Projeto de algoritmos baseados em florestas de posets para o problema de otimização U-curve

Beneficiário: Gustavo Estrela de Matos

Pesquisador Responsável: Marcelo da Silva Reis

Centro de Toxinas, Imuno-resposta e Sinalização Celular (CeTICS)

Laboratório Especial de Ciclo Celular (LECC)

Instituto Butantan, São Paulo, 21 de dezembro de 2016.

Resumo

O problema U-curve é uma formulação de um problema de otimização que pode ser utilizado na etapa de seleção de características em Aprendizado de Máquina, com aplicações em desenho de modelos computacionais de sistemas biológicos. Não obstante, soluções propostas até o presente momento para atacar esse problema têm limitações do ponto de vista de consumo de tempo computacional e/ou de memória, o que implica na necessidade do desenvolvimento de novos algoritmos. Nesse sentido, em 2012 foi proposto o algoritmo Poset-Forest-Search (`PFS`), que organiza o espaço de busca em florestas de posets. Esse algoritmo foi implementado e testado, com resultados promissores; todavia, novos melhoramentos são necessários para que o `PFS` se torne uma alternativa competitiva para resolver o problema U-curve. Neste projeto propomos a construção de uma versão paralelizada e escalável do algoritmo `PFS`, utilizando diagramas de decisão binária reduzidos e ordenados. Além disso, propomos adaptar o `PFS` como um algoritmo de aproximação, no qual o critério de aproximação da solução ótima faça uso do teorema da navalha de Ockham. Os algoritmos desenvolvidos serão implementados e testados em instâncias artificiais e também em conjuntos de dados próprios para experimentos comparativos entre diferentes algoritmos de seleção de características.

Design of poset forest-based algorithms for the U-curve optimization problem

Student: Gustavo Estrela de Matos

Supervisor: Marcelo da Silva Reis

Center of Toxins, Immune-response and Cell Signaling (CeTICS)

Laboratório Especial de Toxinologia Aplicada (LETA)

Instituto Butantan, São Paulo, December 21, 2016.

Abstract

The U-curve problem is a formulation of an optimization problem that can be used in the feature selection step of Machine Learning, with applications in the designing of computational models of biological systems. Nevertheless, the solutions so far proposed to tackle this problem have limitations from both required computational time and space points of view, which implies in the need for development of new algorithms. To this end, it was introduced in 2012 the Poset-Forest-Search (`PFS`) algorithm, which organizes the search space in forests of posets. This algorithm was implemented and tested, with promising results; however, new improvements are required until `PFS` becomes a competitive alternative to tackle the U-curve problem. In this project, we propose the design of a parallelized, scalable version of the `PFS` algorithm, using reduced ordered binary decision diagrams. Moreover, we propose to adapt `PFS` as an approximation algorithm, in which the approximation criterion to the optimal solution makes use of the Ockham's razor theorem. The developed algorithms will be implemented and tested on artificial instances and also on collection of datasets that are suitable for benchmarking of feature selection algorithms.

