CLASSROOM ASSIGNMENT PROBLEM

EXPERIÊNCIAS COM SIMULATED ANNEALING E BUSCA TABU

INF 284 2023.1

Descrição do Problema

O problema de alocação de salas (PAS) diz respeito à distribuição de aulas, com horários previamente estabelecidos, a salas, respeitando-se um conjunto de restrições de várias naturezas (Schaefer 1999).

Objetivo: Para encontrar uma solução ideal, é necessário garantir que todos os requisitos considerados essenciais sejam atendidos. Além disso, é importante avaliar cuidadosamente os requisitos não essenciais e procurar uma solução que os atenda com boa qualidade.

Exemplo Aplicado

Considere um instituto que oferta disciplinas com aulas em 9 horários diferentes. As aulas são realizadas de segunda a sábado à tarde, mas a maioria delas está concentrada de terça a quinta-feira. Para a realização das aulas das turmas estão disponíveis 5 salas de aulas. Os horários de aula das turmas são confeccionados previamente pelos departamentos e encaminhados à diretoria do instituto para que esta faça a alocação das turmas às salas.

Exemplo de Restrições

Essenciais:

1. Em uma mesma sala e horário não pode haver mais de uma aula;
2. Uma sala não pode receber uma turma cuja quantidade de alunos seja superior à sua capacidade;
3. Algumas salas têm alguns horários previamente reservados para a realização de outras atividades e nesses horários elas ficam indisponíveis;

Não Essenciais:

4. Certas salas têm restrições de uso e a utilização delas deve ser evitada;
5. Sempre que possível, alocar a uma mesma sala alunos de um mesmo curso e período;
6. Evitar alocar aulas de turmas pequenas em salas de maior capacidade;
7. Cada uma das salas deve ser deixada vazia em pelo menos um horário ao longo do dia, de forma a possibilitar sua limpeza.

Representação de Solução

Uma solução é representada por uma matriz $S = (s_{ij})_{mxn}$, onde **m** representa o número de horários reservados para a realização das aulas e **n** o número de salas disponíveis. Em cada célula $\mathbf{s}_{ij}$ é colocado o número da turma **t** alocada ao horário **i** e sala **j**. Uma célula vazia indica que a sala **j** está desocupada no horário **i**.

