Problemas de Roteamento de Veículos: Abordagens Eficientes Sequenciais e Paralelos em Ambientes Heterogêneos (CPU & GPU).

Autores: *Luiz Satoru Ochi (IC-UFF), Igor Machado Coelho (IME-UERJ), Puca Huachi Vaz Penna (Inf-UFF)*
http://www2.ic.uff.br/~satoru/

Mini Curso – XLVII SBPO

2015

SOBRAPO
Sociedade Brasileira de Pesquisa Operacional

Universidade Federal Fluminense

Instituto de Computação

UERJ
UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

Métodos Eficientes para a solução de Problemas de Roteamento & Scheduling de uma Frota de Veículos

Equipe de Pesquisadores & Colaboradores deste Tema

*Luiz Satoru Ochi (IC-UFF**) - Coordenador***
Anand Subramanian (UFPB)
Christian Prins (University of Technology of Troyes - UTT)
Edcarlos G. dos Santos (IC-UFF)
El-Ghazali Talbi (Polytech'Lille - University of Lille)
Igor Machado Coelho (IME-UERJ & IC-UFF)
Lucídio Formiga Cabral (UFPB)
Luidi Simoneti (COPPE-SISTEMAS/UFRJ)
Marcone Jamilson Freitas Souza (UFOP)
Matheus Nohra Haddad (IC-UFF & University of Vienna)
Marques Moreira de Sousa (IC-UFF)
Nelson Maculan (UFRJ & IFORS)
Philippe Michelon (Université d'Avignon et des Pays de Vaucluse)
Puca Huachi Vaz Penna (IC-UFF-Pádua/RJ & University of Troyes - UTT)
Pablo L Munhoz (IC-UFF & University of Avignon)
Richard Harth (University of Vienna – Austria)
Thibaut Vidal (INF-PUC-Rio)
Yuri Abitibol (IC-UFF)
Grupo de Pesquisa Operacional da Petrobrás/RJ
Grupo de Pesquisa Operacional do IBGE/RJ

Problemas de Roteamento de Veículos (PRV)
The Vehicle Routing Problems (VRP)

O PRV teve sua origem associada ao trabalho desenvolvido em [Dantzig & Ramser, 1959], denominado *The Truck Dispatching Problem*.

Desde então, tem sido, particularmente nas últimas décadas, um dos problemas mais abordados nas áreas de Otimização Combinatória (OC) e Pesquisa Operacional (PO).

Isso se deve, em parte, ao grande desenvolvimento de métodos de solução e da enorme variedade de aplicações existentes para o PRV.

Outro aspecto que tem contribuído de forma significativa para este sucesso, é a eficiência destes métodos no sentido operacional, ou seja, muitas das técnicas desenvolvidas têm se mostrado muito eficientes quando implementadas em situações reais em diferentes empresas das áreas de transporte.

O PRV é uma generalização do Clássico Problema do Caixeiro Viajante (PCV) e pertence a classe *NP-Hard*

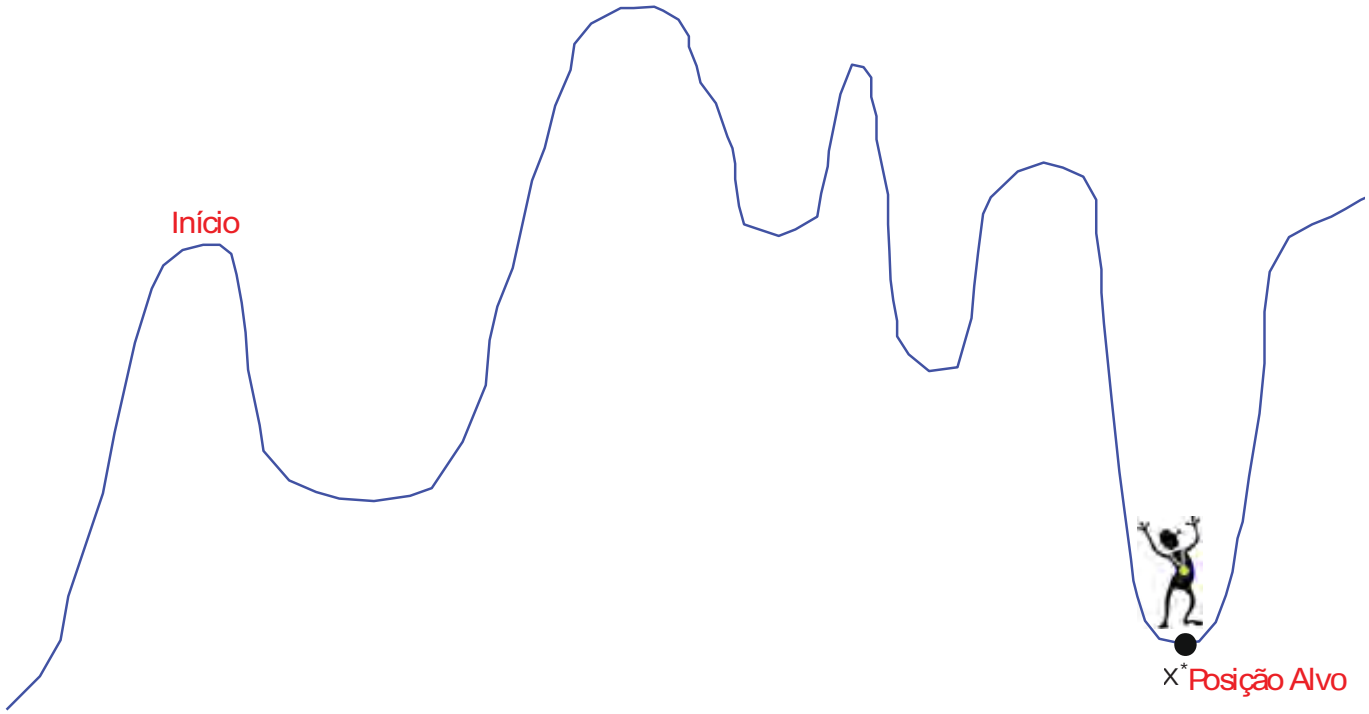
Uma ilustração de uma solução para o PRV

Fonte: Kramer, R.H.F.R *et al*. 2015

Meta-heurísticas

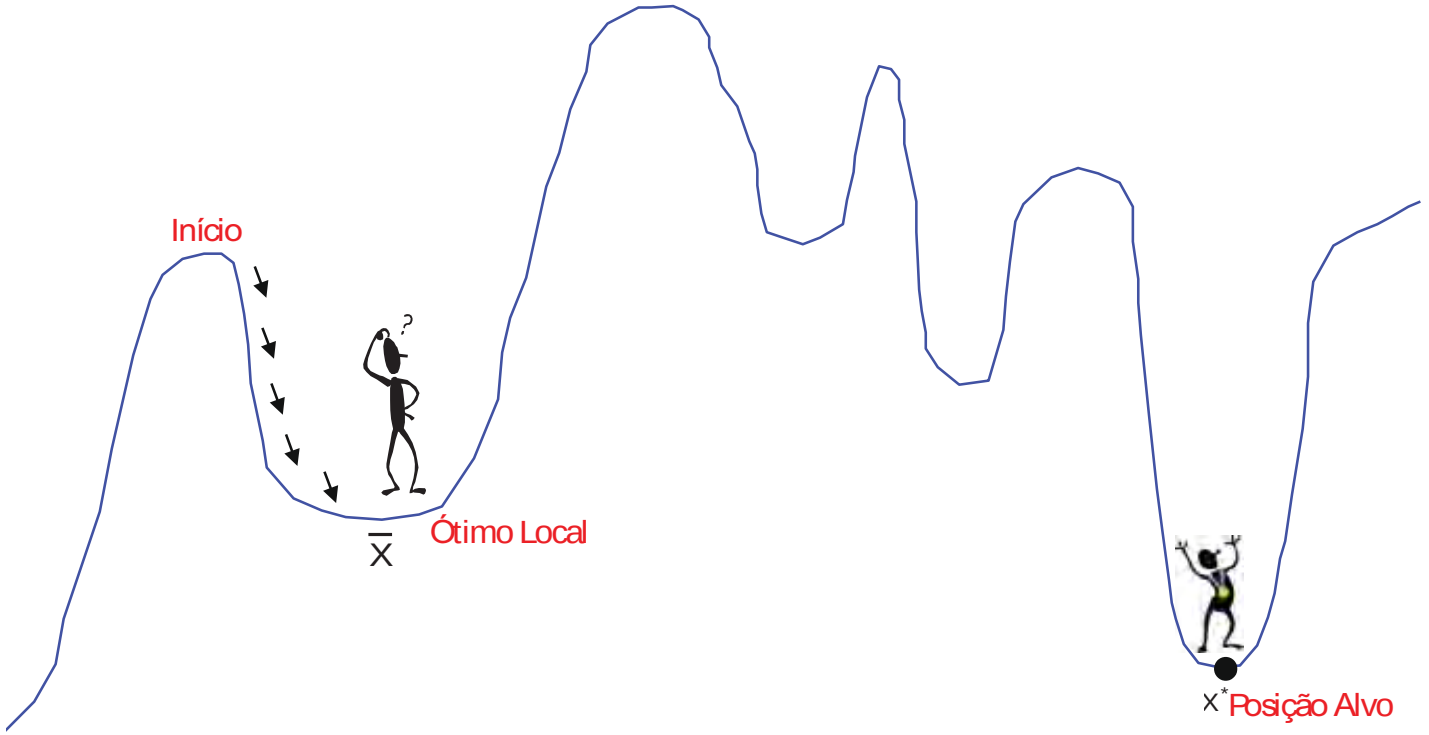
Quais são as principais diferenças entre Heurísticas tradicionais e Meta-heurísticas?

Meta-heurísticas ao contrário das Heurísticas tradicionais possuem ferramentas que ajudam a escapar de ótimos locais ainda distantes de um ótimo global

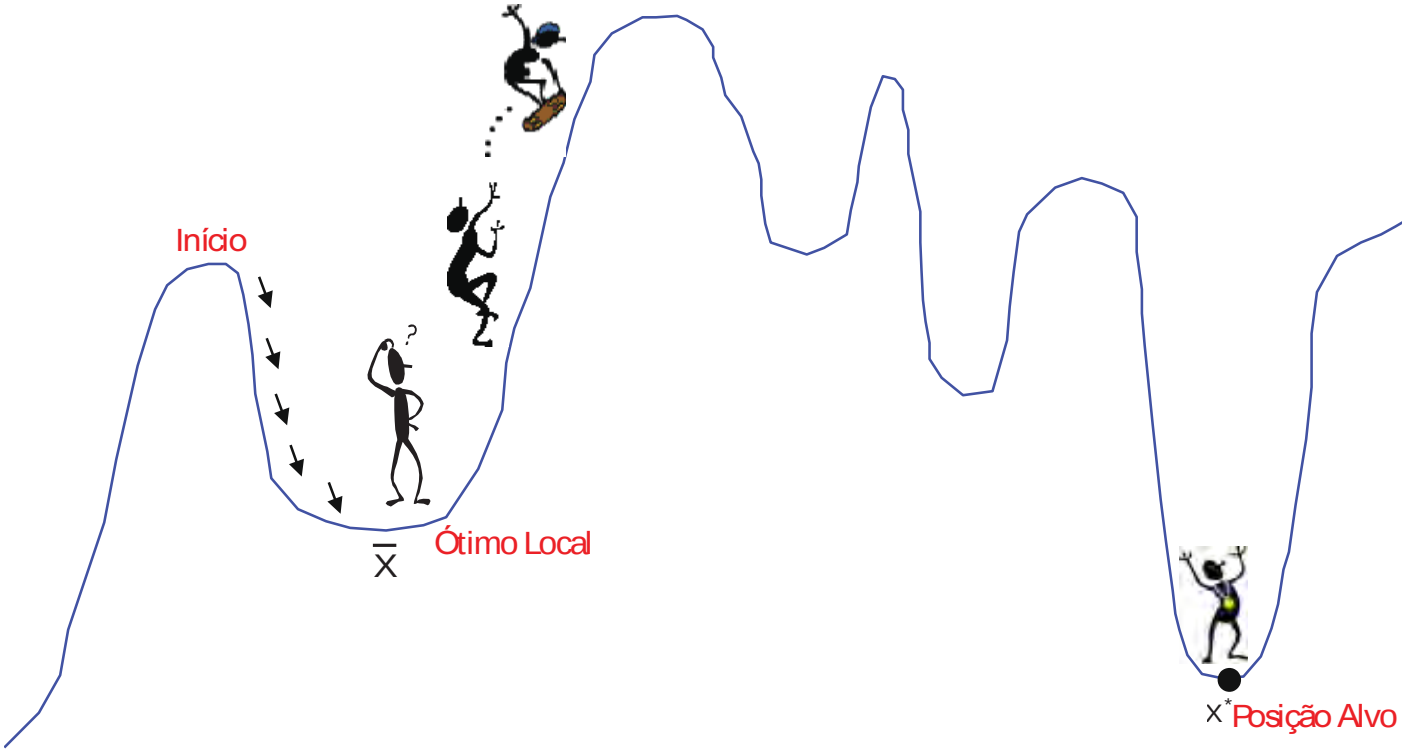


Mas quais são as principais diferenças entre Heurísticas tradicionais e Meta-heurísticas?

Comportamento típico de uma heurística gulosa



Comportamento típico de uma Meta-heurística



Metaheurísticas mais populares atualmente

- Redes Neurais – RN (Neural Networks)
- Simulated Annealing (SA)
- Algoritmos Genéticos - AG(Algoritmos Evolutivos, Programação Genética, Scatter Search, etc)
- Busca Tabu – BT (Tabu Search)
- GRASP (Greedy Randomized Adaptive Search Procedure)
- Iterated Local Search (ILS)
- VNS (Variable Neighborhood Search, VND, RVND, GVNS, etc)
- Ant Colony Optimization (ACO)
- Guide Local Search
-Hyper Heuristics
- outros..

Referências:
1. Handbook of Metaheuristics: Michel Gendreau, Jeaa Yves Potvin (editors)
2. Hybrid Metaheuristics: Christian Blum, M.J.Aguilera,Andrea Roli, M. Sampels(editors).

Tendências

Na área de meta-heurísticas, atualmente existe tendência ao uso de Métodos Híbridos:

H1) *Conjugando várias meta-heurísticas num mesmo algoritmo*: ex:

. AG/AE com população inicial gerado pela etapa de construção da Meta-heurística GRASP.

. AG/AE, com módulo de Busca Local para um subconjunto das melhores soluções geradas (conjunto elite).

. meta-heurísticas com Busca Local tipo VNS, VND, ou RVND.

•*Exemplo bem sucedido:* ***ILS com busca local RVND***.

Na área de meta-heurísticas, atualmente existe tendência ao uso de Métodos Híbridos:

Outra forma de Métodos Híbridos que tem feito sucesso nos últimos anos

H2) Conjugando meta-heurísticas com Métodos Exatos de Programação Matemática: ***Matheuristics.***

Exemplos:

E1. Efetuar uma busca local mais "pesada" utilizando uma formulação matemática do Problema + software CPLEX, XPRESS, outros.

- Exemplo: ***"Local Branching"*** **(LB):** Proposto por Fischetti e Lodi em 2002.

Alguns exemplos bem sucedidos: Conjugando meta-heurísticas com Métodos Exatos de Programação Matemática

1.“A hybrid algorithm for a class of vehicle routing problems”. Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. **Computers & Operations Research, Volume 40, pp. 2519-2533, (2013).**

2.“Large Neighborhoods with implicit customer selection for Prize-Collecting Vehicle Routing Problem and Team-Orienteering Problems”. **Autores**: Thibaut Vidal (MIT), Puca H Vaz Penna(UFF), Nelson Maculan(IFORS), and Luiz Satoru Ochi(UFF). Proc. of the **XLV SBPO** (*Artigo selecionado entre os 5 melhores papers do SBPO **2013***).

3.“GRASP with Path Relinking for the Symmetric Euclidean Clustered Traveling Salesman Problem”. **Autores**: Mário Mestria, Luiz Satoru Ochi, and Simone Lima Martins. In ***Computers & Operations Research* (COR) –** ELSEVIER, 40, pp. 3218-3229 (**2013**).

4.“A Hybrid Algorithm for the Heterogeneous Fleet Vehicle Routing Problem" (2012). **Autores**: Puca Huachi Vaz Penna, Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. **European Journal of Operational Research - EJOR** – ELSEVIER, Volume 221, pp: 285-295, (**2012**).

5."An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem". **Autores**: Puca H V Penna, Luiz Satoru Ochi, and Anand Subramanian. **Journal of Heuristics,** Volume 19(2), pp. 201-232, **2013**

6.“An Iterated Local Search heuristic for the Split Delivery Vehicle Routing Problem”. **Autores**: Marcos Melo Silva, and Luiz Satoru Ochi. **Computers & Operations Research (COR), 2015**.

7.“Large Neighborhoods with Implicit Customer Selection for Vehicle Routing Problems with Profits”. **Autores**: Thibaut Vidal; Nelson Maculan; Luiz Satoru Ochi; and Puca H. V. Penna. **Transportation Science, pp. 234-249, 2015.**

8. "A New Hybrid Heuristic for Replica Placement and Request Distribution in Content Distribution Networks**“**., **Autore**s: Neves,Tiago A , Ochi, Luiz Satoru., Albuquerque, Celio. (2015) **. Optimization Letters - Springer,volume 9(4), pp. 677-692, 2015**

H3) meta-heurísticas Híbridas Paralelas CPU & GPU

•***Algoritmos paralelos utilizando placas gráficas e linguagem CUDA tem trazido contribuições interessantes na área de Pesquisa Operacional.***

Exemplos:

•**"**A hybrid CPU-GPU local search heuristic for the unrelated parallel machine scheduling problem". **Autores**: Igor Coelho Machado, Matheus Haddad, Luiz Satoru Ochi, Marcone J F Souza, Ricardo Farias. Proceedings of the **WAMCA2012,** 3rd Workshop on Applications for Multi-core Architectures – Held in conjunction with the **24th International Symposium on Computer Architecture (WAMCA-SBAC-PAD2012), IEEE Press**, October 2012, NY**.**

•"An Integrated CPU-GPU Heuristic Inspired on Variable Neighborhood Search for the Single Vehicle Routing Problem with Deliveries and Selective Pickups". **Autores**: Igor Coelho Machado , Luiz Satoru Ochi, Marcone J F Souza, Ricardo Farias, Cristiana Bentes. Aceito no **International Journal of Production Research** – IJPR, 2015.

• **Teses de Doutorado** do IC-UFF de: Igor Machado Coelho (2015), e de Eyder Rios (2015-em andamento)

Similaridades do Problema de Roteamento de Veículos com outros Problemas Clássicos da Literatura de Computação:

- Problema de Clusterização em uma Base de Dados (Data Mining).

- Problema de Escalonamento de Tarefas em Múltiplos Processadores.

- Problemas de Roteamento em outros tipos de Redes: Redes de Computadores; ou mais genericamente: Redes de Comunicação de Dados.

⇘ **Problemas de Roteamento pode ser visto como um Problema de primeiro Clusterizar (Agrupar) objetos de um conjunto e numa segunda etapa, resolver um Problema de Sequenciamento de Tarefas de cada Cluster:**

⇘**Clusterização** pode ser entendida como um processo de particionar uma base de dados (objetos, elementos) em grupos (clusters) disjuntos de forma que objetos similares fiquem num mesmo cluster.

⇘ **Clusterização** tem sido muito estudado por matemáticos e estatísticos há dezenas de anos. Mais recentemente este problema passou a ser explorado também na área de computação como um problema de mineração de dados (***Data Mining***).

