Submetido para MO824

Atividade 9 - Grupo 3 - MO824

Ana Clara Zoppi Serpa - RA 165880
Victor Ferreira Ferrari - RA 187890

Resumo. Este trabalho consiste de um estudo de caso para o problema de alocação de professores (PAP). É apresentado um modelo matemático para o problema, e é implementada uma solução heurística baseada no GRASP. A solução direta do modelo com o resolvedor Gurobi foi comparada à metaheurística por meio de perfis de desempenho e tabelas, e os resultados mostram que o Gurobi teve melhor desempenho para este problema, tanto quanto à otimalidade das soluções como ao tempo necessário para encontrá-las. A metaheurística implementada obteve resultados que em geral se aproximam pouco do ótimo. Das configurações testadas, o GRASP com viés linear e $\alpha = 0.1$ foi a melhor.

Palavras-chave. Otimização, alocação, PAP, metaheurística, GRASP, PLI

1. Introdução

Este trabalho consiste de um estudo do problema de alocação de professores (PAP), proposto na Atividade 9 de MO824 2S-2020. Um modelo matemático é apresentado, além da implementação de uma metaheurística. Os métodos são comparados por meio de perfis de desempenho e tabelas.

1.1. Problema

O Problema de Alocação de Professores (PAP) consiste em encontrar uma alocação de P docentes em D disciplinas, que devem ser distribuídas em períodos da semana, tal que certas restrições sejam satisfeitas. Dada a avaliação $a_{pd} \in [0, 100]$ do professor p referente a uma disciplina d, o objetivo é maximizar as avaliações correspondentes aos pares $(professor, disciplina)$ alocados.

Cada disciplina exige $h_d \in [1, 3]$ períodos em uma semana. Cada semana é dividida em cinco dias, e cada dia é separado em quatro períodos. Ou seja, cada semana contém $T = 20$ períodos, que são numerados em ordem cronológica. Não mais que S disciplinas podem ser ministradas no mesmo período, sendo S o número de salas do instituto.

Cada professor possui restrições sobre os períodos, ou seja, se $r_{pt} = 1$, o professor p pode lecionar no período t, e vice-versa. Um professor pode somar até $H = 3$ períodos na semana, e não pode lecionar mais de uma disciplina no mesmo período. Caso uma disciplina não possa ser alocada, uma penalidade de -100 é aplicada na função objetivo.

2. Modelo Matemático

As seguintes variáveis foram usadas no modelo:

x_{pd}: Variável de decisão binária associada à alocação ($x_{pd} = 1$) ou não ($x_{pd} = 0$) do professor p na disciplina d.

y_d: Variável binária associada à alocação ($y_d = 1$) ou não ($y_d = 0$) de uma disciplina (em qualquer período).

z_{pt}: Variável binária associada à alocação ($z_{pt} = 1$) ou não ($z_{pt} = 0$) do professor p no período t.

$$(PAP) \qquad \max \sum_{d=1}^{D}\sum_{p=1}^{P} a_{pd}x_{pd} - \sum_{d=1}^{D} 100(1 - y_d) \tag{2.1}$$

s.a.

$$\sum_{p=1}^{P} x_{pd} = y_d \quad \forall d \in \{1 \dots D\} \tag{2.2}$$

$$\sum_{p=1}^{P} z_{pt} \leq S \quad \forall t \in \{1 \dots T\} \tag{2.3}$$

$$z_{pt} \leq r_{pt} \quad \forall p \in \{1 \dots P\}, t \in \{1 \dots T\} \tag{2.4}$$

$$\sum_{t=1}^{T} z_{pt} \leq H \quad \forall p \in \{1 \dots P\} \tag{2.5}$$

$$\sum_{t=1}^{T} z_{pt} = \sum_{d=1}^{D} h_d x_{pd} \quad \forall p \in \{1 \dots P\} \tag{2.6}$$

$$x_{pd}, y_d, z_{pt} \in \mathbb{B} \quad \forall p \in \{1 \dots P\}, d \in \{1 \dots D\}, t \in \{1 \dots T\} \tag{2.7}$$

onde, como descrito na Seção 1.1, D é a quantidade de disciplinas que necessitam de alocação, P é a quantidade de professores disponíveis, S é a quantidade de salas disponíveis por período, e há $T = 20$ períodos disponíveis por semana. a_{pd} é a avaliação do professor p na disciplina d, h_d é a carga horária semanal (em períodos) exigida pela disciplina d, e r_{pt} é a disponibilidade do professor p no período t. A carga horária máxima H de um professor é de 3 períodos.

