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.

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

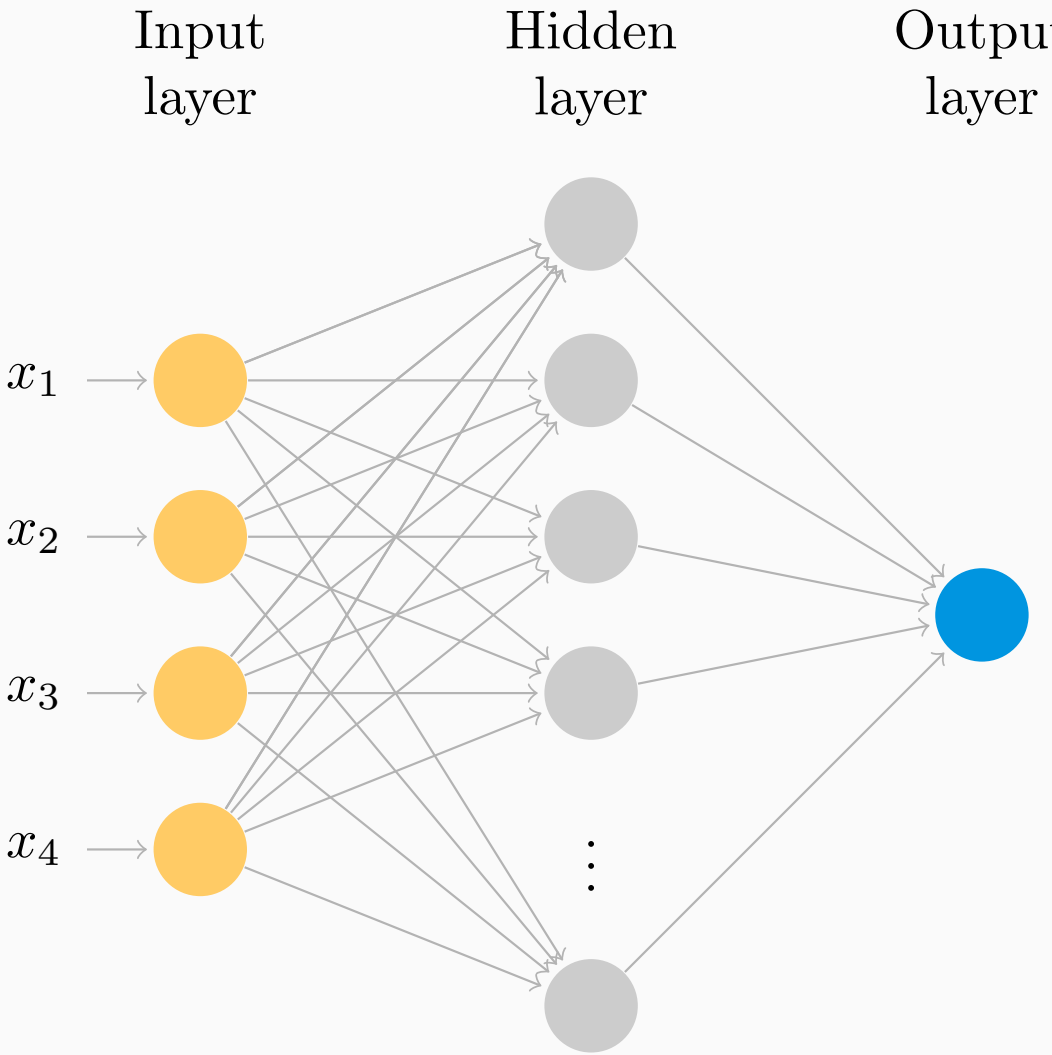
Modelos computacionais

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.

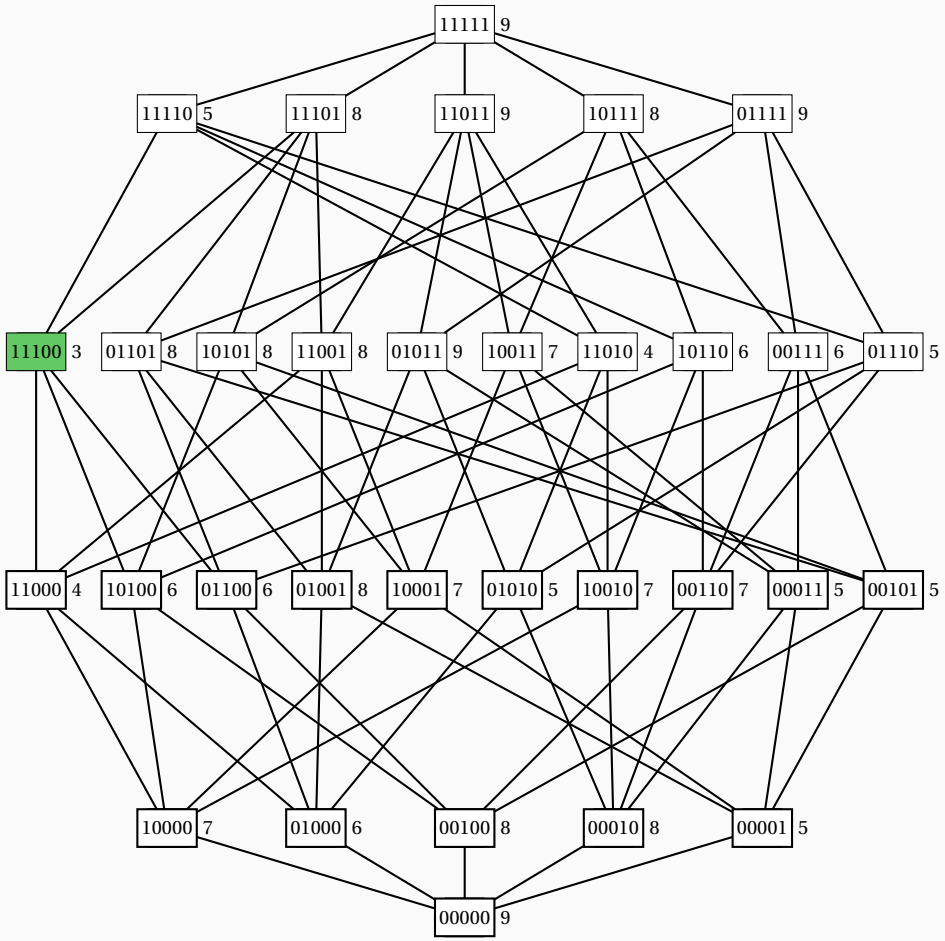
O problema de seleção de características

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 $(\mathcal{P}(S), \subseteq)$.

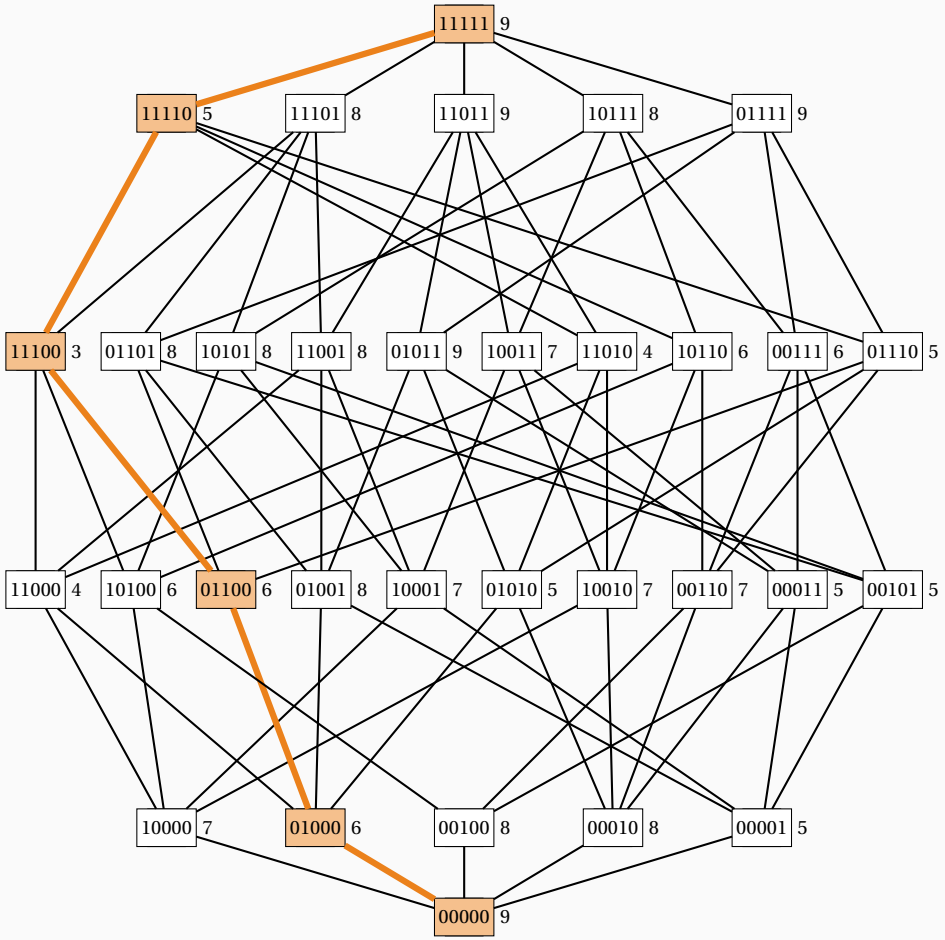


O espaço de busca

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



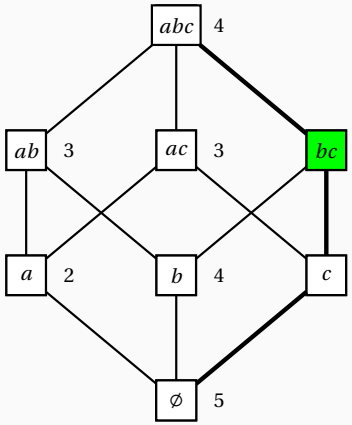
A função de custo

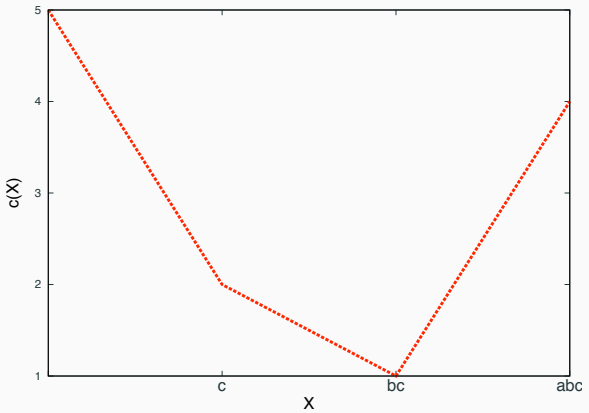
A função de custo c deve refletir a qualidade de um conjunto de características X a ser usado no modelo.

