Projeto de Pesquisa

Prof. Carlos Alberto de Jesus Martinhon (IC/UFF)

Algoritmos Randômicos em Grafos e Otimização

(Atualizado em 03/2005)

Depto de Ciência da Computação – DCC
Instituto de Computação - IC
Universidade Federal Fluminense – UFF

ALGORITMOS RANDÔMICOS EM GRAFOS E OTIMIZAÇÃO

I. Introdução

Embora se conheça aplicações envolvendo algoritmos randômicos desde épocas primitivas (Shallit[1992]) os primeiros artigos sobre este assunto datam do final da década de 70 com os trabalhos de Rabin[1976] e Solavay e Strassen[1977] para o problema do reconhecimento de números primos (*Primality Test*). As décadas de 1980 e 1990 testemunharam, a partir de então, um enorme crescimento da área de algoritmos randômicos. Eles emergiram de aplicações voltadas unicamente à teoria dos números e geometria computacional para problemas nas mais diversas áreas de interesse. Uma gama enorme de pesquisadores tem utilizado, cada vez mais, técnicas e ferramentas oriundas de modelos probabilísticos, sejam eles seqüenciais ou paralelos. Como exemplo, pode-se citar aplicações em algoritmos *on-line*, otimização combinatória, criptografia, geometria computacional, teoria dos números, estrutura de dados, processamento paralelo e distribuído entre outras (Motwani&Raghavan [1995], Karp[1991], Gupta et al.[1994]). Este projeto trata essencialmente, da utilização de algoritmos randômicos aplicados a problemas em grafos e otimização (NP-difíceis ou não).

Problemas pertencentes à Classe P podem ser tratados convenientemente por métodos probabilísticos. Técnicas como a Seleção Randômica (*Random Selection*) ou Amostra Randômica (*Random Sampling*), foram utilizadas com sucesso para problemas clássicos de otimização. O objetivo neste caso, é usufluir da maior flexibilidade existente nos modelos probabilísticos reduzindo ainda mais a complexidade teórica obtida pelos melhores algoritmos determinísticos. Como exemplo, pode-se citar o problema da Árvore Geradora Mínima, Corte-Mínimo em Grafos, Fluxo Máximo (Karger&Stein[1993], Karger[1995], Karger et. al.[1995] e King *et. al*.[1993]), Geração de Orientações Acíclicas em Grafos (Arantes, França e Martinhon[2003]) entre outras.

Problemas combinatórios NP-difíceis, podem também ser resolvidos eficientemente através de técnicas probabilísticas. Analogamente aos algoritmos aproximativos puramente determinísticos, nos algoritmos randômicos aproximativos, buscamos uma solução cuja distância ao valor da solução ótima do problema original possa ser definida a *priori*. Neste caso, espera-se obter um algoritmo randômico cujo tempo esperado seja polinomial e cujo valor esperado da solução heurística seja suficientemente "próximo" do valor ótimo. Nos algoritmos randômicos aproximativos, normalmente, as probabilidades são definidas a partir de relaxações do problema combinatório considerado (Programação Linear ou Semidefinida). Utiliza-se então o método de Monte Carlo para minimização da probabilidade de falha ou constrói-se uma solução determinística através de técnicas de "derandomização" (*Derandomization Techniques*).

Os modelos probabilísticos (Máquina RAM probabilística ou Máquina de Turing probabilística) representam uma extensão natural dos modelos puramente determinísticos. Sua utilização e aplicação na resolução de problemas algorítmicos constitui um tema fascinante de pesquisa e que vem despertando, cada vez mais, o interesse da comunidade científica.