A função objetivo (2.1) procura maximizar a soma das avaliações dos professores nas disciplinas em que são alocados, com uma penalização se a disciplina não foi alocada.

A restrição (2.2) indica que, se uma disciplina for alocada, há exatamente um professor ligado a ela (já que y_d também é binária). A restrição (2.3) limita o número de disciplinas em um período pelo número de salas.

A restrição (2.4) indica que um professor só é alocado em um período se estiver disponível nele. A carga horária de um professor é limitada em $H = 3$ períodos na restrição (2.5). Finalmente, a restrição (2.6) certifica que um professor leciona no máximo 1 disciplina por período, ou seja, o número de períodos em que um professor está alocado é a soma da carga horária das disciplinas ministradas por ele, e a restrição (2.7) restringe o domínio das variáveis.

A função objetivo pode ser simplificada para:

$$\max \sum_{d=1}^{D}(100y_d + \sum_{p=1}^{P} a_{pd}x_{pd}) - 100D \tag{2.8}$$

A variável y também é desnecessária, e pode ser removida do modelo, se a substituirmos pelo somatório da restrição (2.2). Ao substituir e simplificar, a nova função objetivo e restrições alteradas (equivalentes às restrições (2.2) e (2.7)) são:

$$(PAP\_ALT) \qquad \max \sum_{d=1}^{D}(\sum_{p=1}^{P}(a_{pd} + 100)x_{pd}) - 100D \tag{2.9}$$

s.a.

$$\sum_{p=1}^{P} x_{pd} \leq 1 \quad \forall d \in \{1 \dots D\} \tag{2.10}$$

$$x_{pd}, z_{pt} \in \mathbb{B} \quad \forall p \in \{1 \dots P\}, d \in \{1 \dots D\}, t \in \{1 \dots T\} \tag{2.11}$$

O modelo ainda contém as restrições (2.3), (2.4), (2.5) e (2.6).

3. Metaheurística

Neste trabalho foi implementado o GRASP (*greedy randomized adaptive search procedure*) [1] para solucionar o problema, que consiste de uma metaheurística iterativa na qual cada iteração possui uma fase construtiva seguida de uma busca local. Em linhas gerais, uma solução é construída na primeira etapa e sua vizinhança é investigada na segunda, visando encontrar uma solução de melhor custo.

3.1. Heurística Construtiva

Na heurística construtiva, os elementos candidatos a entrarem na solução são avaliados de forma gulosa no que diz respeito aos seus custos de inserção: quanto maior esse custo, mais próximo do máximo a solução se encontra. A cada iteração, um dos elementos (factíveis) é escolhido para a solução.

No entanto, para ampliar a variância entre iterações, ao invés de selecionarmos o elemento cujo custo c melhor afeta a solução, concebemos uma lista de candidatos restrita (RCL, do inglês Restricted Candidate List), que engloba todos elementos

com custo no intervalo $[c^{\min}, c^{\min} + \alpha \cdot (c^{\max} - c^{\min})]$. Um membro da RCL é, então, selecionado aleatoriamente (como detalhado na Seção 3.1.1) para ingressar na solução, e o processo se repete enquanto a inserção de um candidato melhorar a solução ou a lista de candidatos esvaziar. Note que o parâmetro α controla o quão guloso ou aleatório é o algoritmo.

3.1.1. Função de Viés (*Bias*)

Nessa abordagem, a escolha aleatória de um candidato da RCL a ser inserido na solução em construção é feita a partir de uma **função de viés**. Nela, são atribuídas probabilidades para cada elemento com base em uma função $\texttt{bias}(r)$ e no valor de inserção do elemento na solução.

