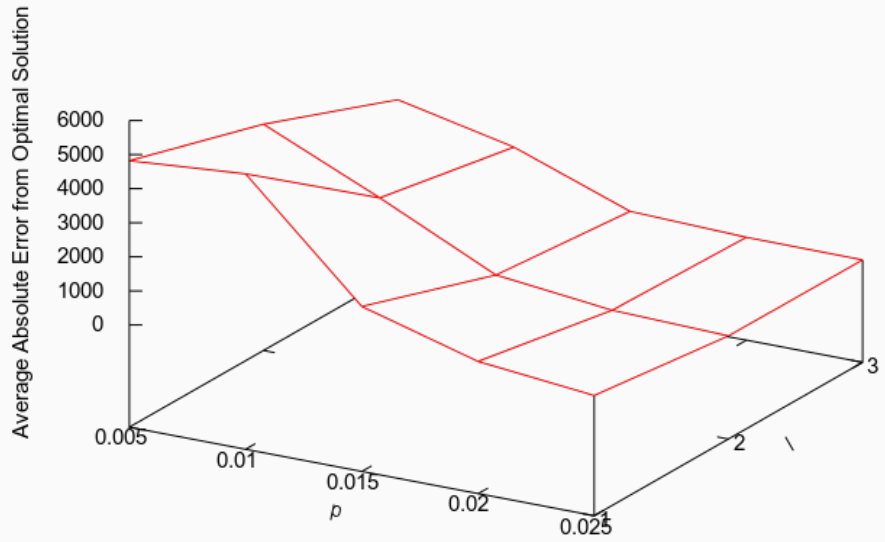
Parâmetros de funcionamento

Quando o algoritmo base utilizado é ótimo, então o PUCS também é ótimo.

Parâmetros de funcionamento

Quando o algoritmo base utilizado é ótimo, então o PUCS também é ótimo.

Caso contrário, o PUCS torna-se um heurística em que os parâmetros p e l influenciam na qualidade da solução.



Resultados do PUCS

Em instâncias pequenas, usamos parâmetros que deixam o algoritmo ótimo.

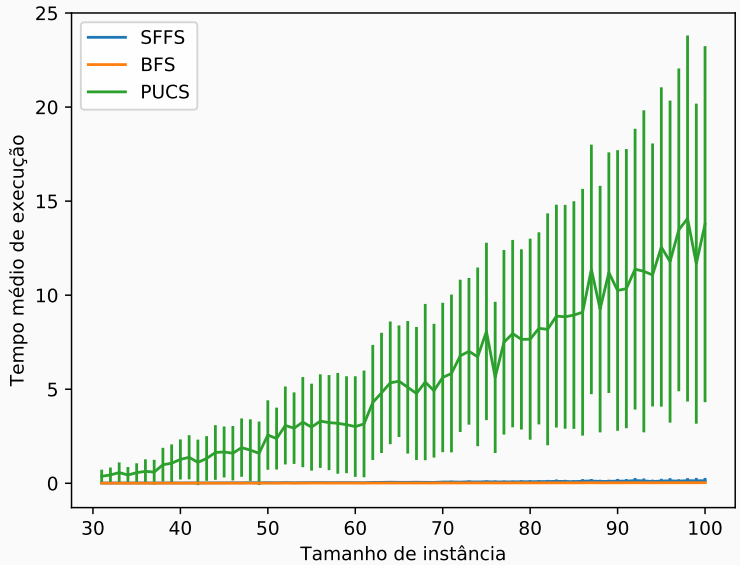
Instance		Total time (sec)			
$\|S\|$	$2^{\|S\|}$	UBB	PFS	UBB-PFS	PUCS
18	262144	**0.319 ± 0.228**	1.512 ± 0.764	0.751 ± 0.338	0.680 ± 0.592
19	524288	**0.684 ± 0.464**	2.875 ± 1.554	1.387 ± 0.707	1.492 ± 1.323
20	1048576	**1.249 ± 0.975**	5.295 ± 3.509	2.594 ± 1.569	2.701 ± 2.908
21	2097152	**2.671 ± 1.948**	11.136 ± 6.947	5.460 ± 3.392	6.118 ± 5.961
22	4194304	**5.420 ± 4.202**	19.825 ± 14.519	9.709 ± 7.319	11.729 ± 11.613