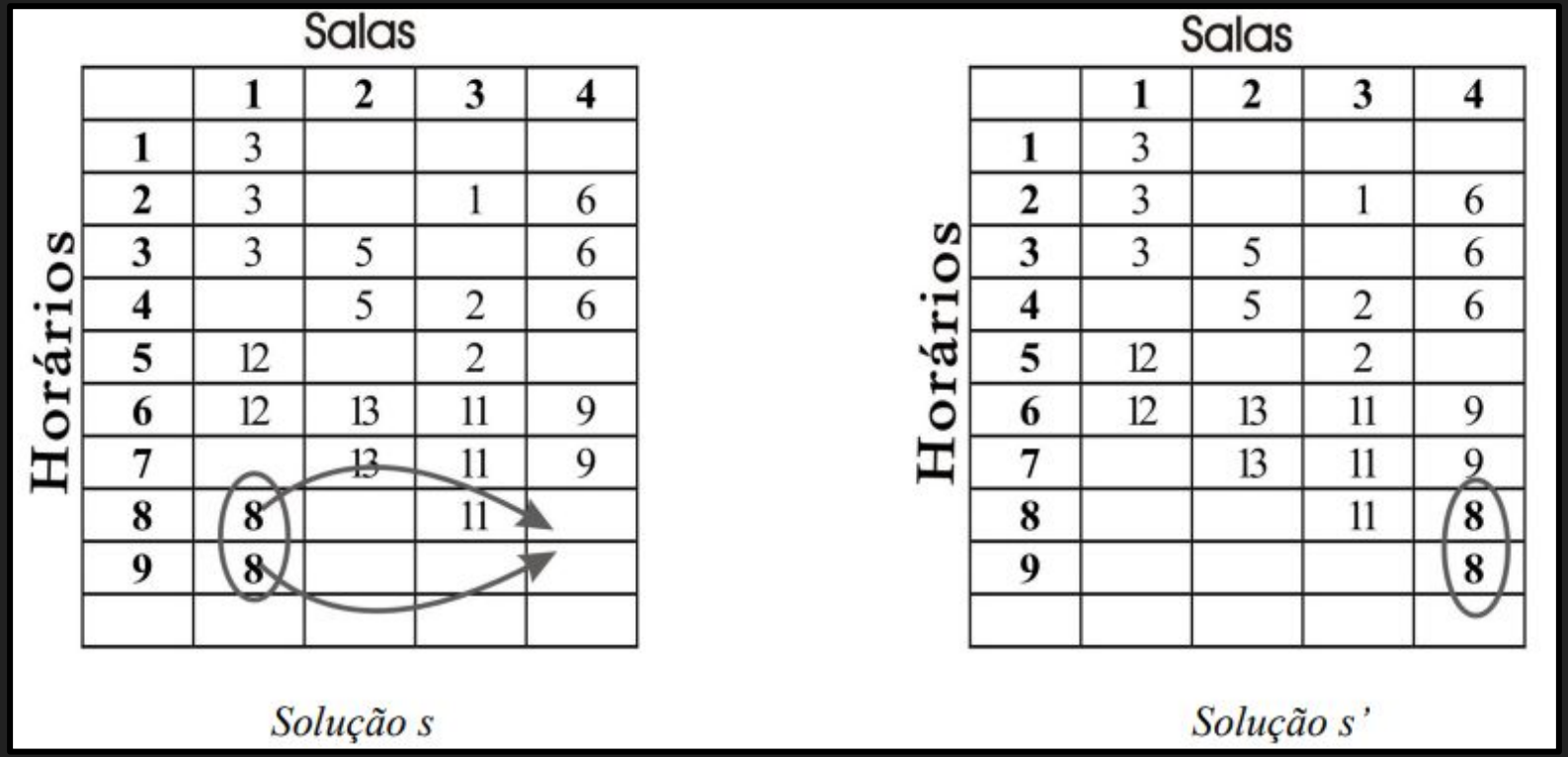
Horários \ Salas	1	2	3	4	5
1	3				4
2	3		1	6	4
3	3	5		6	7
4		5	2	6	7
5	12		2		
6	12	13	11	9	
7		13	11	9	10
8	**8**		11		10
9	**8**				10