O Problema de Clusterização (PC) é muito difícil de se resolver: Porque?????

lQuando o número *k* de clusters é definido como parâmetro de entrada: **Problema de Clusterização - PC**;

$$N(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n$$

Exemplos:

$N(10, 2) = 511$;

$N(100, 5) = 6{,}57384 \times 10^{67}$

$N(100, 2) = 6{,}33825 \times 10^{29}$

$N(1000, 2) = 5{,}3575 \times 10^{300}$

Quando valor K não é conhecido préviamente, temos o Problema de Clusterização Automática (PCA)- Mais DIFÍCIL!!!!!

$$N(n,k) = \sum_{k}(1/k!)\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n$$

Aplicações do PCA desenvolvidas no **LabIC/**IC-UFF

Laboratório de Inteligência Computacional do IC-UFF

- *Projeto de Formação de Células num Sistema de manufatura*

	M1	M2	M3	M4
P1		1	1	
P2	1	1		1
P3		1	1	
P4	1			1
P5	1	1		
P6		1	1	1

	M2	M3	M1	M4
P1	1	1		
P3	1	1		
P6	1	1		1
P2	1		1	1
P4			1	1
P5	1		1	

Aplicações do PCA desenvolvidas no **LabIC/**IC-UFF:

Laboratório de Inteligência Computacional do IC-UFF

-Problema de Escalonamento de Tarefas em Múltiplos Processadores:

Clusterização + Sequenciamento**.**

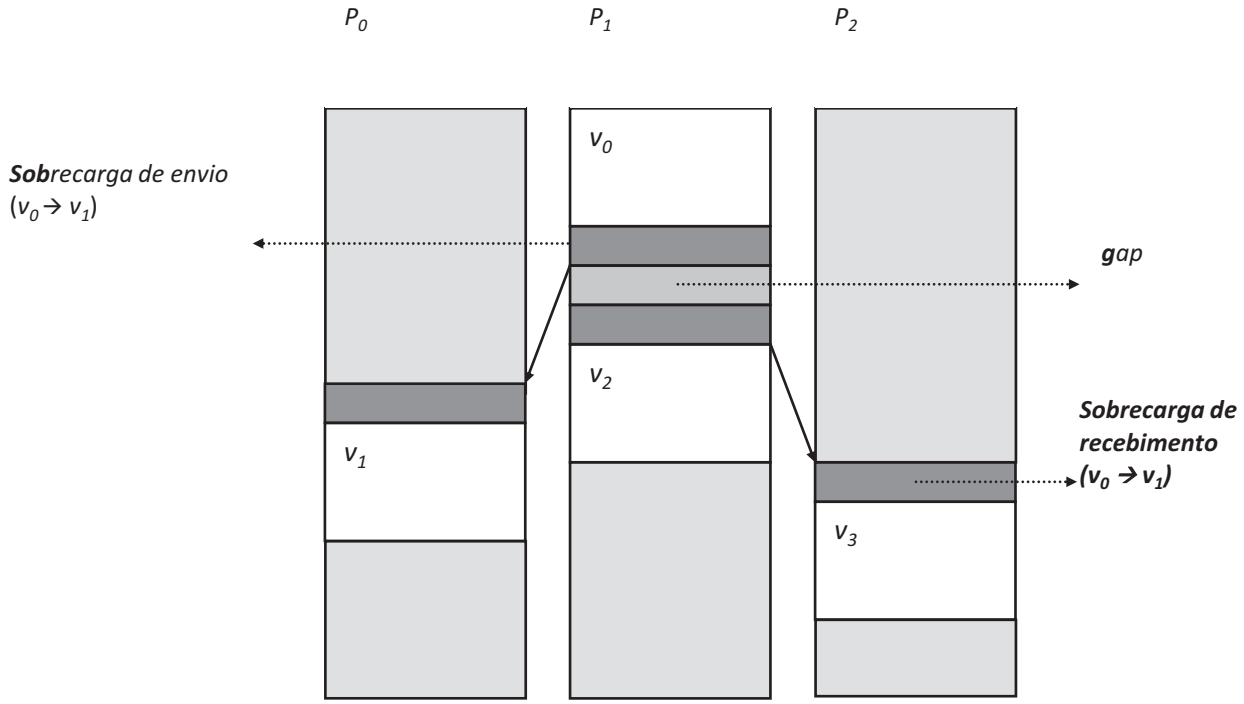
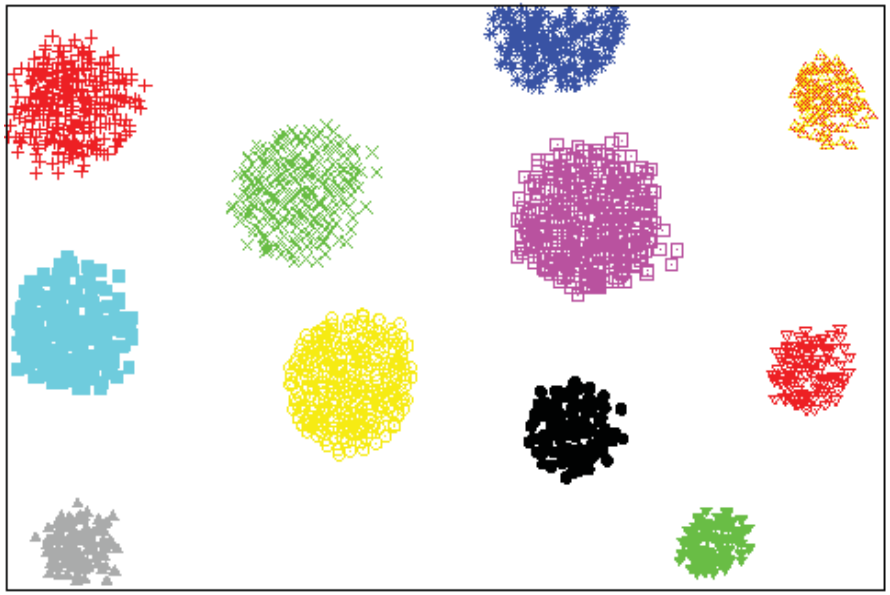


*Figura **:** Possível escalonamento para a aplicação da Figura 1(a), utilizando o modelo LogP*

Meta-heurísticas para o Problema de Clusterização Automática

Instância *2000p11c*

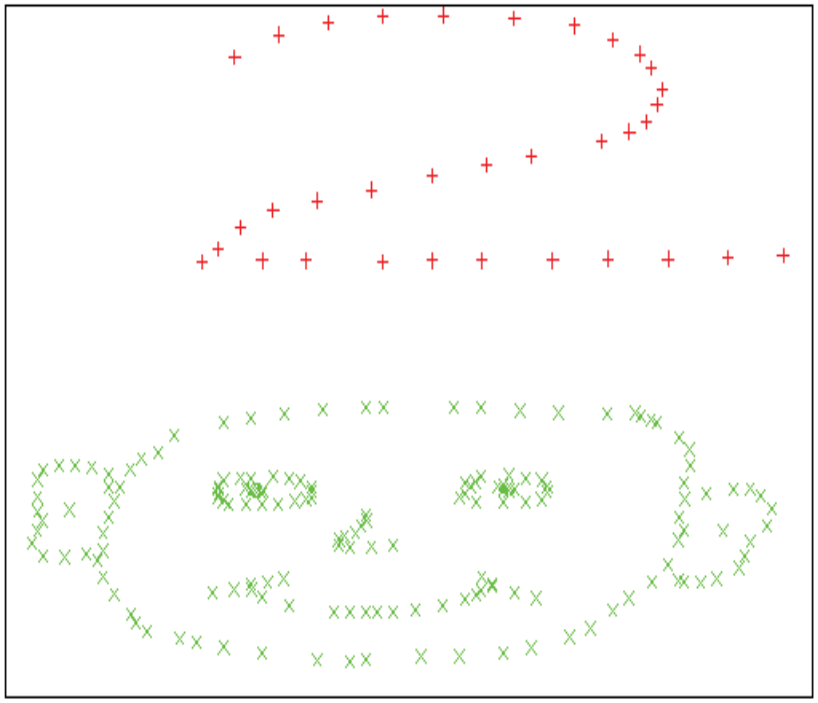


Resultado para o AEC

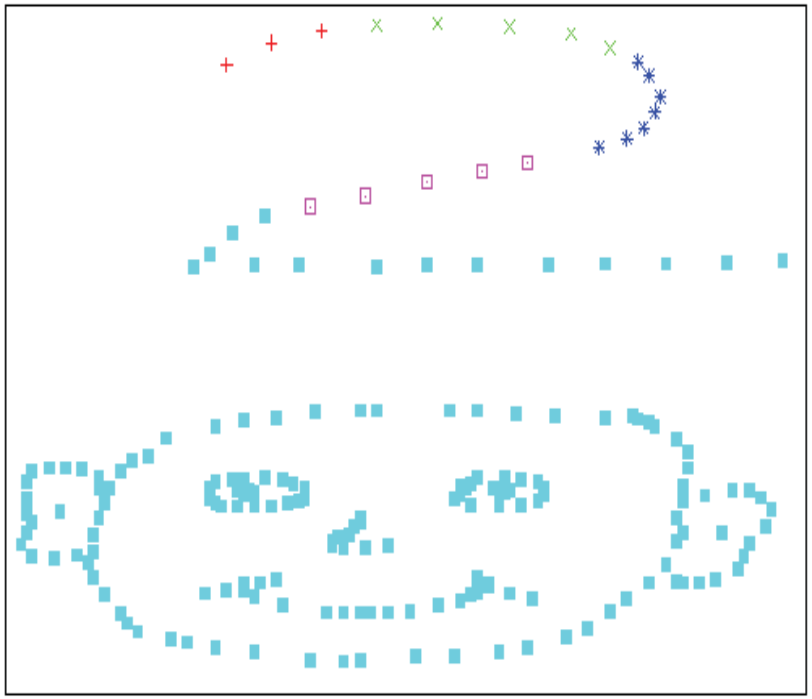
Resultado para o CLUSTERING

Meta-heurísticas para o Problema de Clusterização Automática

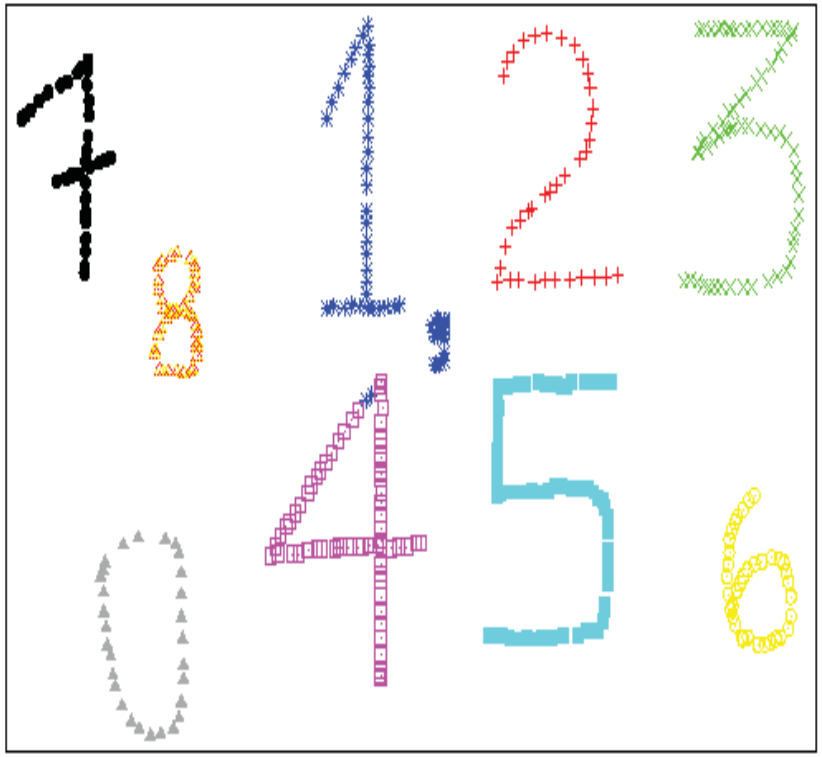
Instância *2face*



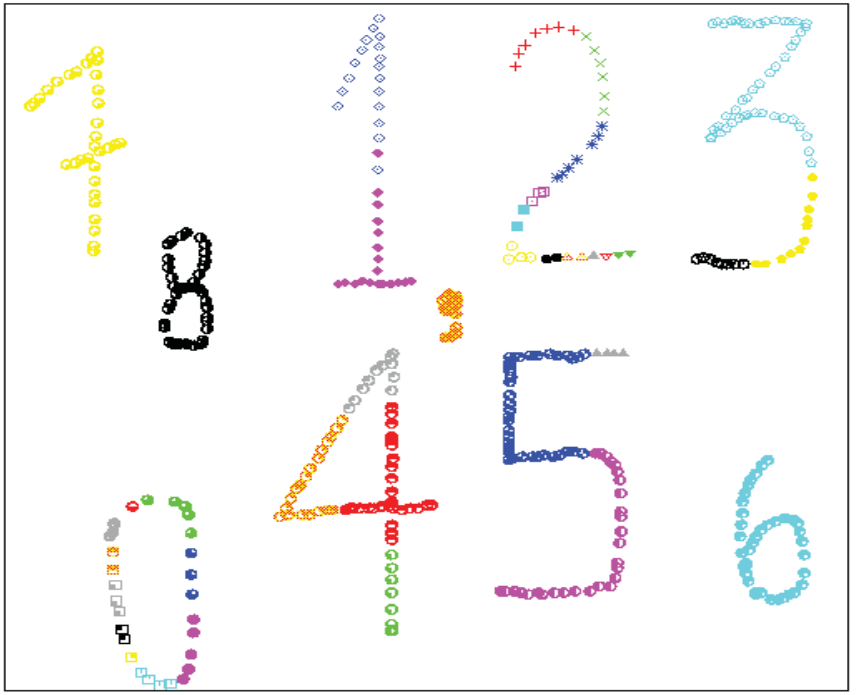
Resultado para o AEC



Resultado para o CLUSTERING



Resultado para o **AEC**



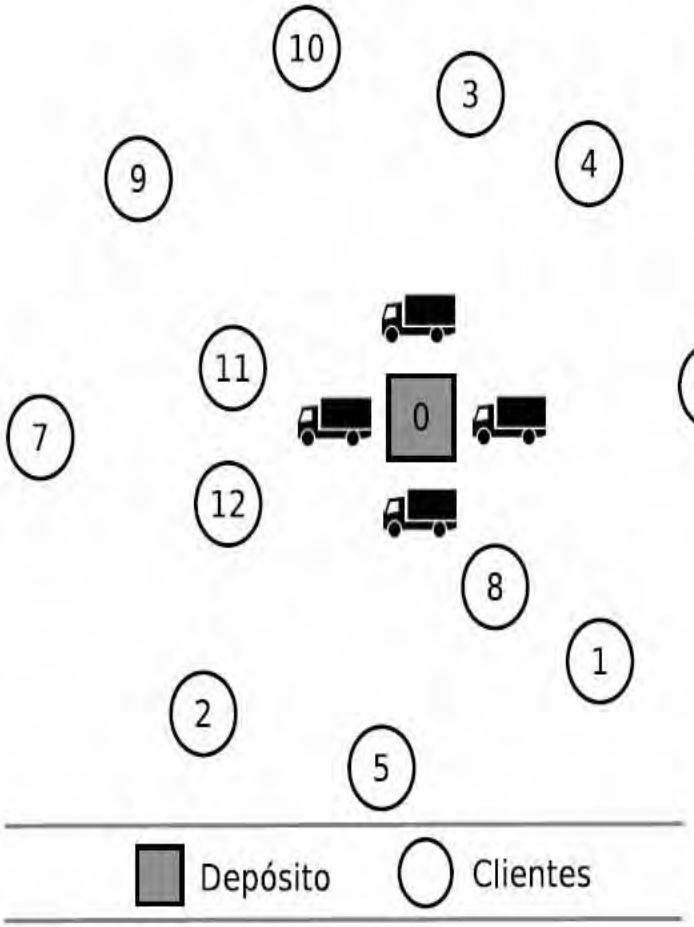
Resultado para o **CLUSTERING**

Alguns Modelos de Problemas de Roteamento de Veículos abordados pelo nosso grupo de pesquisa do : LabIC: *Laboratório de Inteligência Computacional do IC-UFF* :

P1. **Problemas de Roteamento de Veículos com Entregas e Coletas Opcionais**
P2. **Problemas de Roteamento de Veículos com Entregas Fracionárias.**
P3. Problemas de Roteamento de Veículos com Múltiplos Depósitos.
P4. Problemas de Roteamento de Veículos com Coleta e Entrega Simultânea.
P5. Problemas de Roteamento de Veículos com Frota Heterogênea.
P6. Problemas de Roteamento de Veículos Periódico.
P7. Problemas de Roteamento de Veículos com Time-Windows.
P8. Problemas Integrados de Planejamento de Produção, Estoque e Roteamento . de Veículos (*The Inventory Vehicle Routing Problem*)
P9. Algoritmos Paralelos para diferentes modelos de VRP....
P10. Algoritmos Paralelos usando CPU & GPU (Cuda) para VRP e variantes..
P11. Outros modelos de PRV...

Uma Meta-heurística Híbrida Baseada no *Iterated Local Search* (ILS) e *Variable Neighborhood Search* (VNS) para a solução de Problemas de Roteamento de Veículos com Entregas Fracionadas

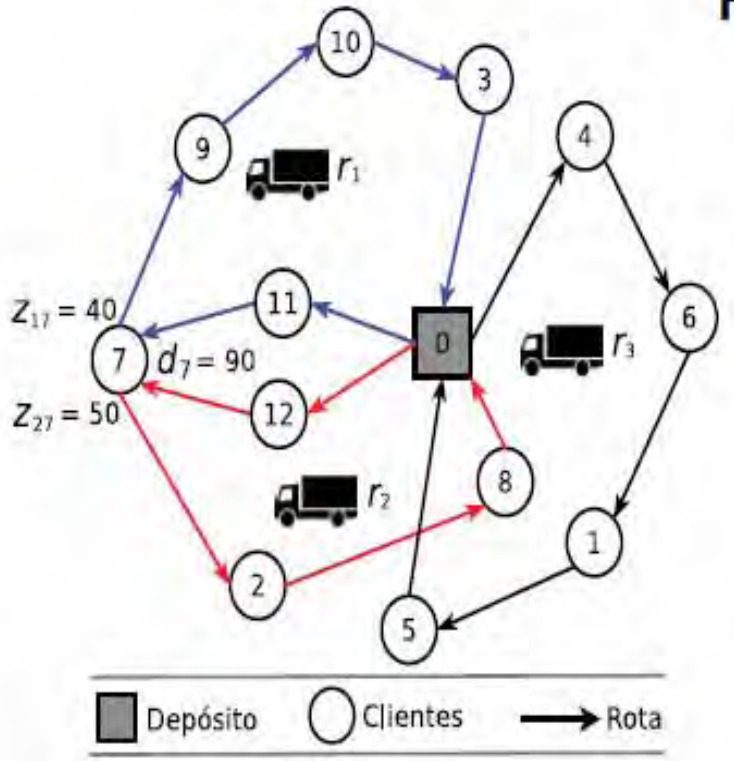
XLVII SBPO
2015



Problema de Roteamento de Veículos (PRV):

- Conjunto de clientes com demandas (d_i) conhecidas;
- Frota homogênea de veículos com capacidade Q

Problema de Roteamento de Veículos com Entregas Fracionárias



Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF):

- - Toda rota inicie e termine no depósito;
- - A soma das demandas dos clientes atendidos por uma dada rota não exceda a capacidade do veículo associado;
- - ~~Cada cliente tenha toda sua demanda atendida em uma única visita~~.

Problema de Roteamento de Veículos com Entregas Fracionárias

- O Problema de Roteamento de Veículos com Entregas Fracionárias (PRVEF) - *Split Delivery Vehicle Routing Problem* (SDVRP) - é uma relaxação do PRV. O PRVEF permite que os clientes tenham suas demandas atendidas por mais de um veículo.
- Apesar de ser uma relaxação do PRV, o PRVEF continua sendo NP-difícil.
- Duas versões do problema: Frota limitada e Frota ilimitada;
- Consequências:
  - A demanda dos clientes pode exceder a capacidade do veículo;
  - Número mínimo de veículos: $K_{min} = \lceil (\sum_{i=1}^{n} d_i)/Q \rceil$

Problema de Roteamento de Veículos com Entregas Fracionárias

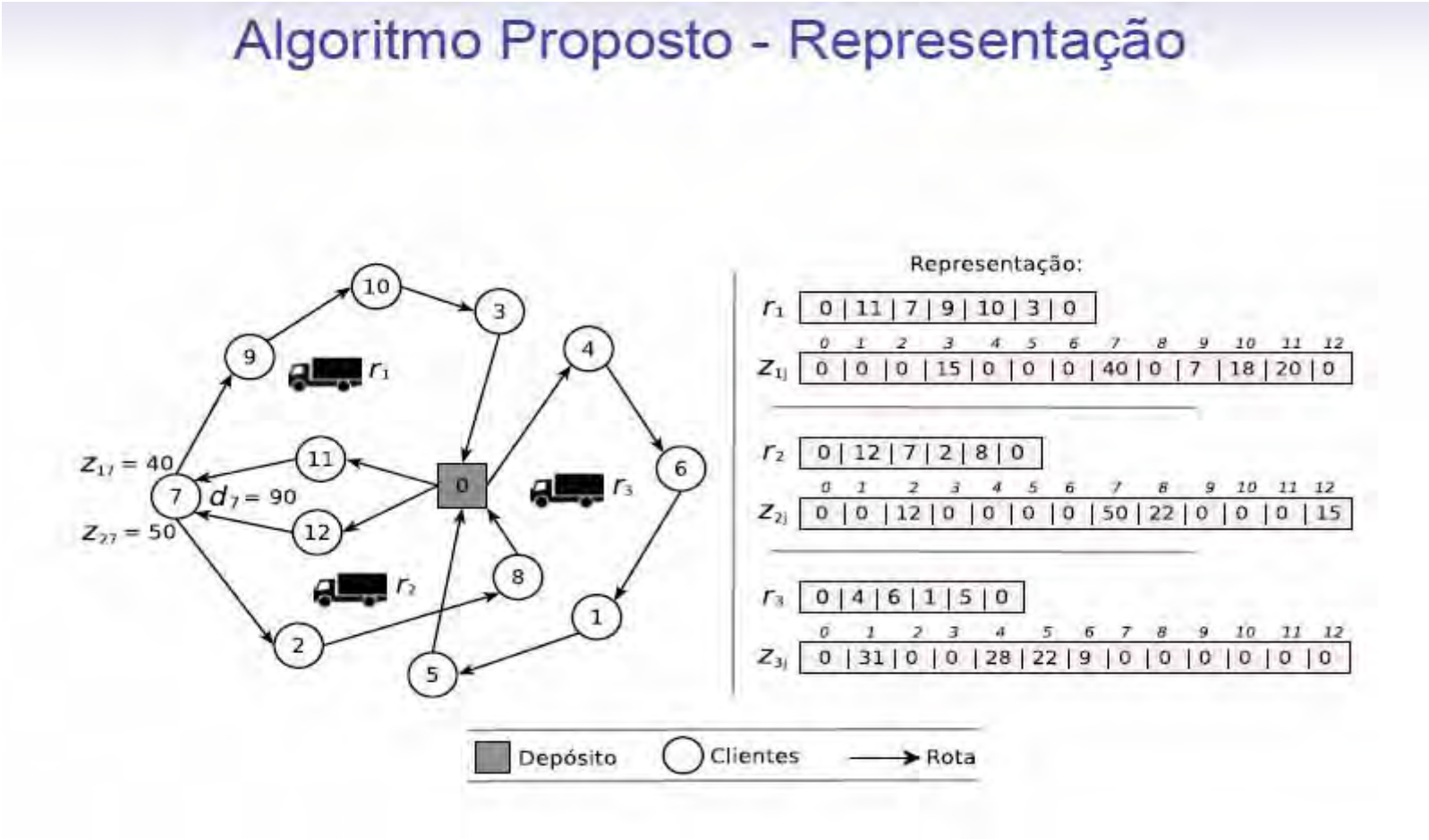
- O PRVEF foi proposto por Dror et. al. (1989)
- Mostraram que as economias obtidas, tanto em relação ao número de veículos utilizados quanto em relação a distância total percorrida, são significativas.
- Vantajoso entregas fracionárias quando a demanda média dos clientes é maior que 10% da capacidade do veículo.

Algoritmo Proposto

Heurística *multi-start* baseada na metaheurística *Iterated Local Search* (ILS);

```
1 Procedimento ILS
2 s0 ← GeraSolucaoInicial();
3 s* ← BuscaLocal(s0);
4 enquanto critério de parada não satisfeito faça
5     s' ← Perturba(s*, historico);
6     s'* ← BuscaLocal(s');
7     s* ← CriterioAceitacao(s*, s'*, historico);
8 fim enquanto
9 fim ILS
```

Início: Como vamos representar uma solução do PRVEF?



Na Meta-heurística ILS, temos que construir uma solução inicial válida para o problema...

