ALGORITMOS, EXPERIMENTAÇÃO E TEORIA EM OTIMIZAÇÃO COMBINATÓRIA

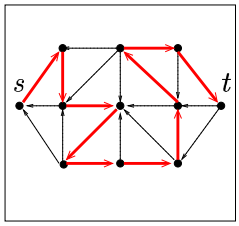
JULIANA BARBY SIMÃO

Orientador: José Coelho de Pina

RESUMO. Neste projeto de iniciação científica pretendemos estudar vários aspectos em otimização combinatória. Desejamos analisar e implementar algoritmos, realizar experimentos e fazer um estudo integrado de algoritmos e teoria (combinatória poliédrica). Esperamos que ao final do projeto os alunos envolvidos tenham bastante familiaridade com algumas das técnicas mais fundamentais em otimização combinatória para um possível futuro mestrado na área.

Observação. Juntamente com este pedido, estamos solicitando bolsa de iniciação científica para o aluno Marcelo Hashimoto.

1. INTRODUÇÃO



Otimização combinatória é um campo da matemática aplicada que usa técnicas de combinatória, programação matemática e teoria de algoritmos para resolver problemas em otimização sobre estruturas discretas.

Problemas em otimização combinatória têm sido um tópico central para a evolução de algoritmos e da teoria de complexidade computacional. Pesquisadores têm apresentado muitas idéias criativas para o projeto de algoritmos eficientes baseados em conceitos e resultados na área. Métodos desenvolvidos para problemas em fluxos em redes, como o primal-dual, têm se mostrado muito úteis no projeto e análise de uma variedade de algoritmos para problemas em outros domínios. Muitas das idéias inovadoras têm se baseado em um conjunto não muito grande de princípios comuns que são simultaneamente simples e poderosos (como *scaling*).

Neste projeto pretendemos estudar algumas das ferramentas mais fundamentais em otimização combinatória. Desejamos analisar e implementar algoritmos para diversos problemas clássicos, principalmente em fluxos em redes e emparelhamentos. O nosso guia nessa jornada será o livro de Ahuja, Magnanti e Orlin [1], com o livro de Cook, Cunningham, Pulleyblank e Schrijver [7] como um guia secundário. Para os tópicos mais teóricos consultaremos os livros de Schrijver [23, 25]. Criaremos um sítio na internet para armazenar todas as implementações, textos e animações produzidas durante a execução do projeto.

O projeto contará ainda com a colaboração dos alunos Marcelo Hashimoto, Nelson Guedes Paulo Junior e Roger Ricardo Flores de Araujo. Todos são

Date: 8 de fevereiro de 2005.

estudantes do quarto e último ano do Bacharelado em Ciência da Computação (BCC) do IME-USP e já iniciaram seus trabalhos neste projeto. O Nelson será o responsável pela animação dos algoritmos e a Juliana, o Marcelo e o Roger farão implementações. Cada um deles deverá redigir um texto que se tranformará em trabalho de formatura.

2. Objetivos

Neste projeto o foco será o estudo, implementação e experimentação de algoritmos para problemas em otimização combinatória, especialmente para problemas de fluxos em redes e emparelhamentos. Desejamos descrever os algoritmos dando especial ênfase a seus invariantes. As implementações serão um rico laboratório para testarmos a nossa compreensão dos algoritmos, métodos e estruturas de dados envolvidas. A nossa filosofia será a de que os programas devem fornecer evidências suficientes que justifiquem as suas respostas e permitam que as mesmas sejam verificadas.

Aspectos de combinatória poliédrica, como o método primal-dual e unimodularidade total, serão considerados.

Criaremos um sítio (*site*) contendo todas as implementações e textos produzidos pelos alunos e *links* para outros sítios de Otimização Combinatória. Ficaríamos muito felizes se produzissemos um sítio nos moldes da *Network Optimization Library* de Andrew V. Goldberg [12].

Na graduação e pós-graduação em Ciência da Computação do IME-USP é oferecida a disciplina MAC0325/MAC5781 Otimização Combinatória. Gostaríamos que as animações feitas pelos alunos envolvidos no projeto pudessem ser usadas nas aulas de MAC0325/MAC5781 para ilustrar o funcionamento dos algoritmos.

Finalmente, é nossa intenção que esta iniciação científica prepare a Juliana para um eventual mestrado em otimização combinatória. Entre os tópicos de mestrado já sugeridos à aluna estão alguns da lista de favoritos do orientador: algoritmos para fluxos submodulares [10, 15] e algoritmos para minimizar funções submodulares [11, 14, 24].

3. Material e métodos

Alguns dos ingredientes do estudo que será feito são brevemente descritos a seguir.

