Algoritmo Grasp para o Problema de Tabela-horário de Universidades

Walace S. Rocha
Universidade Federal do Espírito Santo
Vitóra – ES – Brasil
walacesrocha@yahoo.com.br

Maria C. S. Boeres
Departamento de Informática – Universidade Federal do Espírito Santo
Vitória – ES – Brasil
boeres@inf.ufes.br

Maria C. Rangel
Departamento de Informática – Universidade Federal do Espírito Santo
Vitória – ES – Brasil
crangel@inf.ufes.br

RESUMO

Este modelo resume as normas de formatação para os trabalhos completos a serem publicados nos Anais do XLI SBPO. O Resumo deve ter no máximo 150 palavras. Título, nomes dos autores, sua filiação e endereços, resumo e palavras-chave devem repetir fielmente o que foi informado quando o artigo foi cadastrado através do sistema JEMS. No primeiro upload, por ocasião da submissão do trabalho, exclua desta primeira página os nomes, instituições e endereços dos autores.

PALAVRAS CHAVE. Tabela-horário, Meta-heurística, Grasp, Área de classificação principal (escolher uma na janela de áreas).

ABSTRACT

This model summarizes the formatting rules for full papers accepted for inclusion ion the Proceedings of the XLI SBPO. The Abstract must have no more than 150 words Title, author names, affiliations, addresses, abstract and keywords must precisely mirror the information provided during registration. When you initially submit the paper, suppress author names, affiliations and addresses.

KEYWORDS. Timetabling, Metaheuristic, Grasp, Main area (choose between those in the areas window).

1. Introdução

Um dos problemas de grande interesse na área de otimização combinatória é o problema de tabela-horário. Dado um conjunto de disciplinas, alunos, professores e espaço físico, o problema consiste basicamente em alocar os horários das disciplinas levando-se em conta o espaço e horários disponíveis, além de respeitar algumas restrições divididas em fortes e fracas. Essas restrições surgem das particularidades de cada instituição. As fortes são aquelas que sempre devem ser satisfeitas, como por exemplo, exigir que um aluno não pode assistir mais de uma aula no mesmo horário. Uma tabela-horário é dita viável quando não viola nenhuma restrição forte. As fracas são restrições que não geram inviabilidade, mas refletem algumas preferências dos professores, alunos ou até mesmo da escola, por exemplo, penalizando uma tabela-horário que tenha aulas isoladas. Quanto mais restrições fracas forem satisfeitas, melhor será a tabela-horário. A formulação usado neste trabalho é a adotada pela competição ITC-2007.

A meta-heurística GRASP (Greedy Randomized Adaptive Search Procedure) é uma técnica de destaque no ramo de otimização combinatória. Ela tem sido aplicada em problemas de cobertura de conjuntos [Resende (1998)], satisfabilidade [Festa (2006)], árvore geradora mínima [de Souza (2002)], entre outros. Contudo, há pouca pesquisa desse algoritmo nos problemas de tabela-horário. Foram encontradas apenas implementações para a modelagem de escolas [Souza (2004)], [Vieira (2010)].

Este trabalho implementa um algoritmo GRASP para geração de tabela-horário usando a formulação do ITC-2007. O algoritmo possui uma fase inicial onde é garantido que uma tabela-horário viável será encontrada. Na segunda fase ocorre o melhoramento da tabela-horário através de busca local. Uma terceira fase de intensificação ocorre usando o procedimento *path-relinking*. Esse procedimento de três fases se repete algumas iterações, sempre gerando uma nova tabela. A melhor de todas encontradas é a resposta final.

Os resultados obtidos são comparados com as melhores soluções publicadas na literatura para este problema. Na seção 2. é apresentada a formulação do problema adotada neste trabalho. Na seção 3. são listadas algumas técnicas que tem sido usadas para resolver o problema de tabela-horário de universidades, tanto na formulação do ITC-2007 como em outras formulações. Na seção 4. são apresentados três algoritmos implementados para o problema: algoritmo genético, simulated annealing e o Grasp. Na seção 5. são apresentados os resultados obtidos com as diferentes técnicas. Os resultados são confrontados com as melhores respostas para o problema encontradas na literatura. Por fim, a seção 6. apresenta as conclusões obtidas com o trabalho e lista algumas melhorias futuras para o algoritmo Grasp.