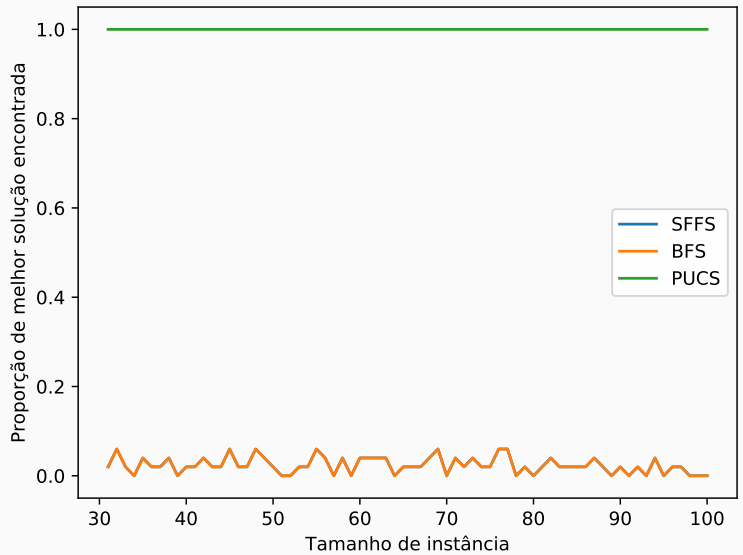
Resultados do PUCS

Instance		# Calls of cost function			
$\|S\|$	$2^{\|S\|}$	UBB	PFS	UBB-PFS	PUCS
16	65536	43529.6 ± 25318.9	26447.0 ± 13446.1	**28783.6 ± 12934.2**	26001.3 ± 21699.6
17	131072	65301.0 ± 56215.8	49694.5 ± 27621.8	**51032.5 ± 29984.3**	50145.2 ± 46799.0
18	262144	145594.5 ± 103597.8	105603.1 ± 52652.2	**110538.0 ± 51589.7**	111296.6 ± 84922.4
19	524288	313096.0 ± 209913.1	194572.5 ± 104802.3	**204604.5 ± 100305.4**	233717.7 ± 186182.0
20	1048576	578319.0 ± 445912.2	340052.5 ± 221271.6	**362007.0 ± 207411.2**	387082.0 ± 389417.4

Resultados do PUCS

Em experimentos sub-ótimos, comparamos o PUCS com as heurísticas Sequential Forward Floating Search (SFFS) e Best-First Search (BFS).





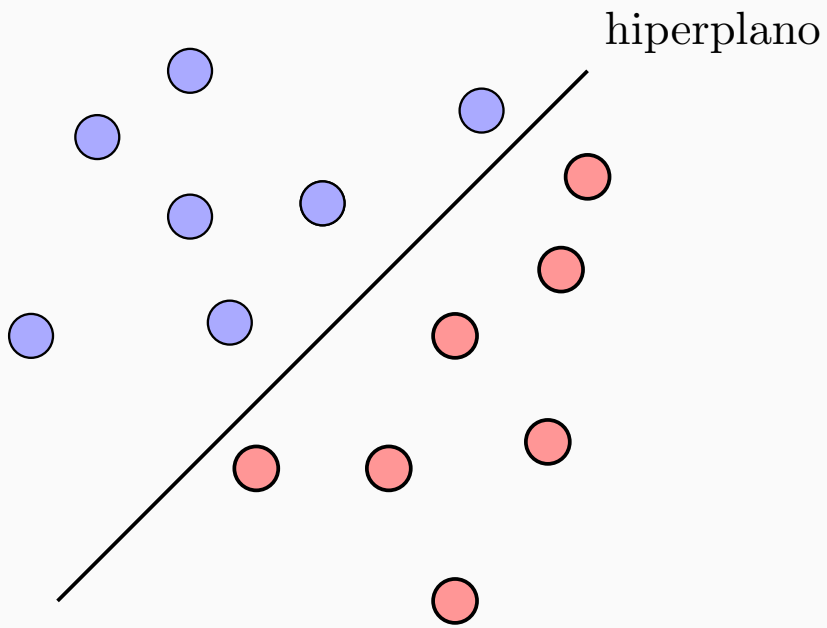
Aplicações instâncias reais

Seleção de características em seleção de modelos

Aplicamos seleção de características na construção de modelos de aprendizado para conjuntos de dados do UCI Machine Learning Repository.