3.1. **Análise experimental.** Recentemente, há um grande interesse em trabalhos relacionados a análise experimental de algoritmos [4, 6, 2, 3, 5, 13, 22]. Algumas conferências foram especialmente criadas para tratar do assunto (ALENEX, WEA).

O interesse em experimentação é devido ao reconhecimento de que os resultados teóricos, freqüentemente, não trazem informações referentes ao desempenho do algoritmo na prática. Segundo Johnson [16], há quatro motivos básicos que levam a realizar um trabalho de implementação de um algoritmo:

(1) Para usar o código em uma aplicação particular, cujo propósito é descrever o impacto do algoritmo em um certo contexto;
(2) Para proporcionar evidências da superioridade de um algoritmo;

(3) Para melhor compreensão dos pontos fortes, fracos e do desempenho das operações algorítmicas na prática; e
(4) Para produzir conjecturas sobre o comportamento do algoritmo no caso-médio sob distribuições específicas de instâncias onde a análise probabilística direta é muito difícil.

Neste projeto o nosso maior interesse é (3).

3.2. **Linguagem algorítmica e invariantes.** A linguagem algorítmica adotada neste projeto será a empregada por Feofiloff [8, 9]. Abaixo encontra-se um exemplo onde esta notação é utilizada.

> **Algoritmo** BUSCA-SEQUENCIAL(n, v, x). Recebe um inteiro positivo n, um vetor de números inteiros $v[1..n]$ e um inteiro x e devolve TRUE se existe um indíce i em $[1..n]$ tal que $v[i] = x$, caso contrario devolve FALSE.

O algoritmo é iterativo e no início de cada iteração tem-se um inteiro i em $[1..n+1]$. No início da primeira iteração $i = 1$.

Cada iteração consiste no seguinte.

Caso 1: $i = n + 1$.
Devolva FALSE e pare.
Caso 2: $i \leq n$.
Caso 2A: $v[i] = x$
Devolva TRUE e pare.
Caso 2B: $v[i] \neq x$
$i' := i + 1$.
Comece uma nova iteração com i' no papel de i. □

A ordem em que os casos são enunciados é irrelevante: em cada iteração, qualquer um dos casos aplicáveis pode ser executado. Os casos podem não ser mutuamente exclusivos, e a definição de um caso *não* supõe implicitamente que os demais não se aplicam. Serão utilizadas ainda expressões como "Escolha um i em $[1..n]$", quando não faz diferença qual o valor escolhido. Portanto, a descrição de um algoritmo não é completamente determinística.

A correção dos algoritmos descritos serão baseadas em demonstrações da validade de invariantes. Estes invariantes são afirmações envolvendo objetos mantidos pelo algoritmo que são válidas no início de cada iteração. Exemplos de invariantes para o algoritmo descrito acima são:

(i1) i é um valor em $[1..(n+1)]$.
(i2) para todo j em $[1..(i-1)]$ vale que $v[j] \neq x$.

Deve-se demonstrar que os invariantes valem no início de cada iteração; não faremos isto para o presente exemplo.

Invariantes nos ajudam a entender e demonstrar a correção de algoritmos. No exemplo em questão vê-se facilmente que quando o algoritmo devolve TRUE, no caso 2A, ele não está mentindo, ou seja, existe i em $[1..n]$ tal que $v[i] = x$. Ademais, quando $i = n + 1$, do invariante (i2), tem-se que $v[j] \neq x$ para todo j em $[1..(i-1)] = [1..n]$, e portanto, no caso 1 o algoritmo corretamente devolve FALSE. Finalmente, o algoritmo pára, já que em cada iteração em que não ocorrem os casos 1 e 2A, o caso 2B ocorre e portanto o valor de i e acrescido de 1.

3.3. **Plataformas combinatórias.** O *Stanford Graph Base* [17] (SGB) e o *Library of Eficient Data Types and Algorithms* [21] (LEDA) são plataformas para algoritmos combinatórios. No SGB um grafo é representado internamente através de listas de adjacências e no LEDA ainda não sabemos como é a representação.

Ainda não decidimos em que linguagem e plataforma serão feitas as implementações. Muito provavelmente utilizaremos C junto com o SGB.

3.4. **Programação literária.** Donald Knuth descreve programação literária da seguinte maneira:

"Programação literária é uma metodologia que combina linguagem de programação com documentação, deste modo, é possível fazer programas mais robustos, mais portáveis, mais facilmente alteráveis, e mais divertidos de se escrever do que programas escritos somente em linguagem de alto nível. A principal idéia é tratar um programa como parte da literatura, direcionado mais aos seres humanos do que aos computadores. O programa também é visto como um documento hipertexto ..."

O sistema de programação literária CWEB concebido por Knuth e Levy [18] combina programação em C com documentação tipografada em LaTeX. Grande parte das implementações da plataforma LEDA, e todas do SGB, foram feitas usando essa metodologia.