2. O Problema de Tabela-horário para Universidades (THU)

Apesar das formulações encontradas na literatura para o problema de tabela-horário (THU) serem bastante variadas, elas podem ser agrupadas em três grupos: tabela-horário de escolas, tabela-horário de universidades e tabela-horário para aplicação de exames. Neste trabalho é desenvolvido um algoritmo para tratar a alocação de tabela-horário de universidades. A formulação do problema é a mesma adotada na competição ITC-2007. A principal vantagem em se adotar esta formulação é que muitos autores convergiram para ela, o que contribuiu grandemente para comparação de resultados de diversos pesquisadores.

O problema de tabela-horário nesta formulação consiste das seguintes entidades:

- Dias, Horários e Períodos: É dado o número de dias na semana em que há aula (geralmente 5 ou 6). Cada dia é dividido em um número fixo de horários que é igual para todos os dias. Um período é um par composto de um dia e um horário. O total de períodos é obtido multiplicando a quantidade de dias pela quantidade de horários do dia.
- Disciplinas e Professores: Cada disciplina possui uma quantidade de aulas semanais que devem ser alocadas em períodos diferentes. É lecionado por um professor e assistida por um dado número de alunos. Cada disciplina deve ser dada em um número mínimo de dias, ou seja, não podem ser lecionadas em poucos dias. Cada disciplina pode ser indisponíveis em alguns períodos.
- Salas: Cada sala possui uma capacidade diferente de assentos.
- Currículo: Um currículo é um grupo de disciplinas que possuem alunos em comum.

A solução do problema é alocação das aulas das diciplinas a um período e uma sala. A solução deve atender a algumas restrições separadas em dois grupos: fortes e fracas.

2.1. Restrições Fortes

As restrições fortes são as seguintes:

- Aulas das Disciplinas: Todas as aulas das disciplinas deve ser alocadas e em períodos diferentes. Uma violação ocorre se uma aula não é alocada ou se duas são alocadas no mesmo período.
- Conflitos: Aulas de disciplinas do mesmo currículo ou lecionadas pelo mesmo professor devem ser alocadas em períodos diferentes.
- Ocupação de Sala: Duas aulas não podem ser dadas na mesma sala e no mesmo período.
- Disponibilidade de Professor: Se um professor é indisponível em um horário, nenhuma disciplina que ele leciona deve ser alocada naquele horário.

2.2. Restrições Fracas

As restrições fracas são as seguintes:

- Capacidade da Sala: O número de alunos da disciplina deve ser menor ou igual ao número de assentos da sala em que a aula for dada. Cada aluno excedente contabiliza uma violação.
- Dias Mínimos de Trabalho: As aulas de cada disciplina devem ser espalhadas por uma quantidade mínima de dias. Cada dia abaixo do mínimo é contado como uma violação.
- Aulas Isoladas: Aulas do mesmo currículo devem ser alocadas em períodos adjacentes. Cada aula isola é contada como uma violação.
- Estabilidade de Sala: Todas as aulas de uma disciplina devem ser dadas na mesma sala. Cada sala usada para aula que é diferente da primeira é contada como uma violação.

  Na contagem total das violações fracas são considerados pesos diferentes para cada tipo de violação. A restrição de dias mínimos possui peso cinco, aulas isoladas peso dois e as demais peso um.

3. Trabalhos Relacionados

As principais técnicas de solução do problema THU recaem basicamente em três grupos: programação matemática, programação em lógica e meta-heurísticas. Por ser um problema complexo, é inviável computacionalmente buscar a solução ótima. Alguns métodos tem alcançado este objetivo, mas apenas para instâncias pequenas. Como pode ser visto em [Schaerf (1995)], este problema é NP-completo, portanto não existe um algoritmo de comlexidade polinomial para encontrar a solução ótima.

Na área de programação matemática, bons resultados têm sido obtidos com programação inteira [Lach (2010)], [van den Broek (2010)] e [Burke (2008)].

Algumas formulações do problema THU usando programação em lógica podem ser vistas em [Gueret (1995)], [Goltz (1999)] e [Asín Achá (2010)].