II – Algoritmos Randômicos

Pode-se dizer que um algoritmo determinístico se caracteriza, essencialmente, por uma seqüência finita de "passos elementares" voltados para a resolução de um determinado problema. Tipicamente, o tempo de execução de um algoritmo determinístico é sempre fixo, ou seja, sucessivas repetições do algoritmo aplicadas a uma mesma entrada resultam sempre em uma mesma saída com tempo de processamento sempre idêntico. O mesmo não ocorre, por exemplo, com os algoritmos randômicos ou probabilísticos onde cada execução poderá produzir uma saída diferente baseada em eventos aleatórios (tempo esperado de processamento). Nesta situação, uma fonte de *bits* randômicos é utilizada com o propósito de realizar escolhas aleatoriamente. Em um algoritmo randômico, o tempo de processamento e/ou os resultados obtidos são "função randômica" da entrada. Surpreendentemente, para uma grande quantidade de problemas, a utilização de algoritmos randômicos se constitui na forma mais simples e/ou mais rápida de implementação! Nestes casos, sua utilização implica em uma melhora de desempenho e eficiência quando comparada aos algoritmos puramente determinísticos!

Essencialmente, dois tipos principais de algoritmos randômicos são referenciados na literatura: o Método de Monte Carlo e o Método de Las Vegas. No método de Monte Carlo, pode-se gerar soluções com probabilidade de falha arbitrariamente pequena, ao contrário do Método de Las Vegas onde uma resposta correta deverá ser sempre obtida com tempo esperado de processamento idealmente polinomial.

Vale destacar que o comportamento de um algoritmo randômico poderá variar, mesmo quando aplicado repetidas vezes para uma mesma entrada do problema! Desta forma, o tempo de processamento se torna uma variável aleatória e a análise de tempo de processamento exige que se tenha uma maior compreensão da sua distribuição de probabilidade associada. Finalmente, pode-se dizer que, em um algoritmo randômico, o valor esperado irá depender de escolhas aleatórias feitas no algoritmo e não de distribuições impostas sobre a entrada de dados. Este comportamento diferencia o tempo esperado de processamento, presente nos algoritmos randômicos, da complexidade de caso médio, normalmente utilizada na análise de algoritmos determinísticos.

II.1 – Métodos de Monte Carlo e Las Vegas

Além dos comandos e estruturas usuais de repetição, decisão e atribuição presente nos algoritmos determinísticos, os algoritmos randômicos ou probabilísticos se caracterizam, fundamentalmente, pela geração de números "aleatórios" ou mais precisamente, pela geração de números pseudo-aleatórios. Como discutido anteriormente, este conceito de algoritmo vai contra àquele de algoritmo puramente determinístico ou de função, onde a cada elemento do domínio (*input*) corresponde um único elemento da imagem (*output*). Nos algoritmos determinísticos, a solução obtida e o tempo de processamento nunca se alteram a cada nova repetição do algoritmo. Ao contrário, nos algoritmos randômicos, a qualidade da solução, o tempo de processamento, ou ambos, definem variáveis aleatórias obedecendo algum tipo de distribuição estatística.

O método de Las Vegas sempre produz uma solução correta com erro *zero*. O método de Monte Carlo, por outro lado, retorna uma resposta correta com probabilidade de fracasso arbitrariamente pequena. Desta forma, ao permitir uma probabilidade de erro não nula, estaremos gozando de uma maior flexibilidade no cálculo da expressão de complexidade.

O método de Las Vegas sempre assegura uma solução correta para o problema considerado. Entretanto, ela nem sempre é obtida dentro de um tempo de execução suportável. Essa abordagem é

interessante para aqueles problemas cujo tempo esperado de processamento seja suficientemente pequeno (ou polinomial) para a aplicação em questão. No método de Las Vegas, executamos sucessivas repetições do mesmo algoritmo até que uma solução correta seja encontrada. Desta forma, seu tempo de processamento pode ser definido convenientemente por uma variável aleatória com distribuição geométrica. Assim, cada nova repetição do algoritmo deverá conter uma etapa de exibição aleatória da solução e uma etapa de reconhecimento desta solução. Desta forma, se a probalidade de sucesso é igual a p, seu tempo esperado de processamento será igual a $1/p$ (valor esperado da distribuição geométrica). Um algoritmo de Las Vegas será considerado *eficiente* se seu tempo esperado (complexidade esperada) for polinomial no tamanho do problema.