Algoritmo Proposto - Construção de solução inicial

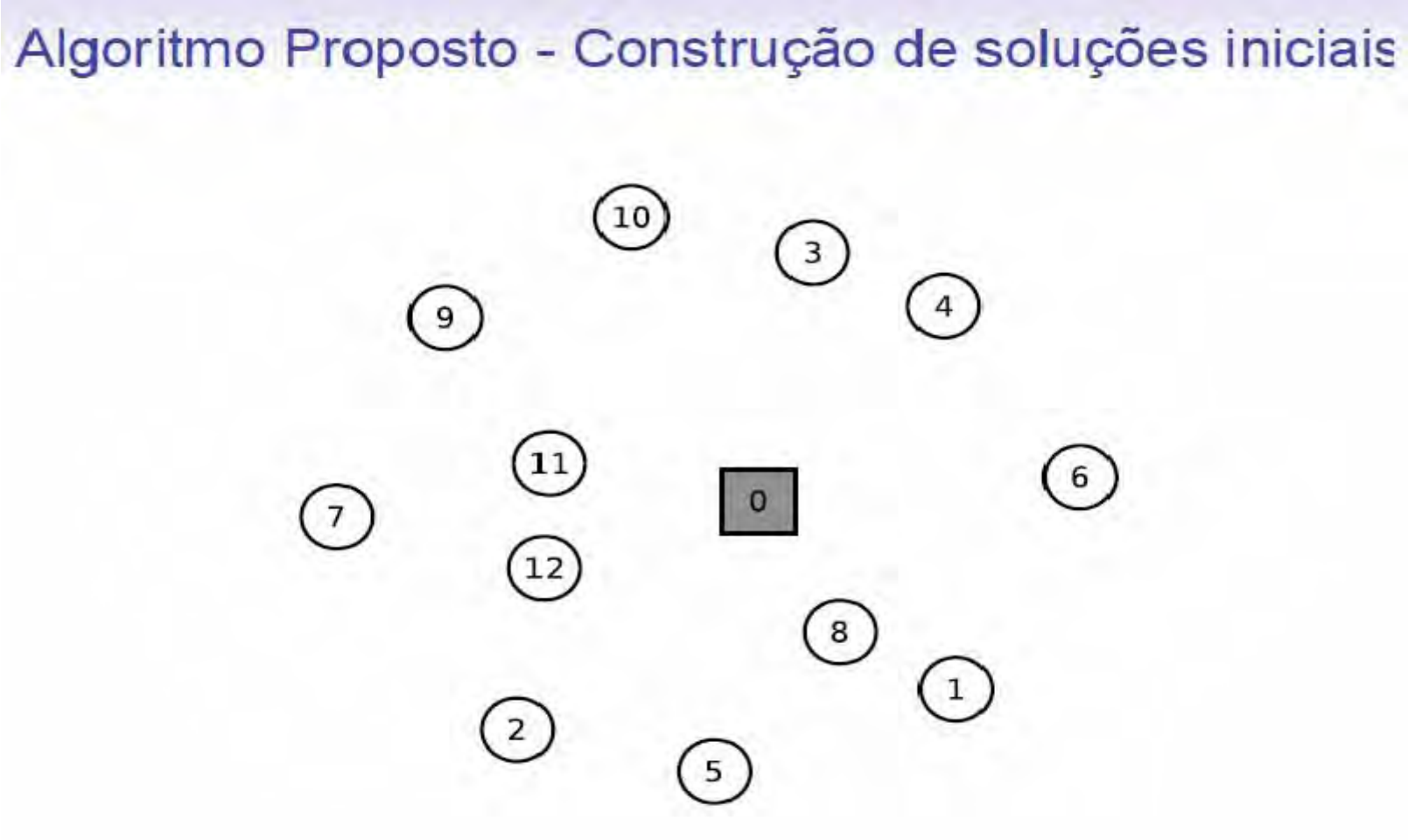
- Algoritmo de Inserção;
  - Estratégias de inserção
    - Sequencial;
    - Paralelo;
  - Critérios de inserção: $min\{g(i)|i \in LC\}$
    - Mais Barata Viável Modificada (MBVM):

$$g(i) = (c^r_{ji} + c^r_{ik} - c^r_{jk}) - \gamma(c^r_{0i} + c^r_{i0})$$

    - Mais Próxima Viável (MPV):

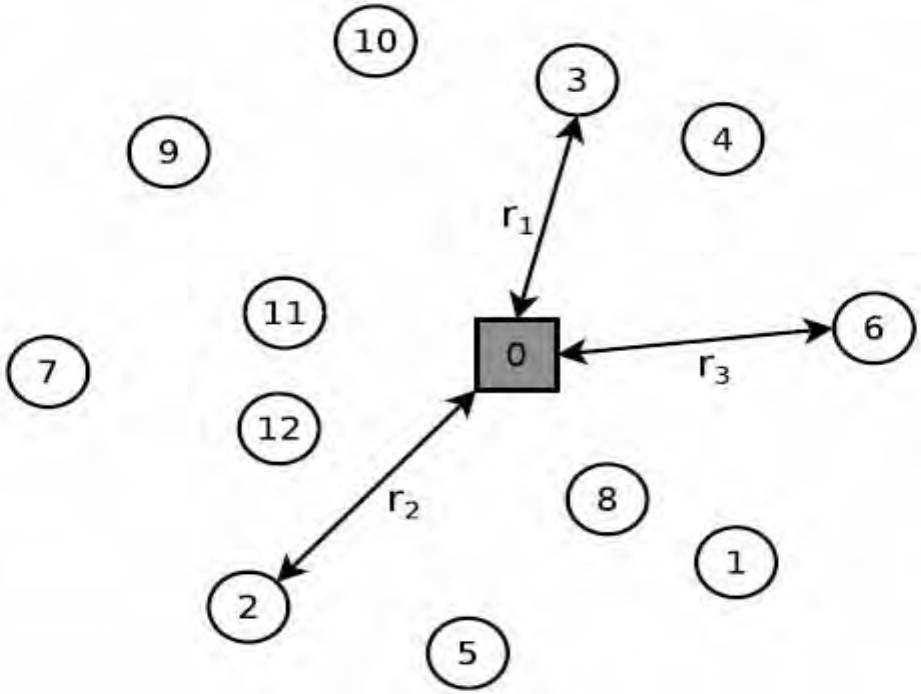
$$g(i) = c^r_{ji}$$

Passo a passo de como construir uma solução inicial para este problema



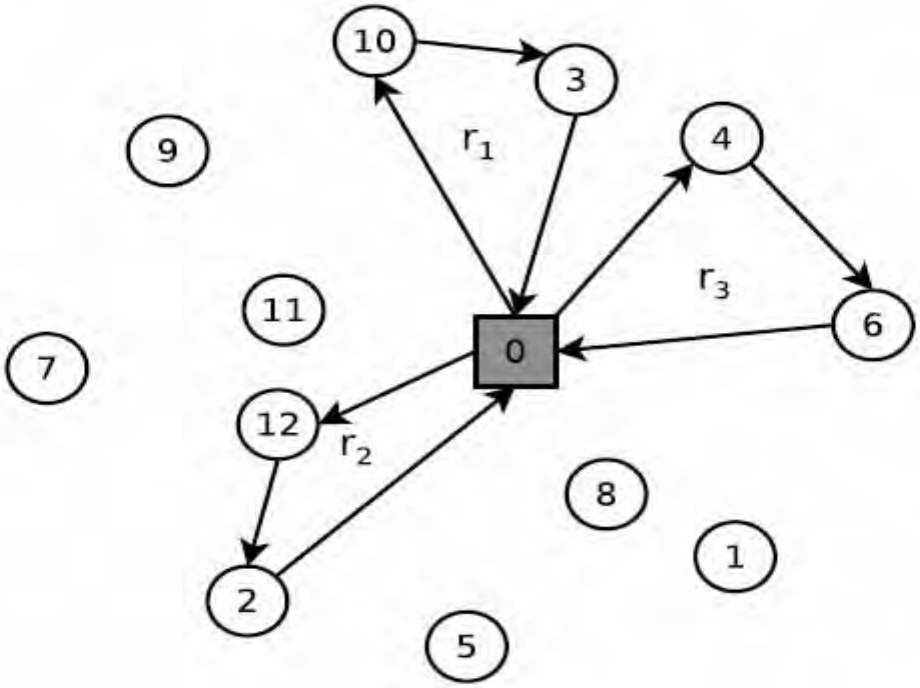
Passo a passo de como construir uma solução inicial para este problema

Algoritmo Proposto - Construção de soluções iniciais

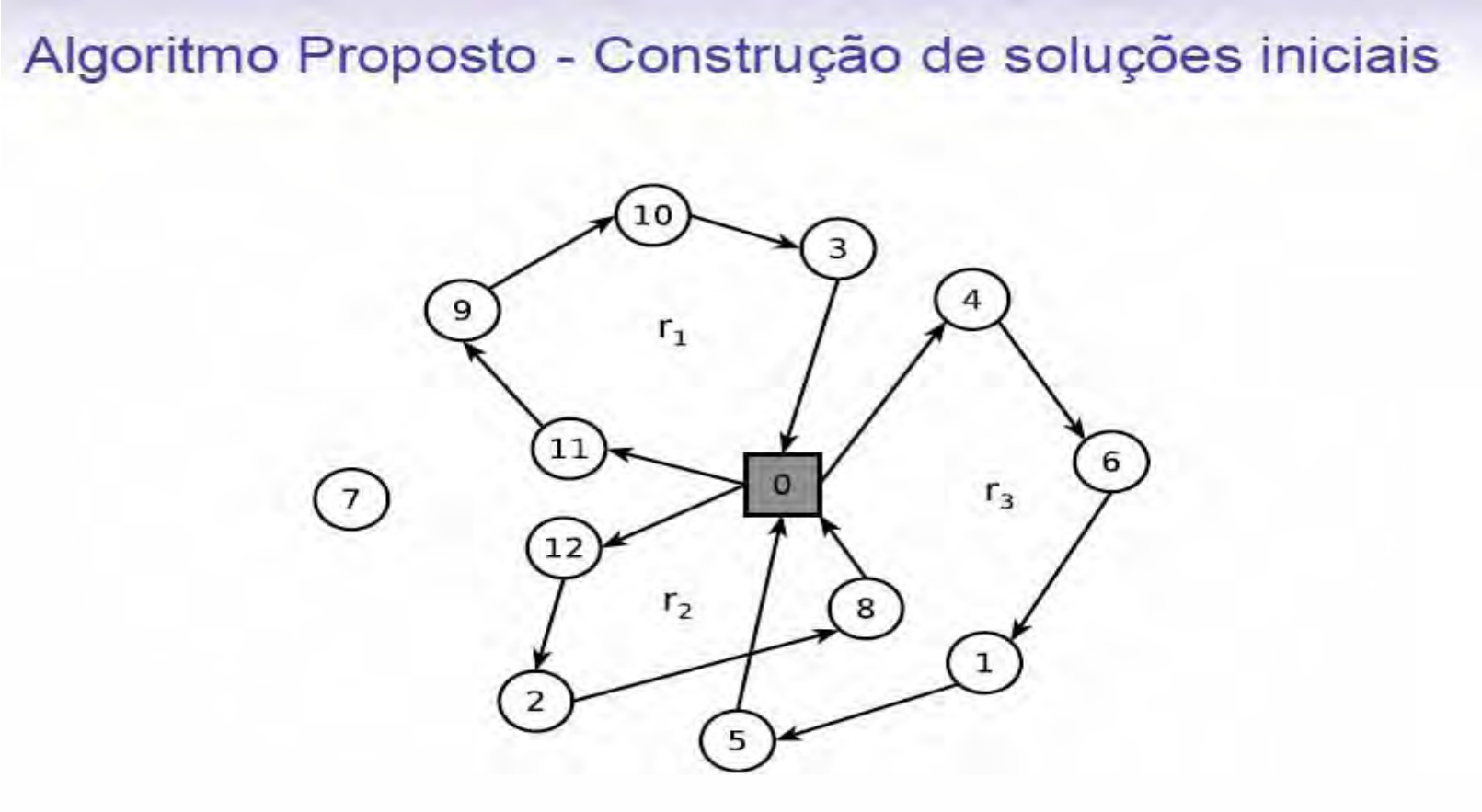


Passo a passo de como construir uma solução inicial para este problema

Algoritmo Proposto - Construção de soluções iniciais

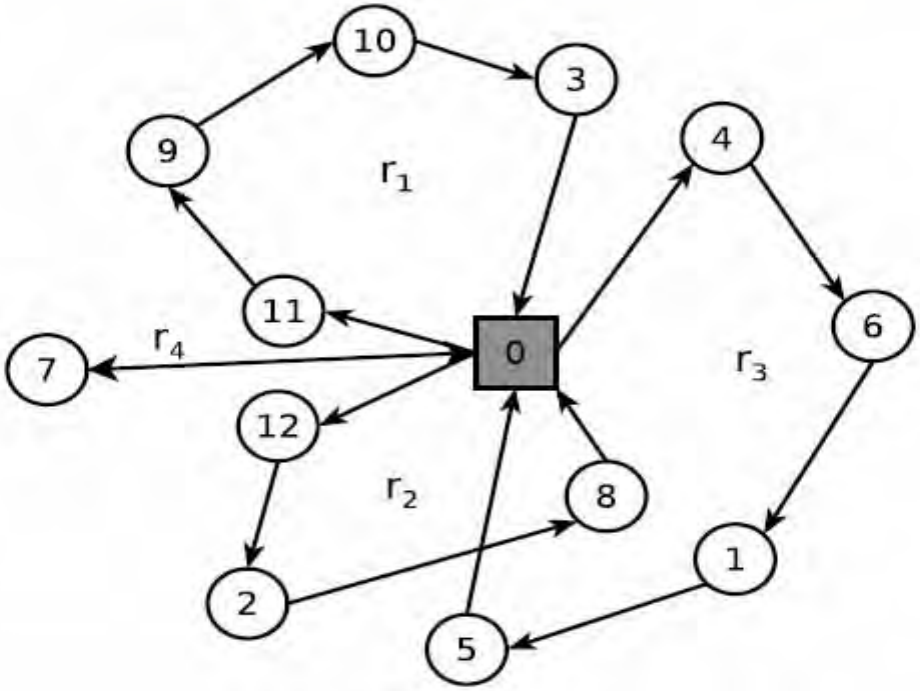


Passo a passo de como construir uma solução inicial para este problema



Passo a passo de como construir uma solução inicial para este problema

Algoritmo Proposto - Construção de soluções iniciais



Além disso, é possível usar alguns módulos mais sofisticados para melhorar a qualidade de uma solução, mas isso vamos deixar de lado agora, quem tiver interesse em como isso é feito, entre em contato conosco.

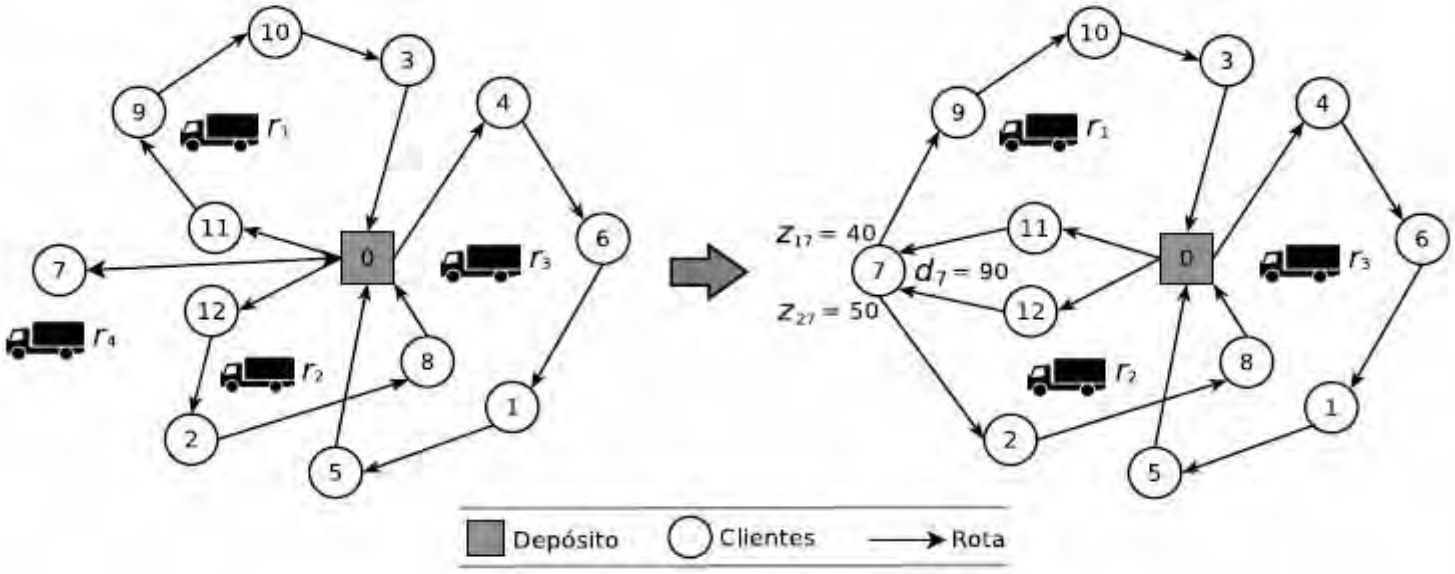
Algoritmo Proposto - Métodos Auxiliares

Algoritmo 1: ReinsertSingleClient

```
1
2  Procedimento ReinsertSingleClient(s)
3  aplicarSplit ← falso;
4
5  f'' ← f(s);
6  para r ← 1, . . . , |s| faça
7      se |r| = 1 então
8          i ← único cliente em r;
9          s' ← SplitReinsertion(s, i, d_i, r);
10         se f(s') < f'' então
11             s'' ← s';
12             f'' ← f(s');
13             aplicarSplit ← verdadeiro;
14         fim se
15     fim se
16 fim para
17 se aplicarSplit = verdadeiro então
18     s ← s'';
19     Vá para a linha 1;
20 fim se
21 retorne s;
22 fim
```

Estes procedimentos adicionais, podem por exemplo, obter nova solução usando <u>menos veículos</u>, economia!!!

Algoritmo Proposto - Construção de soluções iniciais



Agora vamos para um módulo FUNDAMENTAL de uma Meta-heurística: *A fase de Refinamento de uma solução (Busca Local)*

Algoritmo Proposto

```
1 Procedimento ILS
2 s0 ← GeraSolucaoInicial();
3 s* ← BuscaLocal(s0);
4 enquanto critério de parada não satisfeito faça
5     s' ← Perturba(s*, historico);
6     s'* ← BuscaLocal(s');
7     s* ← CriterioAceitacao(s*, s'*, historico);
8 fim enquanto
9 fim ILS
```

Para tentar melhorar uma solução inicial, vamos usar uma variante de uma Meta-heurística conhecido como VNS (*Variable Neighborhood Search*)

Algoritmo Proposto - Busca Local

- Baseia-se no procedimento *Variable Neighborhood Descent (VND)* com ordem aleatória da estrutura de vizinhança (RVND)
- Cada estrutura de vizinhança é examinada exaustivamente

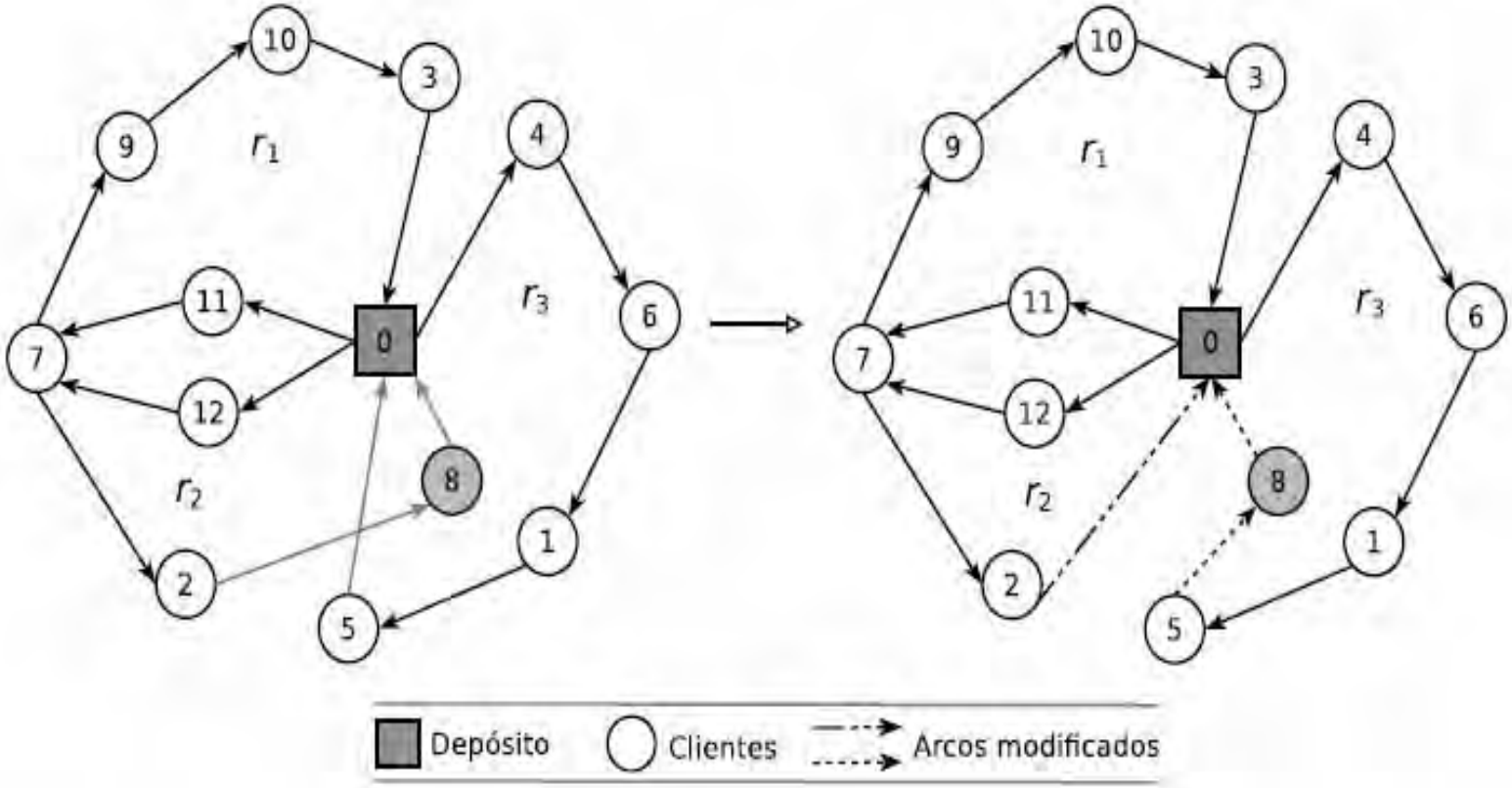
Na Fase de Busca Local (Refinamento) em Problemas de Roteamento de Veículos (PRV), costumamos usar com sucesso, vizinhanças clássicas baseadas em conceitos de permutar 2 clientes de 2 rotas; tirar um cliente de uma rota e colocar noutra rota; mudar a posição de um cliente numa rota, etc.