Vizinhança

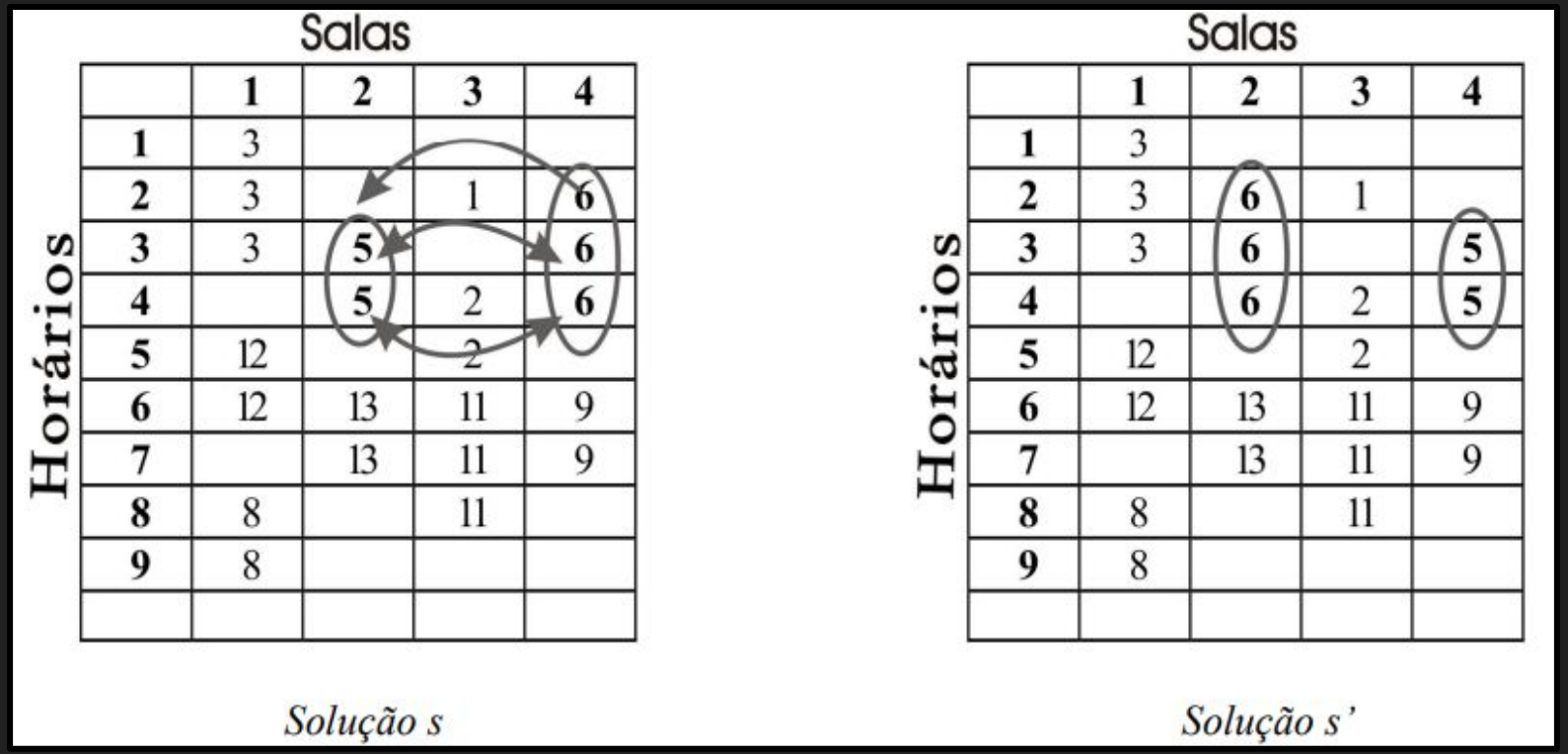
Uma solução $s' \in N(s)$ é um vizinho de s se ela pode ser acessada a partir desta através de um movimento ou de alocação ou de troca.

Movimento Alocação: O movimento de alocação consiste em realocar as aulas de uma dada turma e sala a uma outra sala que esteja vazia nos horários de realização das aulas. Para a realização desse movimento é exigido que a sala que receberá as aulas de uma turma esteja disponível nos horários das aulas.

Movimento Troca: Já o movimento de troca consiste em trocar de sala as aulas de duas turmas realizadas em um mesmo bloco de horários.



Movimento de Alocação



Movimento de Troca

Função de Avaliação

Uma solução s pode ser medida com base em duas componentes, uma de inviabilidade g(s), a qual mede o não atendimento aos requisitos essenciais, e outra de qualidade h(s), a qual avalia o não atendimento aos requisitos considerados não-essenciais. A função objetivo f que associa cada solução s do espaço de soluções a um número real f(s), e que deve ser minimizada, pode ser expressada, portanto, na forma:

$$f(s) = g(s) + h(s)$$

Função de Avaliação

$$g(s) = \sum_{k=1}^{K} \alpha_k I_k$$

- K: número de medidas de inviabilidade;
- I_k: valor da k-ésima medida de inviabilidade;
- α_k: peso associado à k-ésima medida de inviabilidade.

$$h(s) = \sum_{l=1}^{L} \beta_l Q_l$$

- L: número de medidas de qualidade;
- Q_l: valor da l-ésima medida de qualidade;
- β_l: peso associado à l-ésima medida de qualidade.

Solução Inicial

Gerada por um procedimento construtivo que segue as idéias da fase de construção do método GRASP (Feo and Resende 1995). Inicialmente, toma-se a aula ainda não alocada da turma com maior demanda e constrói-se uma lista restrita de candidatos (LRC) das salas vagas nos horários da aula, ordenadas pela capacidade. Caso não exista uma sala vaga no horário, uma sala virtual é criada e a lista se resume a essa sala. A seguir, uma dessas |LRC| salas é escolhida aleatoriamente para receber a aula. Esse procedimento continua até que todas as aulas sejam alocadas.