No caso de problemas de decisão, essencialmente, existem dois tipos de algoritmo de Monte Carlo: algoritmos com erro unilateral (*one-sided error*) e algoritmos com erro bilateral (*two-sided error*). Um algoritmo de Monte Carlo tem erro unilateral, se a probabilidade de fracasso é *zero*, para pelo menos uma das saídas *Sim* ou *Não*. Se a probabilidade de falha é não-nula para as saídas *Sim* e *Não*, o algoritmo possui erro bilateral. Podemos dizer que o método de Las Vegas aplicado a problemas de decisão, possui erro *zero* para *Sim* e *Não* respectivamente (*zero-sided error*). Problemas de decisão resolvidos por algoritmos polinomiais com erro unilateral para as respostas *Sim* e *Não*, pertencem às classes RP e co-RP respectivamente (*Randomized Polinomial Classes*), já os problemas resolvidos por algoritmos com erro bilateral pertencem à Classe BPP dos problemas de decisão (*Bounded Probabilistic Polinomial time*). Problemas resolvidos polinomialmente por algoritmos de Las Vegas (erro zero) pertencem à Classe ZPP (*Zero-error Probabilistic Polinomial time*)

Outra característica interessante a ser destacada, é que o método de Las Vegas pode ser visto como um caso particular do método de Monte Carlo. Isto ocorre sempre que a probabilidade de fracasso for igual a *zero*. Quando a probabilidade de falha no método de Monte Carlo for estritamente positiva, pode-se transformar Monte Carlo em Las Vegas adicionando-se um procedimento (idealmente polinomial) de checagem da solução. Sempre que a solução estiver incorreta, o processo será repetido até que uma solução exata seja encontrada.

Para exemplificar a situação descrita acima, considere um problema π com uma instância I de tamanho n. Além disso, considere um algoritmo de Monte Carlo A com tempo esperado de processamento $T_1(n)$ e probabilidade de acerto igual a $p(n)$. Suponha que, gerada uma solução para π, a corretude ou não desta solução possa ser constatada em tempo $T_2(n)$. O algoritmo de Las Vegas, neste caso, consiste na simples repetição de A até que uma resposta correta seja obtida. Em outras palavras, uma solução correta (sucesso) é gerada satisfazendo uma distribuição geométrica com valor esperado $1/p(n)$. Como o tempo de processamento associado a cada repetição é $T_1(n)+T_2(n)$ tem-se que $(T_1(n)+T_2(n))/p(n)$ irá representar o tempo esperado de processamento no método de Las Vegas. Apesar de interessante, esse tipo de abordagem nem sempre pode ser aplicado na construção do método de Las Vegas (a partir do método de Monte Carlo). Na verdade, não existe um procedimento padrão que se aplique a todos os casos. Já a transformação de Las Vegas em Monte Carlo é mais simples e pode ser realizada, interrompendo-se o método de Las Vegas em uma dada iteração do procedimento.

III – Algoritmos Aproximativos em Otimização Combinatória

Em função da elevada complexidade computacional necessária na solução de uma grande classe de problemas combinatórios, denominados problemas NP-árduos ou NP-difíceis, muitos dos algoritmos propostos para sua solução podem ou não buscar diretamente um ótimo global (solução

ótima exata). Nestes casos, deve-se adotar uma estratégia que vise equilibrar, essencialmente, a qualidade da solução obtida com o tempo total de processamento.

