VARIABLE NEIGHBORHOOD SEARCH

RICARDO MATOS

MARCOS GROSS

GIOVANNI GUIMARÃES

VARIABLE NEIGHBORHOOD SEARCH

Proposta em 1997 por Hansen e Mladnovic

Metaheurística que explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança.

Explora vizinhanças cada vez mais distantes da solução atual.

VARIABLE NEIGHBORHOOD SEARCH

Baseada em três fatos simples:

Um mínimo local em relação a uma vizinhança não necessariamente corresponde a um mínimo local em relação a outra vizinhança.

Um mínimo global é um mínimo local em relação a todas estruturas de vizinhança.

Para muitos problemas, mínimos locais em relação a uma ou várias vizinhanças são relativamente próximos uns dos outros.

VARIABLE NEIGHBORHOOD SEARCH - 4 COMPONENTES

Gerador de solução inicial

Procedimento de perturbação (shaking), que gera uma perturbação na k-ésima vizinhança da solução atual.

Procedimento de busca local, para refinar a solução.

Procedimento de mudança de vizinhança, que define a próxima vizinhança a ser explorada.

VNS BÁSICA (BVNS)

 $\underline{\mathit{Inicializa}$ ção. Selecione um conjunto de estruturas de vizinhança N_k para $k = 1, ..., k_{max}$, que será utilizada na busca; encontre uma solução inicial x; escolha uma condição de parada; $\underline{\mathit{Repita}}$ a seguinte sequência até a condição de parada ser satisfeita:

- (1) $k \leftarrow 1$;
- (2) *Repita* os seguintes passos até $k = k_{max}$:
 - (a) Pertubação. Gere um ponto x' aleatoriamente da k-ésima vizinhança de x $(x' \in N_k(x));$
 - (b) Busca Local. Aplique um método de busca local com x' como solução inicial; Indique como x" o mínimo local encontrado;
 - (c) Mudança. Se o mínimo local x" for melhor do que a solução x, mover para lá $(x \leftarrow x)$, e continuar a busca com $N_1(k \leftarrow 1)$. Caso contrário, definir $k \leftarrow k+1$;

VARIABLE NEIGHBORHOOD DESCENT (VND)

Método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente.

Realiza a troca de vizinhanças de forma determinística.

O VND é baseada no fato de que "Um mínimo local em relação a uma estrutura de vizinhança não necessariamente corresponde a um mínimo local para outra vizinhança"

VND

 $\underline{\mathit{Inicialização}}$ Selecione um conjunto de estruturas de vizinhança N_L , para $L=1,\ldots,L_{max}$, que será usado na descida; encontre uma solução inicial x (ou aplicar a uma dada solução x).;

Repita a seguinte sequência até que nenhuma melhoria seja obtida:

- $(1) L \leftarrow 1;$
- (2) *Repita* os seguintes passos até $L = L_{max}$:
 - (a) Exploração de vizinhança. Encontre o melhor x' do vizinho de x (x' \in N_L(x));
 - (b) *Mudança*. Se a solução x' for melhor do que x, então x ← x' e L ← 1. Caso contrário, L ← L+1;

GENERAL
VARIABLE
NEIGHBORHOOD
SEARCH
(GVNS)

 $\underline{Inicialização}$. Selecione um conjunto de estruturas de vizinhança N_k para $k=1, \ldots, k_{max}$, que será usado na fase de perturbação, e um conjunto de estruturas de vizinhança N_L , para $L=1,\ldots, L_{max}$ que será usado na busca local. Encontrar uma solução inicial. Escolher um critério de parada;

Repita a seguinte sequência até a condição de parada ser satisfeita:

- $(1) k \leftarrow 1;$
- (2) <u>Repita</u> os seguintes passos até $k = k_{max}$:
 - (a) *Perturbação*. Gere um ponto x' aleatoriamente na k-ésima vizinhança de x;
 - (b) Busca local com VND.

$$(b1) L \leftarrow 1;$$

- (b2) Repita os seguintes passos até que $L = L_{max}$:
 - Exploração de vizinhança. Encontre o melhor x" vizinho de x' em N_L(x');
 - Mudança. Se f(x") < f(x'), então x' ← x" e L ← 1; Caso contrário,
 L ← L + 1;
- (c) Mudança. Se o mínimo local x" for melhor do que a solução corrente x, então $x \leftarrow x$ " e a busca continua com N_1 ($k \leftarrow 1$); Caso contrário, $k \leftarrow k + 1$;

APLICAÇÃO

PROBLEMA DE ALOCAÇÃO DE SALAS

PROBLEMA DE ALOCAÇÃO DE SALAS

A VNS foi aplicada por (Sousa et al. 2002) no Problema de alocação de salas.

Esse problema diz respeito à distruibuição de aulas, com horários previamente estabelecidos, a salas de aulas.

Souza considerou o contexto do Instituto de Ciências Exatas e Biológicas da UFOP, à epoca (2002) com 1500 alunos, 250 turmas distribuídas em 20 salas e 29 laboratórios.