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

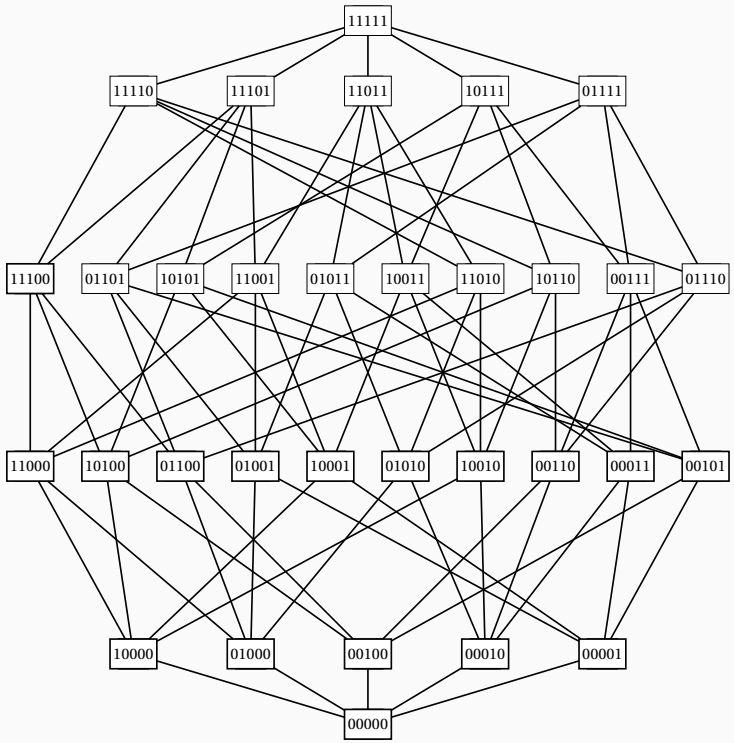
Definição (Problema U-Curve)

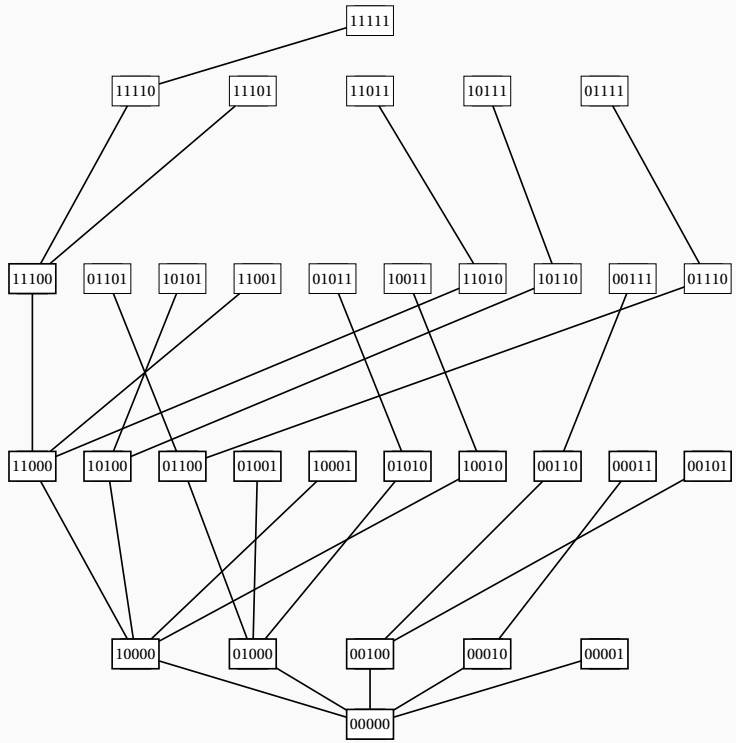
Dados um conjunto finito e não-vazio S e uma função de custo c decomponível em curvas U, encontrar um subconjunto $X \in \mathcal{P}(S)$ tal que $c(X)$ é mínimo.

Algoritmos baseados em florestas

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

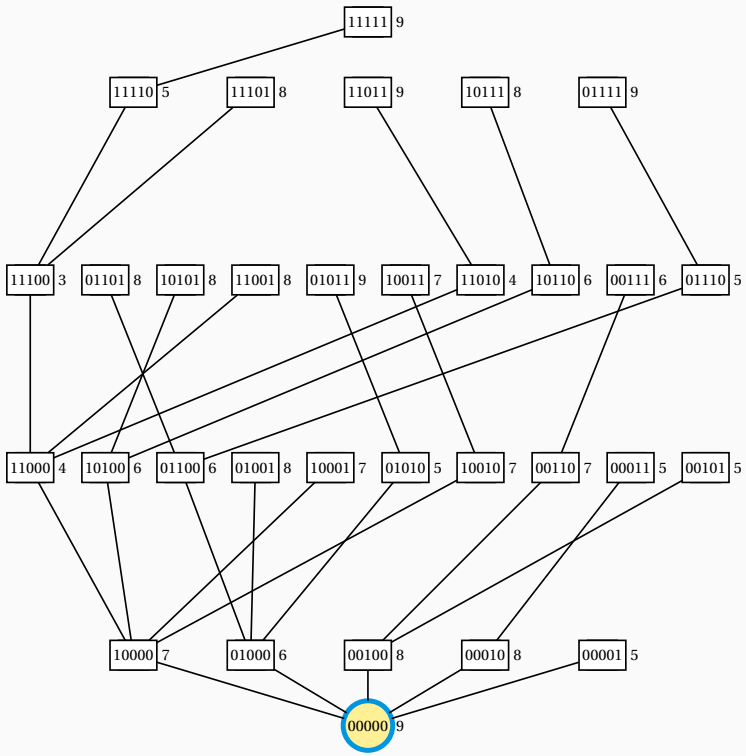
O algoritmo `U-Curve-Branch-and-Bound` (UBB) organiza o espaço de busca em uma árvore.





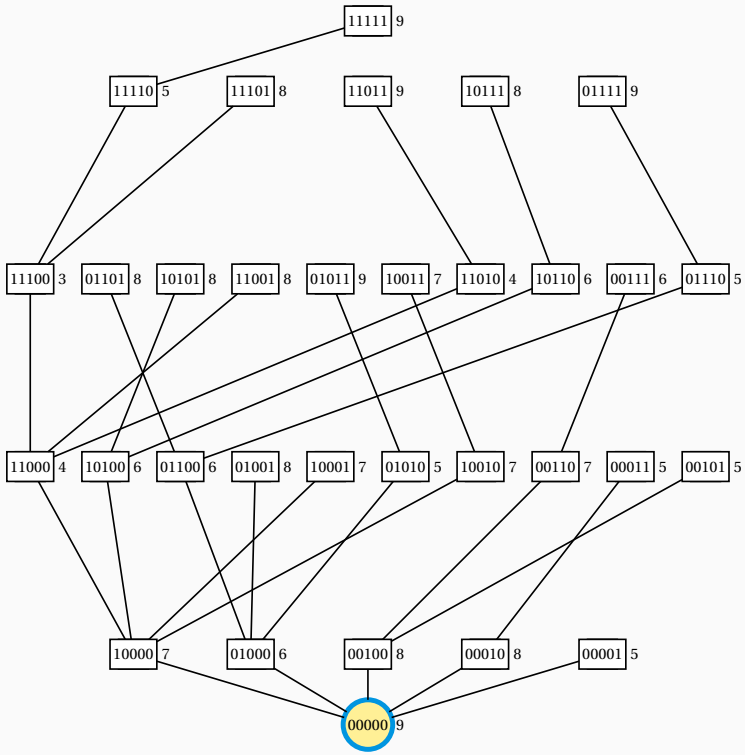
O algoritmo `U-Curve-Branch-and-Bound`

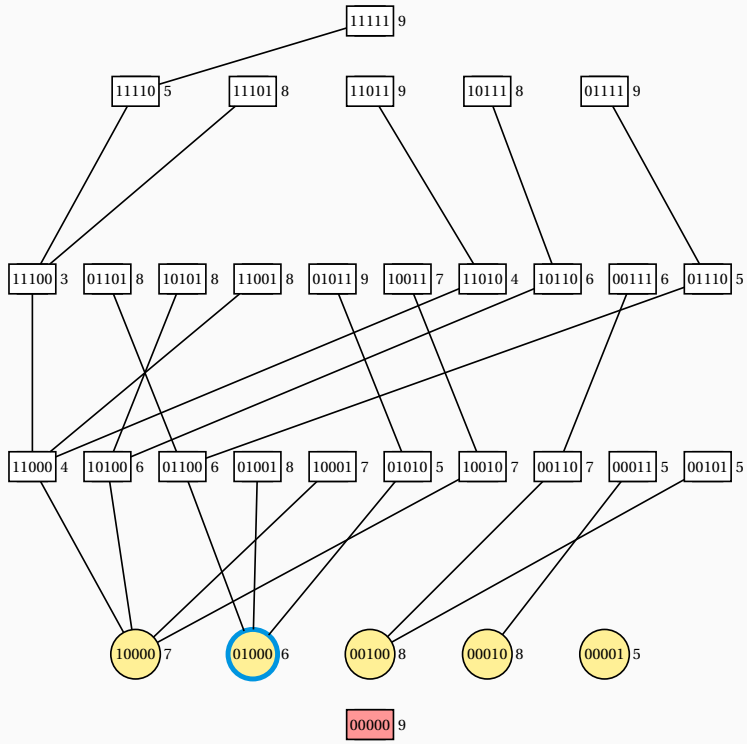
Este algoritmo busca o mínimo global ramificando na árvore como em uma busca em profundidade e faz podas sempre que o custo aumenta.



