Recomendação de parâmetros para o *COIN-OR Branch and Cut*

Rafael de Sousa Oliveira Martins
Departamento de Computação e Sistemas
Rua Trinta e seis, 115 – Bairro Loanda – CEP 35931-008 – João Monlevade – MG – Brasil
martins.rso@gmail.com

Janniele Aparecida Soares Araujo
Departamento de Computação e Sistemas
Rua Trinta e seis, 115 – Bairro Loanda – CEP 35931-008 – João Monlevade – MG – Brasil
janniele@decsi.ufop.br

Samuel Souza Brito
Departamento de Computação e Sistemas
Rua Trinta e seis, 115 – Bairro Loanda – CEP 35931-008 – João Monlevade – MG – Brasil
samuelbrito@decsi.ufop.br

Haroldo Gambini Santos
Departamento de Computação
Rua Quatro, 786 – Bairro Bauxita – CEP 35400-000 – Ouro Preto – MG - Brasil
haroldo@iceb.ufop.br

RESUMO

Este trabalho propõe a recomendação de parâmetros para o resolvedor *COIN-OR Branch and Cut (CBC)*, com o intuito de obter a melhor configuração de parâmetros para problemas de otimização e melhorar a eficiência na busca da solução ótima. Tal situação pode ser resolvida por meio da aplicação do processo de Descoberta de Conhecimento em Base de Dados (*Knowledge Discovery in Databases - KDD*). Os resultados obtidos foram satisfatórios, conseguiu-se encontrar um modelo genérico em árvore, para diferentes tipos de problemas, em que a indicação de parâmetros seja feita de forma confiável.

PALAVRAS CHAVE. COIN-OR Branch and Cut. Otimização Combinatória. Recomendação de Parâmetros. Descoberta de conhecimento em Base de Dados.

Tópicos (indique, em ordem de PRIORIDADE, o(s) tópicos(s) de seu artigo)

ABSTRACT

This paper proposes the recommendation of parameters to the resolver COIN-OR Branch and Cut (CBC), in order to get the best parameter settings for optimization problems and improve efficiency to find the optimal solution. This situation can be resolved by the application of process Knowledge Discovery in Databases (KDD). The results were satisfactory, it was possible to find a generic model tree for different types of problems, in which the parameter indication is made in way trustworthy.

KEYWORDS. COIN-OR Branch and Cut. Combinatorial Optmization. Recommendation parameters. Knowledge Discovery in Databases.

Paper topics (indicate in order of PRIORITY the paper topic(s))

1. Introdução

A Programação Linear Inteira, também referida como Programação Inteira (PI), pode ser vista como um problema de Programação Matemática em que a função objetivo, bem como as restrições, são lineares, porém uma ou mais variáveis de decisão podem apenas assumir valores inteiros. O modelo formal de um problema de Programação Inteira pode ser expresso como:

$$\begin{aligned} Max\ (ou\ Min)\ z = \quad & c^T x + h^T y \\ Sujeito\ a: \quad & Ax + Gy \leq b \\ & x \geq 0,\ y \geq 0 \\ & x \in \Re^n,\ y \in \mathbb{Z}^p \end{aligned}$$

Neste modelo, x representa um conjunto de variáveis de decisão contínuas de dimensão n e y um conjunto de variáveis de decisão inteiras de dimensão p.

Uma forma de solução de problemas de PI se dá por meio da utilização do resolvedor de código aberto *COIN-OR Branch and Cut* (CBC)[1] [Lougee-Heimer, 2003]. Escrito na linguagem C++, o resolvedor possui um conjunto de bibliotecas que permitem a leitura, criação e manipulação de problemas dessa natureza. Assim, o CBC pode ser utilizado tanto como um resolvedor *stand-alone* quanto como uma biblioteca para auxiliar no desenvolvimento de algoritmos *branch-and-cut* customizáveis.

O CBC possui uma série diversificada de parâmetros, que podem influenciar diretamente no desempenho da resolução de um problema. Dessa forma, a tarefa de identificar e utilizar os melhores parâmetros para um dado problema é complexa, porém essencial. Tal situação pode ser resolvida por meio da aplicação do processo de Descoberta de Conhecimento em Base de Dados (*Knowledge-Discovery in Databases - KDD*), que segundo Fayyad et al. [1996] é o processo que visa o descobrimento de conhecimento válido, relevante e desconhecido em base de dados. Esse processo é dividido em 5 etapas: seleção, pré-processamento, transformação, mineração de dados e interpretação.