Tentaremos fazer as implementações utilizando a metodologia de programação literária.

3.5. **Estruturas de dados.** Algoritmos combinatórios exigem a manutenção e manipulação eficiente de dados como conjuntos, grafos, filas, etc. A escolha do esquema de representação dos dados na memória do computador tem um efeito considerável no consumo de tempo de algoritmo e de suas implementações. Os algoritmos que serão estudados utilizam várias estruturas de dados importantes, tais como filas (*d-heaps, Fibonacci Heaps, buckets*), tabelas de espalhamento (*hash tables*) e árvores dinâmicas (*dynamic trees*) [26].

3.6. **Combinatória poliédrica.** Combinatória poliédrica trata das aplicações de métodos de programação linear, especialmente dualidade, à resolução de problemas em otimização combinatória. O teorema da dualidade de programação linear é uma ferramenta extremamente útil para tratar problemas práticos e teóricos. Sempre que um problema pode ser formulado como um programa linear, o teorema da dualidade dá uma nova perpectiva do problema. Aqueles problemas oriundos de combinatória têm a restrição adicional de que as váriáveis devem ser inteiras. Muitos resultados clássicos em combinatória afirmam que para certas classes de programas inteiros o teorema da dualidade permanece válido. O teorema de Hoffman-Kruskal sobre matrizes unimodulares é um exemplo de obtenção de diferentes 'resultados minmax' em combinatória de uma só vez. Neste projeto, sempre que for oportuno, consideraremos questões do tipo: "Se um programa linear e alguns programas relacionados admitem solução inteira então o programa dual admite solução inteira". Muitos teoremas minmax em combinatória podem ser obtidos dessa maneira [19, 20, 25].

4. Plano de trabalho e cronograma de atividades

O desenvolvimento deste projeto não será verdadeiramente 'linear'. Ramificações em várias direções são sempre possíveis, freqüentemente desejáveis e muitas vezes inevitáveis. Dependendo do interesse, cada aluno pode dedicar-se mais a implementação, descrição dos algoritmos e seus invariantes ou ainda combinatória poliédrica.

O "livro texto" deste projeto é *Network Flows* de Ahuja, Magnati e Orlin [1], com o apóio de alguns outros livros [7, 23, 25]. Estudaremos e implementaremos algoritmos contidos nos seguintes capítulos:

6 Maximum flows: basic ideas
7 Maximum flows: polynomial algorithms
9 Minimum cost flows: basic algorithms
10 Minimum cost flows: polynomial algorithms
11 Minimum cost flows: network simplex
13 Minimum spanning trees
14 Convex cost flows
18 Computational testing of algorithms

Para os primeiros seis meses o plano de trabalho da Juliana e do Marcelo são idênticos. Ambos estudarão os capítulos $6, 7, 9, 10, 11$ e 18 do livro *Network Flows*. As implementações dos algoritmos estudados será dividida entre eles.

Nos seis meses finais do projeto a Juliana concentrará seus estudos nos capítulos 13 e 14 (enquanto o Marcelo se dedicará ao estudo dos capítulos 15 Generalized flows e 16 Lagragian relaxation and network optimization).

Na tabela a seguir apresentamos um cronograma mostrando uma expectativa de cumprimento do estudo dos capítulos acima, nos diversos meses do projeto.

Atividade (capítulo)	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ	JAN	FEV
6	√	√										
7		√	√									
9			√	√								
10				√	√							
11					√	√	√					
13							√	√	√			
14										√	√	√
18	√	√										

5. Sobre a estudante

A Juliana é uma excelente estudante do quarto ano do Bacharelado em Ciência da Computação (BCC) do IME-USP, um dos cursos mais concorridos do vestibular da FUVEST. Ela tem média aproximadamente 9,5 nas disciplinas que cursou e não obteve nenhuma reprovação, o que, pelos padrões do BCC, é tido como um desempenho muito bom. No 2o. semestre de 2003 a Juliana trancou a disciplina optativa MAC0436 Tópicos em Matemática Discreta, pois já estava cursando outras 7 disciplinas.

A Juliana se graduará ao final de 2004. Já no 1o. semestre deste ano ela cursará, como aluna especial, a disciplina de pós-graduação MAC5770 Introdução à Teoria dos Grafos.

As disciplinas do BCC que já foram concluídas pela Juliana a qualificam plenamente para levar adiante um projeto como este. Entre estas disciplinas incluiem-se:

MAC0122 Príncipios de Desenvolvimento de Algoritmos;
MAC0315 Programação Linear;
MAC0211 Laboratório de Programação;
MAC0323 Estruturas de Dados;
MAC0328 Algoritmos em Grafos; e
MAC0338 Análise de Algoritmos.