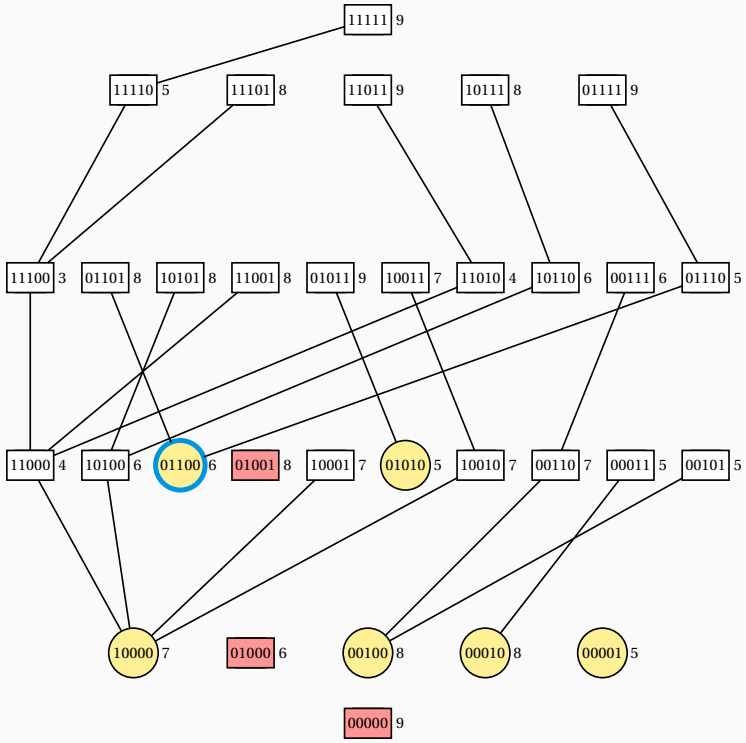
O algoritmo `U-Curve-Branch-and-Bound`

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

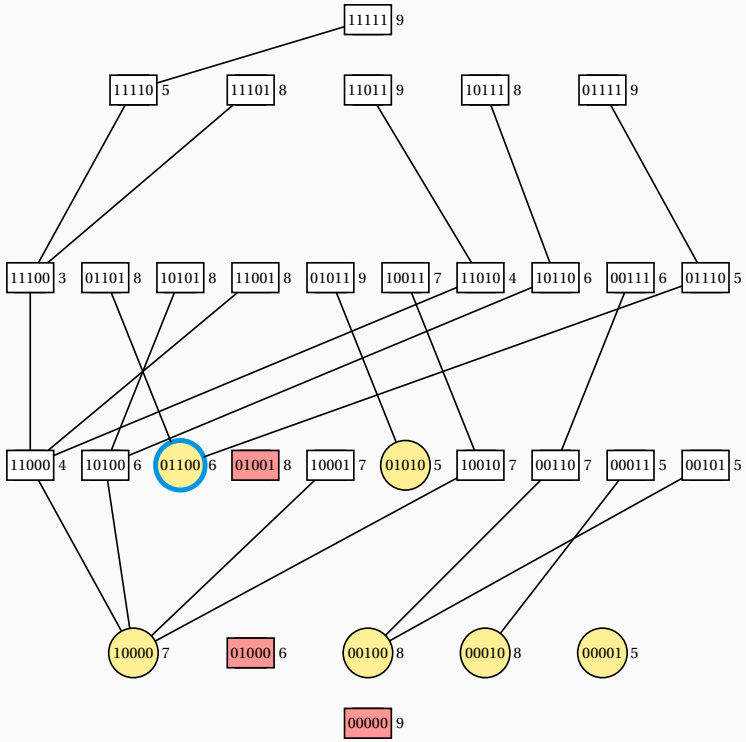


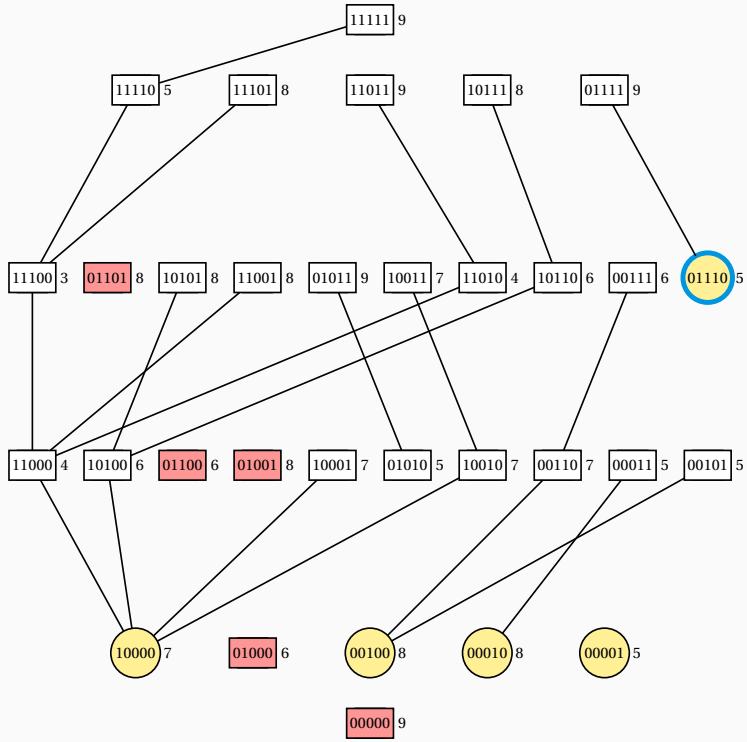


O algoritmo U-Curve-Branch-and-Bound

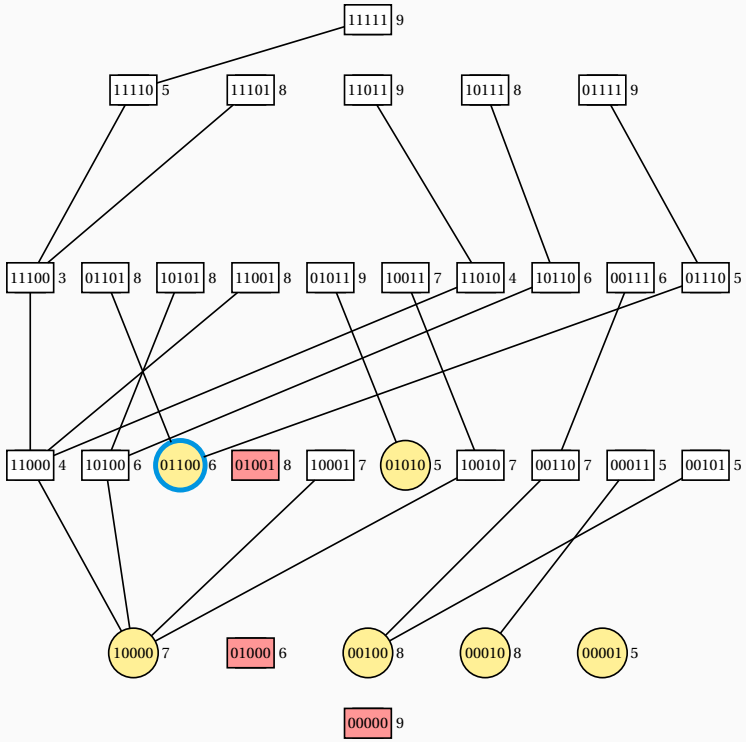


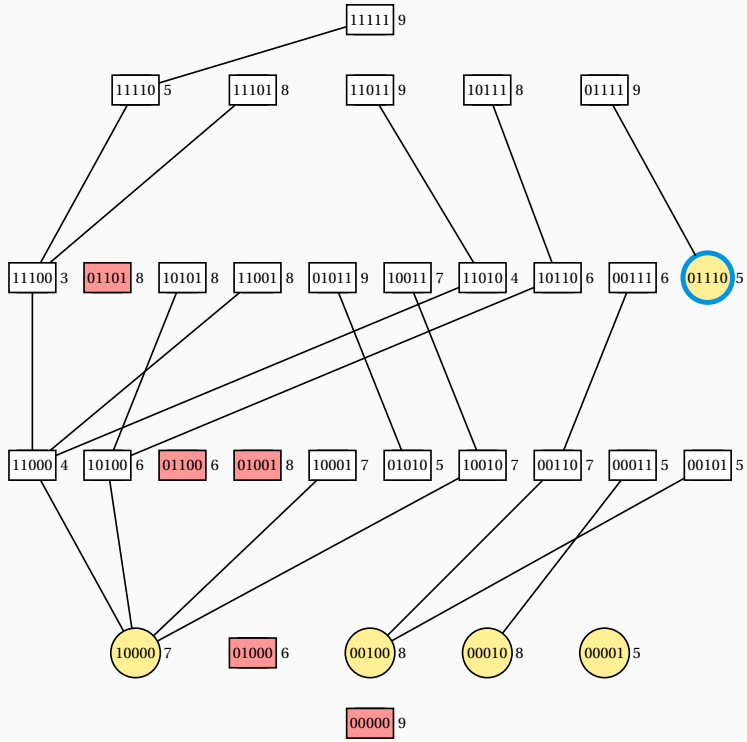
O algoritmo U-Curve-Branch-and-Bound





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.

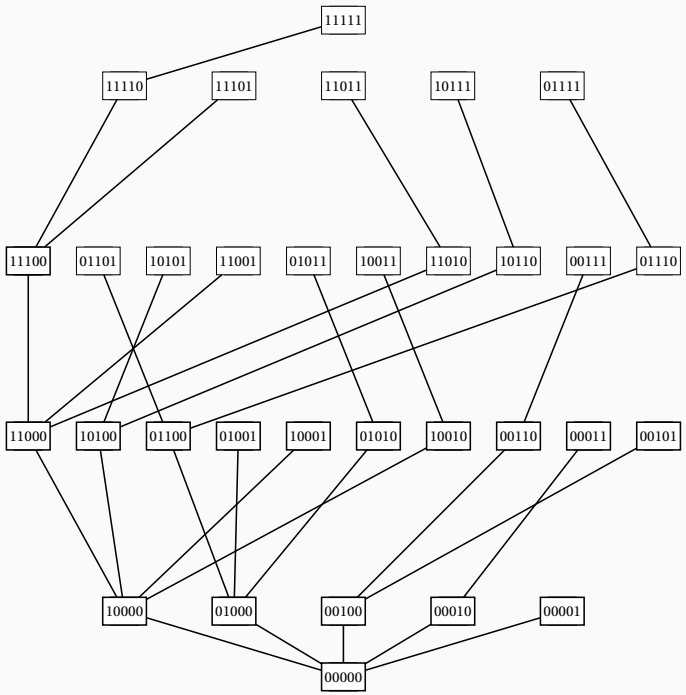
O algoritmo `Poset-Forest-Search`

Solução: percorrer o espaço de busca em duas direções.

O algoritmo `Poset-Forest-Search`

Solução: percorrer o espaço de busca em duas direções.

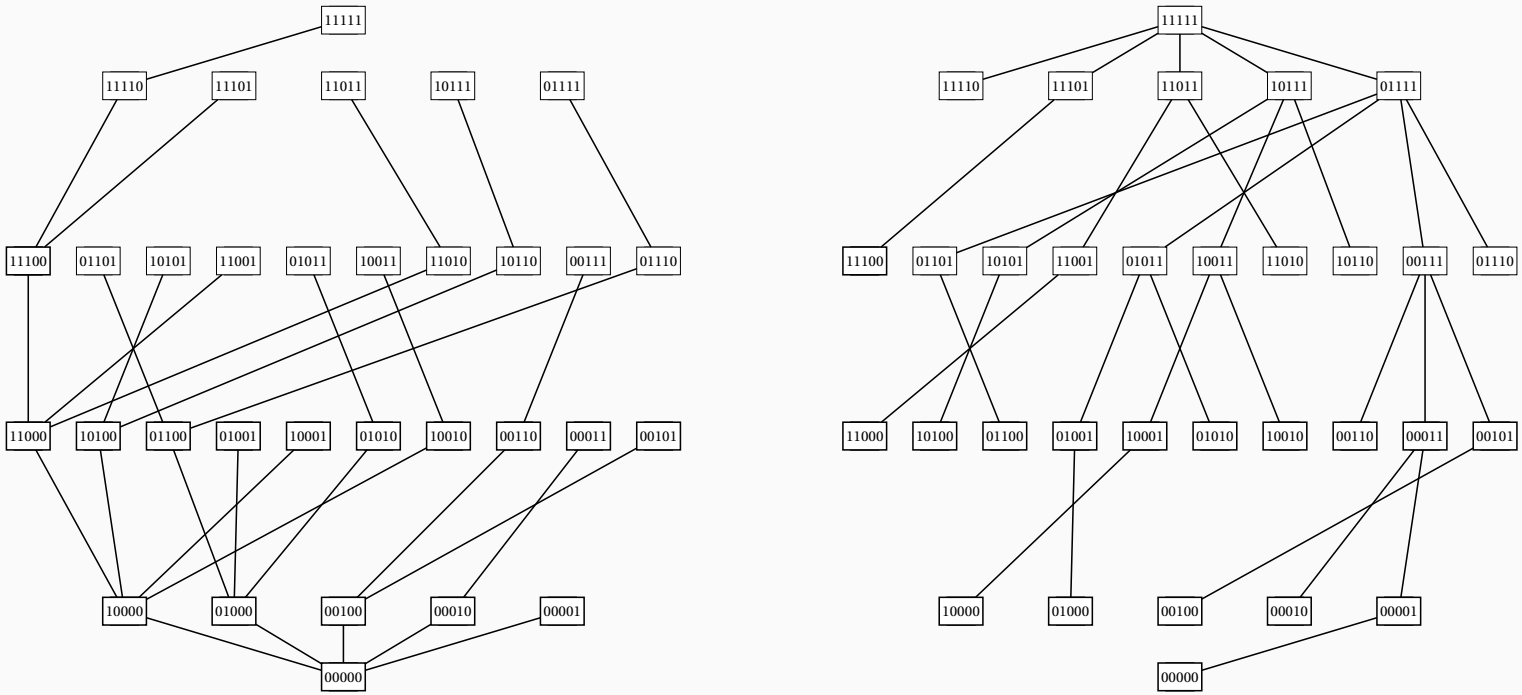
O algoritmo `Poset-Fores-Search` (`PFS`) pode fazer isso porque decompõe o espaço em duas árvores.