Neste trabalho são utilizadas técnicas de descoberta de padrões através da análise do comportamento histórico dos dados, possibilitando que novas instâncias de problemas sejam executadas de forma eficiente no CBC.

2. O problema

Para que o resolvedor CBC possa obter resultados satisfatórios na solução de problemas de PI é preciso que sejam definidos bons parâmetros para cada problema de entrada. A escolha de parâmetros pode influenciar diretamente no desempenho do resolvedor, o que torna esse passo uma atividade essencial na construção da solução.

O objetivo deste trabalho é a recomendação de um conjunto parâmetros a ser utilizado pelo CBC na resolução de problemas de PI. Para isso, se faz necessário o conhecimento prévio de uma base de dados diversificada, analisar as características de problemas já executados pelo CBC e construir um modelo de classificação confiável.

Em problemas de otimização podemos considerar o *GAP* como uma medida da qualidade de uma solução, que representa o percentual da distância entre a solução encontrada e a solução ótima de um problema. Por meio do comportamento histórico dos dados, podemos analisar as características das instâncias e as configurações de parâmetros aplicadas em que se obteve o melhor *GAP*. Assim, através de um modelo de classificação, pode-se indicar de maneira fidedigna, a configuração de parâmetros que retorne um *GAP* ideal, próximo de zero.

[1] `https://projects.coin-or.org/Cbc`

3. Desenvolvimento

No processo de descoberta de conhecimento é realizada uma busca efetiva por conhecimentos no contexto da aplicação. Esse processo envolve a aplicação de algoritmos sobre os dados em busca de conhecimentos implícitos e úteis. Existem várias técnicas e algoritmos que podem ser utilizados no problema em questão, tais como redes neurais, algoritmos genéticos, modelos estatísticos e probabilísticos. Neste artigo foi utilizado a tarefa de classificação que consiste em descobrir uma função que mapeie um conjunto de registros em um conjunto de rótulos categóricos predefinidos, denominados classe.

No problema abordado, o conjunto de registros são as instâncias que possuem os atributos dos problemas de otimização combinatória e linear executados no CBC. Os atributos utilizados são relacionados aos principais componentes de um problema de PI:

- **variáveis**: contém atributos que apresentam o número total de variáveis presentes no problema, bem como o detalhamento da quantidade de variáveis binárias, inteiras e contínuas;
- **restrições**: contém atributos que contabilizam o número total de restrições do problema e os principais tipos de restrições (*set partition*, *set packing*, *set cover*, *cardinality*, *invariant knapsack*, *knapsack*, *binary flow conservation*, *integer flow conservation* e *continous flow conservation*);
- **matriz de incidência**: contém os valores mínimo e máximo dos coeficientes da matriz de incidência do problema.
- **conjunto de Parâmetros**: contém uma *string* que é o conjunto de parâmetros a serem utilizadas, passados como parâmetro para execução do problema no resolvedor CBC.

Além dos atributos das instâncias, temos a lista de alguns parâmetros que podem ser utilizados pelo resolvedor CBC. Abaixo segue a lista de parâmetros utilizados neste trabalho:

- **diveopt**: ativa ou desativa as heurísticas de mergulho (*diving*). No caso da ativação, decide em quais variáveis essas heurísticas serão aplicadas.
- **zero**: ativa ou desativa os cortes *Zero-Half* $(0/\frac{1}{2})$ [Caprara e Fischetti, 1996]. No caso da ativação, decide em quais ramos da árvore esses cortes serão gerados e inseridos.
- **strong**: número máximo de variáveis candidatas para aplicar *strong branching* [Achterberg et al., 2005]. Por padrão, o valor desse parâmetro é igual a 5.
- **trust**: número máximo de ramificações utilizando seleção de variáveis por *strong branching*. Por padrão, o valor desse parâmetro é igual a 5.
- **tunep**: parâmetro usado para controlar a etapa de pré-processamento. O pré-processamento inclui estratégias para alterar coeficientes e fixar os limites de algumas variáveis, diminuir o valor do lado direito de algumas restrições, etc.
- **proximity**: ativa ou desativa a heurística *Proximity Search* [Fischetti e Monaci, 2014].

A classe considerada neste trabalho representa o *GAP*, medida percentual da distância entre a solução encontrada e a solução ótima. Assim será identificada a melhor relação entre os atributos das instâncias e as configurações dos parâmetros que se aproxime do *GAP* ideal, próximo a zero.