A maioria das técnicas de solução deste problema utilizam meta-heurísticas. Ainda não foi identificada uma meta-heurística que seja considerada a melhor. Algoritmos eficientes que produzem bons resultados têm sido encontrados usando Simulated Annealing [Kostuch (2006)], [Bai (2012)], [Elmohamed (1998)], Algoritmo Genético [Erben (1995)], [Suyanto (2010)], [Kanoh (2008)], Busca Tabu [Elloumi (2008)], entre outras mais recentes como a Busca de Harmonia [Al-Betar (2012)].

Há também algumas técnicas híbridas em que são usadas mais de uma meta-heurística ou elas são aplicadas de forma diferente. Exemplos deste tipo de implementação podem ser visto em [Massoodian (2008)] em que o algoritmo genético é combinado com um algoritmo de busca local para melhorar a qualidade das soluções. Em [Kostuch (2006)] há um algoritmo que constrói a tebela-horário em três etapas, cada uma executando um procedimento de Simulated Annealing independente.

Este trabalho propõe um algoritmo GRASP para o problema de tabela-horário de universidades. Há na literatura algumas propostas do GRASP, mas para tabela-horário de escolas [Souza (2004)], [Vieira (2010)].

4. Algoritmos para o THU

Como foi apresentado na seção anterior, muitas meta-heurísticas têm sido usadas para resolver o problema de THU. Dentre elas, Algoritmo Genético e Simulated Annealing têm se destacado pela quantidade de propostas que implementam essas técnicas. Dentre estas propostas, foram escolhidas duas consideradas promissoras, com o intuito de estudá-las mais profundamente e implementar melhorias para produzir soluções melhores.

Os algoritmos escolhidos e as melhorias implementadas são apresentados nas próximas seções.

4.1. Algoritmo Genético

O algoritmo genético escolhido para estudo é o proposto em [Massoodian (2008)]. Após a geração de uma população inicial, são realizadas as iterações (gerações) onde são aplicados os procedimentos de seleção, cruzamento e mutação. Ao final de cada geração, uma busca local é feita no melhor indivíduo da população.

A primeira melhoria foi na geração da população inicial. Na versão original, somente uma restrição forte era levada em conta. No algoritmo modificado tenta-se respeitar

todas as restrções fortes. Devido à complexidade das instâncias nem sempre é possível respeitar todas elas, mas esta modificação produziu soluções mais próximas da viabilidade do que o original.

Uma falha identificada no procedimento de cruzamento original é que, em geral, ele produz filhos com muitas violações das restrições fortes. Para sanar este problema foi implementado um procedimento de reparação do filho gerado. Ele verifica as aulas que estão em horários inviáveis e procura um novo horário em que ela pode ser inserida.

O procedimento de mutação também foi modificado para não permitir uma alteração no indivíduo que produza novas inviabilidades. Esta alteração fez com que o algoritmo genético encontrasse uma solução viável com menos iterações.

4.2. Simulated Annealing

4.3. GRASP - com a primeira busca local

A meta-heurística GRASP [Resende (2012)] possui basicamente duas etapas: na primeira uma solução inicial é construída enquanto na segunda a solução é melhorada através de algum algoritmo de busca local. Essas duas etapas se repetem de forma independente por um número máximo de iterações. A melhor solução encontrada em uma destas etapas é a solução final.

4.3.1. Geração da Solução Inicial

A geração da solução inicial do GRASP é um algoritmo construtivo baseado no princípio guloso aleatório. Partindo de uma tabela-horário vazia, as aulas são acrescentadas uma a uma até que todas estejam alocadas. A escolha é tanto gulosa (para produzir soluções de boa qualidade) quanto aleatória (para produzir soluções diversificadas).

O principal objetivo deste procedimento é produzir uma solução viável, ou seja, sem violações fortes. Com intuito de alcançar este objetivo, é adotada uma estratégia de alocar as aulas mas conflitantes primeiro. Poucos horários são viáveis para as disciplinas mais conflitantes, portanto, é melhor alocá-las quando a tabela está mais vazia.

As aulas que ainda estão desalocadas são mantidas numa lista ordenada pela "dificuldade". Quanto menos horários disponíveis existem para a aula e mais presente ela está em currículos diferentes, mais difícil é encontrar um horário viável.