Sumário

1 Introdução

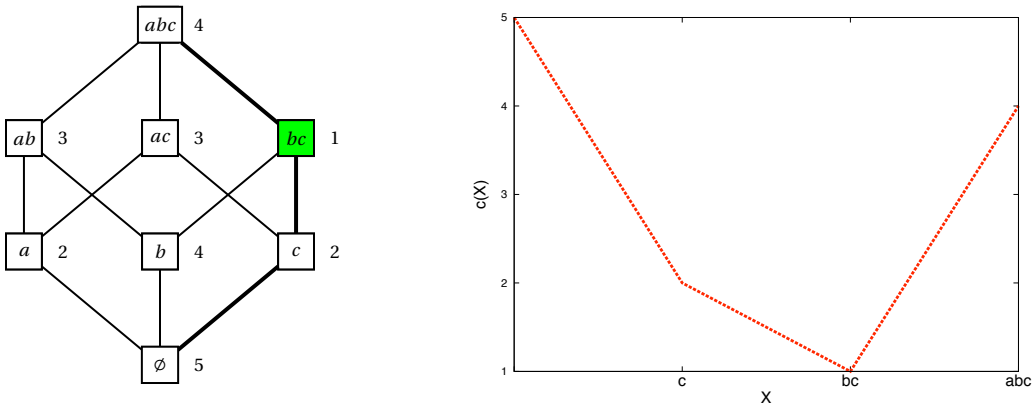
1.1 O problema U-curve

O problema de seleção de característica consiste em, dado um conjunto S de características, escolher um subconjunto de S que seja ótimo de acordo com alguma métrica. A solução desse problema tem aplicações em Aprendizado de Máquina e Reconhecimento de Padrões, com aplicações práticas em identificação de sistemas biológicos – por exemplo, na estimação de redes gênicas regulatórias. Formalmente, podemos definir o problema de seleção de características como um problema de busca, no qual procuramos por um subconjunto $X \in \mathcal{P}(S)$ que minimiza uma função de custo $c : \mathcal{P}(S) \to \mathbb{R}^+$. O espaço de busca do problema de seleção de características pode ser estruturado como um reticulado Booleano $(\mathcal{P}(S), \subseteq)$, no qual cada elemento é um subconjunto de características. Em funções custo c que dependem da estimação de uma distribuição conjunta a partir de um número limitado de amostras, é comum que o custo das cadeias do reticulado, quando restritas a c, descrevam curvas em U (figura 1); esse comportamento pode ser intuitivamente explicado se considerarmos que o erro de estimação diminui ao adicionarmos novas características, até um ponto em que a limitação de amostras faz com que acrescentar características leve a um aumento no erro de estimação.

O problema U-curve é um caso particular do problema de seleção de características no qual todas as cadeias do espaço de busca descrevem curvas em U quando avaliadas pela função de custo (figura 1). Existem algoritmos ótimos para solução do problema U-curve, tais como o U-curve-Branch-and-Bound (`UBB`) e o Poset-Forest-Search (`PFS`) [1]; os princípios de funcionamento deste último serão apresentados a seguir.

1.2 O algoritmo Poset-Forest-Search (`PFS`)

O algoritmo `PFS` é um algoritmo ótimo do tipo *branch-and-bound*. A ideia básica é generalizar o algoritmo `UBB`, que é um *branch and bound* unidirecional tanto em termos de percorrimento



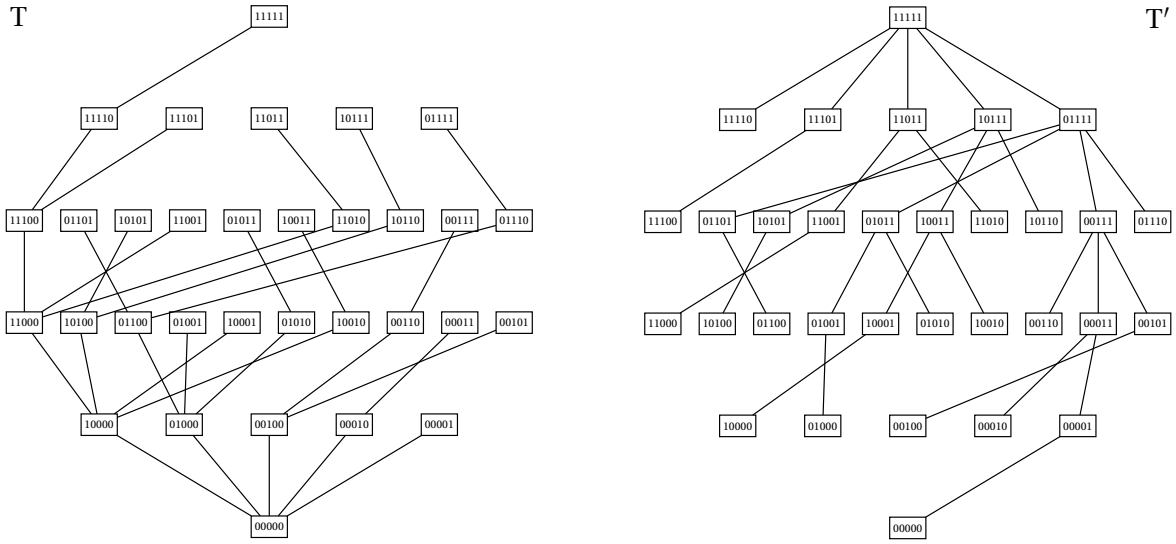
(a) (b)