Algoritmo Proposto - Busca Local

- Vizinhanças Inter-rotas:
- PRV:
  - ***Shift(1,0)***.
  - ***Swap(1,1)***.
  - ***Shift(2,0)***.
  - ***Swap(2,1)***.
  - ***Swap(2,2)***.
  - ***Cross***.
- PRVEF:
  - ***Swap*(1,1)***.
  - ***Swap*(2,1)***.
  - ***RouteAddition***.
  - ***k-Split***.

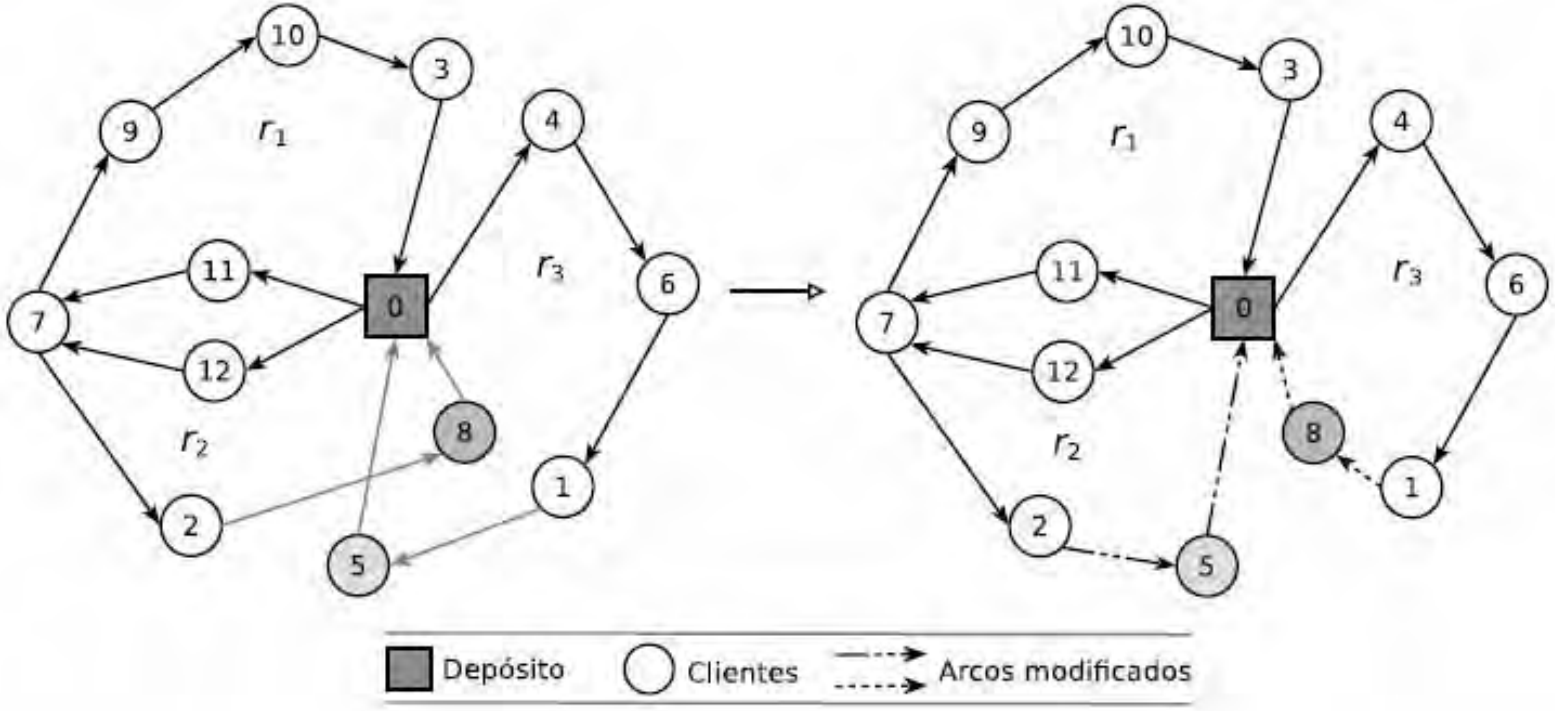
Algoritmo Proposto - Busca Local

- ***Shift(1,0)***.



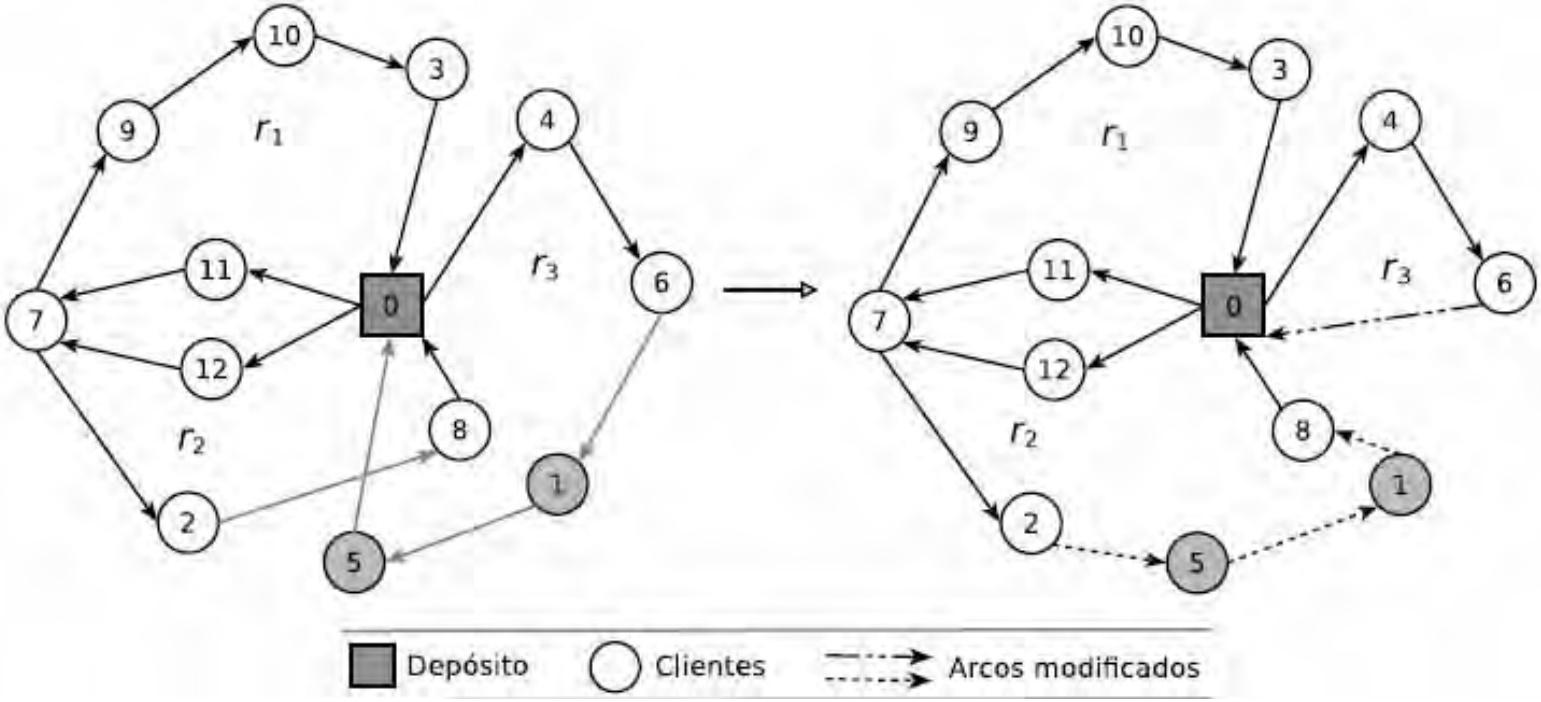
Algoritmo Proposto - Busca Local

- ***Swap(1,1)***.



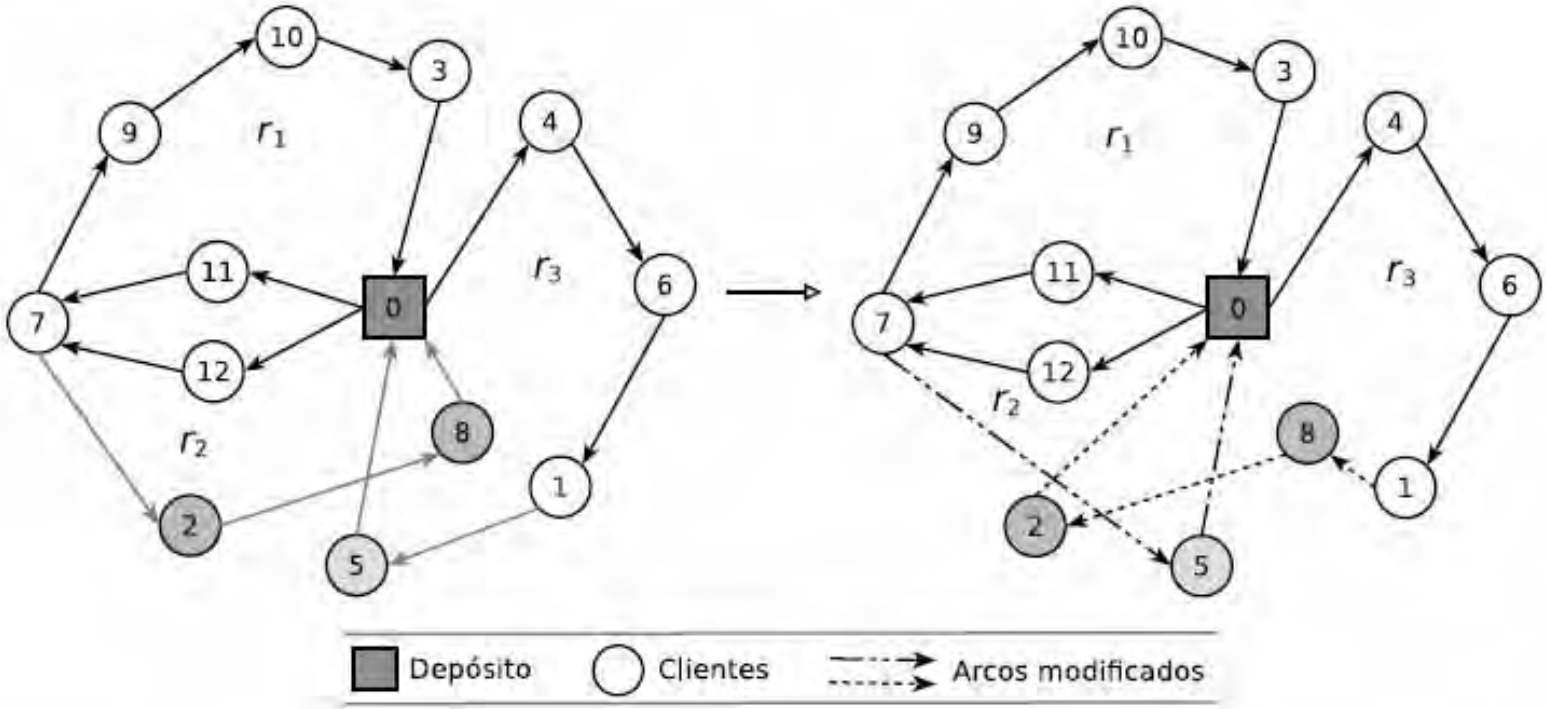
Algoritmo Proposto - Busca Local

- ***Shift(2,0)***.



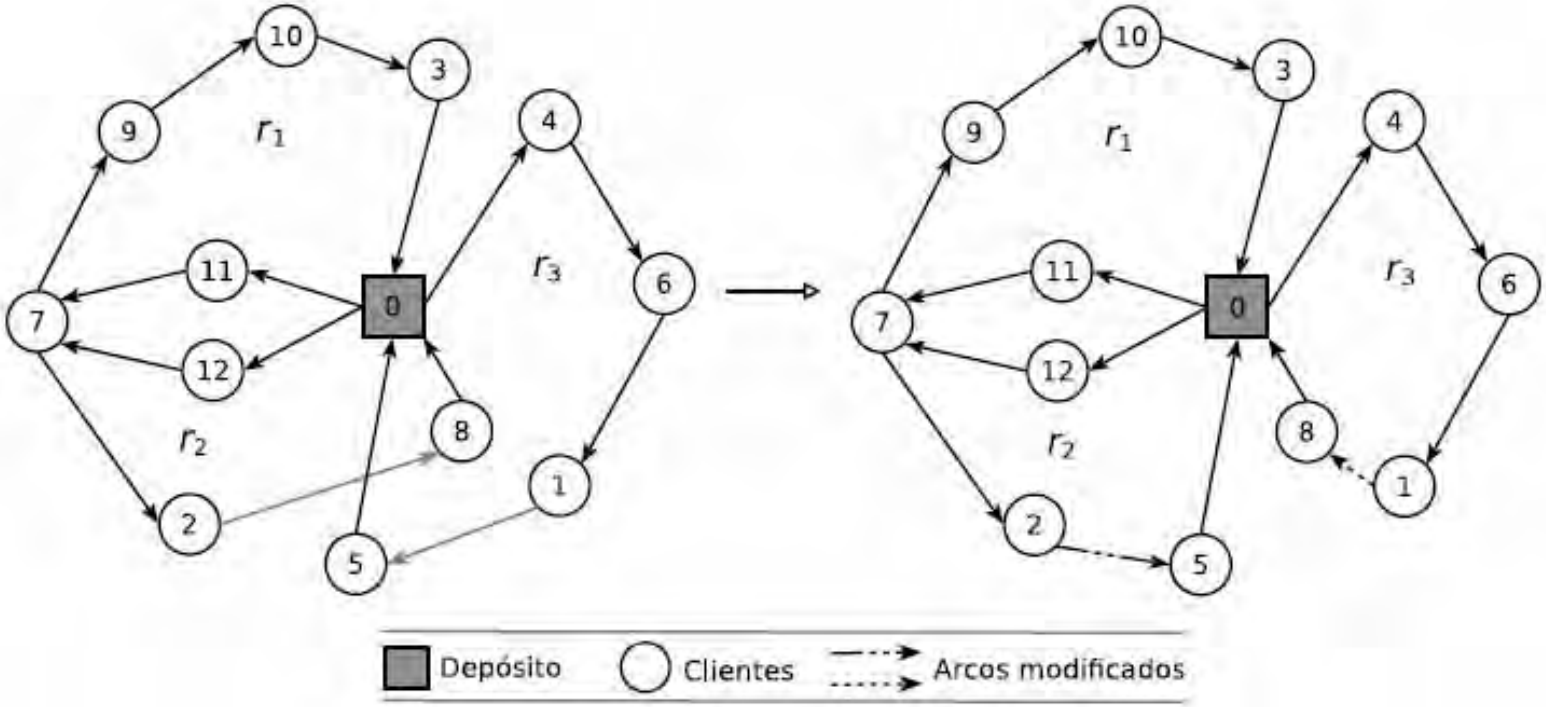
Algoritmo Proposto - Busca Local

- ***Swap(2,1)***.



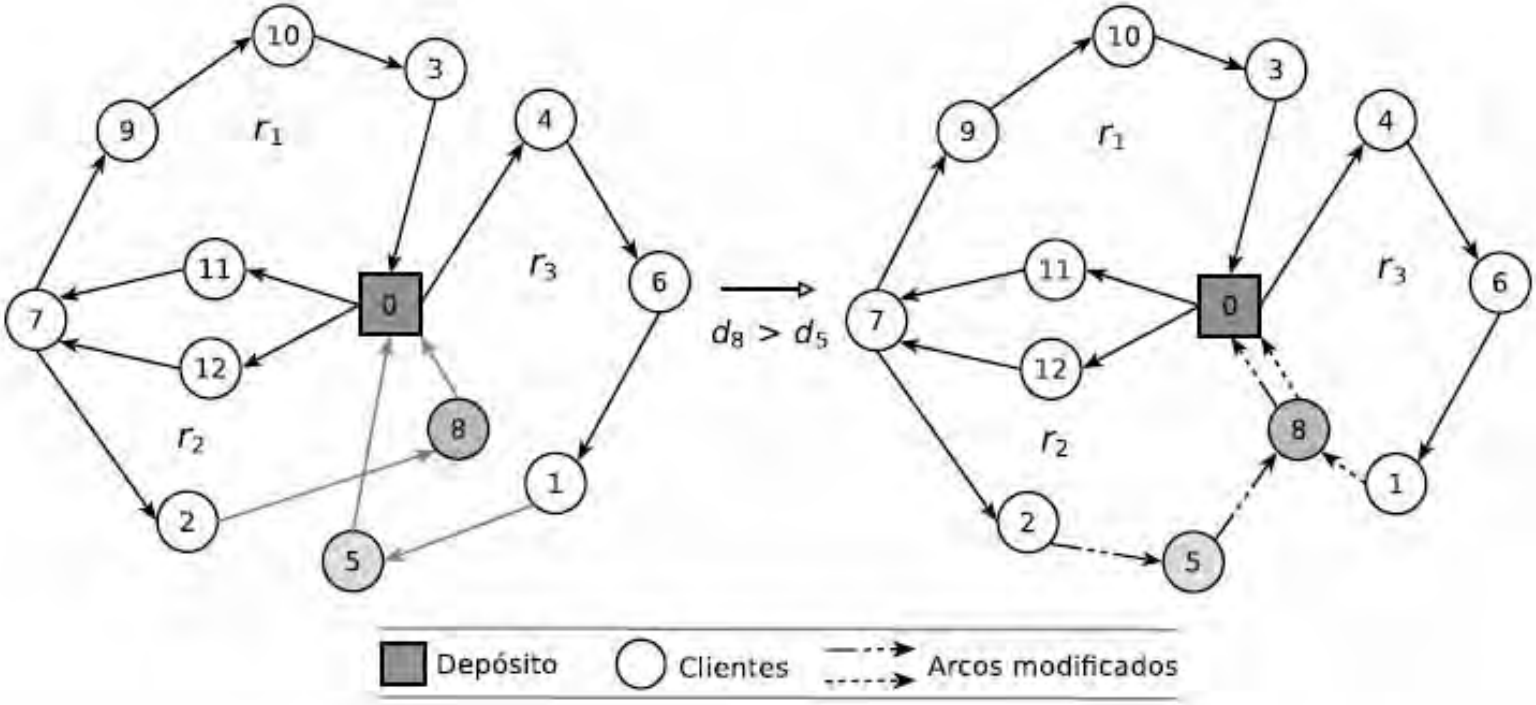
Algoritmo Proposto - Busca Local

- ***Cross***.



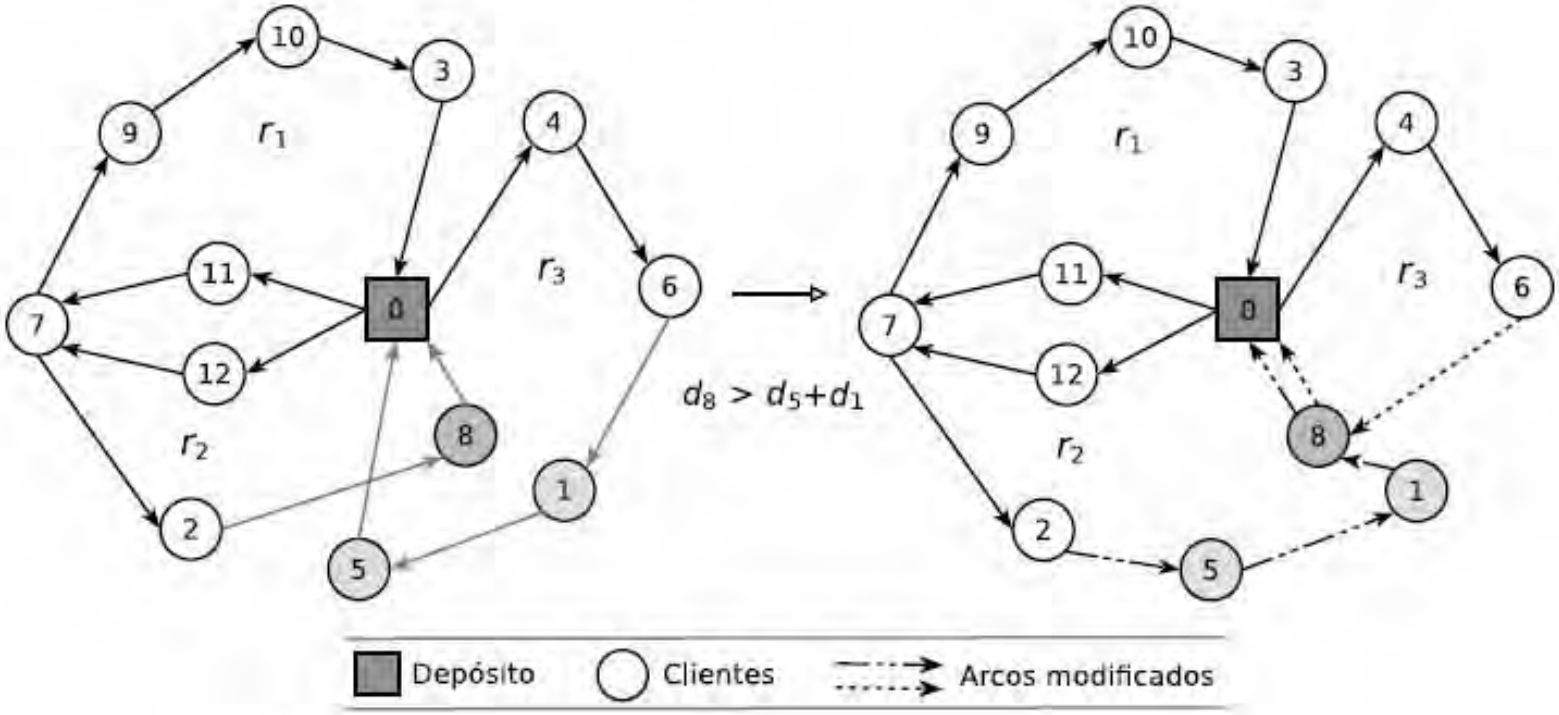
Algoritmo Proposto - Busca Local

- ***Swap*(1,1)***.