A utilização de métodos exatos, por exemplo, se justifica apenas em circunstâncias especiais, onde as instâncias consideradas para um determinado problema são suficientemente pequenas ou o tempo de processamento que se tem disponível é adequado o bastante para a aplicação considerada. Além disso, em muitos problemas práticos, os dados de entrada são conhecidos apenas parcialmente. Isto significa que uma solução ótima dispendiosa em termos de tempo não se justifica em relação às soluções suficientemente "próximas" da ótima e obtidas a um baixo custo computacional. Já nas heurísticas ou metaheurísticas, uma solução viável obtida em tempo polinomial e idealmente "próxima" da solução ótima é procurada. Embora possuam uma abordagem bastante interessante, e sejam utilizadas eficientemente em uma grande quantidade de problemas práticos, elas carecem ainda de formalismo matemático. Os trabalhos de Dell'Amico, Maffioli e Martello[1997] e Aarts&Lenstra[1997] representam uma ótima compilação de bibliografias destas técnicas aplicadas a problemas de Otimização Combinatória.

Para uma grande quantidade de problemas NP-difíceis, será possível constatar a qualidade da solução apresentada utilizando-se técnicas apropriadas como os algoritmos determinísticos ou randômicos. Neste caso, o que se pretende é definir uma "distância" do valor ótimo mesmo sem conhecê-lo explicitamente (razão de aproximação). Nos algoritmos aproximativos, estaremos interessados na determinação de soluções heurísticas polinomiais cuja a razão de aproximação seja a menor possível, independentemente das instâncias consideradas.

Os algoritmos aproximativos (especialmente os determinísticos aproximativos), foram introduzidos por Johnson[1974], e são algoritmos polinomiais que buscam sacrificar o mínimo possível da qualidade, obtida nos métodos exatos, ganhando, simultaneamente, o máximo possível em eficiência (tempo polinomial). Como discutido em Hochbaum[1997], a busca do equilíbrio entre estas situações conflitantes é o grande paradigma dos algoritmos aproximativos.

Antes do surgimento dos algoritmos aproximativos a análise de desempenho dos métodos heurísticos se baseava, simplesmente, em sua execução para um conjunto de finito de instâncias (*benchmark*). A performance da heurística era então comparada com a de outras heurísticas para o mesmo conjunto de instâncias considerado. Este tipo de comparação, ainda hoje bastante utilizado, retorna apenas uma medida parcial de desempenho já que o conjunto de instâncias é normalmente pequeno, além de não representar, satisfatoriamente, o conjunto de todas as instâncias associadas ao problema. Em outras palavras, uma heurística com um bom desempenho para um dado conjunto de instâncias não mantém, necessariamente, a mesma performance quando aplicada a outro conjunto de instâncias com características distintas.

Os algoritmos aproximativos podem ser subdivididos, basicamente, em determinísticos ou randômicos. A diferença central nestes dois tipos de abordagem reside no fato de que, no caso determinístico, execuções adicionais do procedimento aplicadas a uma mesma instância produzem sempre uma mesma saída com tempo de processamento sempre idêntico. Por outro lado, no caso probabilístico, solução gerada e tempo de processamento se modificam a cada nova repetição do procedimento, sendo, por isso, melhor representadas por variáveis aleatórias discretas.

III.1 – Algoritmos Aproximativos Determinísticos

Seja Π um problema de otimização combinatória, e I, uma instância qualquer de Π. Se A é um algoritmo aproximativo (determinístico ou randômico) para Π, representaremos por $x_A(I)$ o valor da

função objetivo gerado por A, $\forall$ I$\in\Pi$. O valor da solução ótima associado será representado por $x^*(I)$.

Garey, Graham e Ullman [1972] e posteriormente Johnson [1974] formalizaram o conceito de algoritmos aproximativos. Como discutido anteriormente, um algoritmo aproximativo deverá, necessariamente, ser polinomial no tamanho de qualquer instância para o problema. Mais formalmente, um algoritmo determinístico A com solução $x_A(I)$ é *d-aproximado* para um problema de minimização (de maximização) Π se, e somente se, qualquer que seja a instância I do problema Π, a solução obtida é no máximo (no mínimo) d vezes o valor da solução ótima $x^*(I)$. Normalmente, em problemas de maximização assume-se que $x^*(I)>0$.