Figura 1: exemplo de instância do problema U-curve. Figura 1(a): o diagrama de Hasse de um reticulado Booleano de grau 3 – as cadeias do reticulado, cujos custos de seus elementos são definidos através dos números ao lado dos elementos, descrevem curvas em U; a cadeia maximal $\{\emptyset, c, bc, abc\}$ está destacada em negrito. O elemento bc, destacado em verde, é mínimo na cadeia (e também no reticulado Booleano). Figura 1(b): o gráfico dos custos em função dos elementos da cadeia maximal destacada em negrito, mostrando sua curva em U. Figuras extraídas de Reis [1].

do espaço de busca quanto de podas, em um algoritmo bidirecional em ambos os procedimentos. Podas bidirecionais em um reticulado Booleano completo resultam em uma coleção de posets, de onde vem o nome do algoritmo.

Antes da inicialização do algoritmo, o espaço de busca é representado em duas árvores T e T' tais que T' é uma árvore complementar a T, ou seja, para qualquer aresta $\{X, Y\}$ na árvore T existe uma aresta $\{X^c, Y^c\}$ na árvore T' (figuras 2(a) e 2(b)).

O algoritmo inicia a execução escolhendo uma das duas árvores; sem perda de generalidade, digamos que T foi escolhida. A partir do conjunto vazio, uma cadeia é percorrida até que o custo de um elemento seja maior que o do elemento X que o precedeu na cadeia (figura 3(a)); nesse caso, são eliminados do espaço de busca todos os elementos que se ramificam na árvore T a partir de X, assim como as arestas percorridas desde o conjunto vazio. Esse procedimento resulta em uma floresta $\mathcal{A}$ composta por três árvores (figura 3(a)). Ao removermos de T' os mesmos elementos que foram removidos de T, o grafo induzido pelas arestas



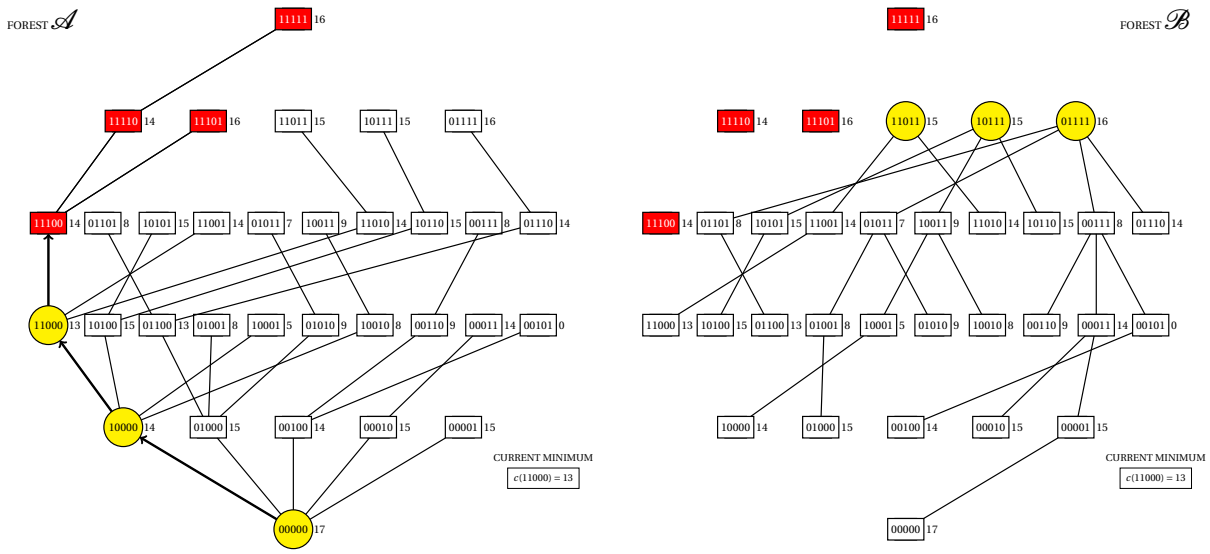
(a) (b)