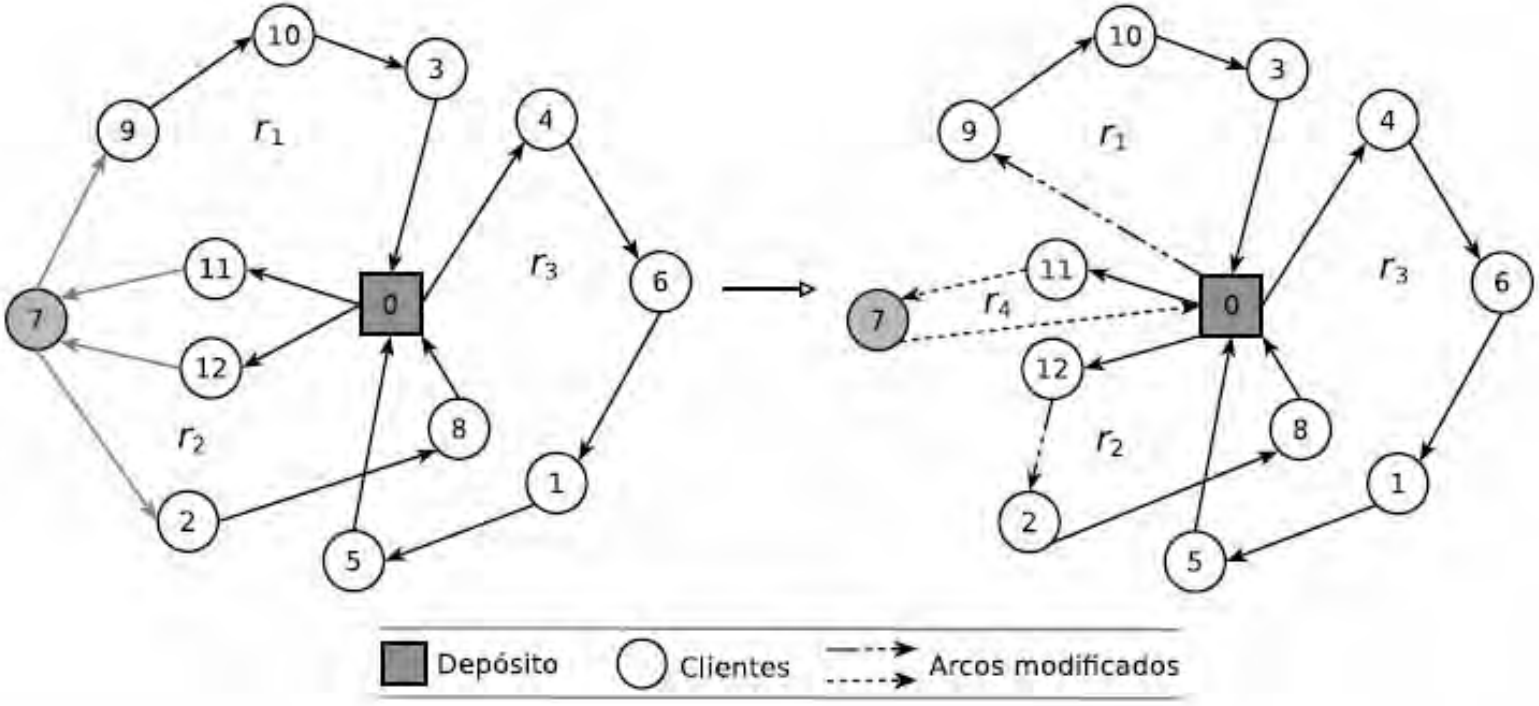
Algoritmo Proposto - Busca Local

- ***Swap*(2,1)***.



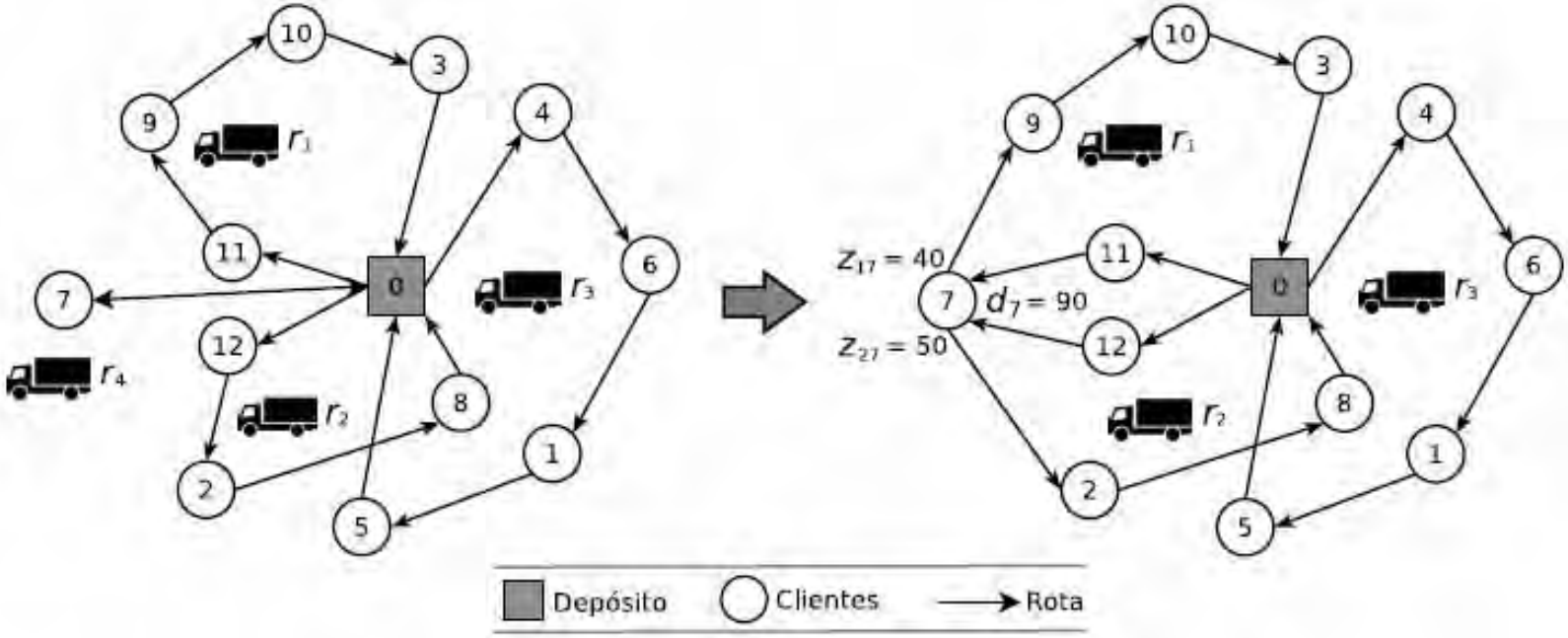
Algoritmo Proposto - Busca Local

- ***RouteAddition***.



Algoritmo Proposto - Busca Local

- ***k-Split***.

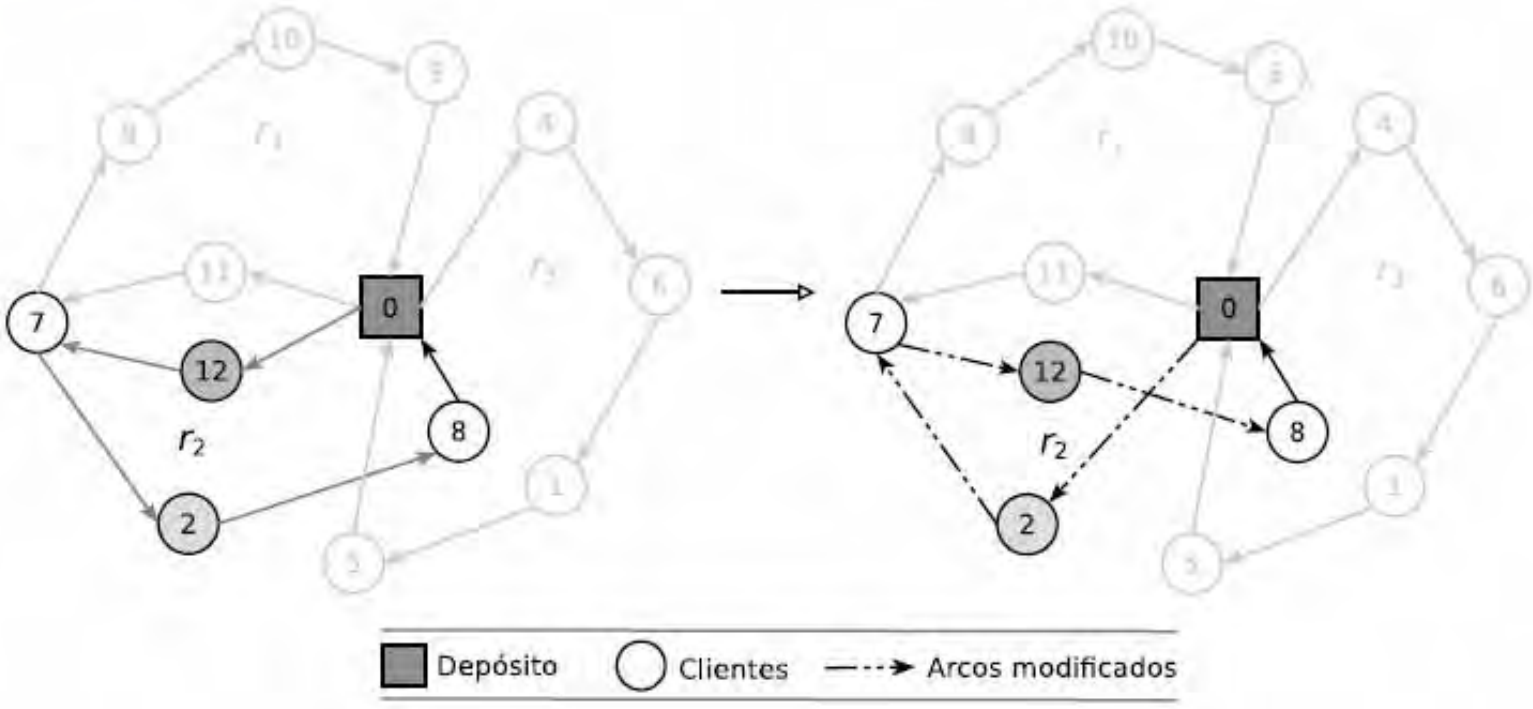


Algoritmo Proposto - Busca Local

- Vizinhanças Intra-rota:
  - ***Exchange***.
  - **2**-*opt*.
  - **Reinserção**.
  - **Or**-*opt2*.
  - **Or**-*opt3*.

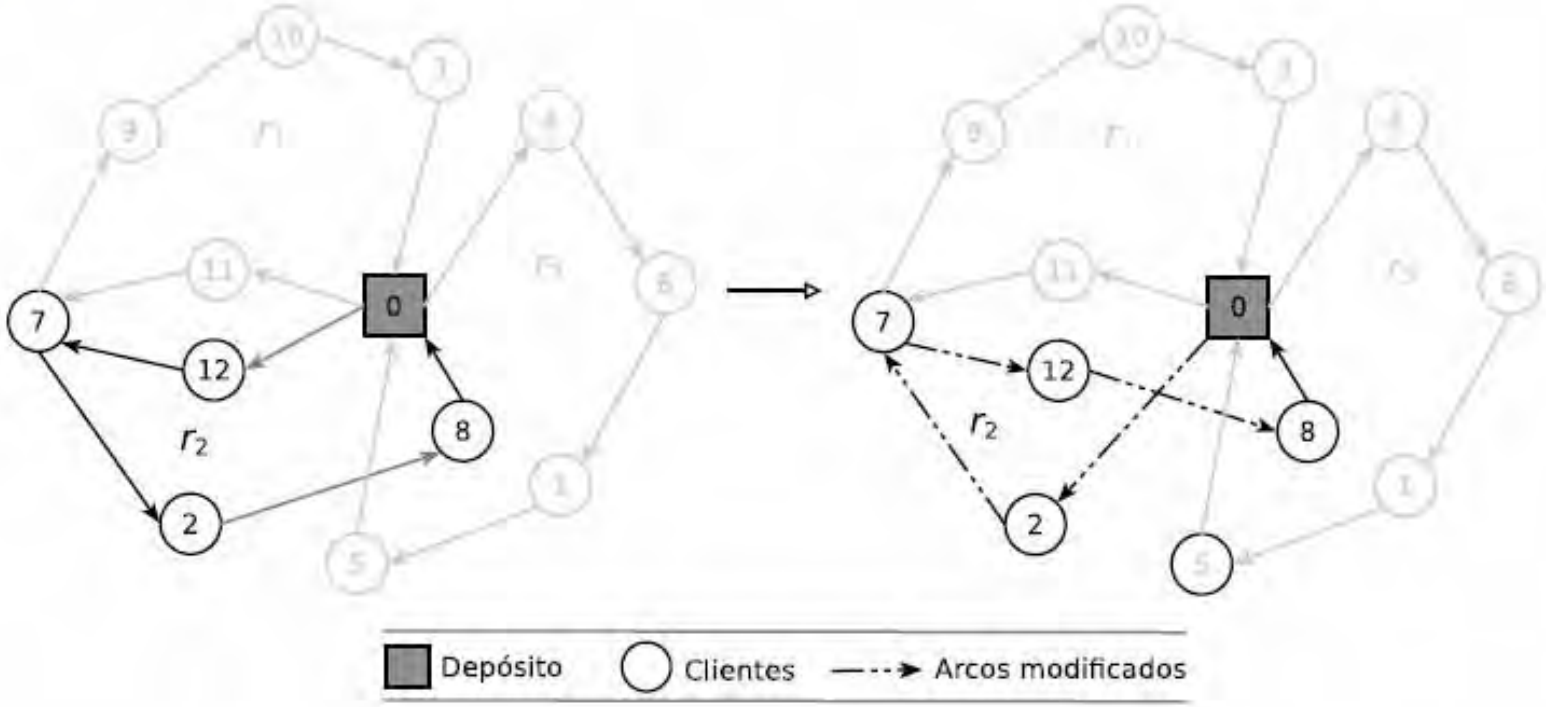
Algoritmo Proposto - Busca Local

- ***Exchange***.



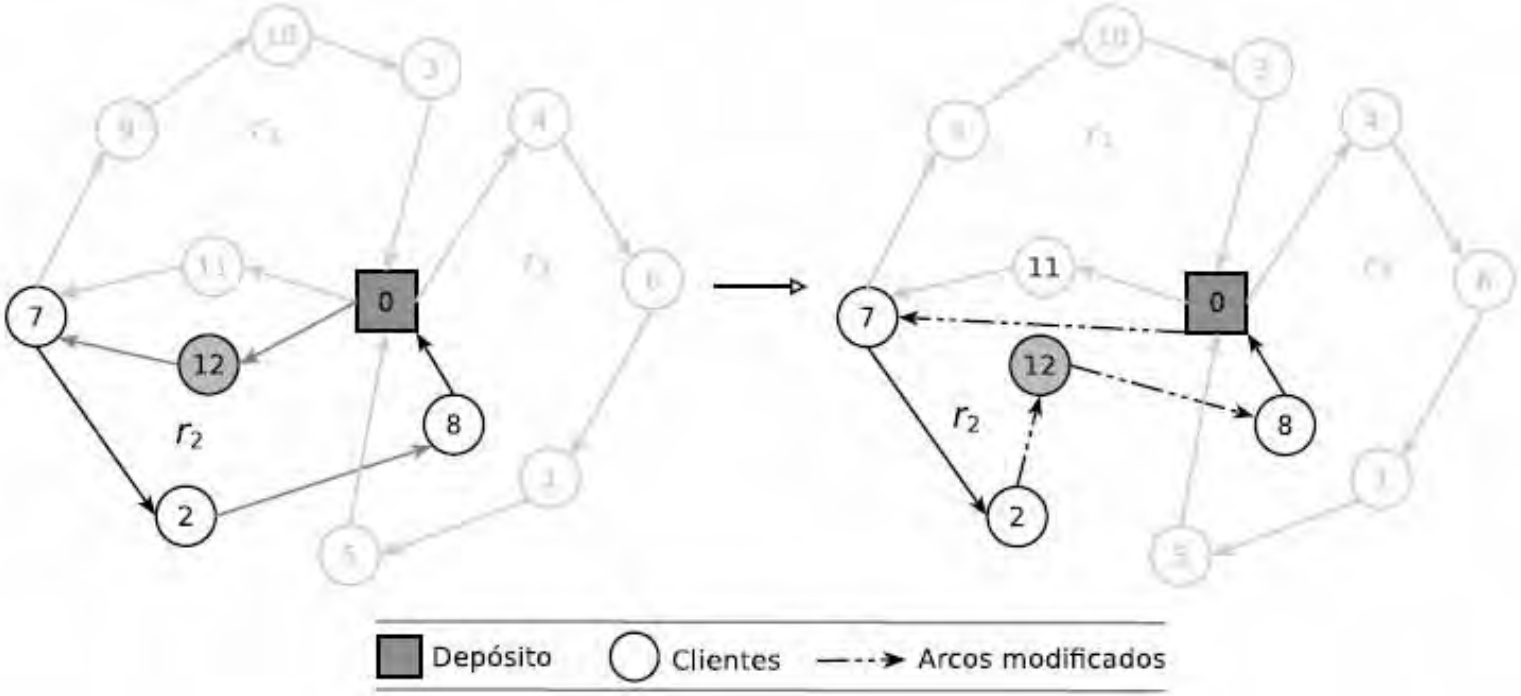
Algoritmo Proposto - Busca Local

- **2**-*opt*.



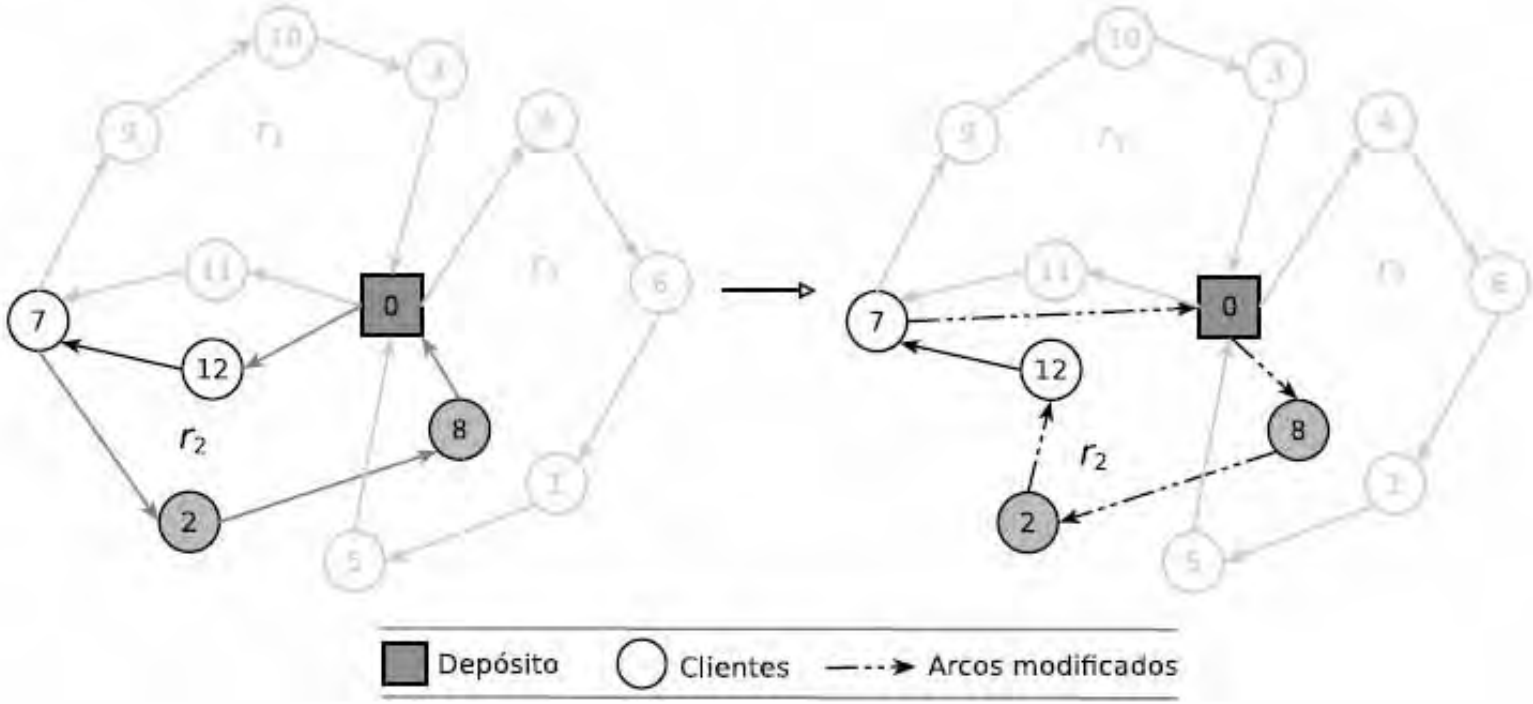
Algoritmo Proposto - Busca Local

- **Reinserção**.



Algoritmo Proposto - Busca Local

- **Or**-*opt2*.



Algoritmo Proposto - Busca Local

```
1  Procedimento RVND(s)
2  Inicializar lista de vizinhança LV;
3  enquanto LV ≠ ∅ faça
4        Selecione aleatoriamente a vizinhança N^(η) ∈ LV;
5        Encontre o melhor vizinho s' de s ∈ N^(η);
6        se f(s') < f(s) então
7              s ← s';
8              s ← BuscaIntraRota(s);
9              f(s) ← f(s');
10             Atualiza LV;
11       senão
12             Remova N^(η) de LV;
13       fim se
14 fim enquanto
15 retorne s;
16 fim RVND
```

ETAPA DE PERTURBAÇÃO NA Meta-heurística ILS

Algoritmo Proposto

```
1 Procedimento ILS
2 s0 ← GeraSolucaoInicial();
3 s* ← BuscaLocal(s0);
4 enquanto critério de parada não satisfeito faça
5     s' ← Perturba(s*, historico);
6     s'* ← BuscaLocal(s');
7     s* ← CriterioAceitacao(s*, s'*, historico);
8 fim enquanto
9 fim ILS
```

Algoritmo Proposto - Perturbação

- Mecanismos de perturbação:
  - ***PRouteAdition***.
  - ***Multiple-Split*** – O número de vezes que o movimento será aplicado é selecionado do intervalo [5, 6, 7].
  - ***Multiple-Swap*(1,1)*** – O número de vezes que o movimento será aplicado é selecionado do intervalo [4, 5, 6].