Modelos de aprendizado

Os modelos que utilizamos para o treinamento e classificação são do tipo Support Vector Machine.



Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine
- Thoracic Surgery

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine
- Thoracic Surgery
- Zoo

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine
- Thoracic Surgery
- Zoo
- Breast Cancer

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine
- Thoracic Surgery
- Zoo
- Breast Cancer
- Lung Cancer

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine
- Thoracic Surgery
- Zoo
- Breast Cancer
- Lung Cancer
- Promoters

Conjuntos de dados testados

Fizemos o treinamento e validação de modelos de aprendizado nos seguintes conjuntos de dados:

- Iris
- Wine
- Thoracic Surgery
- Zoo
- Breast Cancer
- Lung Cancer
- Promoters

Validação cruzada

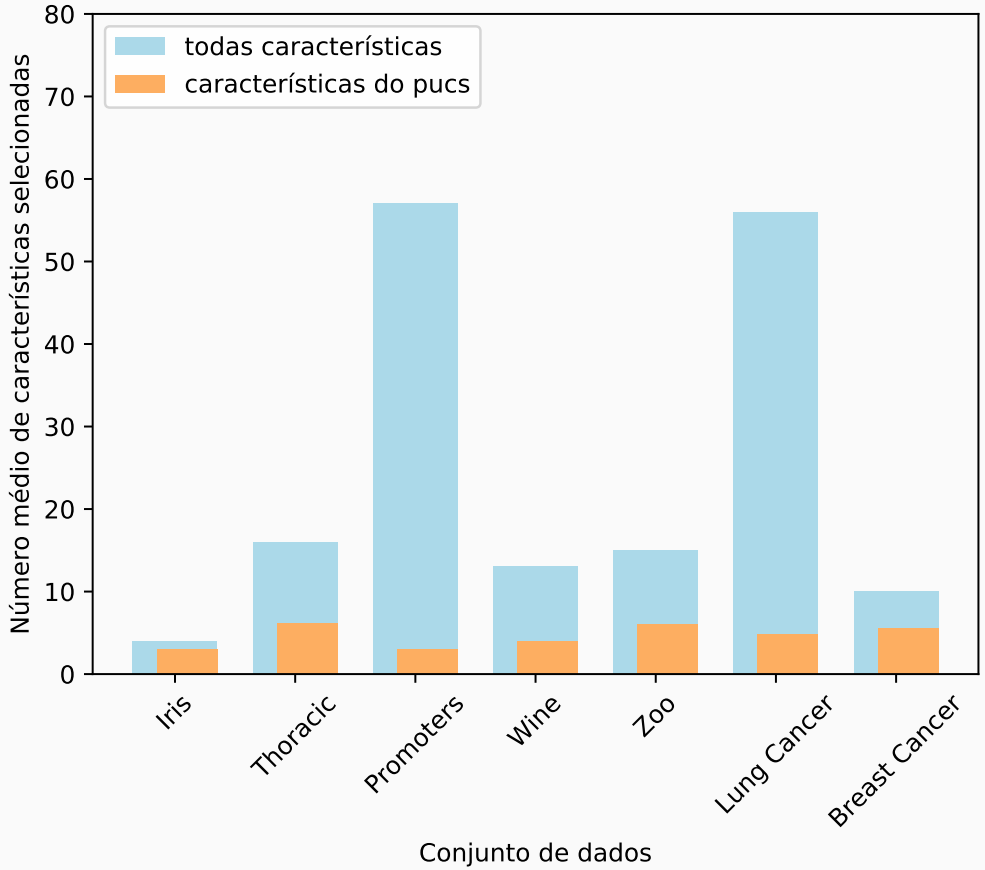
Para avaliar a seleção de características fizemos a validação cruzada de modelos com todas características e a de modelos apenas com características selecionadas.

Resultados

O número de características selecionadas é, de fato, menor do que o conjunto inteiro.

Resultados

O número de características selecionadas é, de fato, menor do que o conjunto inteiro.