Figura 2: exemplo de enumerações de um reticulado Booleano, no qual a árvore T (figura 2(a)) é complementar à árvore T' (figura 2(b)). Figuras extraídas de Reis [1].

remanescentes resulta em uma floresta $\mathcal{B}$ também composta por três árvores (figura 3(b)).

O algoritmo `PFS` inicia então uma nova iteração, escolhendo uma das florestas e em seguida repetindo o procedimento descrito acima para uma das árvores que a compõem. Se uma árvore contém apenas um único elemento X (i.e., uma árvore trivial, composta apenas por uma raiz), então X é removido da floresta escolhida, e a outra floresta é atualizada eliminando-se X e as arestas com uma das pontas ligada a esse elemento. O algoritmo `PFS` itera até que todo o espaço de busca seja esgotado.

Uma versão preliminar do `PFS` já foi implementada e mostrou resultados promissores, principalmente quando analisado o número de elementos do reticulado que são visitados [1]. Apesar disso, o algoritmo apresenta propriedades interessantes que foram pouco exploradas nesse trabalho original; por exemplo, o gerenciamento do espaço de busca através de florestas poderia ser aproveitado para o desenvolvimento de algoritmos paralelizados ótimos para o problema U-curve, com maior escalabilidade em relação a variantes sequenciais. Ademais, seria interessante testar a aplicabilidade desse algoritmo em um conjunto abrangente de instâncias de interesse prático.



(a) (b)

Figura 3: exemplo de um percorrimento e poda realizados na árvore T, no qual é eliminado do espaço de busca o intervalo $[11100, 11111]$ e produzida uma floresta $\mathcal{A}$ contendo três árvores, cujas raízes estão destacadas em amarelo (figura 3(a)). A eliminação desse intervalo em T' produz uma floresta $\mathcal{B}$ também com três árvores (figura 3(b)). Figuras extraídas de Reis [1].

2 Objetivos

Neste trabalho, propomos atingir três objetivos:

1. Implementação de uma versão do algoritmo sequencial do algoritmo `PFS` utilizando diagramas de decisão binária reduzidos e ordenados (*Reduced Ordered Binary Decision Diagrams* – ROBDDs) para representar as listas de raízes da floresta que representa o espaço de busca no algoritmo `PFS`. ROBDD é uma estrutura de dados eficiente para gerenciar coleções de elementos de um reticulado Booleano [2].

2. Desenho de uma versão escalável do `PFS`. Para este fim, paralelizaremos o percorrimento das árvores nas florestas que representam o espaço de busca, com o programa principal gerenciando a escolha das raízes (i.e., início de um percorrimento), guardando o mínimo corrente e centralizando a atualização das podas.

3. Desenvolvimento de uma versão do `PFS` que funcione como algoritmo de aproximação

para o problema U-curve, utilizando como critério de aproximação da solução ótima o teorema da navalha de Ockham: dado um espaço de hipóteses H (i.e., espaço de busca), o número mínimo de amostras necessário para se obter uma solução que erra no máximo ϵ com $1 - \delta$ de probabilidade é expresso por:

$$m(\delta, \epsilon) \geq \frac{1}{\epsilon} log(\frac{|H|}{\delta}). \quad (1)$$

Este resultado da equação 1, que é bem conhecido em teoria de Aprendizado de Máquina [3], será aproveitado para o uso das variantes do algoritmo `PFS` para resolver de forma aproximada instâncias de tamanho proibitivo mesmo para versões escaláveis do algoritmo ótimo.