O algoritmo `Poset-Forest-Search`

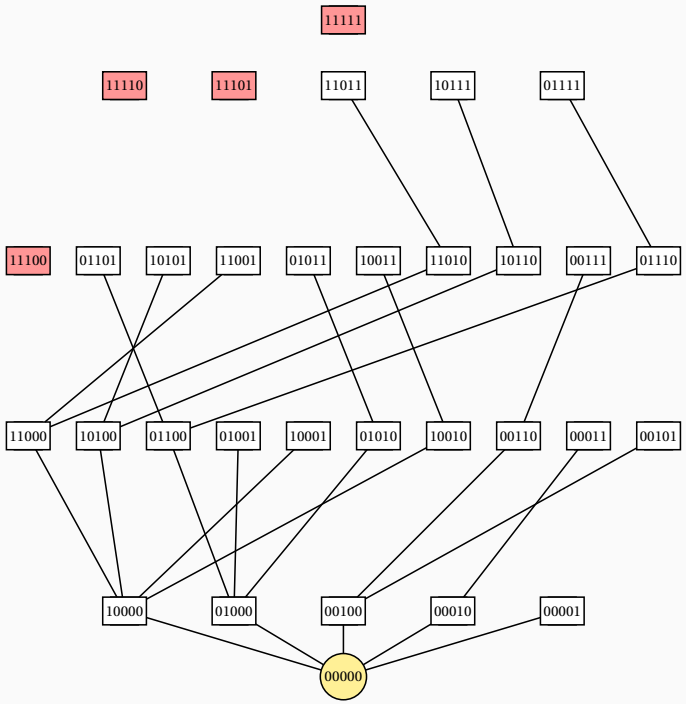
Solução: percorrer o espaço de busca em duas direções.

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



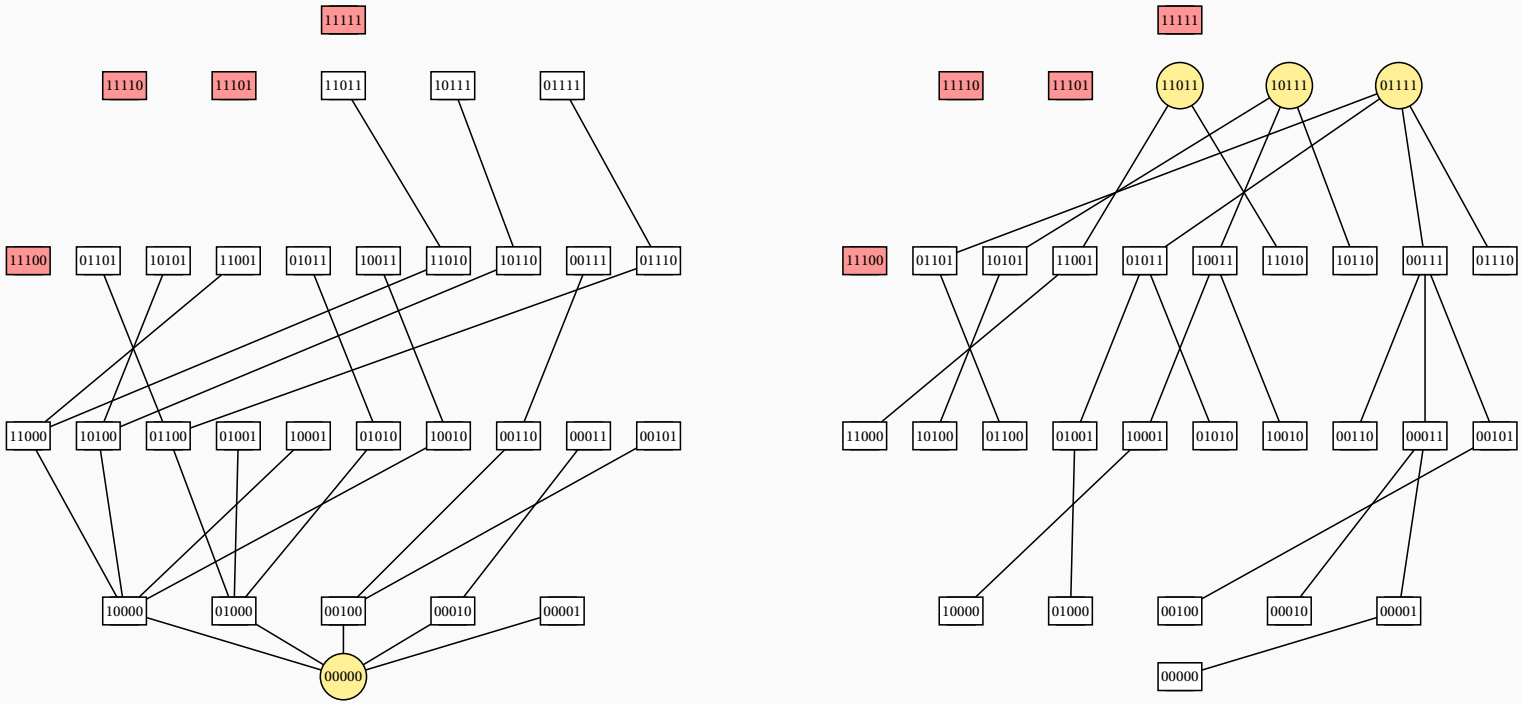
O algoritmo `Poset-Forest-Search`

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:

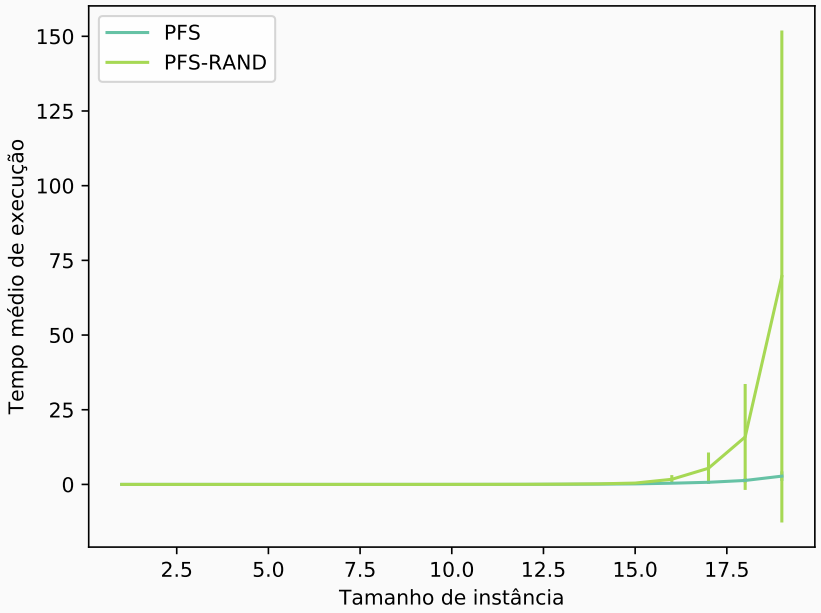
- escolha aleatória e uniforme entre raízes;
- escolha da raiz com maior sub-árvore.

Resultados da mudança de escolha de raízes

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

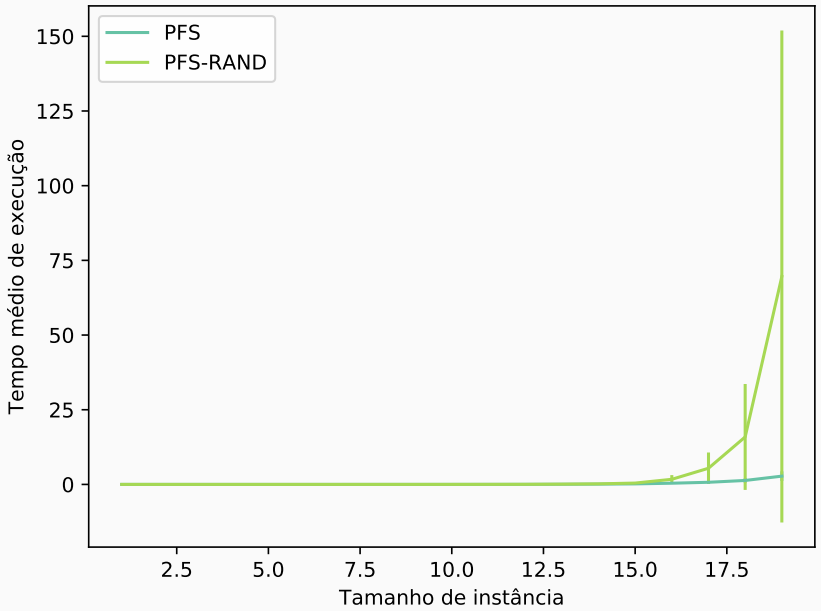
Resultados da mudança de escolha de raízes

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



Resultados da mudança de escolha de raízes

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



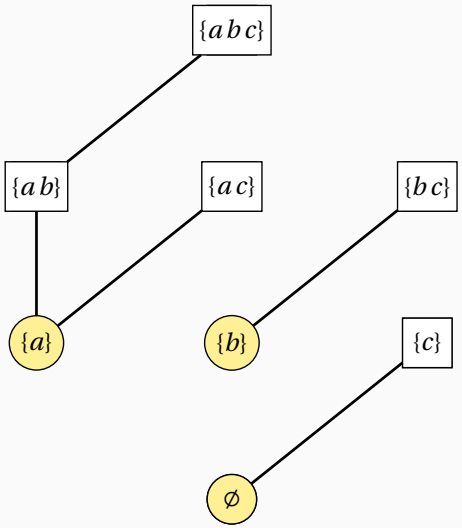
A versão que escolhia a raiz com maior sub-árvore não teve bom desempenho.

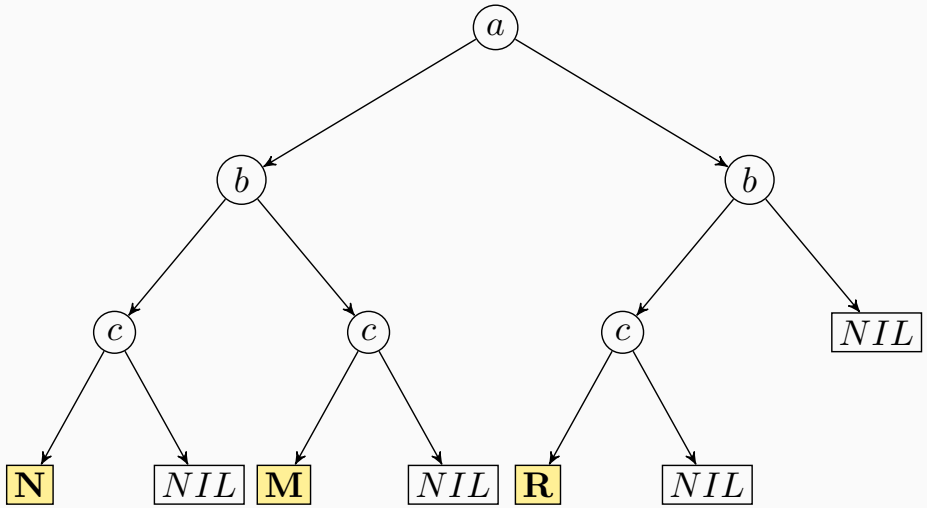
Melhoramentos na estrutura de armazenamento de raízes

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).