Os algoritmos de classificação utilizados neste trabalho são implementados e executados pela ferramenta *Waikato Environment for Knowledge Analysis* (WEKA)[2], que é uma coleção de

[2] `http://www.cs.waikato.ac.nz/ml/weka/`

algoritmos de aprendizado de máquina e de pré processamento de dados. Esta inclui basicamente os algoritmos mais utilizados na literatura. Os algoritmos utilizados no trabalho são divididos em classes, sendo estes de árvore de classificação, aprendizagem preguiçosa (*lazy*), baseado em funções e os meta algoritmos. A Tabela 1 apresenta uma breve explicação sobre os algoritmos utilizados.

Nome	Função
RandomCommitee	Cria um conjunto de classificadores básicos aleatorizados. A previsão final é uma média linear das previsões geradas pelos classificadores de base.
RandomTree	Constrói uma árvore que considera um dado número de atributos aleatórios em cada nó.
RandomForest	Cria várias árvores de classificação A floresta escolhe a classificação com maior pontuação.
KStar	Utiliza o método do Vizinho mais próximo com função de distância generalizada.
J48	Utiliza o algoritmo de árvore de decisão c4.5.
Bagging	Meta algoritmo que melhora a estabilidade e a precisão para reduz a variância. Geralmente aplicado a métodos de árvore de decisão.
REPTree	Método de árvore de decisão rápido que usa poda reduzida de erros.
RandomSubSpace	Constrói um conjunto de classificadores de base com diferentes conjuntos de atributos.
ClassificationRegression	Executa classificação usando um método de regressão.
LMT	Constrói modelos de árvore de regressão logística.
LogitBoost	Executa regressão logística aditiva.
SimpleLogistic	Constrói modelos de regressão logística linear com seleção de atributos embutida.
MultiClassClassifier	Usa um classificador de duas classes para conjuntos de dados multiclasse.
Ibk	Utiliza um classificador baseado em K-vizinhos mais próximos.
LWL	Algoritmo geral para aprendizagem localmente ponderada.
SMO	Algoritmo de otimização mínima sequencial para a classificação de vetores de suporte.

Tabela 1: Tabela modificada de Witten et al. [2011]

3.1. Métricas de Avaliação

Métricas de avaliação são utilizadas para identificar os melhores algoritmos a serem aplicados sobre o problema em questão. Para tal avaliação, os algoritmos da Tabela 1, utilizados para a descoberta de conhecimento em base de dados, são comparados de acordo com métricas baseadas em erros e correlação entre os dados. As métricas utilizadas são: coeficiente de clusterização (CC), média do erro absoluto (MAE), raiz quadrada do quadrado do erro médio (RSME), erro relativo absoluto (RAE), raiz quadrada do erro relativo (RRSE) e o classificações corretas (CCI).

Formalmente o valor de θ é denotado como o valor verdadeiro de interesse, e $\hat{\theta}$ é denotado como valor estimado.

A métrica CC mostra a relação entre o valor verdadeiro e o valor estimado de uma classificação, com isso esse valor varia estritamente entre $-1\theta, \hat{\theta} \leq 1$. Quanto mais próximo os valores de θ e $\hat{\theta}$ de 0 a relação é basicamente inexistente e quanto mais próximo aos extremos a relação é muito forte, sendo, no caso negativo, uma relação muito forte inversa.

A métrica MAE (1), é uma das medidas mais comuns de erro de previsão. Essa medida não leva em conta se o erro foi sobrestimado ou subestimado, caracterizando-se por ser a média dos erros cometidos pelo modelo de previsão durante uma série de execuções. Para calcular, subtrai-se o valor da previsão ao valor verdadeiro em cada período de execução, o resultado deverá sempre ser positivo, sempre em módulo, soma-se e divide-se pelo número de valores que usado para obter a soma, representado a seguir formalmente:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{\theta}_i - \theta_i| \tag{1}$$

A métrica RMSE (2), também é usada como uma medida do erro de previsão. É determinado a soma dos erros de previsão ao quadrado e dividindo pelo número de erros usado no cálculo. o RMSE pode ser expresso formalmente:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{\theta}_i - \theta_i\right)^2} \tag{2}$$

A métrica RRSE (3), segundo Witten et al. [2011] refere-se a uma medida um pouco diferente, o cálculo do erro e feito em relação ao que teria sido um preditor simples utilizado. O preditor simples em questão é apenas a média dos valores reais dos dados, denotados por 1, assim o RRSE normaliza dividindo-se pelo erro quadrado total do indicador padrão, representado a seguir formalmente:

$$RRSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\hat{\theta}_i - \theta_i\right)^2}{\sum_{i=1}^{N}\left(\overline{\theta} - \theta_i\right)^2}}, \text{sendo } \overline{\theta} \text{ a média dos valores de } \theta \tag{3}$$

A métrica RAE (4), segundo Witten et al. [2011] é apenas o erro absoluto total, sendo o mesmo tipo de normalização do RSSE, os error são normalizados pelo erros do preditor simples que prevê os valores médios, representado formalmente:

$$RAE = \frac{\sum_{i=1}^{N}|\hat{\theta}_i - \theta_i|}{\sum_{i=1}^{N}|\overline{\theta} - \theta_i|}, \text{sendo } \overline{\theta} \text{ a média dos valores de } \theta \tag{4}$$

A métrica de CCI a partir do resultado das classificações ele mostra em porcentagem a quantidade de instâncias que foram corretamente classificadas.