Os programas destas disciplinas e a grade curricular do BCC podem ser vistos em `http://www.ime.usp.br/dcc/grad/`

Referências

[1] R.K. Ahuja, T.L. Magnanti e J. Orlin, *Network flows: Theory, algorithms, and applications*, Practice Hall, 1993.

[2] C.S. Chekuri, A.V. Goldberg, D.R. Karger, M.S. Levine e C. Stein, *Experimental study of minimum cut algorithms*, Tech. Report Technical Report 96-132, NEC Research Institute, Inc, 1996, `http://www.avglab.com/andrew/soft.html`.

[3] B.V. Cherkassky e A.V. Goldberg, *Negative-cycle detection algorithms*, Tech. Report Technical Report 96-029, NEC Research Institute, 1996, `http://www.avglab.com/andrew/soft.html`.

[4] B.V. Cherkassky, A.V. Goldberg e T. Radzik, Shortest paths algorithms: Theory and experimental evaluation, *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[5] B.V. Cherkassky, A.V. Goldberg, J.C. Setubal e J. Stolfi, *Augment or push? a computational study of bipartite matching and unit capacity flow algorithms*, Tech. Report Technical Report 96-029, NEC Research Institut, 1998, `http://www.avglab.com/andrew/soft.html`.

[6] B.V. Cherkassky, A.V. Goldberg e C. Silverstein, Buckets, heaps, lists and monotone priority queues, *SIAM J. Comput.* **28** (1999), 1326–1346.

[7] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank e A. Schrijver, *Combinatorial otimization*, Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Son's, New York, 1998.

[8] P. Feofiloff, *Notas de aula de MAC 5781 otimização combinatória*, `"http://www.ime.usp.br/~pf/"`, 1997.

[9] ———, *Algoritmos de programação linear*, EDUSP, 2000.

[10] L. Fleischer, S. Iwata e S.T. McCormick, A faster capacity scaling algorithm for minimum cost submodular flow, *Mathematical Programmming Ser. A* **92** (2002), no. 1, 119–139.

[11] S. Fujishige, Submodular function minimization and related topics, *The Second Japanese-Sino Optimization Meeting, Part I* (Kyoto), 2003, pp. 167–180.

[12] A.V. Goldberg, *Network otimization library*, http://www.avglab.com/andrew/soft.html.

[13] A.V. Goldberg e C. Silverstein, *Implementation of Dijkstra's algorithm based on multilevel buckets*, Tech. report, NEC Research Institute, Princeton, NJ, 1995.

[14] S. Iwata, A fully combinatorial algorithm for submodular function minimization, *Journal Combinatorial Theory Ser. B* (2002), no. 2, 203–212.

[15] S. Iwata, S.T. McCormick e M. Shigero, A fast cost scaling algorithm for submodular flow, *Information Processing Letters* **74** (2000), no. 3-4, 123–128.

[16] D.S. Johnson, A theoretician's guide to the experimental analysis of algorithms, *To appear in Proceedings of the 5th and 6th DIMACS Implementation Challenges*, 2002.

[17] D.E. Knuth, *The Stanford GraphBase: A plataform for combinatorial computing*, ACM Press, 1993.

[18] D.E. Knuth e S. Levy, *The CWEB System of Structured Documentation*, Addison-Wesley, 1994.

[19] L. Lovász, Certain duality principles in integer programming, *Studies in integer programming* (B.H. Korte P.L. Hammer, E.L. Johnson e G.L. Nemhauser., eds.), Annals of Discrete Mathematics, vol. 1, North-Holland, 1977, pp. 363–374.

[20] ———, Graph theory and integer programming, *Discrete Optimization I* (P.L. Hammer, E.L. Johnson e B.H. Korte, eds.), Annals of Discrete Mathematics, vol. 4, North-Holland, 1979, pp. 146–158.

[21] K. Mehlhorn e St. Näher, *The LEDA platform of combinatorial and geometric computing*, Cambridge Press, 1997, `http://www.mpi-sb.mpg.de/~mehlhorn/LEDAbook.html`.

[22] S. Pettie, V. Ramachandran e S. Sridhar, Experimental evaluation of a new shortest path algorithm, *4th Workshop on Algorithm Engineering and Experiments (ALENEX'02)*, 2002, pp. ??–??

[23] A. Schrijver, *Theory of integer and linear programming*, John Wiley & Son's, New York, 1986.
[24] ———, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *Journal of Combinatorial Theory Ser. B* **80** (2000), no. 2, 346–355.
[25] ———, *Combinatorial optimization: Polyhedra and efficiency*, Algorithms and Combinatorics, vol. 24, Springer, 2003.
[26] R.E. Tarjan, *Data structures and network algorithms*, BMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA, 1983.

José Coelho de Pina

Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010, 05508–900 São Paulo, SP
E-mail address: julianab@linux.ime.usp.br, coelho@ime.usp.br