Dado um candidato σ, podemos definir um *rank* $r(\sigma)$ para o elemento dentro do conjunto C. Assim, a probabilidade de σ é dada pela Equação (3.1).

$$\pi(\sigma) = \frac{\texttt{bias}(r(\sigma))}{\sum_{\sigma' \in C} \texttt{bias}(r(\sigma'))} \tag{3.1}$$

Esse método não elimina diversidade, pois a escolha ainda é feita aleatoriamente, mas fornece um viés para essa escolha, tentando aumentar a qualidade média das soluções.

A escolha padrão do GRASP ainda pode ser feita, definindo a função `bias` constante igual a 1, independente do *rank*, e assim as probabilidades são iguais para todos os elementos da RCL. Porém, outras combinações podem ser testadas, como funções lineares, logarítmicas, exponenciais e polinomiais [1]. Após diversos testes com todas as funções citadas, escolhemos focar em uma função de viés **linear** (além da padrão).

3.2. Busca Local

A etapa de busca local consiste na exploração da vizinhança da solução S construída anteriormente. Uma vizinhança imediata consiste nas soluções obtidas ao **inserir** ou **remover** um elemento em S, ou **trocar** um elemento de S por outro. Essas operações são chamadas de *movimentos de vizinhança*. A lista de candidatos é atualizada a cada iteração, para conter apenas elementos factíveis. O processo de atualização da CL e os movimentos de vizinhança podem ser vistos na Seção 3.3.

Dados os movimentos, a vizinhança é explorada até que um ótimo local seja encontrado, sendo este retornado para atualização da solução titular. Essa busca segue a estratégia *best-improving*, que trata de avaliar todos os movimentos possíveis a partir da solução atual, e escolher aquele que fornecer o melhor custo. Em outras palavras, todas as inserções, remoções e trocas são avaliadas, e o melhor movimento é escolhido.

3.3. Modelagem do Problema

Cada elemento da solução é um par (p, d), o que equivale à decisão de que um professor p lecionará a disciplina d. Além disso, outras estruturas auxiliares são usadas para armazenar as combinações (d, t), ou seja, a alocação das disciplinas d nos períodos t disponíveis, para verificação de viabilidade das soluções.

O operador de inserção consiste em alocar um professor p a uma disciplina d, e alocar os períodos necessários por *first-fit* — ou seja, no primeiro período disponível. O operador de remoção envolve remover um professor de disciplina, e seus períodos. O operador de troca consiste em uma remoção seguida de uma inserção.

A cada iteração da heurística construtiva e da busca local, um processo de viabilização da lista de candidatos é feito. A CL é reiniciada, e para cada disciplina que ainda não foi alocada, todos os pares (p, d) possíveis são avaliados quanto à viabilidade, seguindo as restrições do modelo (PAP_ALT). Assim, um elemento só está na lista de candidatos se sua inserção é segura.

A presença das estruturas auxiliares permite a expansão da solução final, para incluir todas as triplas (p, d, t).

4. Implementação e Avaliação

A implementação da solução heurística foi feita em *Java*, com base nos *frameworks* fornecidos, com as modificações citadas na Seção 3 e outras necessárias para obtenção dos dados e realização dos experimentos. Foi utilizado como *software* de resolução dos modelos matemáticos o *solver* Gurobi Optimizer V9.0.3, com a API para *Java*.

Dois modelos para o problema foram implementados: "GUROBI" se refere ao uso do modelo com as variáveis y_d para o PAP (modelo PAP), e "GUROBI ALT" se refere ao uso do modelo no qual as variáveis y_d são substituídas pelo somatório (modelo PAP_ALT). Quatro configurações do GRASP foram testadas:

1. PADRÃO: $\alpha_1 = 0.25$, e *bias* aleatório.
2. PADRÃO+LINEAR: $\alpha_1 = 0.25$, e *bias* linear.
3. PADRÃO+LINEAR+ALPHA2: $\alpha_2 = 0.1$, e *bias* linear.
4. PADRÃO+LINEAR+ALPHA3: $\alpha_3 = 0.4$, e *bias* linear.