PROBLEMA DE ALOCAÇÃO DE SALAS

No processo de alocação de salas, são observados vários requisitos, como:

Em uma mesma sala e horário não pode haver mais de uma aula

Uma sala não pode receber uma turma cuja quantidade de alunos seja superior à sua capacidade

Sempre que possível, alocar a uma mesma sala alunos de um mesmo curso e período

Utilizar o espaço das salas eficientemente

REPRESENTAÇÃO DO PROBLEMA

	Salas					
		1	2	3	4	5
Horários	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

ESTRUTURAS DE VIZINHANÇA

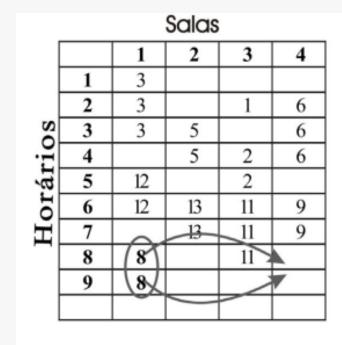
Dada uma solução s, para atingir uma solução s', onde s' é dito vizinho de s, são usados dois tipos de movimento: Alocação e Troca.

A partir desses movimentos, são definidas três estruturas de vizinhança: N⁽¹⁾ (s), N⁽²⁾(s) e N⁽³⁾(s)

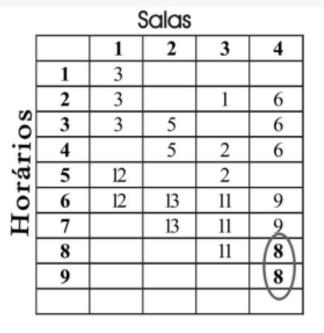
O movimento de Alocação (chamado de 1-optimal) consiste em realocar as aulas de uma dada turma e sala a uma outra sala que esteja vazia nos horários das aulas.

O movimento de Troca (chamado de 2-optimal) consiste em trocar de sala as aulas de duas turmas realizadas no mesmo bloco de horários.

MOVIMENTO DE ALOCAÇÃO

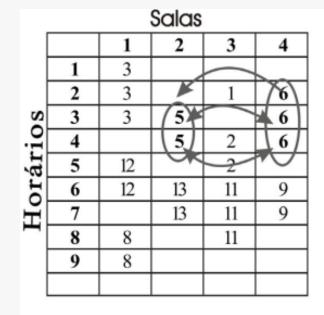


C 1	~	
Sol	ução	S
		-



Solução s'

MOVIMENTO DE TROCA



Solução s

	Salas				
		1	2	3	4
Horários	1	3			
	2	3	(6)	1	
	3	3	6		(5)
	4		(6/	2	(5)
ár	5	12		2	
or	6	12	13	11	9
Ĕ	7		13	11	9
	8	8		11	
	9	8			

Salas

Solução s'

(Souza et al. 2002) definiu o problema de alocação de salas como um problema de decisão multicritério, pois para determinar a qualidade de uma alocação é necessário considerar diferentes objetivos.

Para avaliar uma alocação, os requisitos do problema foram divididos em duas categorias:

- (1) Requisitos essenciais: se não forem satisfeitos, geram uma solução inviável. Por exemplo, uma sala com duas turmas no mesmo horário.
- (2) Requisitos não-essenciais: São desejáveis, mas não geram soluções inviáveis se não forem satisfeitos.

Desse modo, uma alocação (solução) pode ser medida com base em duas componentes:

- (1) Uma de inviabilidade (g(s)), que mede o atendimento aos requisitos essenciais
- (2) Uma de qualidade (h(s)), que mede o atendimento aos requisitos não essenciais.

A função objetivo será uma f(s) = g(s) + h(s)

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

K representa o número de medidas de inviabilidade I_k O valor da k-ésima medida de inviabilidade α_k o peso associado à essa k-ésima medida

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

L representa o número de medidas de qualidade Q_l é valor da l-ésima medida de qualidade β_l o peso associado à essa l-ésima medida

GERAÇÃO DE SOLUÇÃO INICIAL

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.

A seguir, uma dessas salas é escolhida aleatoriamente para receber a aula. Esse procedimento continua até que todas as aulas sejam alocadas.

ALGORITMOS ANALISADOS

Foram analisadas duas variantes do VNS:

GVNS (VNS+VND)

VNS+MV, na qual a busca local consiste em determinar o melhor vizinho da solução corrente usando a estrutura corrente de vizinhança

RESULTADOS COMPUTACIONAIS

Os algoritmos foram implementados na linguagem C++

Microcomputador PC AMD Athlon, 800 MHz, 128 MB de RAM, sob o sistema operacional Windows 2000

RESULTADOS COMPUTACIONAIS

Instância	Número	Número	Número de
	de salas	de turmas	horas-aula
Teste17	17	214	713
Testereal	20	233	763
Teste22	22	281	938

RESULTADOS COMPUTACIONAIS

Instância	Algoritmo	Melhor	Média
		Solução	
	VNS+VND	7330	8378
Teste17	VNS+MV	7750	8978
	VNS+VND	10733	15280
Testereal	VNS+MV	11210	16273
	VNS+VND	31339	38035
Teste22	VNS+MV	32189	39581

Para cada teste, foram realizadas 5 execuções de cada algoritmo. Utilizou-se como critério de parada um tempo de execução de 5000 segundos.

OBRIGADO!