4. Experimentos Computacionais

As instâncias a serem utilizadas compreendem uma base de teste criada pelo Grupo de Otimização e Algoritmos (GOAL) da Universidade Federal de Ouro Preto. Essa base consiste de 199 problemas de Programação Inteira, incluindo instâncias da Mixed Integer Problem Library (MIPLIB) [Koch et al., 2011] e de problemas de escalonamento em geral.

Para obter uma base de dados de teste foram realizadas dezesseis execuções no CBC, para cada instância do conjunto. Essas execuções utilizam diferentes combinações de parâmetros deste resolvedor (ver Seção 3). Informações sobre cada uma das instâncias, além do detalhamento completo das execuções realizadas no CBC podem ser encontradas no sítio disponibilizado pelo GOAL [3].

A ferramenta *Waikato Environment for Knowledge Analysis (WEKA)* utilizada, foi projetada para que seja possível a experimentação rápida de métodos existentes em novos conjuntos de dados de maneira flexível Witten et al. [2011].

Sobre a base de testes, foram executados todos os algoritmos apresentados na Tabela 1, assim é possível avaliá-los em relação às métricas de avaliação apresentadas na subseção 3.1.

Analisando a Tabela 2 pode-se perceber, que para o problema em questão, os algoritmos que trabalham com fatores aleatórios, como *Random Commitee* e *Random Tree*, possuem desempenho melhor do que algoritmos que trabalham de forma sequencial, como por exemplo o SMO.

Os algoritmos *Random Commitee* e *Random Tree* obtiveram os resultados mais satisfatórios, ou seja, com os menores erros em relação às métricas de avaliação. Os mesmos possuem erros bem menores que os demais algoritmos, eles apresentam um coeficiente de clusterização(CC) próximo a 1, (ver Subseção 3.1), e classificam corretamente 98% da instâncias executadas.

A Tabela 2 apresenta os algoritmos ordenados de acordo com os menores erros, inciando com o MAE, RMSE, RAE e RRSE.

[3] http://cbc.decom.ufop.br/

Algoritmo	CC	MAE(%)	RMSE(%)	RAE(%)	RRSE(%)	CCI(%)
RandomCommitee	0.980	0.002	0.032	2.076	14.325	98.30
RandomTree	0.979	0.002	0.033	2.194	14.817	98.18
RandomForest	0.980	0.028	0.090	28.090	40.233	98.30
KStar	0.885	0.045	0.129	44.800	57.699	90.23
J48	0.578	0.056	0.167	55.814	74.730	64.45
Bagging	0.556	0.065	0.172	65.177	76.959	62.69
REPTree	0.461	0.069	0.186	69.390	83.325	54.55
RandomSubSpace	0.473	0.072	0.183	71.764	81.964	55.75
ClassificationRegression	0.415	0.080	0.193	79.577	86.385	51.22
LMT	0.313	0.080	0.205	80.279	91.605	43.59
LogitBoost	0.239	0.091	0.210	90.882	94.019	39.35
SimpleLogistic	0.071	0.096	0.218	95.688	97.670	29.40
MultiClassClassifier	0.064	0.096	0.218	95.815	97.757	28.89
Ibk	0.066	0.097	0.220	97.346	98.558	28.83
LWL	0.005	0.099	0.222	99.055	99.459	26.22
SMO	0.046	0.106	0.228	106.114	102.011	28.67

Tabela 2: Tabela de Comparação dos Algoritmos de Classificação

Na figura 1, é apresentado o modelo simplificado da árvore de classificação gerada pelo algoritmo *Random Tree*, apresentando somente os nós iniciais. O algoritmo que obteve o melhor desempenho, *Random Committee*, utiliza como classificador o *Random Tree* e encontra a média final das previsões geradas por esse classificador variando somente as bases.

