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

Aluno: Gustavo Estrela de Matos

Orientador: Marcelo da Silva Reis

14 de Dezembro de 2016

Resumo

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

Student: Gustavo Estrela de Matos

Supervisor: Marcelo da Silva Reis

14 de Dezembro de 2016

Abstract

Conteúdo

1 Introdução

1.1 O problema U-Curve

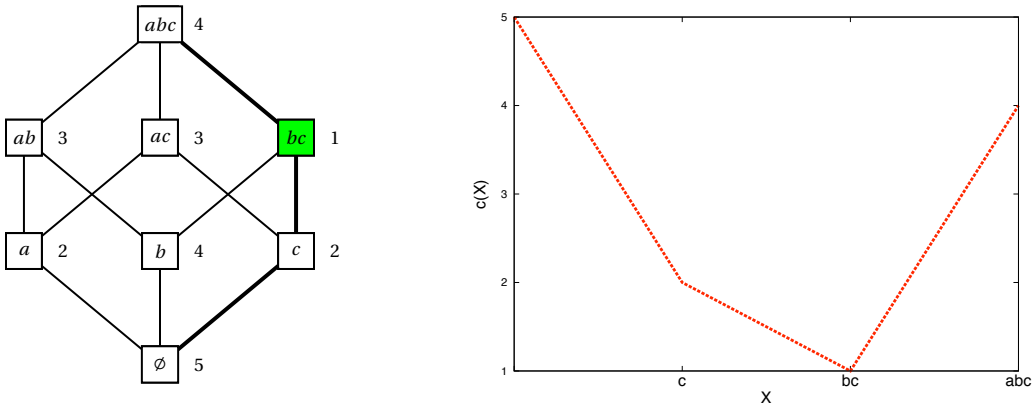
O problema de seleção de característica consiste em, dado um cojunto S de características, escolher um subconjunto de características que seja ótimo. A solução desse problema tem aplicações na construção de modelos para aprendizado de máquina e reconhecimento de padrões, que dependem da escolha de um subconjunto de características que seja o mais relevante possível (ótimo), de acordo com alguma métrica. Formalmente, podemos definir o problema de seleção de características como um problema de busca, no qual procuramos 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 visto como um reticulado booleano $(\mathcal{P}(S), \subseteq)$, onde cada nó é um conjunto de características, também chamado de classificador. É comum nesse problema que as cadeias do reticulado descrevam "curvas em u"quando avaliadas pela função de custo c. Esse comportamento pode ser intuitivamente explicado se considerarmos que um classificador melhora ao adicionarmos novas características até um ponto em que o grande número de características causa grandes erros de estimação, piorando o classificador.

O problema U-Curve é um caso particular do problema de seleção de características em que todas as cadeias do espaço de busca descrevem "curvas em u"quando avaliadas pela função de custo. Existem algoritmos ótimos para solução do problema U-Curve, como o Poset-Forest Search (`PFS`) e U-Curve Search (`UCS`) [1]. Além disso, em outra oportunidade de iniciação científica, estudamos o uso de diagramas de decisão binários ordenados e reduzidos (ROBDDs) como uma estrutura de dados eficiente para o controle do espaço de busca [2].

O uso de ROBDDs aliado a mudanças à dinâmica do `UCS` nos levaram a criação do algoritmo `UCSR`. Esse novo algoritmo trouxe melhoras no tempo de execução, pois permite consultas rápidas ao espaço de busca, o que era mais custoso no algoritmo `UCS`. Porém, as melhorias obtidas foram limitadas, uma vez que manter a estrutura de ROBDD, em alguns

casos, demandava grande processamento e uso de memória. Portanto, torna-se necessário a criação de novos algoritmos para resolver o problema, o que nos leva ao estudo do algoritmo Poset-Fores Search (`PFS`) [1].



(a) (b)

Figura 1: um 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 nós, 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. Figura extraída de Reis [1].

1.2 O algoritmo Poset-Forest Search

O algoritmo `PFS` é um algoritmo ótimo do tipo *branch-and-bound*. Esse algoritmo define duas florestas que são subgrafos do reticulado booleano que define o espaço de busca. Essas florestas são compostas por árvores que são definidas por uma enumeração arbitrária dos elementos do conjunto de característica. A ideia principal do algoritmo é, partindo de um raíz, percorrer um caminho dentro de uma árvore, fazendo podas no espaço de busca quando possível.

Uma versão preliminar desse algoritmo já foi implementada [1] e mostrou resultados promissores, principalmente quando analizado o número de nós do reticulado que são visitados.

Apesar disso, o algoritmo ainda pode ser estudado e apresenta características que podem ser exploradas para melhorar seu desempenho.

O primeiro ponto do algoritmo que pode ser explorado é a escolha da raíz de um caminho a ser percorrido no espaço. Quando se utiliza ROBDDs como estrutura de dados, torna-se fácil representar uma função booleana que recebe como entrada um elemento do espaço de busca. Portanto, ao invés de listar as raízes do espaço de busca, poderíamos usar um ROBDD para representar esses elementos.