Alguns autores ressaltam que esta medida (convencional) de aproximação pode não ser muito precisa. Suponha, por exemplo, que a razão entre a pior solução e a melhor solução associada a uma dada instância I seja no máximo uma constante δ. Neste caso um algoritmo d-aproximado para o problema pode significar na verdade a pior solução (Hassin e Khuller[2001])! Para contornar esse problema existente na definição de medida convencional de aproximação, alguns autores propuseram independentemente outras medidas de performance, como a aproximação diferencial ou z-aproximação (vide, Demange e Paschos[1996], Demange et. al.[1998], Hassin e Khuller [2001]).

Maiores detalhes sobre a utilização de algoritmos aproximativos determinísticos na solução de problemas combinatórios podem ser encontrados em Hochbaum[1997]. O trabalho de Ausiello *et.al.*[1995] é uma ótima referência sobre classes de complexidade em algoritmos aproximativos.

III.2 – Algoritmos Randômicos Aproximativos:

Nos algoritmos randômicos aproximativos, solução esperada e tempo esperado de processamento (representados por variáveis aleatórias discretas) são parâmetros observados conjuntamente. Neste caso, um algoritmo randômico A de complexidade polinomial é δ-aproximado para um problema de minimização Π se, e somente se, $E(x_A(I)) \leq \delta.x^*(I)$, onde $\delta \geq 1$. Se Π é um problema de maximização então A é δ-aproximado se e somente se $E(x_A(I)) \geq \delta.x^*(I)$, onde $\delta \leq 1$.

Na determinação de uma solução randômica $x_A(I)$, pode-se, simplesmente, atribuir valores aleatóriamente às variáveis associadas ao problema original satisfazendo, obviamente, alguma distribuição estatística. Outra possibilidade bastante utilizada é a modelagem via Programação Linear ou Programação Semidefinida para a determinação de bons limitantes para o problema (inferiores ou superiores conforme o caso). Estes dois modelos, podem ser resolvidos por métodos de pontos interiores de complexidade polinomial (vide Karmarkar[1984], Wright[1997]) e as soluções relaxadas em ambos os casos, definirão probabilidades a serem utilizadas na etapa randômica.

Na etapa randômica são geradas soluções inteiras que poderão ser viáveis ou não. Neste caso, precisaremos garantir apenas que a probabilidade de obtenção de uma solução viável (probabilidade de sucesso) seja estritamente maior que *zero*. Sucessivas repetições da etapa randômica (utilizando-se as mesmas probabilidades definidas na relaxação) reduzem arbitrariamente a probabilidade de falha (método de Monte Carlo).

Esta técnica que é conhecida na literatura como Arredondamento Randômico (*Randomized Rounding*) e foi introduzida inicialmente por Raghavan&Thompson[1987]. Outra possibilidade (discutida mais adiante), é a construção de soluções determinísticas através do método

probabilístico introduzido por Erdos&Spencer[1974] (*Derandomization Techniques*). Neste caso, as probabilidades obtidas na relaxação são utilizadas diretamente na determinação de uma solução viável e com probabilidade de erro igual a *zero*. Os trabalhos de Srinavasan[1995], [2001] e Bertismas&Voha[1998], Motwani *et.al.*[1995] são ótimas referências sobre Arredondamento Randômico e Relaxação Linear na solução de problemas combinatórios. Maiores detalhes sobre a utilização de programação semidefinida e algoritmos randômicos aplicada a problemas de otimização combinatória podem ser obtidos em Goemans& Willianson[1994], [1995], Anderson[2000].