Para todos os algoritmos propostos, faremos implementação e testes empíricos, para isso utilizando tanto instâncias artificiais quanto conjuntos de dados próprios para *benchmarking* de algoritmos de seleção de características.

3 Plano de trabalho

A pesquisa se iniciará com o estudo do algoritmo `PFS`. Uma implementação desse algoritmo atualmente encontra-se disponível para o arcabouço featsel, e será usado como base para os futuros algoritmos desse projeto. O próximo passo envolverá a construção de uma modificação do algoritmo `PFS` que utiliza a estrutura de dados ROBDD para guardar as raízes das florestas que representam o espaço de busca do problema.

Estudaremos em seguida a biblioteca OpenMP, escolhida para paralelização do algoritmo através do uso de memória compartilhada. Após concluídos os estudos dessa biblioteca, devemos analisar a dinâmica do algoritmo `PFS` e reescrevê-la com as necessárias modificações para que partes do algoritmo possam ser executadas em paralelo. A princípio, a paralelização do algoritmo se dará no percorrimento de caminhos nas florestas que compõem o espaço de busca, que será atualizado em uma área da memória comum a todas as linhas de execução.

Ao final dessa etapa, teremos um novo algoritmo, que será testado e comparado com outros algoritmos do arcabouço featsel, tais como o `UBB` e o `PFS` original.

A próxima etapa do nosso trabalho será o estudo de algoritmos de aproximação e do *Probably Approximately Correct* (PAC learning), que é o modelo de aprendizado no qual se aplica o teorema da navalha de Ockham. Com isso, pretendemos construir uma variante do algoritmo `PFS` que deixa de buscar uma solução ótima para buscar uma solução provavelmente aproximadamente correta. Após implementarmos o algoritmo de aproximação para o problema U-curve, poderemos testar o seu desempenho de forma análoga ao que será feito com a versão paralelizada com OpenMP.

Por fim, testaremos todos os algoritmos desenvolvidos durante o projeto com instâncias artificiais e reais do problema U-curve. Para gerar instâncias artificiais, empregaremos a redução polinomial do problema *subset sum* para instâncias do problema U-curve apresentada em Reis [1], assim como outras funções custo que o beneficiário utilizou durante seu estágio na Universidade Texas A&M (EUA), nas quais são introduzidas oscilações nas curvas em U das cadeias do reticulado Booleano. Já para instâncias reais, utilizaremos conjuntos de dados próprios para benchmarking de algoritmos, tais como os abrigados no UCI Machine Learning Repository.

A organização do plano de trabalho em uma lista de atividades é apresentada na na seção 3.1, enquanto que um cronograma para execução dessas atividades é proposto na seção seguinte (tabela 1).

3.1 Descrição de atividades

- **Atividade 1.** Estudo do algoritmo `PFS`.
- **Atividade 2.** Implementação de uma variante do `PFS` que usa ROBDDs como estrutura de dados para representar as coleções de raízes das árvores das florestas.
- **Atividade 3.** Estudo da biblioteca OpenMP, para paralelização do `PFS`.

- **Atividade 4.** Implementação de uma versão paralela do algoritmo `PFS` com o uso de ROBDDs como estrutura de dados.

- **Atividade 5.** Testes da primeira versão do `PFS` paralelizado com instâncias artificiais.

- **Atividade 6.** Estudos do *PAC learning* e do teorema da navalha de Ockham.

- **Atividade 7.** Implementação de versões do `PFS`, sequencial e paralelizada, que se comportem como algoritmos de aproximação.

- **Atividade 8.** Testes com instâncias artificiais dos algoritmos de aproximação.

- **Atividade 9.** Estudo comparativo entre os algoritmos produzidos, incluindo testes com instâncias artificiais e também de problemas reais.

3.2 Cronograma

Tabela 1: cronograma de atividades previstas nesta proposta de projeto.