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários

	1	2
1		
2		
3		
4		
5		

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários	1	2
1		
2		
3		
4		
5		

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários	1	2
1		
2		
3		
4		
5		

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários	1	2
1		
2		
3	2	
4	2	
5	2	

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários

	1	2
1		
2		
3	2	
4	2	
5	2	

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários	1	2
1		
2		
3	2	
4	2	
5	2	

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários

	1	2
1		1
2		1
3	2	
4	2	
5	2	

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários

	1	2
1		1
2		1
3	2	
4	2	
5	2	

Turma	1	2	3
Demanda	2	3	2
Horário	1-2	3-5	2-3

Salas

Horários

	1	2	sala virtual
1		1	
2		1	3
3	2		3
4	2		
5	2		

Simulated Annealing

Método de busca local que aceita movimentos de piora para escapar de ótimos locais. O processo se inicia com uma solução inicial e seleciona um de seus vizinhos randomicamente. Se este vizinho for melhor que o original ele é aceito e substitui a solução corrente. Se ele for pior por uma quantidade Δ, ele é aceito com uma probabilidade $e^{-\Delta/T}$, onde T decresce gradualmente conforme o progresso do algoritmo. Esse processo é repetido até que T seja tão pequeno que mais nenhum movimento seja aceito.

```
1.  Seja s0 uma solução inicial, T0 a temperatura inicial, α a taxa de resfriamento e SAmax o
    número máximo de iterações para se atingir o equilíbrio térmico;
2.  s ← s0;                 {Solução corrente}
3.  s' ← s;                 {Melhor solução obtida até então}
4.  T ← T0;                 {Temperatura corrente}
5.  IterT ← 0;              {Número de iterações na temperatura T}
6.  enquanto (T > 0) faça
7.     enquanto (IterT < SAmax) faça
8.        IterT ← IterT + 1;
9.        Gere um vizinho qualquer s' ∈ N (s);
10.       Δ = f(s') – f(s);
11.       se (Δ < 0)
12.          então
13.             s ← s';
14.             se f(s') < f(s*) então s* ← s';
15.          senão
16.             Tome x ∈ [0,1];
17.             se x < e^(-Δ/T) então s ← s';
18.       fim-se;
19.    fim-enquanto;
20.    T ← α ×T;
21.    IterT ← 0;
22. fim-enquanto;
23. Retorne s*;
fim SA;
```

1. Seja s_0 uma solução inicial, T_0 a temperatura inicial, α a taxa de resfriamento e SAmax o número máximo de iterações para se atingir o equilíbrio térmico;
2. $s \leftarrow s_0$; {Solução corrente}
3. $s' \leftarrow s$; {Melhor solução obtida até então}
4. $T \leftarrow T_0$; {Temperatura corrente}
5. IterT $\leftarrow$ 0; {Número de [illegible]T}
6. enquanto (T > 0) faça
7. enquanto (IterT < SAmax) faça
8. IterT $\leftarrow$ IterT + 1;
9. Gere um vizinho qualquer $s' \in N(s)$;
10. $\Delta = f(s') - f(s)$;
11. se ($\Delta < 0$)
12. então
13. $s \leftarrow s'$;
14. se $f(s') < f(s^*)$ então $s^* \leftarrow s'$;
15. senão
16. Tome x $\in$ [0,1];
17. se x < $e^{-\Delta/T}$ então $s \leftarrow s'$;
18. fim-se;
19. fim-enquanto;
20. T $\leftarrow \alpha \times$T;
21. IterT $\leftarrow$ 0;
22. fim-enquanto;
23. Retorne s^*;

fim *SA;*

Troca ou alocação

Troca ou alocação

Se s' melhor que s

Movimento de piora

Controle de temperatura

Busca Tabu

Procedimento de otimização local que admite soluções de piora. Em sua forma clássica, a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho s’ de um subconjunto V da vizinhança N(s) da solução corrente s. Independentemente de f(s’) ser melhor ou pior que f(s), s’ será sempre a nova solução corrente. Entretanto, apenas esse mecanismo não é suficiente para escapar de ótimos locais, uma vez que pode haver retorno a uma solução previamente gerada. Para evitar isso, o algoritmo usa o conceito de lista tabu. Esta lista define todos os movimentos que têm um certo atributo como sendo tabu por um determinado número de iterações, conhecido como tempo tabu. Tais movimentos são proibidos a menos que a solução satisfaça a um certo critério de aspiração A, em geral que essa solução seja melhor que a melhor solução encontrada até então.