III.3 – Técnicas de *Derandomização*

Em algumas situações, será possível construir algoritmos determinísticos, com o auxílio de técnicas probabilísticas. Em outras palavras, deseja-se construir um algoritmo determinístico sem que se sacrifique muito da qualidade da solução e/ou o tempo de processamento obtidos no procedimento randômico. Apesar de não existir um termo apropriado em português para esta técnica, talvez o mais conveniente seja chama-la simplesmente de *derandomização* (do termo inglês *derandomization*).

De maneira geral, se um algoritmo randômico A possui um espaço de probabilidade (Ω,P) associado (onde Ω representa o espaço amostral e P é alguma medida de probabilidade), cada ponto em Ω irá corresponder a uma seqüência de *bits* executada pelo algoritmo. Seja $A(I,w)$, a execução de *A* para uma instância I onde *A* seleciona aleatóriamente $w \in \Omega$. Uma seqüência $w \in \Omega$ será considerada *boa* para uma dada instância I, se $A(I,w)$ calcula uma solução correta para o problema considerado. Assume-se geralmente que, se *w* é escolhida aleatóriamente satisfazendo uma distribuição uniforme, então a probabilidade de *w* ser *boa* para I é suficientemente alta. Nos métodos de *derandomização*, essencialmente, deseja-se determinar uma estratégia de busca em Ω de uma *boa* seqüência *w* com respeito a instância I. O problema, geralmente, é que o espaço de busca é exponencialmente grande tornando a busca exaustiva inviável.

Entre as técnicas de *derandomização* mais conhecidas podemos citar o método das probabilidades condicionais. Quando as probabilidades condicionais condicionais não puderem ser determinadas diretamente, pode-se utilizar então o método dos estimadores pessimístas introduzido por Raghavan[1988]. Esta técnica poderá ser bastante interessante principalmente quando combinada às desigualdades de Chernoff e Hoffding (Motwani&Raghavan[1995]).

Outras técnicas importantes de *derandomização* poderão ser utilizadas, como o método das expectâncias condicionais, *k-wise independence* entre outros. Para maiores detalhes vide Raghavan[1988], Erdös&Spencer[1974], Alon&Spencer[1992].

IV – Objetivos

Neste projeto, daremos uma atenção especial à utilização de algoritimos randômicos em grafos e otimização combinatória. Nos dedicaremos essencialmente ao estudo de técnicas e métodos probabilísticos aplicados a problemas de biologia computacional, Escalonamento de Tarefas, Redes Sociais, *Network Design*, entre outros.

Uma grande quantidade de aplicações importantes em Otimização Combinatória podem ser modeladas com o auxílio de grafos sanduíche. Dados dois grafos $G_1=(V,E_1)$ e $G_2=(V,E_2)$ tal que E_1

$\subseteq E_2$, dizemos que $G=(V,E)$ (onde $E_1 \subseteq E \subseteq E_2$) é grafo sanduíche para alguma propriedade π se $G=(V,E)$ satisfaz π. Uma grande quantidade de propriedades distintas, envolvendo problemas de decisão e otimização, podem ser consideradas dentro deste contexto (Golumbic *et.al.*[1995]). Como exemplo podemos citar o trabalho de Kaplan&Shamir[1996] para determinação do mapeamento físico de DNA (via grafos de intervalo sanduíche), o trabalho de Martinhon &Protti[2004] para determinação do Maior Conjunto Controlado-PMCC (*Max-Controlled Set Problem*), entre outros. Dando continuidade ao PMCC, pretendemos trabalhar ainda no desenvolvimento de um Esquema de Aproximação Polinomial para o problema. Neste caso, espera-se que uma solução arbitrariamente próxima do valor ótimo seja obtida em tempo polinomial no tamanho do problema.