Em cada etapa a aula mais difícil é escolhida para ser alocada. Existem diferentes combinações de horários e salas para fazer a alocação. Os custos de todas essas combinações são calculados levando se em conta as penalizações das restrições fracas. As combinações que possuem horários inviáveis são descartadas. Com base no menor e maior custo de adição (c^{min} e c^{max}) é construída a lista restrita de candidatos (LRC). Estarão na LRC as aulas cujo custo estejam no intervalo $[c^{min}, c^{min} + \alpha(c^{max} - c^{min})]$. Com $\alpha = 0$, o algoritmo é puramente guloso, enquanto com $\alpha = 1$ a construção é alaeatória. Uma aula é escolhida aleatoriamente da LRC e acrescentada à solução.

Em alguns casos ocorre que ao tentar inserir uma aula na tabela não há uma posição viável. Para contornar esta situação foi implementado um procedimento denominado “explosão”. É uma estratégia que retira da tabela uma aula alocada anteriormente para abrir

espaço para a aula que não está sendo possível alocar. Essa estratégia mostrou-se bastante eficaz, tornando possível gerar uma tabela viável para todas as instâncias testadas.

4.3.2. Busca Local

As soluções geradas pela fase inicial não são necessariamente ótimas. Uma busca local é usada para explorar a vizinhança e encontrar soluções melhores. O fator determinante nesta etapa é como a vizinhança será explorada, ou que movimentos serão aplicados para se encontrar um vizinho.

Neste trabalho um vizinho é gerado aplicando dois movimentos sucessivos:

- **MOVE**: Uma aula é movida para uma posição vazia da tabela.
- **SWAP**: Duas aulas são trocadas de posição na tabela.

O segundo movimento (*SWAP*) é aplicado com uma certa probabilidade, portanto alguns vizinhos são gerados apenas com *MOVE*, outros com *MOVE* seguido de *SWAP*. A taxa de *SWAP* é parametrizada. As escolhas de aulas e posições vazias para aplicação dos movimentos são todas aleatórias.

A busca local parte da solução inicial e em cada iteração gera um vizinho da solução atual. É contado quantas iterações já passaram sem melhora. Sempre que um vizinho com melhor função objetivo é encontrado ele passa a ser a solução atual e a contagem de iterações sem melhora é zerada. O teste de parada da busca local é o número de iterações sem melhora. A quantidade máxima é parametrizada.

4.4. Path-Relinking

A proposta original do GRASP é considerada sem memória, pelo fato das soluções obtidas nas iterações não serem compartilhadas entre si para produzir melhores soluções. O Path-Relinking trabalha com duas soluções: uma ótima local e outra elite. Ele tenta melhorar a solução ótima local usando a estrutura da solução elite. Para aplicar o Path-Relinking, o GRASP precisa manter um conjunto de soluções elite.

Este procedimento constrói um caminho conectando uma solução inicial e uma solução alvo. Esse caminho é feito aplicando movimentos que façam com que a solução inicial fique igual à solução alvo. Durante o percurso melhores soluções podem ser encontradas. No sentido *para frente* a solução inicial é a ótima local e a alvo uma solução elite. No sentido *para trás*, o caminho é percorrido da solução elite para a ótima local.

Testes mostraram que o sentido *para trás* produzem melhores resultados para o problema em questão. [Resende (2012)] comenta que, em geral, boas soluções estão mais próximas às soluções elite do que às soluções ótimas locais.

5. Resultados Computacionais

Os resultados...

6. Conclusões e Trabalhos Futuros

As conclusões ....

Instância	AG	SA	GRASP	CTT	
comp01	15	6		5	
comp02	234	116		24	
comp03	233	116		66	
comp04	137	76		35	
comp05	818	429		290	
comp06	215	132		27	
comp07	205	99		6	
comp08	169	84		37	
comp09	229	137		96	
comp10	183	67		4	
comp11	13	0		0	
comp12	544	407		300	
comp13	206	106		59	
comp14	173	90		51	
comp15	225	120		66	
comp16	198	91		18	
comp17	206	122		56	
comp18	153	133		62	
comp19	207	111		57	
comp20	257	130		4	
comp21	276	151		76	

Tabela 1. Resultados computacionais - Quantidade de penalizações obtidas pelo algoritmo genético (AG), simulated annealing (SA), Grasp e as melhores soluções disponíveis no site CTT

Referências

Al-Betar, M. and Khader, A. (2012). A harmony search algorithm for university course timetabling. *Annals of Operations Research*, 194:3–31. 10.1007/s10479-010-0769-z.