Todos os métodos foram executados com limite de 1800 segundos (30 minutos) e sem limite de memória. Para o GRASP, o máximo de iterações em todos os experimentos foi 1000. As instâncias utilizadas em nossos experimentos foram fornecidas previamente no pacote da atividade.

As especificações de hardware do computador no qual foram feitas as execuções estão na Tabela 1. O sistema operacional utilizado foi o Windows 10 (64 bits).

Tabela 1: Condições de Execução

Modelo da CPU	Intel(R) Core(TM) i7-7500 CPU
Frequência do Clock da CPU	2.70 GHz - 2.90 GHz
RAM	8,00 GB/2133 MHz

5. Resultados e Análise

A tabela 2 mostra os resultados obtidos com o resolvedor Gurobi para cada instância, sendo as colunas LB e UB referentes aos limitantes inferior e superior indicados pelo resolvedor para o custo (valor da função objetivo) da solução, respectivamente. O tempo de execução foi medido em segundos. Os casos em que a solução encontrada é ótima estão destacados em negrito na tabela 2.

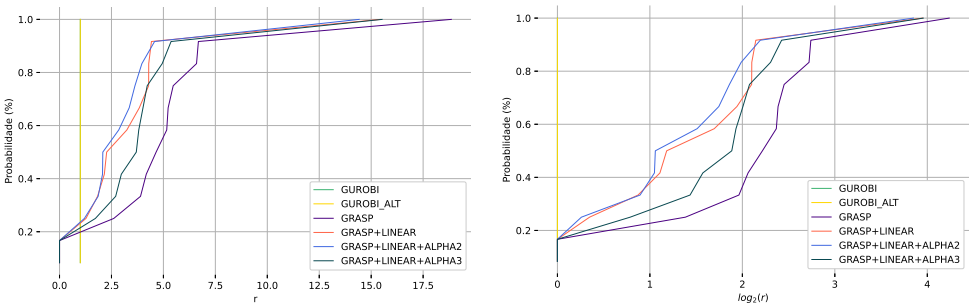
Os resultados para a resolução direta dos modelos foram muito bons. Foram encontradas soluções ótimas para todas as instâncias com o modelo GUROBI. Já com GUROBI_ALT, não foi possível confirmar a otimalidade das soluções para as instâncias P50D50S3 e P100D150S10, no tempo alocado. Ainda assim, as soluções adquiridas são ótimas. Quanto ao tempo de execução, o modelo GUROBI foi mais lento que o modelo GUROBI_ALT somente para a instância P100D150S20. Esses resultados sugerem que o modelo com as variáveis y_d é melhor tanto em termos de tempo de execução quanto na capacidade de encontrar soluções garantidamente ótimas para o problema PAP.

As Tabelas 3 e 4 mostram os resultados obtidos com a metauheurística GRASP para os diferentes valores de α (0.1, 0.25, 0.4) e as funções de viés (linear e aleatória) exploradas. As configurações, salvo dois casos, ultrapassaram o limite de 30 minutos estipulado para a execução.

Em geral, os resultados do GRASP não alcançaram os modelos em termos de qualidade. Nenhuma solução ótima foi encontrada, embora nas configurações com viés linear soluções próximas das ótimas tenham sido encontradas para algumas instâncias. O valor $\alpha = 0.1$ com bias linear parece ter sido a melhor abordagem, obtendo soluções mais próximas das ótimas pontadas pelo resolvedor Gurobi em mais instâncias do que as demais configurações.

A Figura 1 contém a análise de perfis de desempenho para todos os métodos testados. Pode-se ver que as curvas dos modelos (sobrepostas) claramente dominam as outras, atingindo as melhores soluções em todas as instâncias. Entre os métodos heurísticos, se confirma a conclusão anterior: a configuração com viés linear e $\alpha = 0.1$ tem a curva dominante. Por outro lado, a configuração com viés aleatório retornou a pior curva, ou seja, a mudança no viés melhorou consideravelmente a qualidade das soluções encontradas.