Outro tema bastante interessante a ser explorado é o estudo de algoritmos randômicos aproximativos dentro do contexto de novas medidas de aproximação, como descritas em Demange e Paschos[1996], Demange et. al.[1998], Hassin e Khuller[2001]. Dentro desta linha, já tendo obtido alguns resultados preliminares, daremos continuidade ao *Fixed Linear Crossing Number Problem*, introduzido por em Masuda et. al. [1990] com várias aplicações no *layout* de circuítos. Podemos destacar ainda o estudo do teorema PCP – *Probabilistic Checkble Proofs* (Arora e Safra[1992]) e suas consequências na área de algoritmos aproximativos.

Esperamos finalmente, com este projeto, que novos resultados continuem sendo obtidos e que ele possa contribuir para que novos alunos e pesquisadores se dediquem ao estudo e desenvolvimento de técnicas e métodos probabilísticos em Ciência da Computação.

V - Referências Bibliográficas

[01] E. Aarts, J. Lenstra [1997] "Local Search in Combinatorial Optimization", John Wiley&Sons.

[02] N. Alon, J. Spencer [1992] "The Probabilistic Method", Wiley, New York.

[03] G. Andersson [2000] "Some New Randomized Approximation Algorithms", Doctoral Dissertation, Royal Institute of Technology - Stockholm / Swedan, Department of Numerical Analyis and Computer Science.

[04] G. Arantes, F. França, C. Martinhon [2002] "Algoritmos Randômicos para Geração de Orientações Acíclicas em Sistemas Distribuídos", Trabalho aceito para publicação na Revista Operacional.

[05] S. Arora, S. Safra [1992] "Probabilistic checking of proofs; a new caracterization of NP", Proc. 33rd Annual IEEE Symp. Found. Comput. Sci., 2-13.

[06] G. Ausiello, P. Crescenzi, M. Protasi [1995] "Approximate solution of NP optimization problems", Theoretical Computer Science 150, 1-55.

[07] Bertisimas and Voha [1998] "Rounding Algorithms for Covering Problems", Math. Programming, v.80 pp. 63-89.

[08] Bruneman[1974] "

[09] W.J. Cook, W.H. Cunninghan, W.R. Pulleyblank, A Schrijver [1998] "Combinatorial Optimization", Wiley-Interscience Series in Discrete Mathematics and Optimization.

[10] M. Demange, V. Paschos [1996] "On na approximation measure founded on the links between optimization and polynomial approximation theory", Theoretical Computer Science 158, 117-141.

[11] M. Demange, P. Grisoni and V. Paschos [1998] "Differential approximation algorithms for some combinatorial optimization problems", Theoretical Computer Science 209, Nos. 1 and 2, 107-122.

[12] P. Erdös, J. Spencer [1974] "The Probabilistic Method in Combinatorics", Academic Press, San Diego.

[13] J. Gill [1977] "Computational complexity of probabilistic Turing machines", SIAM J.