Algoritmo Proposto - SPLITILS

```
1  Procedimento SplitILS(I_Max, I_ILS)
2  f* ← ∞;
3  para i ← 1, . . . , I_Max faça
4        s ← GSolInicial();
5        s' ← s;
6        iterILS ← 0;
7        enquanto iterILS < I_ILS faça
8              s ← RVND(s);
9              se PRVEF-FL então
10                   s ← EmptyRoutes(s);
11             fim se
12             se f(s) < f(s') então
13                   s' ← s;
14                   iterILS ← 0;
15             fim se
16             s ← Perturba(s');
17             iterILS ← iterILS + 1;
18       fim enquanto
19       se f(s') < f* então
20             s* ← s';
21             f* ← f(s');
22       fim se
23 fim para
24 retorne s*;
25 fim
```

Resultados

- O algoritmo foi implementado na linguagem C/C++

- Ambiente Computacional:
  - PC Intel® Core™ i7 com 2.93 GHz
  - 8.0 GB de memória RAM
  - Sistema operacional GNU/Linux Ubuntu 10.04 (*kernel* 2.6.32-25)
  - Uma única *thread* foi utilizada

Resultados - Parametrização

- Parametrização:
  - Número de iterações *multi-start* (I_{max}): 10
  - Número de perturbações (I_{ILS}): $\min\{K_{min} \times n, 5000\}$;
  - Construtivo: $\gamma = [0.1, 0.2, \ldots, 1)$
- O algoritmo foi executado 20 vezes para cada problema-teste

Resultados - Instâncias

- Testes em quatro conjuntos de instâncias disponíveis na literatura:
  1. Archetti *et. al.* (2008): 42 problemas-teste (50 a 199 clientes)
  2. Belenguer *et. al.* (2000): 25 problemas-teste (21 a 100 clientes)
  3. Campos *et. al.* (2008): 49 problemas-teste (50 a 199 clientes)
  4. Chen *et. al.* (2007): 21 problemas-teste (8 a 288 clientes)

- Demandas selecionadas aleatoriamente no intervalos ($Q = 160$):
  - D1: $[\lceil 0.01Q \rceil, \lfloor 0.1Q \rfloor]$
  - D2: $[\lceil 0.1Q \rceil, \lfloor 0.3Q \rfloor]$
  - D3: $[\lceil 0.1Q \rceil, \lfloor 0.5Q \rfloor]$
  - D4: $[\lceil 0.1Q \rceil, \lfloor 0.9Q \rfloor]$
  - D5: $[\lceil 0.3Q \rceil, \lfloor 0.7Q \rfloor]$
  - D6: $[\lceil 0.7Q \rceil, \lfloor 0.9Q \rfloor]$.

Resultados - Instâncias de Belenguer et. al. (2000)

- Custos Arredondados para o inteiro mais próximo;

			Aleman et al.[c]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	375,00	*375[a]	**375**	4,19	**375**	**375,00**	0,00	0,00	0,13	0,01	375
eil23	569,00	*569[a]	570	3,42	**569**	**569,00**	0,00	0,00	0,09	0,00	569
eil30	510,00	*510[a]	**510**	14,47	**510**	**510,00**	0,00	0,00	0,30	0,00	510
eil33	834,70	*835[a]	851	14,03	**835**	**835,00**	0,04	0,00	0,39	0,03	835
eil51	517,80	*521[a]	**521**	54,91	**521**	521,55	0,72	0,11	1,63	0,14	521
eilA76	807,60	832[a]	847	83,28	**818**	**820,45**	1,59	-1,39	25,68	0,35	818
eilB76	981,40	1023[a]	1055	79,00	**1002**	**1005,80**	2,49	-1,68	38,05	0,33	1002
eilC76	717,80	735[a]	746	148,20	**733**	**733,55**	2,19	-0,20	15,17	1,78	732
eilD76	666,10	683[a]	695	140,83	**681**	**683,00**	2,54	0,00	11,02	2,87	681
eilA101	799,80	817[a]	843	319,33	**815**	**815,85**	2,01	-0,14	32,70	8,57	814
eilB101	1040,60	1077[a]	1122	185,84	**1061**	**1065,40**	2,38	-1,08	75,43	8,34	1061

Resultados - Instâncias de Belenguer et. al. (2000)

			Aleman et al.[c]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
S51D1	454,40	458[a]	466	40,53	**458**	**458,00**	0,79	0,00	1,21	0,09	458
S51D2	694,20	725[b]	**725**	28,34	**703**	**704,65**	1,51	-2,81	8,32	0,06	703
S51D3	930,70	972[a]	994	14,70	**943**	**944,20**	1,45	-2,86	13,58	0,03	943
S51D4	1539,00	1672[b]	**1672**	16,53	**1552**	**1555,55**	1,08	-6,96	47,34	0,01	1551
S51D5	1313,40	1385[b]	**1385**	13,94	**1328**	**1329,15**	1,20	-4,03	33,46	0,02	1328
S51D6	2141,70	2211[b]	**2211**	16,83	**2163**	**2165,70**	1,12	-2,05	65,68	0,01	2162
S76D1	586,60	594[a]	600	476,27	**592**	**592,45**	1,00	-0,26	4,75	1,33	592
S76D2	1061,10	1138[b]	**1138**	46,94	**1081**	**1083,35**	2,10	-4,80	59,20	0,03	1080
S76D3	1395,90	1474[a]	1485	53,34	**1419**	**1422,05**	1,87	-3,52	68,07	0,03	1419
S76D4	2046,10	2160[b]	**2160**	51,84	**2071**	**2074,30**	1,38	-3,97	148,48	0,01	2071
S101D1	704,90	716[a]	740	2125,58	**716**	718,40	1,92	0,34	14,17	13,00	716
S101D2	1337,10	1393[a]	1426	217,91	**1364**	**1370,95**	2,53	-1,58	116,33	0,45	1364
S101D3	1832,20	1974[b]	1974	146,61	**1859**	**1868,75**	1,99	-5,33	233,36	0,01	1859
S101D5	2737,10	2915[a]	2970	104,05	**2772**	**2779,65**	1,55	-4,64	579,68	0,06	2772
Média				176,04			1,42	-1,87	63,77	1,50	

∗ - Solução ótima
a - Belenguer et al. [5]
b - Aleman et al. [2]
c - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 73,68

Resultados - Instâncias de Belenguer et. al. (2000)

			Aleman et al.[d]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
eil22	-	375, 28[b]	**375,28**	4,19	**375,28**	**375,28**	-	0,00	0,14	0,00	375,28
eil23	525,65	569, 75[b]	**569,75**	3,42	**568,56**	**568,56**	8,16	-0,21	0,12	0,00	568,56
eil30	-	512, 72[b]	**512,72**	14,47	**512,72**	**512,72**	-	0,00	0,32	0,00	512,72
eil33	-	853, 10[b]	**853,10**	14,03	**837,06**	**837,06**	-	-1,88	0,45	0,00	837,06
eil51	518,26	524, 61[b]	**524,61**	54,91	**524,61**	**524,61**	1,23	0,00	1,63	0,24	524,61
eilA76	809,67	829, 58[a]	851,24	83,28	**823,89**	**825,22**	1,92	-0,53	27,25	3,19	823,89
eilB76	985,42	1023, 23[a]	1059,57	79,00	**1009,04**	**1011,19**	2,62	-1,18	44,98	0,15	1009,04
eilC76	723,55	753, 29[b]	**753,29**	148,20	**738,67**	**739,83**	2,25	-1,79	15,68	0,01	738,67
eilD76	672,54	699, 35[b]	**699,35**	140,83	**687,60**	**688,37**	2,35	-1,57	9,92	0,03	687,60
eilA101	803,62	851, 39[a]	852,74	319,33	**826,14**	**826,26**	2,82	-2,95	36,59	0,01	826,14
eilB101	1055,40	1118, 93[a]	1139,27	185,84	**1076,26**	**1078,58**	2,20	-3,61	101,26	0,06	1076,26

Resultados - Instâncias de Belenguer et. al. (2000)

			Aleman et al.[d]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
S51D1	457,10	464, 78[a]	471,92	40,53	**459,50**	**459,50**	0,53	-1,14	1,07	0,02	459,50
S51D2	700,40	719, 82[a]	731,01	28,34	**708,42**	**709,54**	1,30	-1,43	9,98	0,15	708,42
S51D3	938,50	955, 56[a]	1001,22	14,70	**948,01**	**949,96**	1,22	-0,59	14,15	0,34	947,97
S51D4	1549,70	1614, 36[a]	1680,66	16,53	**1561,01**	**1563,25**	0,87	-3,17	59,96	0,01	1560,92
S51D5	1318,90	1343, 07[a]	1389,40	13,94	**1333,67**	**1333,85**	1,13	-0,69	32,41	0,21	1333,67
S51D6	2154,70	2179, 65[a]	2218,23	16,83	**2169,10**	**2174,71**	0,93	-0,23	83,79	30,90	2169,10
S76D1	592,60	606, 47[b]	**606,47**	476,27	**598,94**	**598,98**	1,08	-1,24	4,54	0,25	598,94
S76D2	1071,30	1129, 06[a]	1143,36	46,94	**1087,99**	**1089,69**	1,72	-3,49	74,51	0,08	1087,99
S76D3	1407,54	1442, 08[a]	1490,08	53,34	**1427,81**	**1429,01**	1,53	-0,91	88,72	1,76	1427,81
S76D4	2059,80	2100, 15[a]	2173,61	51,84	**2079,76**	**2080,76**	1,02	-0,92	173,55	2,69	2079,76
S101D1	716,80	749, 19[b]	**749,19**	2125,58	**726,59**	**728,44**	1,62	-2,77	14,16	0,01	726,59
S101D2	1358,90	1417, 87[a]	1443,44	217,91	**1383,35**	**1386,45**	2,03	-2,22	129,94	0,06	1378,19
S101D3	1853,10	1906, 28[a]	1988,78	146,61	**1876,97**	**1881,26**	1,52	-1,31	277,62	1,84	1876,48
S101D5	2767,60	2873, 08[c]	2984,48	104,05	**2792,01**	**2795,73**	1,02	-2,69	696,64	0,38	2789,54
Média				176,04			1,87	-1,46	75,97	1,69	

a - Archetti et al. [3]
b - Aleman et al. [2]
c - Wilck IV e Cavalier [12], Xeon 2,49 GHz, uma única execução
d - Pentium 4, 2,8 GHz, uma única execução, Avg. cTime (s): 73,68

Resultados - Instâncias de Campos et. al. (2008)

		Campos et al.[c]		Aleman et al.[d]		SplitILS					
Problem	BKS	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	524,61[a]	**524,61**	49,70	**524,61**	54,91	**524,61**	**524,61**	0,00	1,87	0,22	524,61
p02-75	829,01[a]	**829,01**	166,50	851,24	83,28	**823,89**	**824,77**	-0,51	30,84	6,72	823,89
p03-100	829,45[a]	**829,45**	276,10	852,74	319,33	**826,14**	**826,39**	-0,37	40,81	3,20	826,14
p04-150	1045,22[a]	**1045,22**	527,10	1074,11	1361,16	**1024,59**	**1026,60**	-1,78	251,66	1,06	1023,66
p05-199	1324,73[a]	**1324,73**	588,30	1368,67	3284,64	**1289,40**	**1296,37**	-2,14	1594,46	10,70	1286,08
p06-120	1042,12[a]	**1042,12**	270,30	1201,83	3414,41	**1037,88**	1043,41	0,12	90,06	20,49	1037,88
p07-100	819,56[a]	**819,56**	192,40	824,78	126,08	**819,56**	**819,56**	0,00	27,99	0,13	819,56
p01-50D1	460,79[a]	**460,79**	51,80	471,92	33,70	**460,79**	**460,79**	0,00	1,16	0,02	460,79
p02-75D1	596,99[a]	**596,99**	144,00	597,46	303,77	**596,25**	**596,25**	-0,12	5,00	0,61	596,25
p03-100D1	726,81[a]	**726,81**	272,10	745,35	2194,23	**726,81**	730,01	0,44	17,72	13,70	726,81
p04-150D1	871,26[a]	**871,26**	743,30	891,98	3461,44	**866,31**	**866,31**	-0,57	119,63	2,77	866,31
p05-199D1	1023,14[a]	**1023,14**	1874,80	1073,55	15505,22	**1017,30**	**1018,40**	-0,46	438,21	27,99	1017,28
p06-120D1	976,57[a]	**976,57**	370,90	1087,80	3952,67	**975,96**	**976,42**	-0,02	46,16	8,15	975,96
p07-100D1	636,00[a]	**636,00**	166,50	673,54	1207,42	**632,63**	**633,11**	-0,45	14,38	2,75	632,63
p01-50D2	741,06[a]	**741,06**	66,40	766,19	19,77	**741,06**	741,26	0,03	9,87	4,12	741,06
p02-75D2	1071,87[a]	**1071,87**	143,80	1099,47	73,05	**1064,49**	**1066,87**	-0,47	53,42	3,14	1064,49
p03-100D2	1397,50[a]	**1397,50**	305,10	1425,90	190,53	**1376,09**	**1380,28**	-1,23	182,16	1,85	1374,19
p04-150D2	1937,20[a]	**1937,20**	326,60	1978,01	878,55	**1861,63**	**1866,48**	-3,65	1055,54	0,29	1861,53
p05-199D2	2433,17[a]	**2433,17**	32,10	2464,65	1457,16	**2307,82**	**2313,37**	-4,92	2440,32	0,16	2306,80
p06-120D2	2742,60[a]	**2742,60**	380,80	2806,92	558,56	**2703,75**	**2708,51**	-1,24	762,81	6,43	2703,75
p07-100D2	1418,81[a]	**1418,81**	206,30	1428,27	123,00	**1413,85**	**1413,91**	-0,35	130,70	1,21	1413,85

Resultados - Instâncias de Campos et. al. (2008)

		Campos et al.[c]		Aleman et al.[d]		SplitILS					
Problem	BKS	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50D3	997, 83[a]	**997,83**	87,10	1039,89	18,16	**982,77**	**983,70**	-1,42	18,44	0,40	982,77
p02-75D3	1463, 60[a]	**1463,60**	126,80	1478,67	67,80	**1393,11**	**1393,11**	-4,82	101,77	0,04	1393,11
p03-100D3	1908, 02[a]	**1908,02**	225,20	1956,13	154,47	**1823,17**	**1827,47**	-4,22	326,55	0,06	1822,36
p04-150D3	2649, 97[a]	**2649,97**	21,30	2671,62	625,83	**2528,51**	**2531,79**	-4,46	1514,55	0,08	2525,51
p05-199D3	3291, 96[a]	**3291,96**	31,20	3411,38	2173,84	**3153,01**	**3163,89**	-3,89	3895,07	0,32	3153,01
p06-120D3	3979, 67[a]	**3979,67**	329,00	4026,53	358,56	**3907,27**	**3910,03**	-1,75	1543,98	26,71	3907,27
p07-100D3	1995, 34[a]	**1995,34**	266,50	2007,11	107,47	**1967,41**	**1967,93**	-1,37	260,65	0,41	1967,41
p01-50D4	1522, 43[b]	1554,38	92,60	**1522,43**	16,36	**1456,00**	**1456,87**	-4,31	46,74	0,01	1456,00
p02-75D4	2182, 34[a]	**2182,34**	119,90	2200,51	71,11	**2081,38**	**2084,62**	-4,48	219,74	0,02	2081,37
p03-100D4	2865, 86[b]	2894,21	177,90	**2865,86**	126,52	**2751,13**	**2754,52**	-3,89	629,59	0,03	2749,05
p04-150D4	4062, 88[a]	**4062,88**	50,40	4165,18	671,36	**3988,06**	**3997,49**	-1,61	1986,49	2,41	3988,06
p05-199D4	5074, 57[a]	**5074,57**	50,70	5184,57	3650,59	**4844,58**	**4855,82**	-4,31	3806,84	0,25	4842,97
p06-120D4	6357, 33[a]	**6357,33**	20,60	6364,87	458,91	**6201,66**	**6215,87**	-2,23	1975,24	0,57	6196,48
p07-100D4	3156, 31[b]	3166,31	272,70	**3156,31**	96,98	**3087,75**	**3088,96**	-2,13	435,09	0,04	3087,75
p01-50D5	1532, 19[a]	**1532,19**	92,40	1540,39	15,33	**1467,47**	**1467,47**	-4,22	48,93	0,01	1467,47
p02-75D5	2228, 90[a]	**2228,90**	11,10	2238,98	80,30	**2112,19**	**2113,38**	-5,18	267,72	0,02	2110,20
p03-100D5	2941, 64[b]	2986,33	17,00	**2941,64**	103,94	**2813,82**	**2817,05**	-4,24	737,35	0,04	2813,35
p04-150D5	4165, 18[b]	4185,68	23,00	**4165,18**	675,39	**3986,49**	**3996,85**	-4,04	2076,38	0,16	3986,49
p05-199D5	5265, 01[a]	**5265,01**	327,30	5363,65	3026,22	**5061,25**	**5070,77**	-3,69	4570,46	0,33	5061,25
p06-120D5	6481, 09[a]	**6481,09**	20,50	6545,50	469,17	**6372,58**	**6375,64**	-1,63	2289,79	1,93	6370,97
p07-100D5	3225, 63[b]	3248,76	16,00	**3225,63**	110,05	**3125,47**	**3126,22**	-3,08	619,62	0,03	3125,47

Resultados - Instâncias de Campos et. al. (2008)

		Campos et al.[c]		Aleman et al.[d]		SplitILS					
Problem	BKS	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50D6	2215, 34[b]	2312,48	5,80	**2215,34**	18,70	**2154,21**	**2154,51**	-2,75	83,85	0,01	2154,21
p02-75D6	3304, 24[b]	3387,86	10,50	**3304,24**	58,05	**3179,20**	**3181,30**	-3,72	441,77	0,01	3179,08
p03-100D6	4429, 21[b]	4576,13	38,30	**4429,21**	94,98	**4294,12**	**4298,50**	-2,95	731,49	0,03	4294,12
p04-150D6	6479, 46[a]	**6479,46**	30,50	6482,11	584,84	**6231,01**	**6233,76**	-3,79	1660,06	0,09	6230,49
p05-199D6	8323, 72[a]	**8323,72**	215,00	8329,55	2124,66	**8045,18**	**8047,68**	-3,32	4718,09	0,25	8041,01
p06-120D6	10158, 32[a]	**10158,32**	20,40	10302,16	636,72	**10001,95**	**10005,18**	-1,51	2209,90	0,08	10001,95
p07-100D6	5028, 78[b]	5065,26	13,80	**5028,78**	178,19	**4902,81**	**4907,00**	-2,42	823,19	0,04	4902,37
Média			201,40		1130,15			-2,23	925,59	3,06	

a - Campos et al. [7]
b - Aleman et al. [2]
c - Pentium 4, 2,4 GHz; uma única execução; Avg. cTime (s): 65,91
d - Pentium 4, 2,8 GHz; uma única execução; Avg. cTime (s): 473,04

Resultados - Instâncias de Archetti et. al. (2008)

			SplitILS						
Problem	LB[a]	BKS[a]	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50	-	-	524,61	524,61	-	-	1,83	-	524,61
p02-75	-	-	823,89	824,39	-	-	30,31	-	823,89
p03-100	-	-	826,14	826,45	-	-	42,16	-	826,14
p04-150	-	-	1023,87	1026,48	-	-	243,93	-	1023,66
p05-199	-	-	1285,79	1292,79	-	-	1672,72	-	1285,25
p11-120	-	-	1037,88	1038,68	-	-	85,14	-	1037,88
p01-50D1	457,17	459,98	**459,50**	**459,50**	0,51	-0,10	1,21	0,09	459,50
p02-75D1	604,89	637,36	**617,85**	**620,19**	2,53	-2,69	5,72	0,02	617,85
p03-100D1	742,97	794,21	**760,00**	**760,70**	2,39	-4,22	22,00	0,03	760,00
p04-150D1	896,03	1065,94	**921,47**	**923,74**	3,09	-13,34	164,58	0,03	921,44
p05-199D1	1042,37	1188,45	**1074,18**	**1080,65**	3,67	-9,07	629,08	0,04	1074,18
p11-120D1	1023,37	1063,73	**1043,19**	**1043,21**	1,94	-1,93	93,35	8,15	1042,89
p01-50D2	747,25	771,65	**756,71**	**760,52**	1,78	-1,44	14,55	0,14	756,71
p02-75D2	1093,56	1124,70	**1110,43**	**1112,70**	1,75	-1,07	54,11	0,63	1109,62
p03-100D2	1435,23	1492,05	**1458,46**	**1462,37**	1,89	-1,99	200,43	0,48	1458,46
p04-150D2	1986,79	2109,45	**2017,00**	**2021,78**	1,76	-4,16	1156,99	0,13	2016,96
p05-199D2	2423,64	2632,22	**2481,44**	**2487,28**	2,63	-5,51	2846,16	0,14	2478,02
p11-120D2	2879,63	2988,61	**2899,91**	**2905,28**	0,89	-2,79	898,52	0,22	2898,46
p01-50D3	996,93	1011,64	**1005,75**	**1005,93**	0,90	-0,56	21,48	4,19	1005,75
p02-75D3	1483,17	1509,79	**1502,05**	**1503,42**	1,37	-0,42	110,02	2,86	1502,05
p03-100D3	1971,43	2018,09	**1997,76**	**2001,83**	1,54	-0,81	366,31	24,78	1996,76
p04-150D3	2811,64	2956,18	**2849,66**	**2856,41**	1,59	-3,37	1699,36	0,30	2849,66
p05-199D3	3420,17	3548,13	**3472,79**	**3481,37**	1,79	-1,88	3015,92	4,94	3471,71
p11-120D3	4162,99	4308,17	**4219,01**	**4220,59**	1,38	-2,03	2260,39	4,37	4217,59

Resultados - Instâncias de Archetti et. al. (2008)