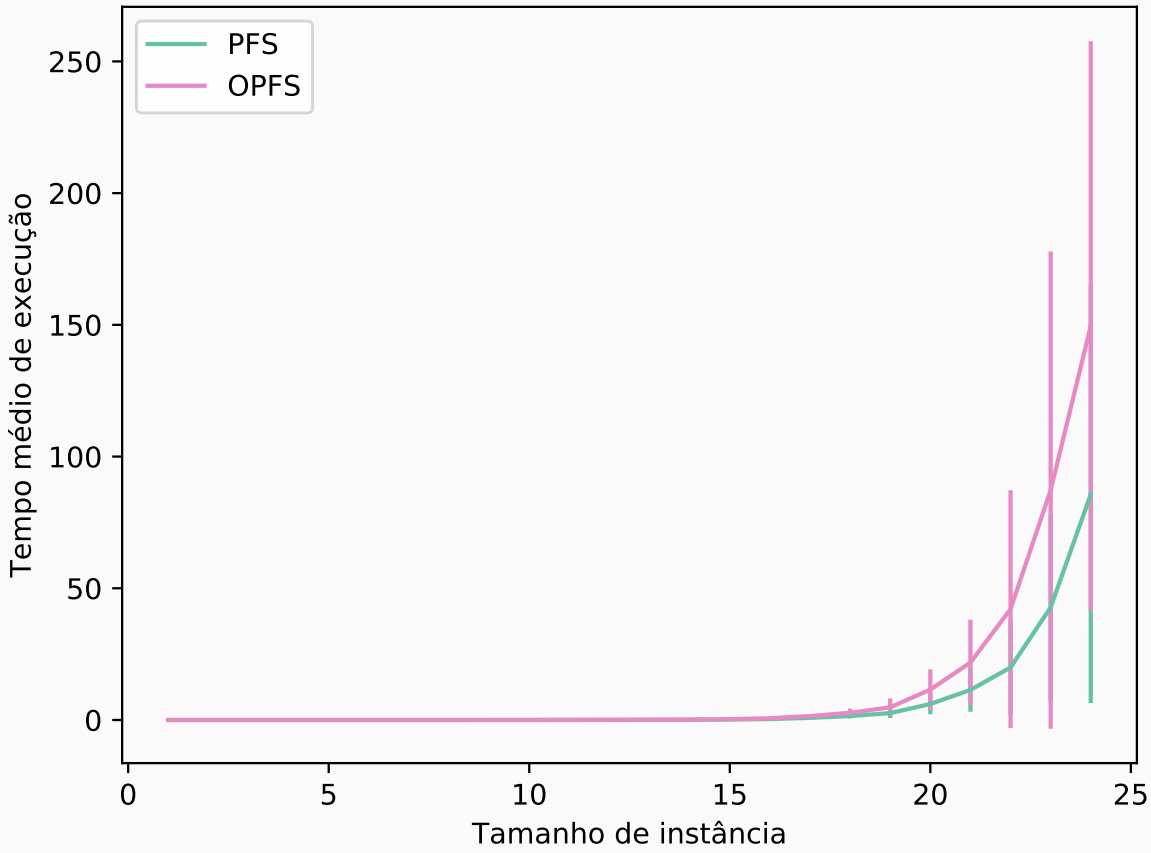


Resultados da mudança de estrutura para armazenamento de raízes

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

Resultados da mudança de estrutura para armazenamento de raízes

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



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:

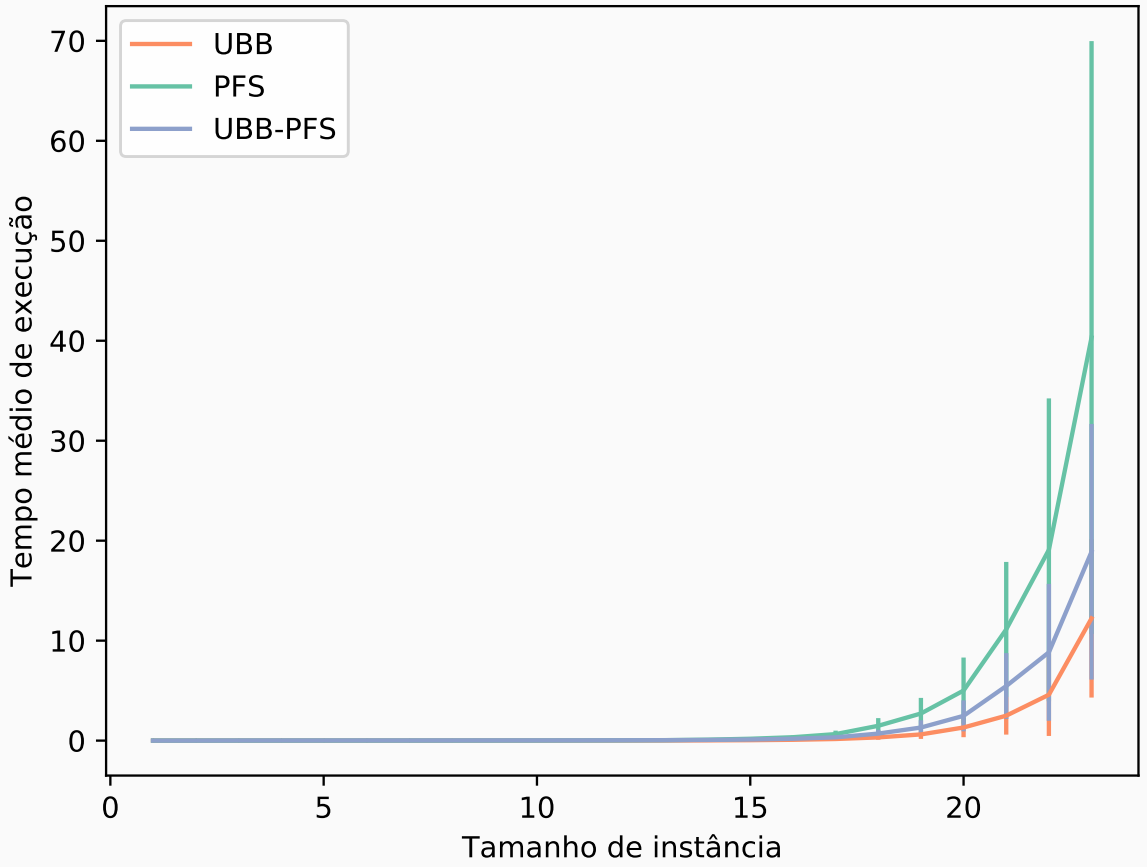
- Percorrimento sequencial: idêntico ao UBB deve criar sub-árvores no espaço enquanto faz uma ramificação do tipo busca em profundidade.
- Solução em paralelo: cada sub-árvore gerada na etapa de ramificação deve ser resolvida por uma chamada do PFS.

Resultados do UBB-PFS

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

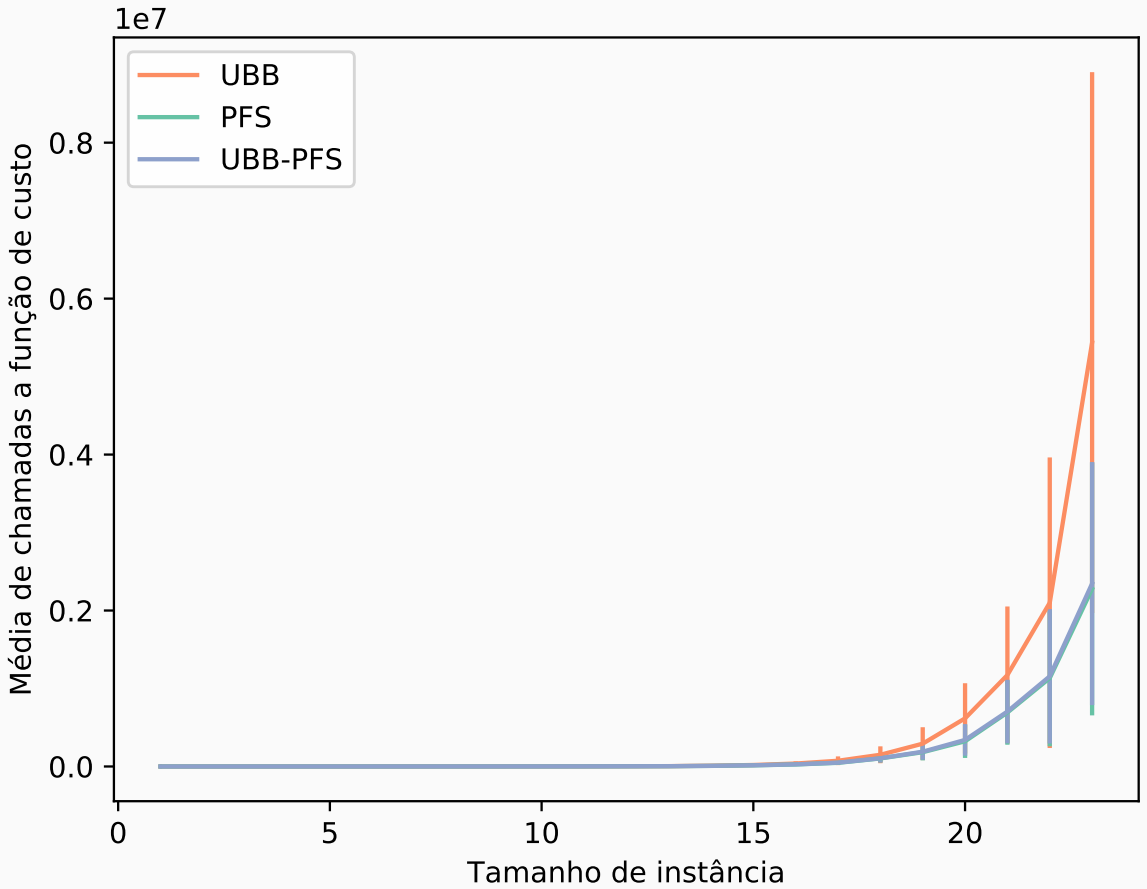
Resultados do UBB-PFS

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



Resultados do UBB-PFS

E computou menos a função custo do que o UBB.



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,

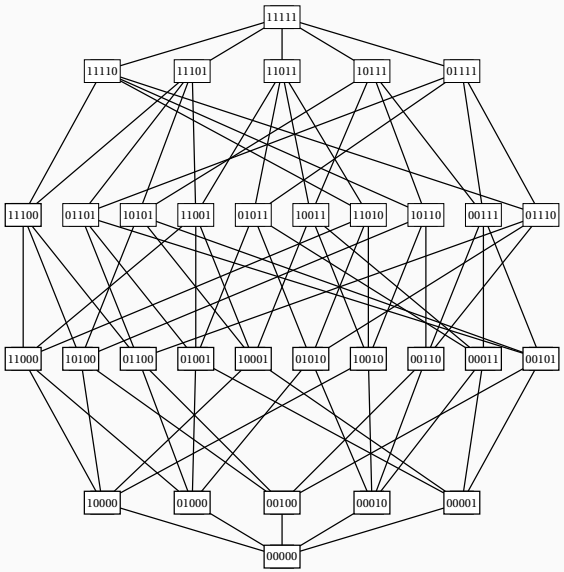
Ideia do `Parallel-U-Curve-Search` (PUCS)

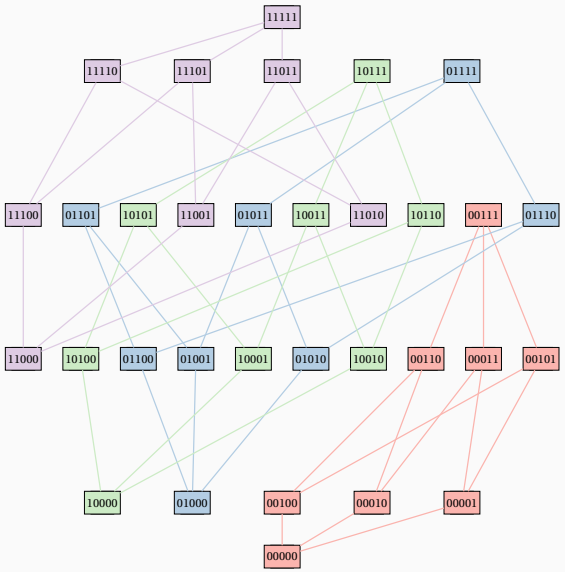
Um algoritmo de fácil paralelização e pouco entrelace de linhas de processamento, e que também distribua o trabalho em partes de tamanho parecido.