Além disso, a floresta que representa o espaço de busca é composta por árvores, isto é, componentes conexas que podem ser percorridas separadamente. Portanto, é natural que percorramos paralelamente as árvores do espaço de busca. Porém, é necessário notar que o controle do espaço de busca deve ser feito de maneira centralizada.

Por último, podemos explorar uma modificação no `PFS` para que ele se torne um algoritmo de aproximação, isto é, um algoritmo que não é ótimo mas garante que a sua solução dista ϵ da solução ótima com probabilidade $1 - \delta$.

2 Objetivos

Neste trabalho, propomos atingir os seguintes objetivos:

1. Utilização dos ROBDDs, implementados na iniciação científica anterior, para representar as listas de raízes da floresta que representa o espaço de busca no algoritmo PFS.

2. Desenho de uma versão paralelizada do PFS, com maior escalabilidade. 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:

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

4. Implementação e testes dos algoritmos propostos, para isso empregando o arcabouço featsel.

3 Plano de trabalho

A pesquisa se iniciará com o estudo do algoritmo `PFS`. Esse algoritmo está atualmente implementado no 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 da floresta que representa o espaço de busca do problema.

Estudaremos em seguida a biblioteca OpenMP, escolhida para paralelização do algoritmo. Após concluidos estudos da biblioteca, devemos analisar a dinâmica do algoritmo `PFS` e reescreve-la com as necessárias modificações para que partes do algoritmo possam ser executados em paralelo. A principio, a paralelização do algoritmo se dará no percorrimento de caminhos das florestas do espaço de busca, que será atualizado em um espaço comum da memória 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.

A próxima etapa do nosso trabalho será o estudo de algoritmos de aproximação e do modelo de aprendizado *Probably Approximately Correct* (PAC learning). Portanto, pretendemos construir uma variante do algoritmo `PFS` que deixa de buscar uma solução ótima para buscar uma solução provavelmente aproximadamente correta. Utilizaremos como critério de aproximação a solução ótima do teorema da navalha de Ockham.

Após implementarmos o algoritmo de aproximação para o problema U-Curve poderemos testar o seu desempenho com instâncias artificiais. Além disso, poderemos testar os algoritmos desenvolvidos durante o projeto com instâncias reais do problema U-Curve. Os resultados obtidos, assim como as descobertas feitas no processo de estudo e criação dos novos algoritmos serão descritas em um artigo que será escrito como ultima atividade desse projeto.

3.1 Cronograma

Tabela listando atividades de janeiro a dezembro de 2017.

Tabela 1: cronograma de atividades previstas neste projeto de Iniciação Científica. As descrições das atividades listadas seguem na seção a seguir.

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**

3.2 Descrição de atividades

- **Atividade 1.** Estudo do algoritmo `PFS`.
- **Atividade 2.** Implementação de uma variação do `PFS` que usa ROBDDs como estrutura de dados para representar início de caminhos.

- **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` paralelo com instâncias artificiais.
- **Atividade 6.** Estudo do teorema da navalha de Ockham.
- **Atividade 7.** Implementação de uma variação do `PFS` paralelo que se comporte como um algoritmo de aproximação.
- **Atividade 8.** Testes com instâncias artificiais do algoritmo de aproximação.
- **Atividade 9.** Estudo comparativo entre os algoritmos produzidos, incluindo testes com instâncias reais.

4 Materiais e métodos

Os algoritmos produzidos nesse trabalho serão implementados junto ao 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, o arcabouço featsel possui a estrutura de dados de ROBDDs, implementada em uma outra oportunidade de iniciação científica.

O arcabouço featsel segue a licença *GNU General Public License* (GNU-GPL), o que nos permite utilizar o código do arcabouço para implementação dos novos algoritmos. Por utilizarmos o arcabouço como base, o código que será implementado nesse trabalho também seguirá a licença GPL.

Também utilizaremos a especificação OpenMP junto ao compilador g++ para compilar os códigos produzidos. A especificação OpenMP determina uma biblioteca que dá suporte

ao programador, por meio de diretivas de compilação, ao determinar como um código pode ser executado em paralelo no processador.

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. As principais métricas a serem consideradas nos algoritmos implementados serão tempo de execução e gasto de memória. As instâncias utilizadas para geração desses dados serão tanto artificiais quanto reais, o que também nos oferecerá uma terceira métrica: a robustez dos algoritmos quando violamos a suposição de que a função custo é decomponível em "curvas em u".

Após testarmos os algoritmos com instâncias reais e artificiais iremos escrever um paper onde descreveremos os avanços que os novos algoritmos trouxeram e quais características do problema U-Curve e de nossa solução podem ser exploradas futuramente para elaboração de novas abordagens para resolver o problema.

Referências

[1] Reis, Marcelo S. "Minimization of decomposable in U-shaped curves functions defined on poset chains–algorithms and applications."PhD thesis, Institute of Mathematics and Statistics, University of São Paulo, Brazil, (2012).

[2]