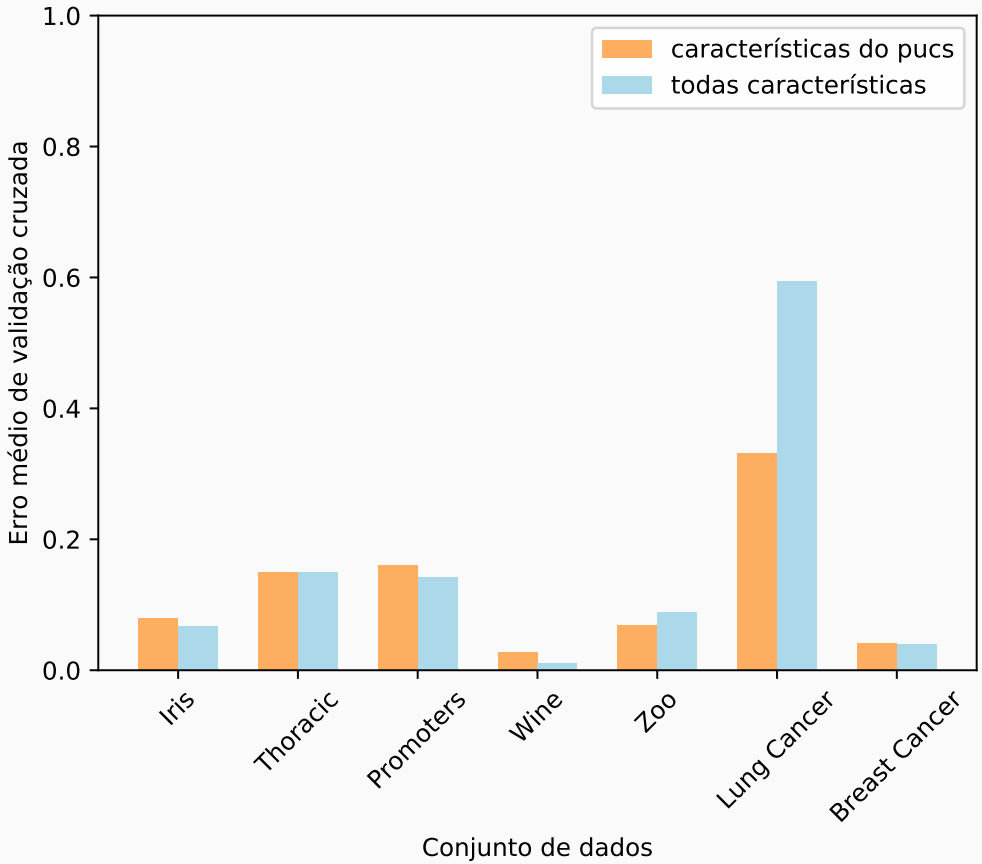


Resultados

Além disso, a qualidade dos modelos não é afetada.

Resultados

Além disso, a qualidade dos modelos não é afetada.



Revisão

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

- Modificações no PFS.
  - Escolha de raízes.
  - Estrutura de dados para armazenamento de raízes.

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

- Modificações no `PFS`.
  - Escolha de raízes.
  - Estrutura de dados para armazenamento de raízes.
- Uma paralelização do `PFS`.

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

- Modificações no `PFS`.
  - Escolha de raízes.
  - Estrutura de dados para armazenamento de raízes.
- Uma paralelização do `PFS`.
- O algoritmo `UBB-PFS`.

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

- Modificações no `PFS`.
  - Escolha de raízes.
  - Estrutura de dados para armazenamento de raízes.
- Uma paralelização do `PFS`.
- O algoritmo `UBB-PFS`.
- O algoritmo `PUCS`.

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

- Modificações no `PFS`.
  - Escolha de raízes.
  - Estrutura de dados para armazenamento de raízes.
- Uma paralelização do `PFS`.
- O algoritmo `UBB-PFS`.
- O algoritmo `PUCS`.
- Testes com instâncias reais.

Revisão dos resultados deste trabalho

Ao longo deste trabalho apresentamos

- Modificações no `PFS`.
  - Escolha de raízes.
  - Estrutura de dados para armazenamento de raízes.
- Uma paralelização do `PFS`.
- O algoritmo `UBB-PFS`.
- O algoritmo `PUCS`.
- Testes com instâncias reais.