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

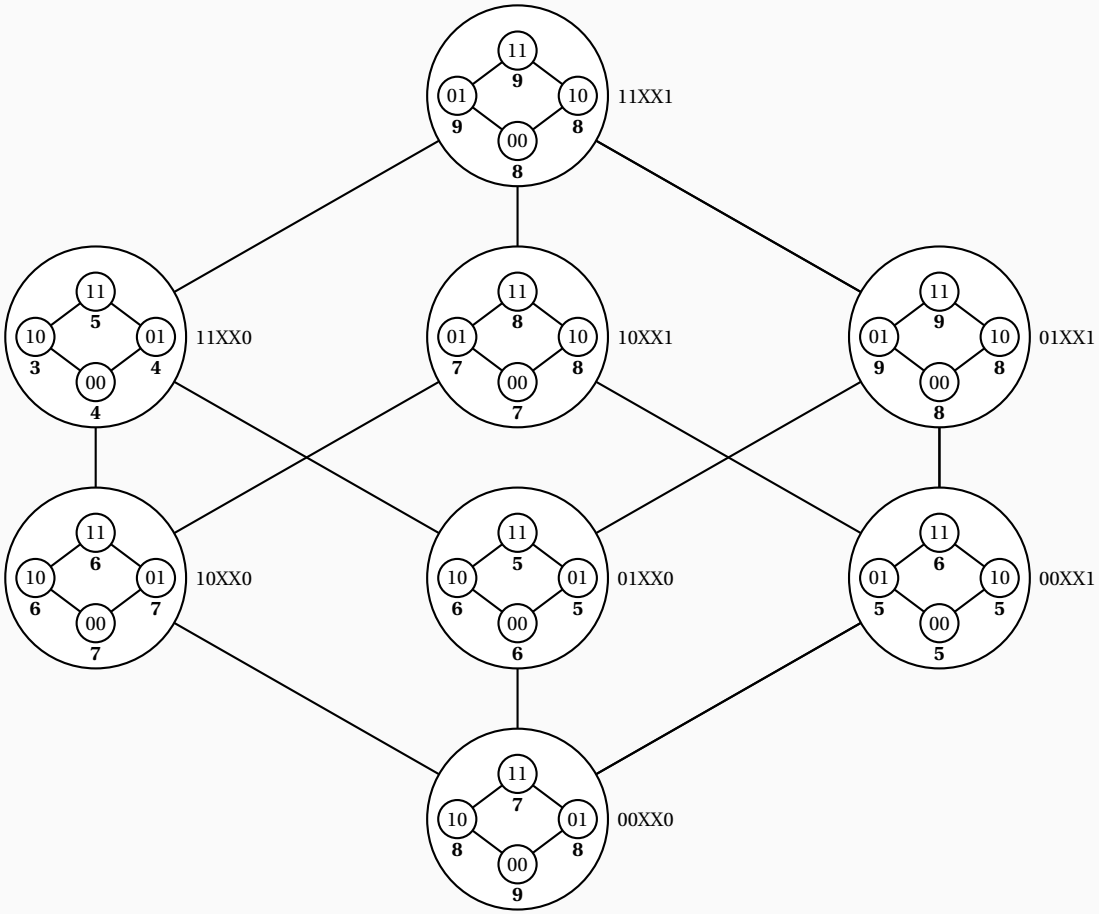
Se representamos cada parte por um subconjunto de características de S', então temos um reticulado Booleano de partes. Chamamos este reticulado de reticulado externo.

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

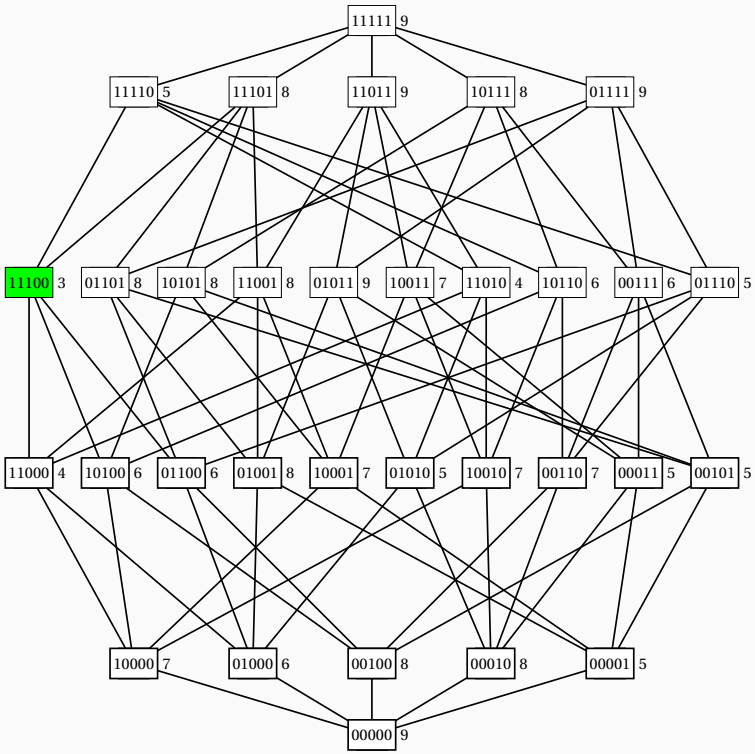


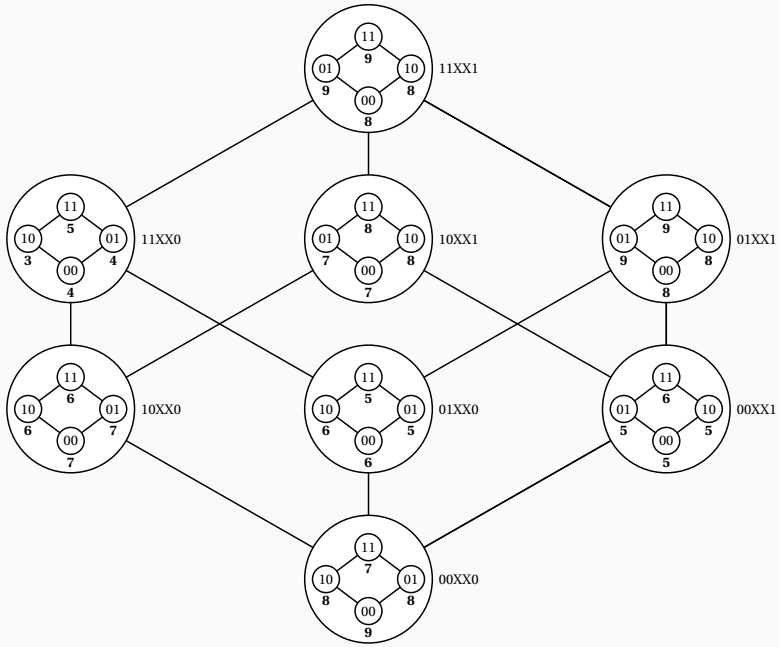
Dinâmica

Podemos resumir a dinâmica deste algoritmo nos passos:

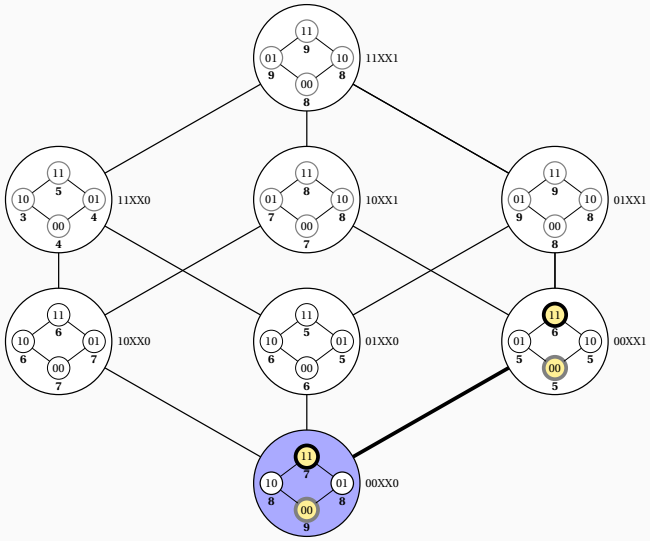
- passeio aleatório no reticulado externo, com podas;
- solução de partes não podadas;
- união de respostas das partes.