Atividade/mês	Jan.17	Fev.17	Mar.17	Abr.17	Mai.17	Jun.17
Atividade 1	**x**	-	-	-	-	
Atividade 2	**x**	**x**	-	-	-	
Atividade 3	-	**x**	**x**	-	-	-
Atividade 4	-	-	**x**	**x**	**x**	-
Atividade 5	-	-	-	-	**x**	**x**
Primeiro Relatório	-	-	-	-	**x**	**x**

Atividade/mês	Jul.17	Ago.17	Set.17	Out.17	Nov.17	Dez.17
Atividade 6	**x**	**x**	-	-	-	-
Atividade 7	-	**x**	**x**	**x**	-	-
Atividade 8	-	-	-	**x**	**x**	-
Atividade 9	-	-	-	-	**x**	**x**
Segundo Relatório	-	-	-	-	**x**	**x**

4 Materiais e métodos

Os algoritmos produzidos serão implementados no arcabouço featsel. Esse arcabouço foi implementado em C++ e conta com diversas estruturas de dados que serão necessárias para implementar os algoritmos sugeridos nesse trabalho. Além disso, featsel já conta com classes que implementam ROBDDs; tal melhoramento foi feito durante a Iniciação Científica anterior do beneficiário, numa tentativa de melhorar o desempenho do algoritmo U-Curve-Search (`UCS`), resultando em ganhos de desempenho do ponto de vista de consumo de tempo computacional [4]. O arcabouço featsel está disponível sob a licença *GNU General Public License* (GNU-GPL), o que nos permite utilizar livremente o código desse arcabouço para implementação dos novos algoritmos.

Também utilizaremos a biblioteca OpenMP junto ao compilador `g++` para compilar os códigos produzidos. Por meio de diretivas de compilação, OpenMP determina como um código pode ser executado em paralelo no processador.

Experimentos computacionais, em particular os que envolverem paralelização de algoritmos, serão feitos em uma servidora do Instituto Butantan que é própria para esse fim: uma Dell PowerEdge 850, com 64 núcleos e 256 GB de memória RAM.

5 Forma de Análise e de Divulgação dos Resultados

Para analisar os resultados obtidos utilizaremos o próprio arcabouço featsel, que já possui métodos que registram dados sobre a execução dos algoritmos, assim como um programa auxiliar para *benchmarking* de algoritmos e de funções custo implementados nessa ferramenta. As principais métricas a serem consideradas nos experimentos serão tempo de execução, consumo de memória e a robustez dos algoritmos quando violamos a suposição de que a função custo é decomponível em curvas em U. O código-fonte final do projeto será disponibilizado para a comunidade acadêmica no repositório web do arcabouço featsel.

Além disso, ao longo de 2017, o beneficiário deverá se matricular em MAC499 – Trabalho

de Formatura Supervisionado, disciplina do IME-USP que exige a preparação de uma monografia ao final do curso e também a apresentação de um pôster contendo resultados finais da Iniciação Científica.

Por fim, projetamos a escrita de um artigo científico, no qual descreveremos os avanços que os novos algoritmos trouxeram, além de eventuais propriedades do problema U-curve e de nossa solução que possam ser exploradas futuramente para elaboração de novas abordagens para resolver esse importante problema de otimização.

Referências

[1] Marcelo S. Reis. "Minimização de funções decomponíveis em curvas em U definidas sobre cadeias de posets – algoritmos e aplicações". Tese de doutorado. Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil (2012).

[2] Randal E. Bryant. "Graph-based algorithms for boolean function manipulation". IEEE Transactions on Computers, v. 100.8, p. 677–691 (1986).

[3] Michael J. Kearns e Umesh V. Vazirani. "An introduction to computational learning theory". MIT Press (1994).

[4] Gustavo E. Matos e Marcelo S. Reis. "Estudos de estruturas de dados eficientes para abordar o problema de otimização U-curve". Relatório científico final FAPESP, Instituto Butantan, Brasil (2015).