Problem	LB[a]	BKS[a]	SplitILS						
			Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
p01-50D4	1470,91	1490,61	**1488,27**	**1489,05**	1,23	-0,10	52,71	3,54	1488,06
p02-75D4	2270,44	2372,22	**2301,61**	**2304,89**	1,52	-2,84	283,77	0,12	2301,15
p03-100D4	3043,27	3167,29	**3090,65**	**3094,91**	1,70	-2,29	746,44	0,14	3090,40
p04-150D4	4474,18	4708,11	**4543,18**	**4550,63**	1,71	-3,34	2467,51	0,21	4542,52
p05-199D4	5425,69	5894,69	**5526,28**	**5530,56**	1,93	-6,18	5799,52	0,34	5524,97
p11-120D4	6808,07	7020,87	**6856,11**	**6863,96**	0,82	-2,23	3363,54	0,79	6854,51
p01-50D5	1467,09	1498,30	**1481,71**	**1484,62**	1,19	-0,91	42,90	0,57	1481,71
p02-75D5	2192,25	2235,61	**2219,52**	**2222,58**	1,38	-0,58	261,25	18,55	2219,11
p03-100D5	2945,76	3044,92	**2991,22**	**2991,89**	1,57	-1,74	756,18	0,38	2990,27
p04-150D5	4269,77	4637,69	**4336,80**	**4342,45**	1,70	-6,37	2366,91	0,08	4332,45
p05-199D5	5306,11	5669,69	**5404,44**	**5415,31**	2,06	-4,49	5706,50	0,32	5399,91
p11-120D5	6584,11	6860,65	**6674,97**	**6678,58**	1,43	-2,65	2306,51	0,07	6662,11
p01-50D6	2133,94	2162,58	**2156,14**	**2160,60**	1,25	-0,09	84,35	19,39	2155,80
p02-75D6	3177,69	3259,82	**3223,06**	**3226,79**	1,55	-1,01	377,02	0,01	3223,06
p03-100D6	4316,42	4494,53	**4387,32**	**4389,19**	1,69	-2,34	719,27	0,03	4386,72
p04-150D6	6287,09	6529,63	**6396,68**	**6402,63**	1,84	-1,94	2180,59	0,09	6396,44
p05-199D6	8062,24	8400,74	**8188,47**	**8195,06**	1,65	-2,45	3528,41	0,22	8188,47
p11-120D6	10111,11	10456,19	**10215,90**	**10218,78**	1,06	-2,27	2006,24	0,07	10202,34
Média					1,69	-2,81	1159,19		

a - Archetti et al. [3]

Resultados - Instâncias de Chen et. al. (2007)

			Wilck et al.[d]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
SD1	228,28	*228, 28[b]	**228,28**	0,27	**228,28**	**228,28**	0,00	0,00	0,05	0,00	228,28
SD2	708,28	*708, 28[b]	**708,28**	1,95	**708,28**	**708,28**	0,00	0,00	0,58	0,00	708,28
SD3	430,40	*430, 40[b]	430,58	1,94	430,58	430,58	0,04	0,04	0,59	-	430,58
SD4	630,62	*630, 62[b]	631,05	6,24	631,05	631,05	0,07	0,07	2,16	-	631,05
SD5	1381,76	1389, 94[b]	1390,57	14,20	1390,57	1390,57	0,64	0,05	5,90	-	1390,57
SD6	830,86	*830, 86[b]	833,58	14,97	831,24	831,24	0,05	0,05	5,62	-	831,24
SD7	3640,00	3640, 00[b]	**3640,00**	28,61	**3640,00**	**3640,00**	0,00	0,00	13,74	0,00	3640,00
SD8	5068,28	*5068, 28[b]	**5068,28**	48,26	**5068,28**	**5068,28**	0,00	0,00	24,07	0,01	5068,28
SD9	2031,22	2042, 88[b]	2054,84	48,91	2044,20	2044,43	0,65	0,08	35,86	-	2044,20
SD10	2679,04	2683, 73[b]	2746,54	114,16	2684,88	2684,88	0,22	0,04	81,76	-	2684,88
SD11	13275,00	13280, 00[b]	**13280,00**	231,64	**13280,00**	**13280,00**	0,04	0,00	136,43	0,01	13280,00
SD12	7175,80	7239, 57[b]	7279,97	227,11	**7213,61**	**7216,34**	0,56	-0,32	179,19	0,02	7213,61
SD13	10053,60	10105, 86[b]	10110,57	421,95	10110,58	10110,58	0,57	0,05	168,07	-	10110,58
SD14	10588,20	10725, 38[b]	10786,52	718,65	**10715,53**	**10722,73**	1,27	-0,02	432,26	196,13	10715,53
SD15	14908,50	15129, 68[b]	15160,04	1278,35	**15093,85**	**15102,85**	1,30	-0,18	658,54	0,06	15093,85

Resultados - Instâncias de Chen et. al. (2007)

			Wilck et al.[d]		SplitILS						
Problem	LB	BKS	Best Sol.	Time (s)	Best Sol.	Avg. Sol.	Avg. Gap LB (%)	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)	Best All Exp.
SD16	3379,33	*3379, 33[b]	3433,83	1225,88	3395,11	3395,16	0,47	0,47	580,27	-	3381,25
SD17	26317,20	26533, 39[b]	26559,92	1722,20	**26493,56**	**26499,23**	0,69	-0,13	484,43	0,10	26493,56
SD18	14029,20	14283, 51[b]	14302,22	1735,83	**14197,80**	**14202,85**	1,24	-0,56	676,77	0,10	14197,80
SD19	19707,20	20152, 53[c]	**20152,53**	3093,17	**19989,95**	**20000,54**	1,49	-0,75	1261,95	0,17	19989,95
SD20	39252,80	39706, 51[c]	**39706,51**	6208,16	**39641,91**	**39648,42**	1,01	-0,15	1518,12	0,33	39637,53
SD21	11271,00	*11271, 00[a]	11461,20	10565,70	11344,96	11357,62	0,77	0,77	4326,99	-	11344,96
Média				1319,44			0,53	-0,02	504,44		

∗ - Solução ótima
a - Moreno et al. [11]
b - Archetti et al. [3]
c - Wilck IV e Cavalier [12]
d - Xeon 2,49 GHz, uma única execução

Resultados - Resumo

Conjunto	#Instâncias	#Melhores	#Empates	Avg. Gap (%)	Avg. Time (s)	Avg. Time BKS (s)
Belenguer [5][a,c]	25	18	7	−1,87	63,77	1,50
Belenguer [5][a]	25	22	3	−1,46	75,97	1,69
Campos [7][a]	49	44	5	−2,23	925,59	3,06
Archetti [4][a]	42	36	-	−2,81	1159,19	2,68
Chen [8][a]	21	7	5	−0,02	504,44	16,41*
Belenguer [5][b,c]	25	13	12	−0,32	56,87	9,47
Belenguer [5][b]	25	22	3	−1,13	74,51	1,39
Campos [7][b]	49	40	9	−0,41	872,02	63,55
Archetti [4][b]	42	36	5	−1,22	1062,86	44,74*
Chen [8][b]	21	5	6	−0,03	591,62	14,67*
Total	324	243	55			

a - PRVEF-FL
b - PRVEF-FI
c - Custos arredondados
* - Desconsiderando as instâncias em que o SplitILS não foi capaz de encontrar ou melhorar a BKS

Problemas de Roteamento de Veículos com Entregas Fracionadas

Conclusões

- Este trabalho propôs uma heurística *multi-start* baseada na metaheurística ILS e no procedimento RVND para o PRVEF;
- A abordagem desenvolvida é simples e sua eficácia, em termos de qualidade das soluções, foi demonstrada por meio de experimentos realizados em quatro conjuntos de instâncias, compostos por 324 problemas-teste com até 288 clientes;
- O método apresentado foi capaz de melhorar o resultado de 243 soluções e de chegar à mesma solução conhecida em outras 55 instâncias.


Parque da Cidade – Niterói/RJ

Vista da trilha do Costão de Itacoatiara – Niterói/RJ

Quem tiver interesse neste ou em outros problemas de Otimização Combinatória e/ou queira fazer uma Pós-Graduação (Mestrado ou Doutorado ou um Pós-Doutorado) na UFF (Niterói/RJ)

Contatos:
Luiz Satoru Ochi
http://www2.ic.uff.br/~satoru/
e-mail: satoru@ic.uff.br, ou,
luiz.satoru@gmail.com

Obrigado!!

Uma Heurística Híbrida Unificada para PRVs

Meta-heurísticas híbridas:
Iterared Local Search (ILS) e *Set Partitioning* (SP)

Métodos de Resolução

- Abordagens Exatas
  - Soluções ótimas.
  - Baseados em programação matemática, enumeração, etc.
  - Utilizados em problemas de pequena dimensão.

- Abordagens Heurísticas/Meta-heurísticas
  - Soluções aproximadas, não há garantia da solução ótima.
  - Podem resolver problemas grandes em um tempo relativamente curto.

- **Abordagens Híbridas (*Matheuristics*)**
  - Combinam características das heurísticas e dos métodos exatos.

Heurísticas Híbridas

- Heurísticas + Heurísticas
  - Unified Hybrid Genetic Search (UHGS)
    - Apresenta um Algorítmo Genético Híbdrido com uma busca local
    - Resolve 29 variantes do PRV

A unified solution framework for multi-attribute vehicle routing problems. Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Christian Prins. European Journal of Operational Research, v. 234, p. 658-673, 2014

Heurísticas Híbridas

- Heurísticas + Métodos Exatos
  - *Iterated Local Search* + *Set Partitioning*
    - Heurística *multi-start* ILS com *Randomized* VND
    - Solver de PIM para resolver o Set Partitioning
    - Resolve 25 variantes do PRV

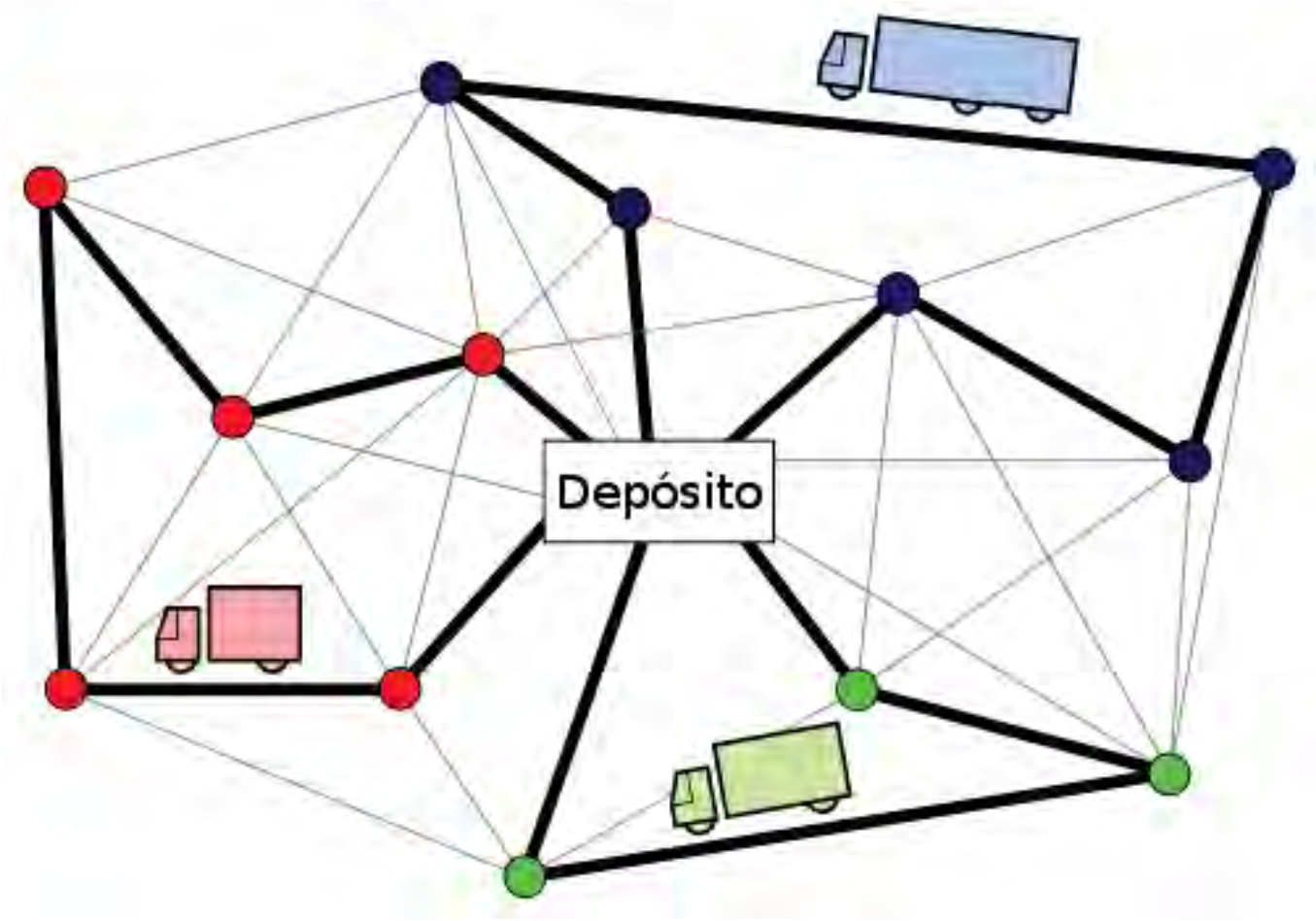
Heurísticas Híbridas

- ILS-RVND-SP
  - –***Um Algoritmo Unificado para uma Classe de Problemas de Roteamento de Veículos com Frota Heterogênea***. Puca Huachi Vaz Penna. Tese de Doutorado. Universidade Federal Fluminense, Niterói – RJ, 2013.
  - –***Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems***. Tese de Doutorado, Universidade Federal Fluminense, Niterói – RJ, 2012.
  - –***A Hybrid Algorithm for the Heterogeneous Fleet Vehicle Routing Problem*.** Anand Subramanian, Puca Huachi Vaz Penna, Eduardo Uchoa and Luiz Satoru Ochi. *European Journal of Operational Research* - EJOR, v. 221, p. 285-295, 2012
  - –***A hybrid algorithm for a class of vehicle routing problems***. Anand Subramanian, Eduardo Uchoa, Luiz Satoru Ochi. *Computers & Operations Research*, v. 40, p. 2519-2531, 2013

PRVFH: Motivação

- Aplicações reais envolvendo roteamento possuem, normalmente, uma enorme quantidade de características e restrições:
  - **Estrutura**: composição da frota e número de depósitos.
  - **Requisitos dos clientes**: visitas dentro de um horário ou múltiplas visitas.
  - **Regras de operações dos veículos**: restrições de carga ou vias, distância máxima, etc.
  - **Decisões de contexto**: tráfego.

PRV com Frota Heterogênea: PRVFH



Descrição do PRVFH

- Características comuns dos PRVFHs
  - único depósito.
  - m diferentes tipos de veículos.
  - n clientes (1, 2, ..., n).
  - q_i, demanda determinística do cliente i.
  - Q_u, capacidade do veículo u (u = 1, ..., m).
  - f_u, custo fixo do veículo u.
  - r_u, custo variável (dependente) por unidade de distância, do veículo u.
  - c_{ij}^u, matriz de custo: $c_{ij}^u = d_{ij} r_u$ sendo d_{ij} a distância entre os clientes (i, j).

Descrição do PRVDFH

- Duas variantes principais
  - PRV com Dimensionamento de Frota Heterogênea (PRVDFH)
  - PRV com Frota Heterogênea Fixa (PRVFHF)

Características do PRVDFH

- PRVDFH
  - Frota ilimitada: $m_u = +\infty, \forall u \in M$.
- Variantes tratadas
  - PRVDFH com custo fixo e variável (PRVDFH-FV)
    - $f_u > 0; r_u > 0, \forall u \in M$.
  - PRVDFH com custo fixo (PRVDFH-F)
    - $f_u > 0; r_u = 1, \forall u \in M$.
  - PRVDFH com custo variável (PRVDFH-V)
    - $f_u = 0; r_u > 0, \forall u \in M$.

Trabalhos para o PRVDFH

Trabalho	Abordagem	Frota	Custo Fixo	Custo Var.	Custo Fixo e Var
[Golden *et al.*, 1984]	heurísticas	Ilimitada	X		
[Ochi *et al.*, 1998a]	AG+SS	Ilimitada	X		
[Ochi *et al.*, 1998b]	AG+SS paralelo	Ilimitada	X		
[Gendreau *et al.*, 1999]	BT+GENIUS+PMA	Ilimitada	X	X	
[Taillard, 1999]	BT+PMA+GC	Ilimitada		X	
[Renaud & Boctor, 2002]	Sweep	Ilimitada	X		
[Lima *et al.*, 2004]	AM	Ilimitada	X		
[Yaman, 2006]	Desigualdades válidas	Ilimitada	X		
[Choi & Tcha, 2007]	GC + SC	Ilimitada	X	X	X
[Lee *et al.*, 2008]	BT+SP	Ilimitada	X	X	
[Pessoa *et al.*, 2008]	BCP	Ilimitada	X		
[Baldacci & Mingozzi, 2009]	SP exato	Ilimitada	X	X	
[Brandão, 2009]	BT determinística	Ilimitada	X	X	
[Liu *et al.*, 2009]	AG híbrido	Ilimitada	X	X	
[Pessoa *et al.*, 2009]	BCP	Ilimitada	X	X	X
[Prins, 2009b]	AM	Ilimitada	X	X	X
[Imran *et al.*, 2009]	VNS	Ilimitada	X	X	X
[Subramanian *et al.*, 2012]	ILS híbrido	Ilimitada	X	X	X
[Penna *et al.*, 2013a]	ILS	Ilimitada	X	X	X

Características do PRVFHF

- PRVFHF
  - Frota Limitada.
- Variantes tratadas
  - PRVDFH com custo fixo e variável (PRVFHF-FV)
  - $f_u > 0; r_u > 0, \forall u \in M$.
- PRVDFH com custo variável (PRVFHF-V)
  - $f_u = 0; r_u > 0, \forall u \in M$.

Trabalhos para PRVFHF

Trabalho	Abordagem	Frota	Custo Var.	Custo Fixo e Var.
[Taillard, 1999]	BT+PMA+GC	Limitada	X	
[Prins, 2002]	Heurísticas	Limitada	X	
[Tarantilis *et al.*, 2003]	TA	Limitada	X	
[Tarantilis *et al.*, 2004]	TA	Limitada	X	
[Gencer *et al.*, 2006]	Heurísticas	Limitada		X
[Li *et al.*, 2007]	RTR	Limitada	X	
[Baldacci & Mingozzi, 2009]	SP exato	Limitada	X	X
[Prins, 2009b]	AM	Limitada	X	
[Li *et al.*, 2010]	PMA+RC+BT	Limitada	X	X
[Brandão, 2011]	BT determinística	Limitada	X	
[Subramanian *et al.*, 2012]	ILS híbrido	Limitada	X	X
[Penna *et al.*, 2013a]	ILS	Limitada	X	X
[Duhamel *et al.*, 2011]	GRASP+BLE	Limitada	X	X
[Duhamel *et al.*, 2013]	GRASP+BLE Paralelo	Limitada	X	X

Heurística Híbrida

- Faz uso do HURVFH.
- Implementa um mecanismo de memória por meio de um método exato baseado no *Set Partitioning* - SP (Problema de Particionamento de Conjuntos).
- O SP é tratado por um resolvedor de Programação Inteira Mista (PIM).
- É uma extensão do algoritmo de nossa autoria descrito em [Subramanian *et al.*, 2012].

Heurística Híbrida

- Utiliza o HURVFH para criar um conjunto de rotas de boa qualidade, que em seguida é utilizado pelo SP na construção da solução do problema. O HURVFH é executado interativamente sempre que uma nova solução é encontrada durante a resolução do SP.

Implementando Memória com o SP

- Conjuntos:
  - V' o conjunto de clientes
  - M o conjunto de tipos de veículos
  - R o conjunto de todas as possíveis rotas para todos os tipos de veículos
  - $R_i \subseteq R$ o subconjunto de rotas que atende o cliente $i \in V'$
  - $R_u \subseteq$ R o conjunto de rotas associadas ao veículo do tipo $u \in M$
- Dados:
  - c_j como o custo associado a rota j
  - m_u número de veículos do tipo u
- Variáveis:
  - y_j variável binária associada à rota $j \in R$, onde $y_j = 1$ se a rota j estiver na solução

Formulação do SP

$$\text{Min} \sum_{j \in \mathcal{R}} c_j y_j \qquad (1)$$

sujeito a

$$\sum_{j \in \mathcal{R}_i} y_j = 1 \qquad (2)$$

$$\sum_{j \in \mathcal{R}_u} y_j \leq m_u \qquad (3)$$

$$y_j \in \{0, 1\}. \qquad (4)$$

Heurística Híbrida Unificada

Algoritmo 2: HURVFH($MaxIterMS$, β, s)

```
1  Início
2      Inicializa frota
3      v ← número total de veículos
4      f(s*) ← ∞
5      para i ← 1 até MaxIterMS faça
6          iterILS ← 0
7          MaxIterILS ← CalcularMaxIterILS(n, v, β)
8          se (não foi fornecida solução inicial) então
9              s ← GeraSoluçãoInicial(v)
10         s*' ← BuscaLocal(s)
11         enquanto (iterILS ≤ MaxIterILS) faça
12             s' ← Perturbação(s*')
13             s'' ← BuscaLocal(s')
14             se (f(s'') < f(s*')) então
15                 s*' ← s''
16                 iterILS ← 0
17             iterILS ← iterILS + 1
18         se (f(s*') < f(s*)) então
19             s* ← s*'
20     retorna s*
21 Fim
```

Heurística Híbrida Unificada

Algoritmo 10: HHURVFH($MaxIterMS$, β, $MaxTempo$, $Tolerancia$, n, $MaxN$)

```
1  Início
2      Inicializa frota
3      v ← número total de veículos
4      f(s*) ← ∞
5      para i ← 1 até MaxIterMS faça
6          iterILS ← 0
7          MaxIterILS ← CalcularMaxIterILS(n, v, β)
8          s ← GeraSoluçãoInicial(v)
9          s'* ← BuscaLocal(s)
10         AdicionaRotasTemporárias(ConjuntoRotas, s*', f(s*), Tolerancia)
11         enquanto (iterILS ≤ MaxIterILS) faça
12             s' ← Perturbação(s*')
13             s'' ← BuscaLocal(s')
14             AdicionaRotasTemporárias(ConjuntoRotas, s'', f(s*), Tolerancia)
15             se (f(s'') < f(s*')) então
16                 s*' ← s''
17                 iterILS ← 0
18             iterILS ← iterILS + 1
19         se (n ≥ MaxN) então
20             s*' ← ResolvePPC(ConjuntoRotas, s*', MaxTempo, Tolerancia)
21         se (f(s*') < f(s*)) então
22             s* ← s*'
23     se (n < MaxN) então
24         s* ← ResolvePPC(ConjuntoRotas, s*, MaxTempo, Tolerancia)
25     retorna s*
26 Fim
```

Heurística Híbrida Unificada

Algoritmo 10: HHURVFH($MaxIterMS$, β, $MaxTempo$, $Tolerancia$, n, $MaxN$)

```
1  Início
2      Inicializa frota
3      v ← número total de veículos
4      f(s*) ← ∞
5      para i ← 1 até MaxIterMS faça
6          iterILS ← 0
7          MaxIterILS ← CalcularMaxIterILS(n, v, β)
8          s ← GeraSoluçãoInicial(v)
9          s'* ← BuscaLocal(s)
10         AdicionaRotasTemporárias(ConjuntoRotas, s*', f(s*), Tolerancia)
11         enquanto (iterILS ≤ MaxIterILS) faça
12             s' ← Perturbação(s*')
13             s'' ← BuscaLocal(s')
14             AdicionaRotasTemporárias(ConjuntoRotas, s'', f(s*), Tolerancia)
15             se (f(s'') < f(s*')) então
16                 s*' ← s''
17                 iterILS ← 0
18             iterILS ← iterILS + 1
19         se (n ≥ MaxN) então
20             s*' ← ResolvePPC(ConjuntoRotas, s*', MaxTempo, Tolerancia)
21         se (f(s*') < f(s*)) então
22             s* ← s*'
23     se (n < MaxN) então
24         s* ← ResolvePPC(ConjuntoRotas, s*, MaxTempo, Tolerancia)
25     retorna s*
26 Fim
```

Heurística Híbrida Unificada

Procedimento 11: ResolvePPC(*ConjuntoRotas*, s^*, *MaxTempo*, *Tolerancia*)

```
1  Início
2      AdicionaRotasPermanentes(ConjuntoRotas, s*)
3      melhorouSolucao ← verdadeiro
4      enquanto (melhorouSolucao) faça
5          modeloPC ← CriaModeloPPC(ConjuntoRotas, v)
6          s' ← ResolvedorPIM(modeloPC, s*, MaxTempo, IncumbentCallback(s*))
7          se (f(s') < f(s*)) então
8              s* ← s'
9              AdicionaRotasPermanentes(ConjuntoRotas, s*)
10         senão
11             RemoveRotasTemporárias(ConjuntoRotas)
12             melhorouSolucao ← falso
13         se (Problema for resolvido na raiz) então
14             Aumenta a Tolerancia em 2%
15         se (Tempo do PIM > MaxTempo) então
16             Diminui a Tolerancia em 2%
17     retorna s*
18 Fim
```

Heurística Híbrida Unificada

Procedimento 12: *IncumbentCallback*(s^*)

```
1 Início
2     s ← Solução Incumbente
3     s' ← HURVFH1(1, β, s, ConjuntoRotas)
4     se f(s') < f(s*) então
5         s* ← s'
6 Fim
```

Resultados para o PRVFH

•O algoritmo HHURVFH foi desenvolvido em C++ usando o compilador g++ 4.6.3.