Simulação de execução

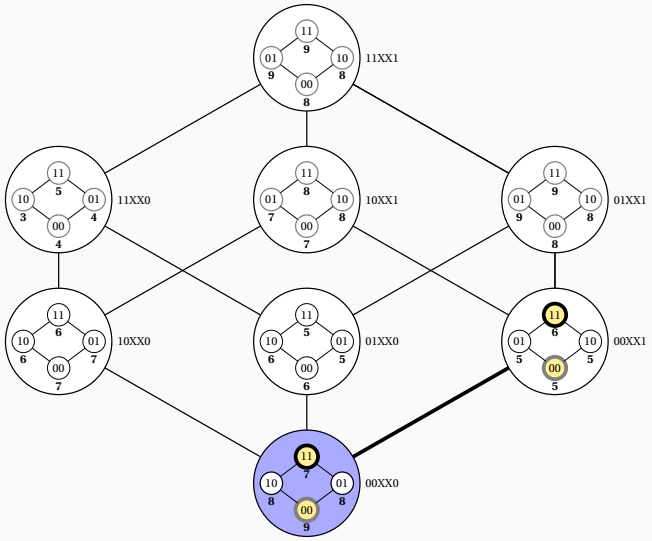


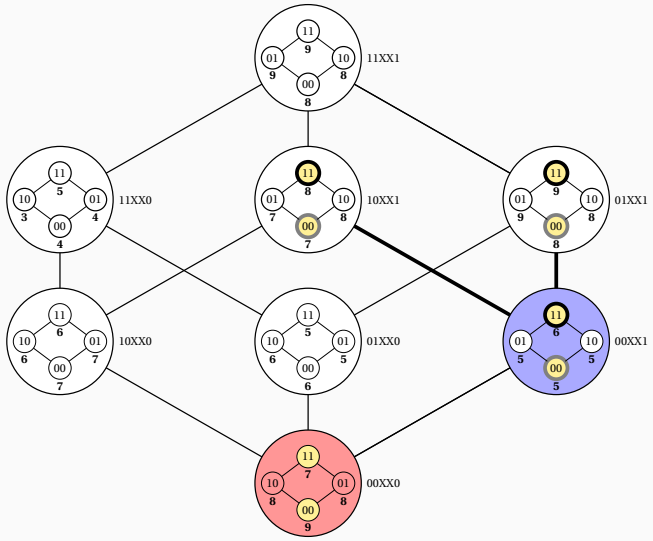


Simulação de execução

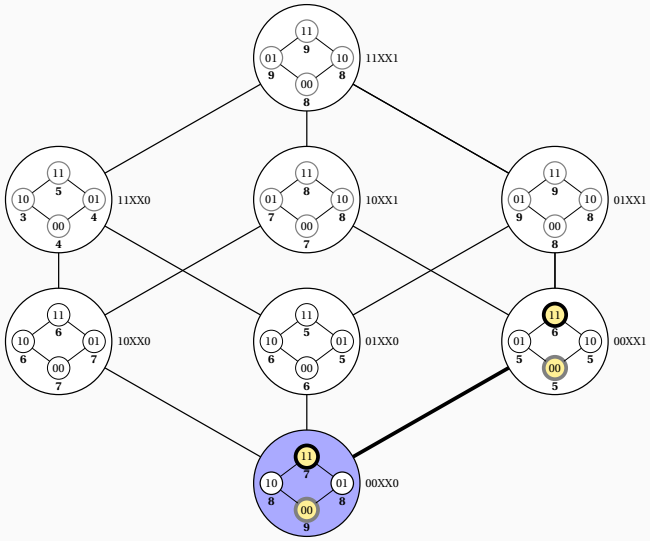


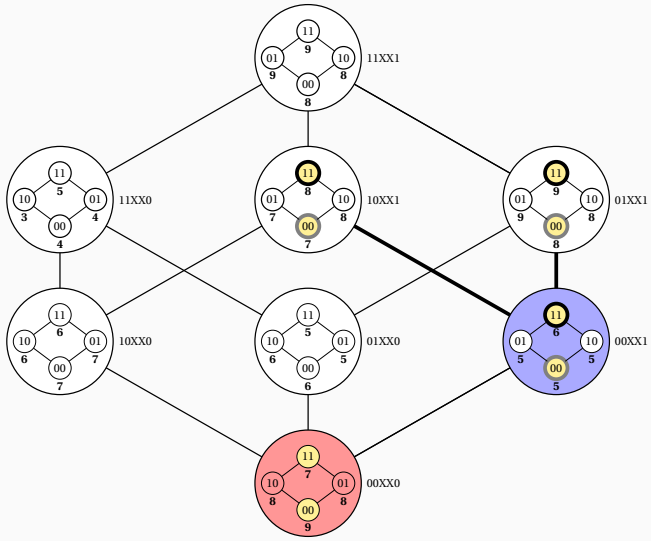
Simulação de execução

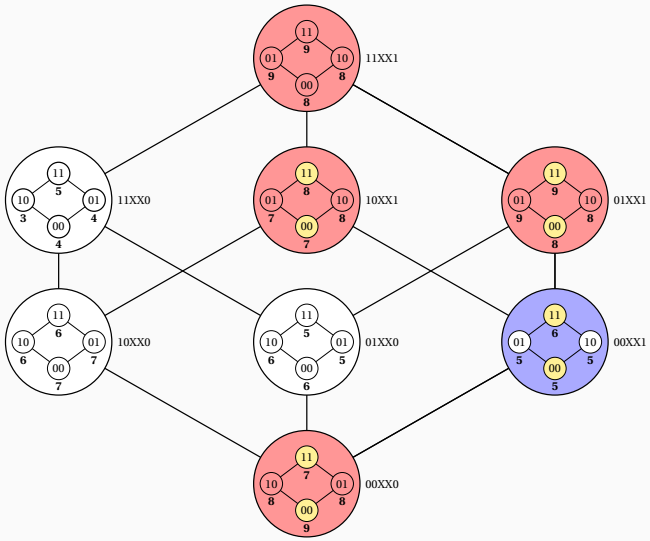




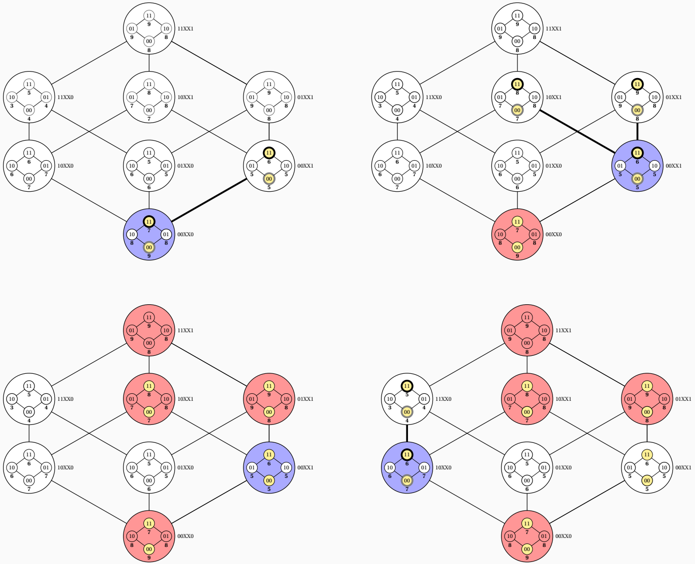
Simulação de execução



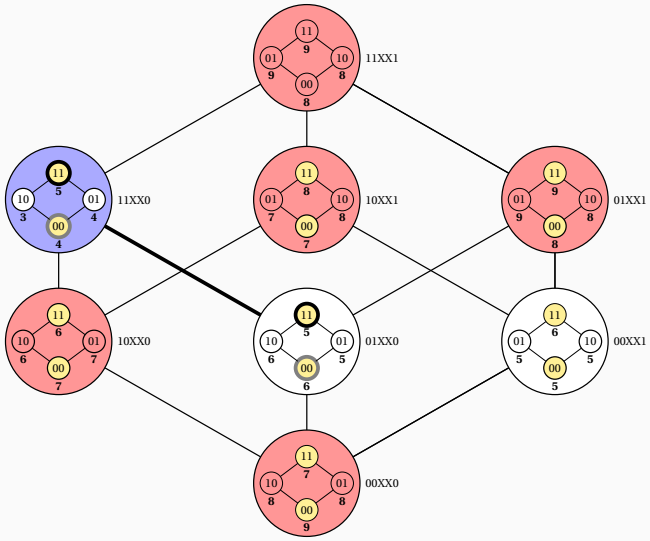




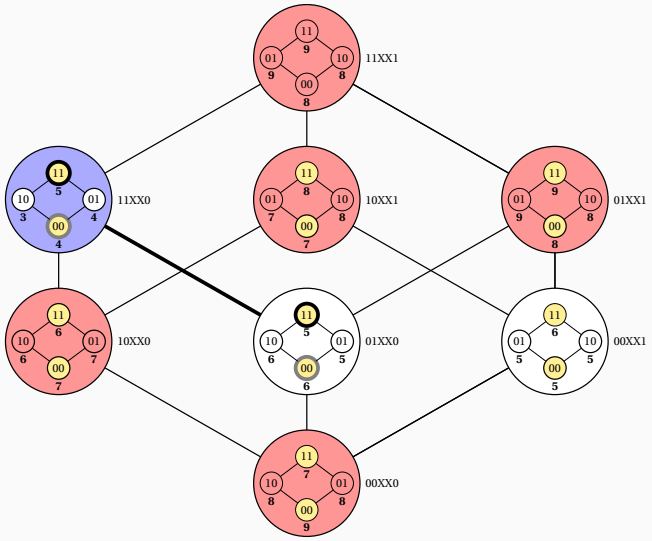
Simulação de execução

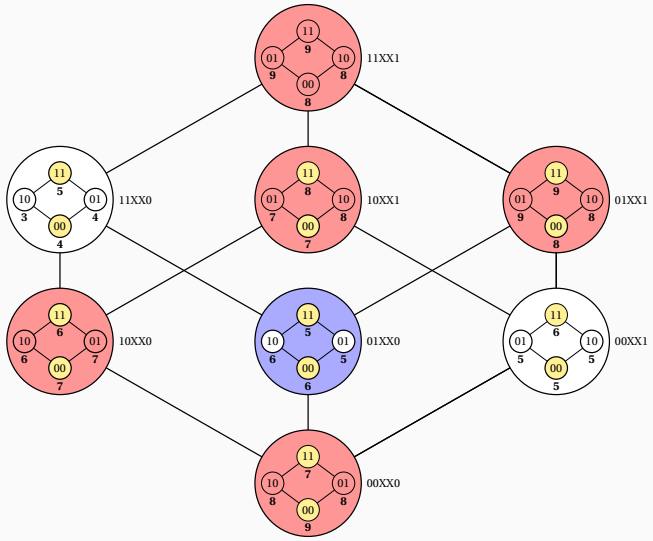


Simulação de execução

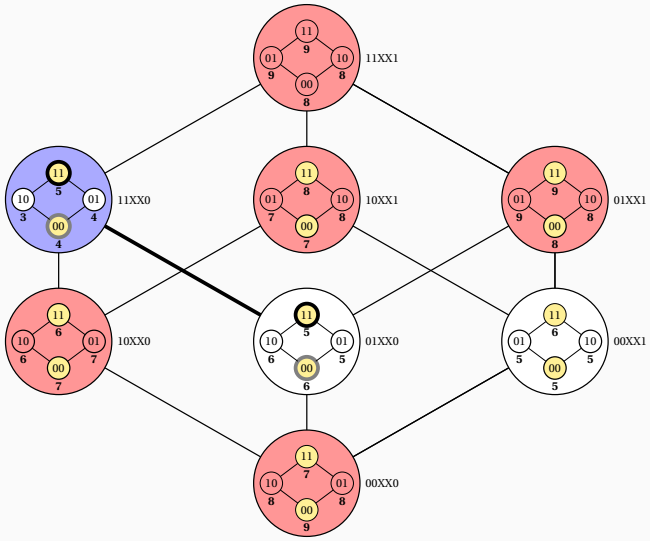


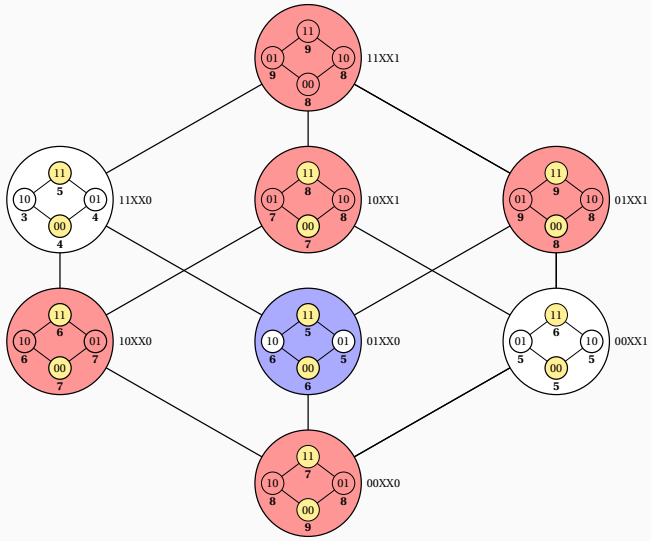
Simulação de execução

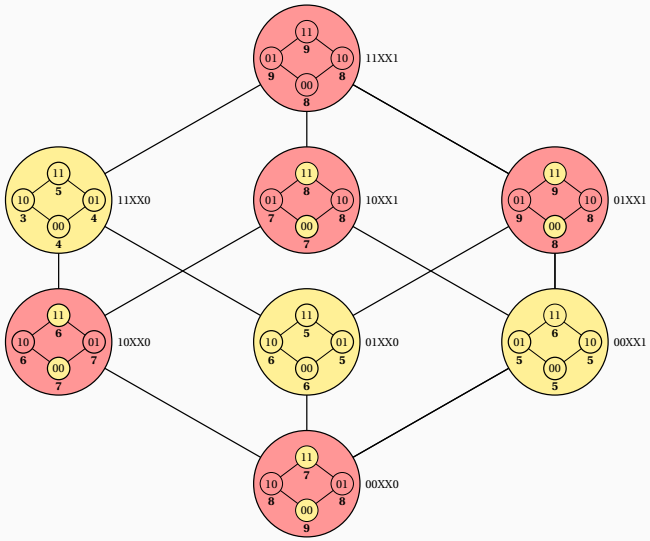




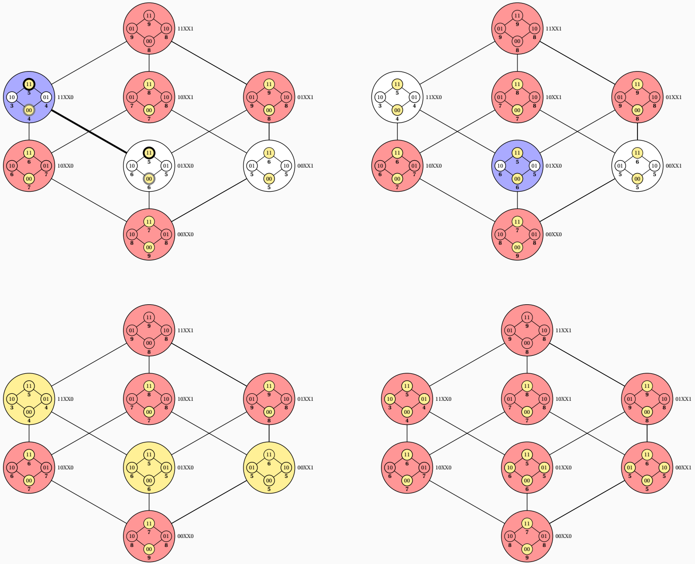
Simulação de execução







Simulação de execução