1. Seja s_0 solução inicial;
2. $s^* \leftarrow s$; {Melhor solução obtida até então}
3. Iter $\leftarrow 0$; {Contador do número de iterações}
4. MelhorIter $\leftarrow 0$; {Iteração mais recente que forneceu s^*}
5. Seja BTmax o número máximo de iterações sem melhora em s^*;
6. T $\leftarrow \varnothing$; {Lista Tabu}
7. Inicialize a função de aspiração A;
8. enquanto (Iter – MelhorIter ≤ BTmax) faça
9. Iter $\leftarrow$ Iter + 1;
10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subseteq N(s)$ tal que o movimento m não seja tabu ($m \notin$ T) ou s' atenda a condição de aspiração ($f(s') < A(f(s))$);
11. T $\leftarrow$ T – {movimento mais antigo} + {movimento que gerou s'};
12. $s \leftarrow s'$;
13. se $f(s) < f(s^*)$ então
14. $s^* \leftarrow s$;
15. MelhorIter $\leftarrow$ Iter ;
16. fim-se;
17. Atualize a função de aspiração A;
18. fim-enquanto;
19. Retorne s^*;

fim *BT*;

1. Seja s_0 solução inicial;
2. $s^* \leftarrow s$; {Melhor solução obtida até então}
3. Iter $\leftarrow 0$; {Contador do número de iterações}
4. MelhorIter $\leftarrow 0$; {Iteração mais recente que forneceu s^*}
5. Seja BTmax o número máximo de iterações sem melhora em s^*;
6. T $\leftarrow \varnothing$; {Lista Tabu}
7. Inicialize a função de aspiração A;
8. enquanto (Iter – MelhorIter ≤ BTmax) faça
9. Iter $\leftarrow$ Iter + 1;
10. Seja $s' \leftarrow s \oplus m$ o melhor elemento de $V \subseteq N(s)$ tal que o movimento m não seja tabu ($m \notin$ T) ou s' atenda a condição de aspiração ($f(s') < A(f(s))$);
11. T $\leftarrow$ T – {movimento mais antigo} + {movimento que gerou s'};
12. $s \leftarrow s'$;
13. se $f(s) < f(s^*)$ então
14. $s^* \leftarrow s$;
15. MelhorIter $\leftarrow$ Iter ;
16. fim-se;
17. Atualize a função de aspiração A;
18. fim-enquanto;
19. Retorne s^*;

fim *BT*;

A: solução melhor que a melhor solução

Janela de iterações (critério de parada)

Controle tabu

Algoritmo Híbrido

Foi apresentada uma heurística híbrida para resolver o problema de alocação de salas. Uma solução inicial é gerada por um procedimento construtivo parcialmente guloso e submetida à heurística Simulated Annealing. A solução final desse método é, então, refinada por um algoritmo de Busca Tabu.

procedimento *SA+BT*
1. $s^0 \leftarrow$ ConstruaSolucaoInicial();
2. $s^1 \leftarrow \text{SA}(s^0)$;
3. $s^* \leftarrow \text{BT}(s^1)$;

fim *SA+BT;*

Resultados

Instância	Algoritmo	Melhor Solução	Tempo de CPU (segundos)	Desvio (%)
Teste17	SA	7320	4767	4.93
	BT	7510	2099	5.94
	SA+BT	**7144**	5199	**1.29**
ICEB2001/2	SA	10600	5358	16.39
	BT	10191	5386	34.81
	SA+BT	**9208**	4532	**8.09**
Teste22	SA	26329	2835	10.87
	BT	28677	5276	27.7
	SA+BT	**25212**	1655	**7.35**

Obrigado pela atenção!

Referências:
- Experiências com simulated annealing e busca tabu na resolução do problema de alocação de salas, Marcone Jamilson Freitas Souza, Alexandre Xavier Martin, Cássio Roberto de Araújo.
http://www.decom.ufop.br/prof/marcone/Publicacoes/SBPO-2002-PAS-TC0106.pdf