Comput. 6, 675-695.
[14] M.X. Goemans, D.P. Williamson [1995] "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming", In J. ACM., 42, 1115-1145.
[15] M. C. Golumbic, H. Kaplan, R. Shamir, [1994] "Graph sandwich problems", Journal of Algorithms, 19, pp. 449-473.
[16] R. Gupta, S.A. Smolka, S. Bhaskar [1994] "On randomization in sequential and distributed algorithms", ACM Computing Surveys, Vol 26, n.1.
[17] R. Hassin, S Khuler [2001] "z-approximations", Journal of Algorithms 41, 429-442.
[18] D.S Hochbaum [1997] "Approximation algorithms for NP-hard problems", PWS Publishing Company.
[19] D.S. Johnson [1974] "Approximation algorithms for combinatorial problems", J. Comput. System Sci., 9, 256-278.
[20] H. Kaplan, R. Shamir, [1996] "Bounded degree interval sandwich problems", Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems – ISTCS, pp. 195-201.
[21] D. R. Karger [1995] "Random sampling in graph optimization problems", PhD Dissertation - Department of Graduate Studies of Stanford University.
[22] D.R. Karger, C. Stein [1993] "An $\tilde{O}(n^2)$ algorithm for minimum cuts", In Proceedings of the 25th Annual ACM Symposium on Theory of Computing, pages 757-765.
[23] D.R. Karger, P. N. Klein, R. E. Tarjan [1995] "A randomized linear-time algorithm for finding minimum spanning trees", Journal of the ACM, 42 (2): 321-328.
[24] N. Karmarkar [1984] "A new polynomial-time algorithm for linear programming", Combinatorica, 4:373-395.
[25] R.M. Karp [1991] "An introduction to randomized algorithms", Discrete Applied Mathematics 34, 165-201, North-Holland.
[26] V.King, S. Rao, R.E. Tarjan [1993] "A faster deterministic maximum flow algorithm", In Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, pages 157-164.
[27] K. Makino, M. Yamashita and T. Kameda [2002] "Max-and min-neighborhood monopolies", Algorithmica, 34, 240-260.
[28] C.A. Martinhon [2002] "Randomized Algorithms in Combinatorial Optimization", (in Portuguese) Short course presented at XXXIV SBPO, pp. 1-122, ISSN 1518-1731.
[29] C.A. Martinhon, F. Protti[2004] "An Improved Derandomized Approximation Algorithm for the Max-Controlled Set Problem", WEA 2004, Lecture Notes in Computer Science 3059, pp. 341-355.
[30] S. Masuda, K Nakajima, T. Kashiwabara, T. Fujisawa [1990], "Crossing Minimization in Linear Embbedings of Graphs", IEEE Transactions on Computers, Vol. 39, n. 1.
[31] R. Motwani, P. Raghavan, [1995] "Randomized Algorithms", Cambridge University Press.
[32] R. Motwani, J. Naor, P. Raghavan,[1997] "Randomized approximation algorithms in combinatorial optimization", in Approximation Algorithms for NP-hard problems, Hochbaum (ed), PWS.
[33] G. Nemhauser, L. Wolsey [1988] "Integer and Combinatorial Optimization", John Wiley&Sons.
[34] M. Rabin [1976] "Probabilistic Algorithms", In J. F. Tranb, editor, Algorithms and complexity: New Directions and Recent results, pages 21-39, Academic Press.
[35] P. Raghavan [1988] "Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs", Journal of Computer and System Sciences 37, 130-143.
[36] P. Raghavan and C.D. Thompson [1987] "Randomized Rounding: Provably good algorithms and algorithmic proofs", Combinatorica 7, 365-374.
[37] J. Shallit [1992] "Randomized algorithms in "primitive cultures"", SIGACT News, 23(4):77-80.

[38] R. Solovay, V. Strassen [1977] “A fast Monte-Carlo test for primality” SIAM J. Comput. 6, 84-85.
[39] Spencer [1987] “Ten Lectures on the Probabilistic Method”, CBMS-NSF Regional Conference Series in App. Math., N.52, SIAM.
[40] A. Srinivasan [1995] “Improved Approximation Guarantees for Packing and Covering Integer Programs”, DIMACS Technical Report 95-37, September 1995.
[41] A. Srinivasan [2001] “Approximation algorithms via randomized rounding: a survey” , Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974-0636, USA.
[42] D.J.A. Welsh [1983] “Randomized algorithms”; Discrete Applied Mathematics 5, 133-145. North-Holland Publishing Company.
[43] L. A. Wolsey [1998] “Integer Programming”, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley&Sons.
[44] S.J. Wright [1997] “Primal-Dual Interior-Point Methods”; SIAM - Society for Industrial and Applied Mathematics.
[45] K. Yamamoto [2004] “A survey of randomized rounding techniques applied to the closest string problem” (in Portuguese), Master Thesis, Advisers: C. Martinhon and Helena Leitão.

Carlos Alberto de Jesus Martinhon
Prof. Adjunto – Detpo. Ciência da Computação
Instituto de Computação – UFF