Isso sugere que o método implementado pode ser melhorado, seja com outros métodos alternativos de construção (já que as funções de viés foram bem-sucedidas) ou com mudanças na modelagem do problema. Seria interessante explorar outras formas de lidar com a alocação dos períodos nos operadores de busca local, em vez



(a) Escala Linear (b) Escala Log

Figura 1: Perfis de Desempenho (Função Objetivo)

de utilizar *first-fit*, seja com aleatoriedade ou uma heurística mais sofisticada.

Em conclusão, comparando os dois métodos estudados (GRASP e Gurobi), o resolvedor Gurobi teve um desempenho claramente melhor, tanto em termos de qualidade das soluções encontradas como de tempo de execução.

Tabela 2: Resultados obtidos com o resolvedor Gurobi

	GUROBI			GUROBI_ALT		
Instância	LB	UB	Tempo (s)	LB	UB	Tempo (s)
P50D50S1	**-1292**	**-1292**	0.062	**-1292**	**-1292**	0.062
P50D50S3	**2271**	**2271**	3.452	2271	2371	1800.079
P50D50S5	**4770**	**4770**	0.141	**4770**	**4770**	1.875
P70D70S1	**-3058**	**-3058**	0.062	**-3058**	**-3058**	0.421
P70D70S3	**924**	**924**	1.109	**924**	**924**	10.123
P70D70S5	**4265**	**4265**	0.219	**4265**	**4266**	2.609
P70D100S6	**4610**	**4610**	0.437	**4610**	**4610**	2.890
P70D100S8	**7177**	**7177**	1.719	**7177**	**7177**	49.316
P70D100S10	**9616**	**9617**	4.671	**9616**	**9617**	23.245
P100D150S10	**7626**	**7626**	315.613	7626	7743	1800.248
P100D150S15	**13259**	**13259**	47.739	**13259**	**13259**	198.282
P100D150S20	**13259**	**13259**	12.169	**13259**	**13259**	11.372

Referências

[1] M. G. C. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications," in *Handbook of Metaheuristics* (M. Gendreau and J.-Y. Potvin, eds.), vol. 146 of *International Series in Operations Research & Management Science*, ch. 10, pp. 283–318, Springer Science+Business Media, 2010.

Tabela 3: Resultados obtidos com a metaheurística GRASP com $\alpha = 0.25$

	PADRÃO		PADRÃO + LINEAR	
Instância	Solução	Tempo(s)	Solução	Tempo(s)
P50D50S1	-2169	0.062	-2112	0.062
P50D50S3	1791	3.452	2127	1800.079
P50D50S5	4386	0.141	4705	1.875
P70D70S1	-3856	0.062	-4241	0.421
P70D70S3	99	1.109	257	10.123
P70D70S5	3588	0.219	4088	2.609
P70D100S6	3300	0.437	3851	2.890
P70D100S8	6137	1.719	6762	49.316
P70D100S10	7616	4.671	8251	23.245
P100D150S10	5493	315.613	6372	1800.248
P100D150S15	10295	47.739	10989	198.282
P100D150S20	10840	12.169	11774	11.372

Tabela 4: Resultados obtidos com a metaheurística GRASP com viés linear, com $\alpha_2 = 0.1$ e $\alpha_3 = 0.4$.

	PADRÃO + LINEAR+ ALPHA2		PADRÃO + LINEAR+ ALPHA3	
Instância	Solução	Tempo(s)	Solução	Tempo(s)
P50D50S1	-2223	0.062	-2116	0.062
P50D50S3	2061	1800.079	1965	1800.079
P50D50S5	4723	1.875	4597	1.875
P70D70S1	-4222	0.421	-4425	0.421
P70D70S3	304	10.123	253	10.123
P70D70S5	4034	2.609	3844	2.609
P70D100S6	3786	2.890	3603	2.890
P70D100S8	6869	49.316	6566	49.316
P70D100S10	8191	23.245	8068	23.245
P100D150S10	6729	1800.248	6123	1800.248
P100D150S15	11515	198.282	11259	198.282
P100D150S20	12550	11.372	11394	11.372