Asín Achá, R. and Nieuwenhuis, R. (2010). Curriculum-based course timetabling with sat and maxsat. *Annals of Operations Research*, pages 1–21. 10.1007/s10479-012-1081-x.

Bai, R., Blazewicz, J., Burke, E., Kendall, G., and McCollum, B. (2012). A simulated annealing hyper-heuristic methodology for flexible decision support. *4OR: A Quarterly Journal of Operations Research*, 10:43–66. 10.1007/s10288-011-0182-8.

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Animation and Visualization*. John Wiley & Sons ltd.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2008). A branch-andcut procedure for the udine course timetabling. In *Problem, Proceedings of the 7th PATAT Conference, 2008*.

de Souza, M. C., Duhamel, C., and Ribeiro, C. C. (2002). A grasp heuristic for the capacitated minimum spanning tree problem using a memory-based local search strategy.

Elloumi, A., Kamoun, H., Ferland, J., and Dammak, A. (2008). A tabu search procedure for course timetabling at a tunisian university. In *Proceedings of the 7th PATAT Conference, 2008*.

Elmohamed, M. A. S., Coddington, P., and Fox, G. (1998). A comparison of annealing techniques for academic course scheduling. In *Lecture Notes in Computer Science*.

Erben, W. and Keppler, J. (1995). A genetic algorithm solving a weekly course-timetabling problem.

Festa, P., Pardalos, P., Pitsoulis, L., and Resende, M. (2006). GRASP with path-relinking for the weighted MAXSAT problem. *ACM J. of Experimental Algorithmics*, 11. article 2.4: 1-16.

Goltz, H.-J. and Matzke, D. (1999). University timetabling using constraint logic programming. In *PRACTICAL ASPECTS OF DECLARATIVE LANGUAGES*, pages 320–334. Springer-Verlag.

Gueret, C., Jussien, N., Boizumault, P., and Prins, C. (1995). Building university timetables using constraint logic programming.

Kanoh, H. and Sakamoto, Y. (2008). Knowledge-based genetic algorithm for university course timetabling problems. *Int. J. Know.-Based Intell. Eng. Syst.*, 12(4):283–294.

Knuth, D. E. (1984). *The TEX Book*. Addison-Wesley, 15th edition.

Kostuch, P. (2006). The university course timetabling problem with a 3-phase approach.

Lach, G. and Lubbecke, M. E. (2010). Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*.

Massoodian, S. and Esteki, A. (2008). A hybrid genetic algorithm for curriculum based course timetabling. In *Proceedings of the 7th PATAT Conference, 2008*.

Resende, M. and Ribeiro, C. (2012). *GRASP: Greedy Randomized Adaptive Search Procedures*. Springer, 2nd edition.

Resende, M. G. C. (1998). Computing approximate solutions of the maximum covering problem using GRASP. *J. of Heuristics*, 4:161–171.

Schaerf, A. (1995). A survey of automated timetabling. *ARTIFICIAL INTELLIGENCE REVIEW*, 13:87–127.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, *Advances in Computer Science*, pages 555–566. Publishing Press.

Souza, M. J. F., Maculan, N., and Ochi, L. S. (2004). Metaheuristics. chapter A GRASP-tabu search algorithm for solving school timetabling problems, pages 659–672. Kluwer Academic Publishers, Norwell, MA, USA.

Suyanto (2010). An informed genetic algorithm for university course and student timetabling problems. In Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L., and Zurada, J., editors, *Artifical Intelligence and Soft Computing*, volume 6114 of *Lecture Notes in Computer Science*, pages 229–236. Springer Berlin / Heidelberg.

van den Broek, J. and Hurkens, C. (2010). An ip-based heuristic for the post enrolment course timetabling problem of the itc2007. *Annals of Operations Research*.

Vieira, A., Rafael, M., and Scaraficci, A. (2010). A grasp strategy for a more constrained school timetabling problem.