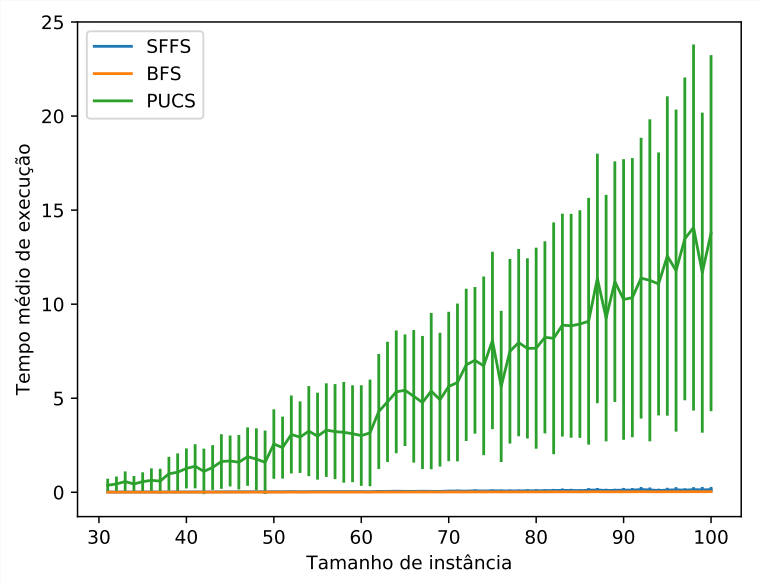


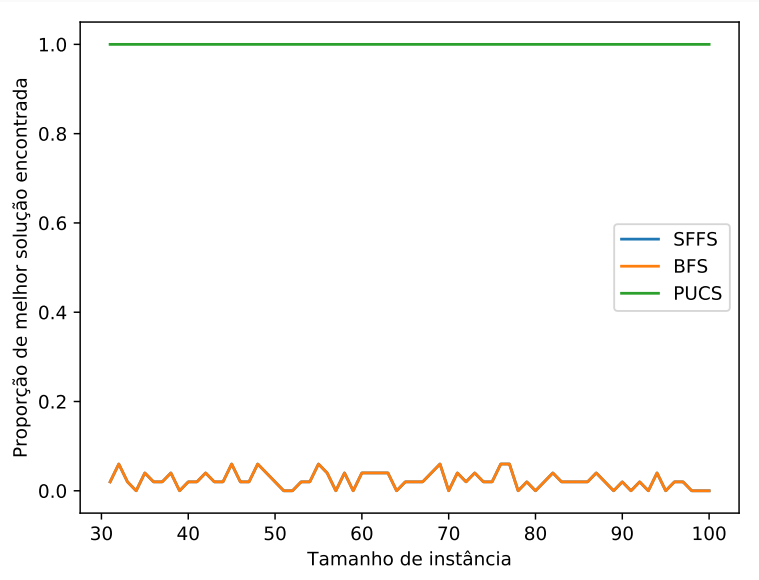
Resultados do PUCS

Em instâncias pequenas, usamos parâmetros que deixam o algoritmo ótimo. O desempenho do PUCS foi similar ao do UBB-PFS.

Resultados do PUCS

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





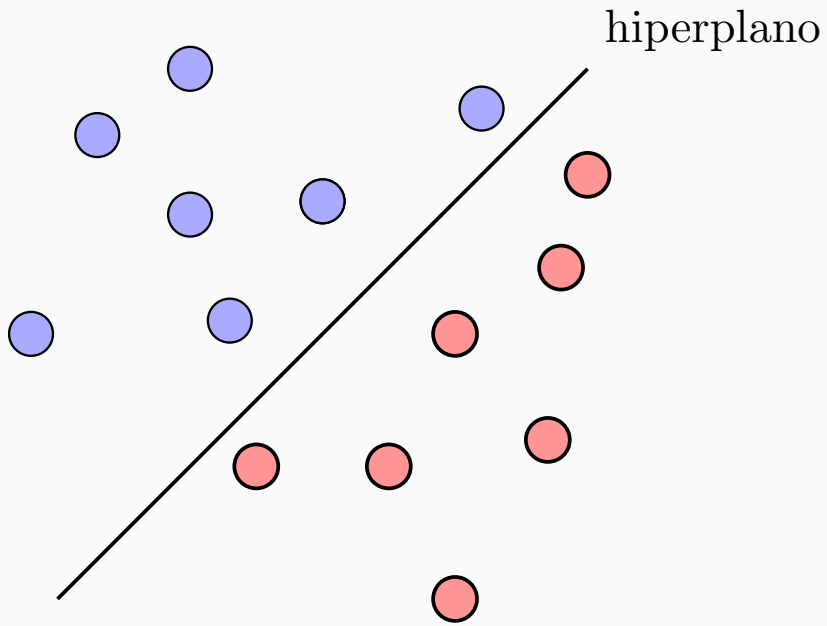
Aplicações instâncias reais

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

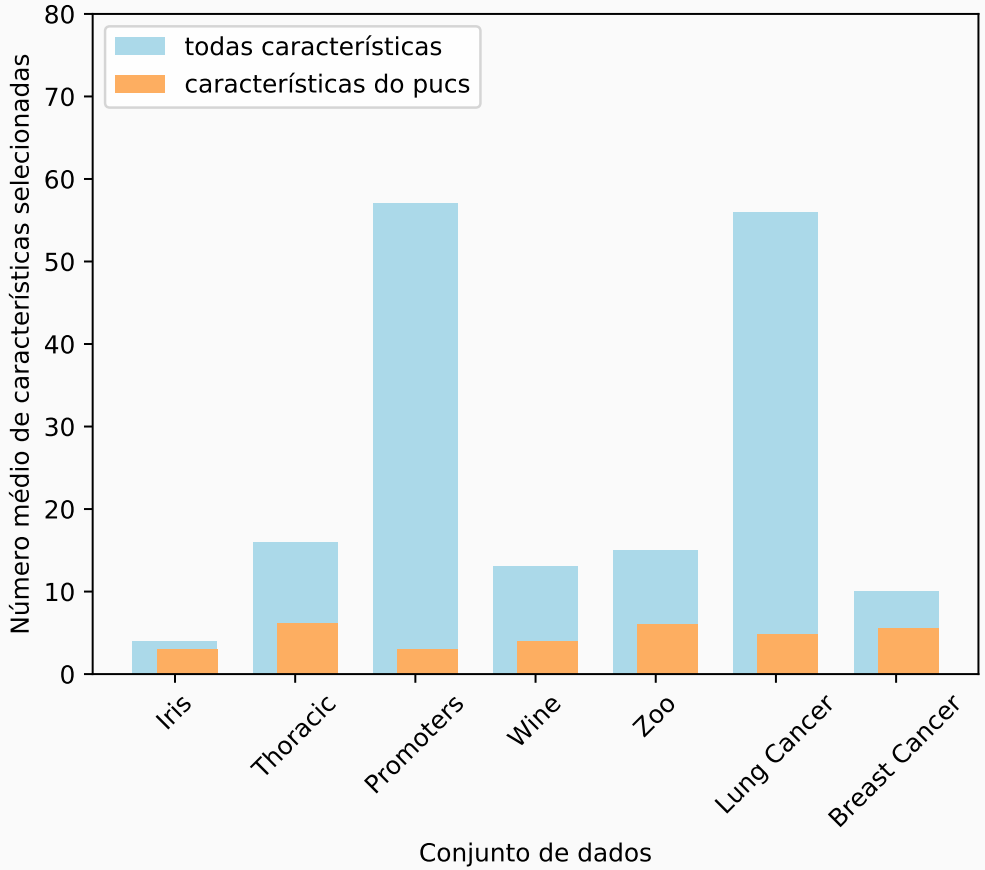
Para avaliar a seleção de características fizemos a validação cruzada de modelos com todas características e a de modelos apenas com características selecionadas.

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.

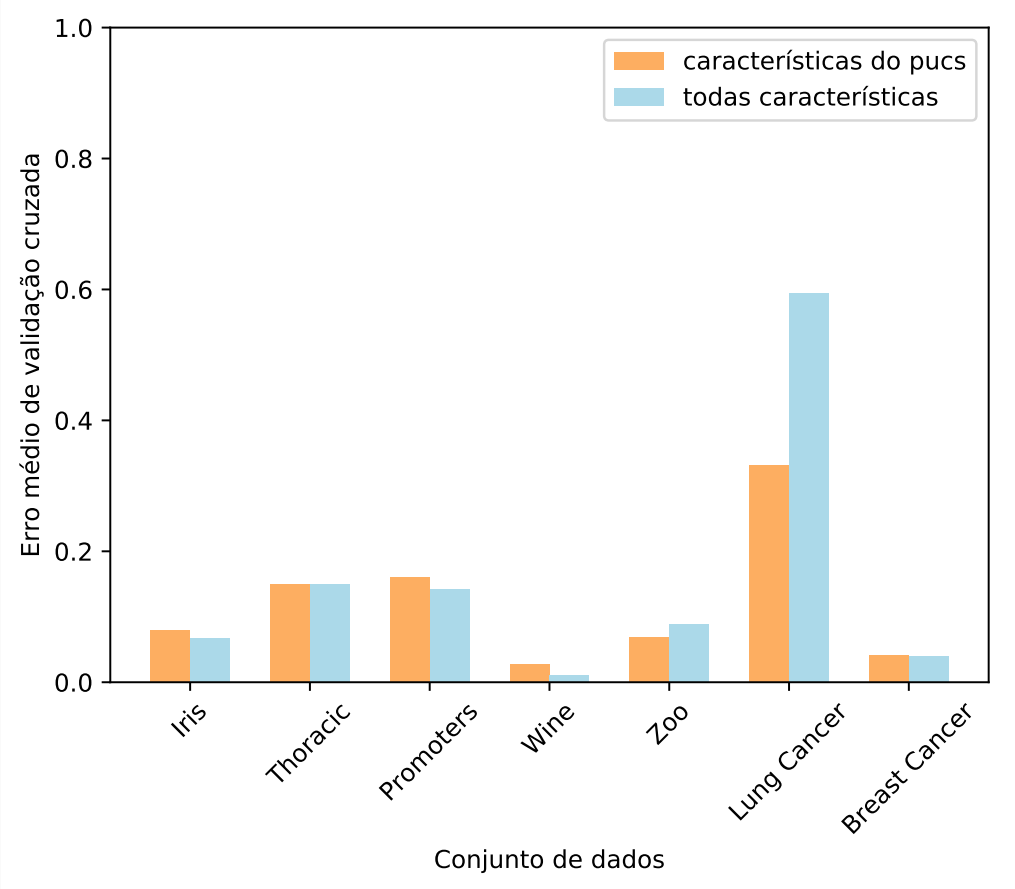


Resultados

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.