Podemos observar que os nós principais correspondem aos atributos e quantidade de variáveis que o problema possui divididos em dois grupos, os que possuem valores menores que 552 e maiores ou iguais a 552. A partir do nó raiz da árvore expande-se uma sequência de sub árvores diferenciando os demais atributos em duas ou mais ramificações até que seja concluído a classificação. Os dois filhos diretos da raiz são *mflow* e *min*. O primeiro, *mflow*, é um atributo relacionado a um tipo de restrição que pode envolver variáveis inteiras e contínuas, cujo objetivo é garantir a conservação do fluxo em uma rede. O segundo, *min*, é o menor coeficiente da matriz do problema. Com isso, consegue-se perceber uma grande variação nas características da instâncias para o problema da recomendação de parâmetros, tornando impossível a percepção de algum conhecimento por meio desses atributos diretamente.

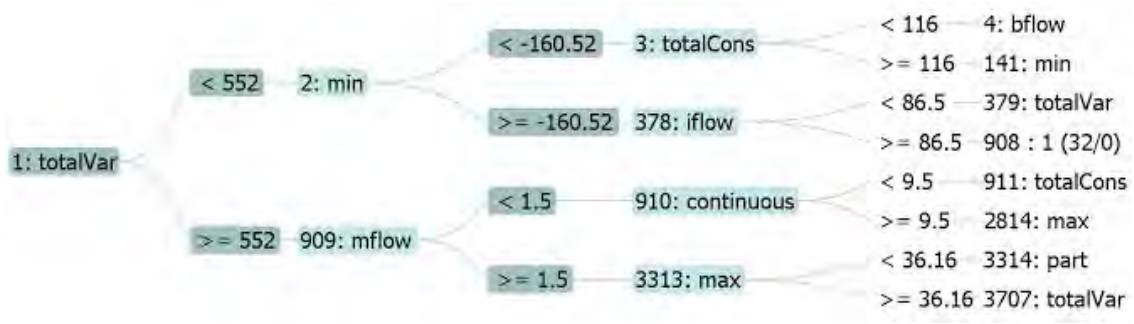

Figura 1: Modelo Simplificado da Árvore gerada pelo *Random Tree*

5. Conclusão e Trabalhos Futuros

Neste trabalho foi apresentada uma abordagem baseada em classificação para a recomendação de parâmetros a serem utilizados pelo *COIN-OR Branch and Cut* na resolução de problemas

de Programação Inteira. Nos experimentos realizados sobre uma base de dados diversificada foi possível obter uma taxa de acerto de 98% por meio da utilização de algoritmos baseados em árvores de decisão (*Random Commitee* e *Random Tree*). Com uma boa acurácia, é possível inferir qual conjunto de parâmetros associado ao CBC levaria à melhor solução possível para um dado problema.

Vale ressaltar que existe uma limitação quanto à recomendação dos parâmetros: foram selecionadas e utilizadas dezesseis combinações entre um vasto conjunto de parâmetros. Nesse sentido, destacam-se como trabalhos futuros a inclusão de novas combinações de parâmetros, a fim de ampliar a base de dados, refinar os resultados obtidos e obter uma boa taxa de acerto para novas instâncias. Além disso, pretende-se desenvolver uma aplicação que faça a recomendação dos parâmetros de forma independente da ferramenta WEKA, para que esse processo possa ser integrado diretamente com o uso do CBC. Por fim, pretende-se investigar a importância de cada atributo no processo de classificação, especialmente cada parâmetro e sua influência direta na produção de uma solução de qualidade.

Referências

Achterberg, T., Koch, T., e Martin, A. (2005). Branching rules revisited. *Operations Research Letters*, 33(1):42 – 54. ISSN 0167-6377. URL http://www.sciencedirect.com/science/article/pii/S0167637704000501.

Caprara, A. e Fischetti, M. (1996). {0, 1/2}-chvátal-gomory cuts. *Mathematical Programming*, 74 (3):221–235. ISSN 1436-4646. URL http://dx.doi.org/10.1007/BF02592196.

Fayyad, U., Piatetsky-Shapiro, G., e Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34. ISSN 0001-0782. URL http://doi.acm.org.ez28.periodicos.capes.gov.br/10.1145/240455.240464.

Fischetti, M. e Monaci, M. (2014). Proximity search for 0-1 mixed-integer convex programming. *Journal of Heuristics*, 20(6):709–731. ISSN 1572-9397. URL http://dx.doi.org/10.1007/s10732-014-9266-x.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. E., e Wolter, K. (2011). MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163.

Lougee-Heimer, R. (2003). The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66. ISSN 0018-8646.

Witten, I. H., Frank, E., e Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition. ISBN 0123748569, 9780123748560.