•O *Cplex* 12.5.1 foi utilizado como resolvedor PIM.

- –Os testes foram executados em um computador Intel Core i7 2,93 GHz com 8 GB de RAM.
- –Sistema operacional *Ubuntu Linux* 12.04 (*kernel* 3.5 – 64 *bits*).
- –Em todas as variantes, cada instância foi testada 10 vezes.
- –*MaxIterMS* = 30, *MaxTempo* = 30 segundos, Tolerancia = 25%.

•Esta versão do ILS-RVND-SP (denominada HHURVRP) foi testado em 17 variantes, sendo 18 conjuntos de problemas-teste

- –Total de 688 problemas
- –Resultados para PRVDFH e PRVFHF

			Melhores Trabalhos da Literatura					
Variante	**Instância**	n	**Autores**	**Gap Mel.**	**Gap Med.**	**Tempo (s)**	**Sol./ Tot.**	**CPU (GHz)**
PRVDFH-FV	[Taillard, 1999]	[20 - 100]	[Prins, 2009b]	0,02		6,86	7/12	P4 1,8
			[Imran *et al.*, 2009]	0,04		117,92	8/12	PM 1,7
			HURVFH	**0,00**	0,09	24,64	11/12	I7 2,93
			HHURVFH	**0,00**	0,01	7,56	12/12	I7 2,93
PRVDFH-F	[Taillard, 1999]	[20 - 100]	[Imran *et al.*, 2009]	0,05		126,60	9/12	PM 1,7
			[Liu *et al.*, 2009]	0,01	0,19	107,96	10/12	P4 3,0
			HURVFH	0,01	0,23	30,48	11/12	I7 2,93
			HHURVFH	**-0,01**	0,11	9,09	12/12	I7 2,93
PRVDFH-V	[Taillard, 1999]	[20 - 100]	**[Choi & Tcha, 2007]**	**0,00**	0,12	21,05	11/12	P4 2,6
			[Imran *et al.*, 2009]	0,02		134,32	7/8	PM 1,7
			HURVFH	**0,00**	0,17	30,38	11/12	I7 2,93
			HHURVFH	**0,00**	0,12	3,33	12/12	I7 2,93
PRVDFH-V	[Brandão, 2011]	[100 - 199]	[Brandão, 2011]	0,00			5/5	P4 2,6
			HURVFH	0,39	0,90	423,68	0/5	I7 2,93
			HHURVFH	**-0,09**	0,15	40,71	5/5	I7 2,93
PRVFHF-FV	[Taillard, 1999]	[50 - 100]	[Li *et al.*, 2010]	0,11	0,22	43,25	7/8	P 2,2
			HURVFH	-0,05	0,24	32,89	8/8	I7 2,93
			HHURVFH	**-0,07**	0,17	7,73	8/8	I7 2,93
PRVFHF-FV	[Duhamel *et al.*, 2011]	[20 - 256]	[Duhamel *et al.*, 2011]	0,86		468,57	7/96	P 2,2
			HURVFH	-0,06	0,23	1684,68	53/96	I7 2,93
			HHURVFH	**-0,12**	0,33	464,94	78/96	I7 2,93
PRVFHF-V	[Taillard, 1999]	[50 - 100]	**[Li *et al.*, 2007]**	**0,03**		57,16	7/8	Ath 1,0
			[Prins, 2009b]	0,08		25,38	6/8	P4 1,8
			HURVFH	**0,03**	0,22	31,89	7/8	I7 2,93
			HHURVFH	**0,03**	0,18	4,03	7/8	I7 2,93
PRVFHF-V	[Brandão, 2011]	[100 - 199]	[Brandão, 2011]	0,00			5/5	P4 2,6
			HURVFH	-0,21	0,03	404,30	5/5	I7 2,93
			HHURVFH	**-0,39**	-0,15	42,25	5/5	I7 2,93
PRVFHF-V	[Li *et al.*, 2007]	[200 - 360]	[Li *et al.*, 2007]	0,28		346,22	0/5	Ath 1,0
			[Brandão, 2011]	**0,09**		1246,28	2/5	P4 2,6
			HURVFH	0,82	1,53	2856,46	1/5	I7 2,93
			HHURVFH	0,34	1,73	551,01	2/5	I7 2,93

Conclusões

- Foi apresentado uma heurística hibrida
- Incorpora um mecanismo de memória por meio de *Set Partitioning*
- O ILS é utilizado para gerar as rotas para o SP
- O SP é resolvido iterativamente pelo *cplex* e sempre que uma nova solução é encontrado o ILS é executado
- Resolve mais de 27 variantes do PRV

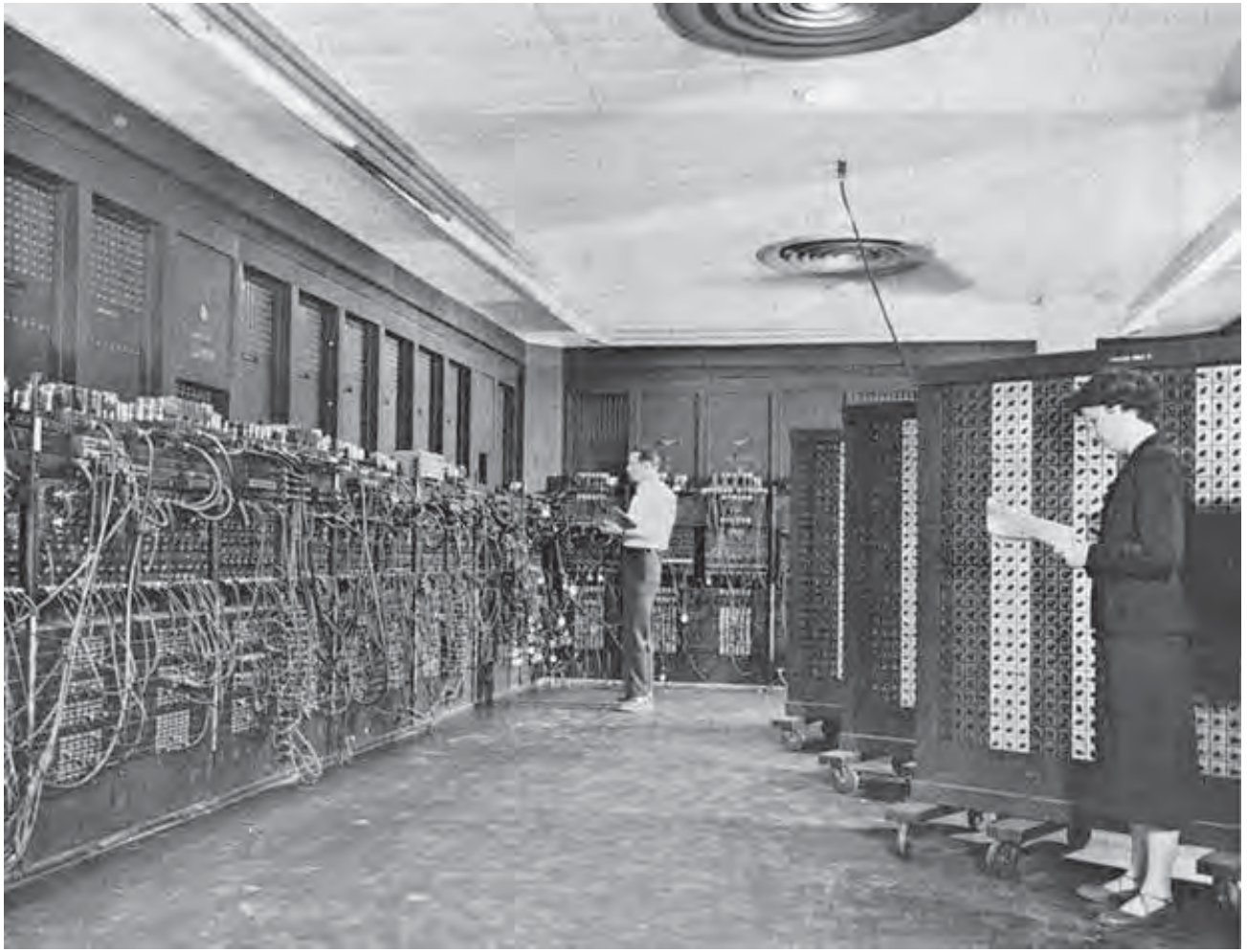
Acelerando algoritmos de otimização com GPUs

- Breve história das Graphics Processing Units (GPUs)
- Arquiteturas heterogêneas em CPU/GPU
- Programação paralela em CUDA
- Problema de roteamento de veículos e *scheduling* em CPU/GPU

Computadores Modernos

- Por mais de duas décadas, computadores baseados em uma **unidade central de processamento** (CPU) impulsionaram o desenvolvimento de aplicações
- Processadores das famílias Intel Pentium e AMD Opteron trouxeram a usuários comuns giga (bilhões) de **operações com ponto flutuante por segundo** (GFLOPS) e centenas de GFLOPS a servidores especializados
- Usuários ficam acostumados com os novos recursos e funcionalidades com interfaces gráficas mais bonitas
- Ciclo benéfico para usuários e indústria de computadores.

História dos Computadores

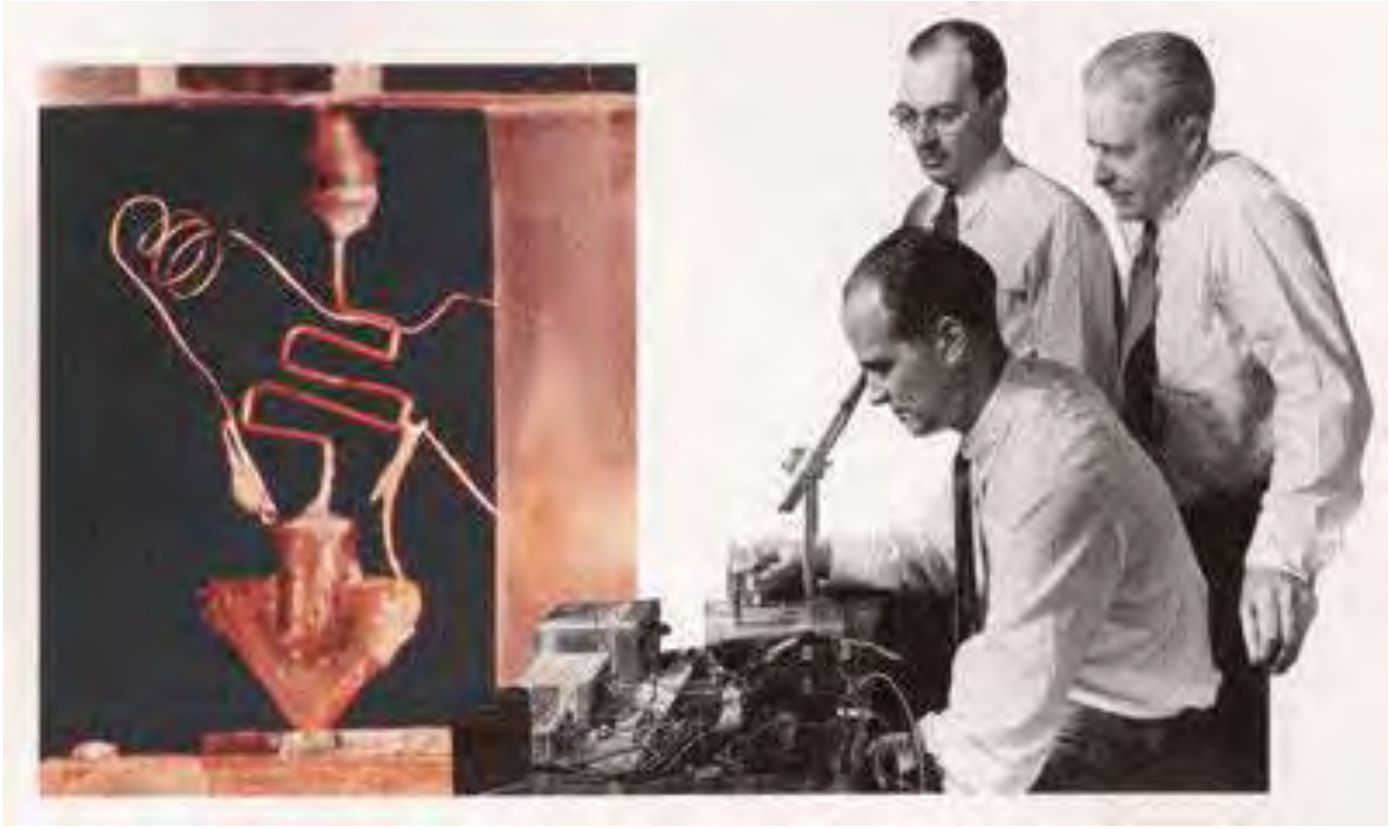


ENIAC (1946)

História dos Computadores

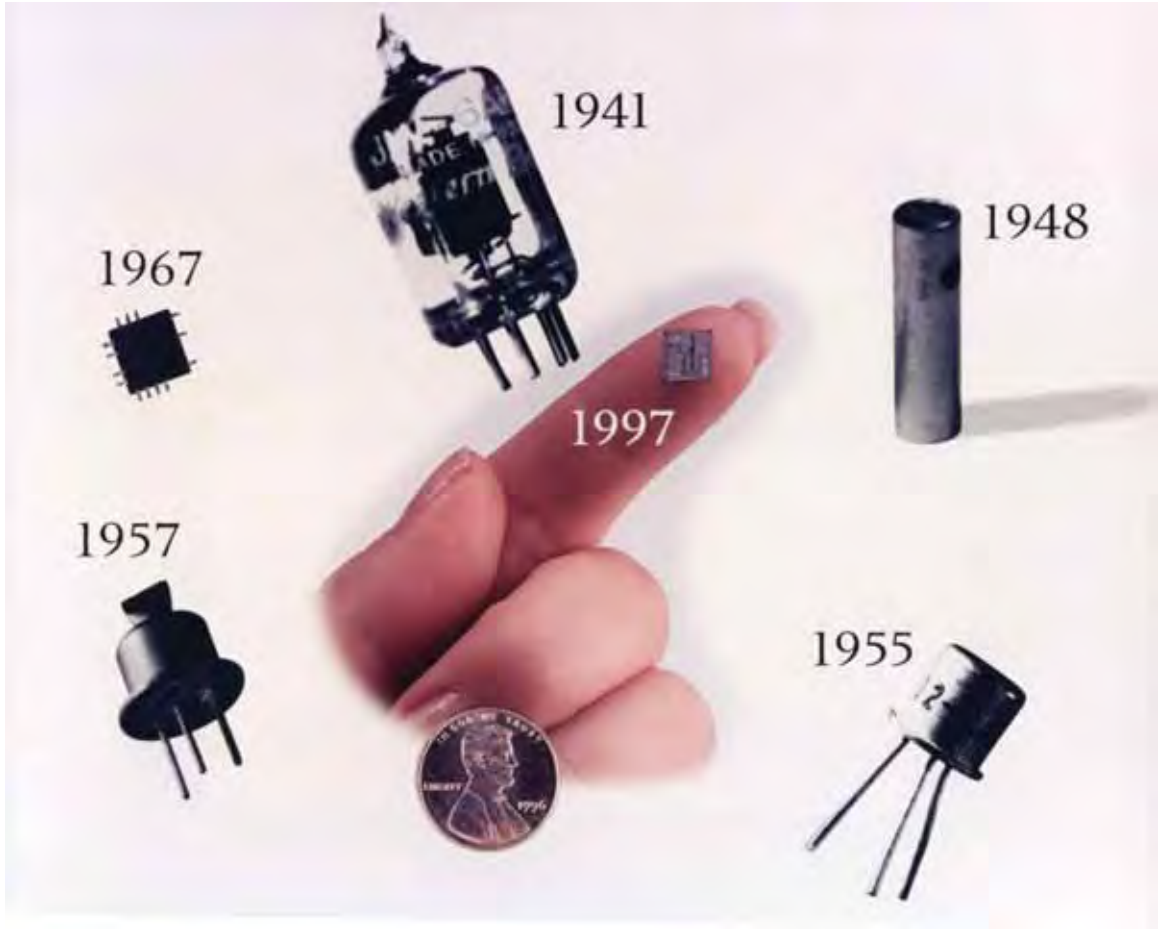
Válvulas ENIAC

História dos Computadores



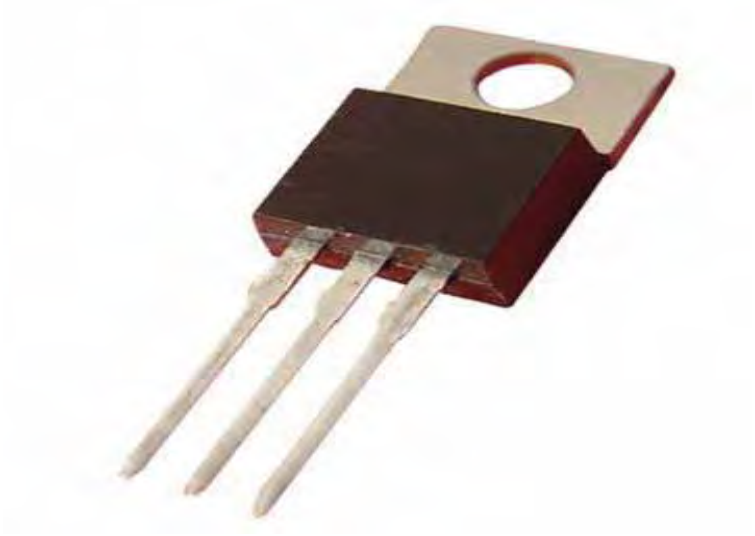
Transistor (1947) – Bell Labs

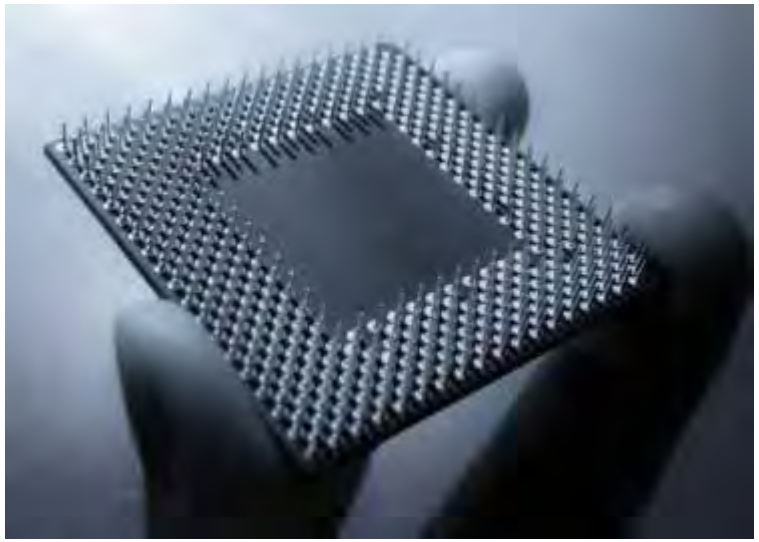
História dos Computadores



Cada vez menores…

História dos Computadores





Milhares em um único chip

Computadores Modernos

- Em 1965, o então presidente da Intel Gordon G. Moore fez uma previsão (que acabou ficando conhecida como Lei de Moore) de que a cada 18 meses o número de transistores dos chips dobraria de tamanho
- Porém, em 2003 a evolução do desenvolvimento de processadores com uma única CPU foi dificultada por problemas de alto consumo de energia e dissipação de calor
- Processadores migraram para modelos baseados em cores, causando grande impacto na comunidade de desenvolvimento de software

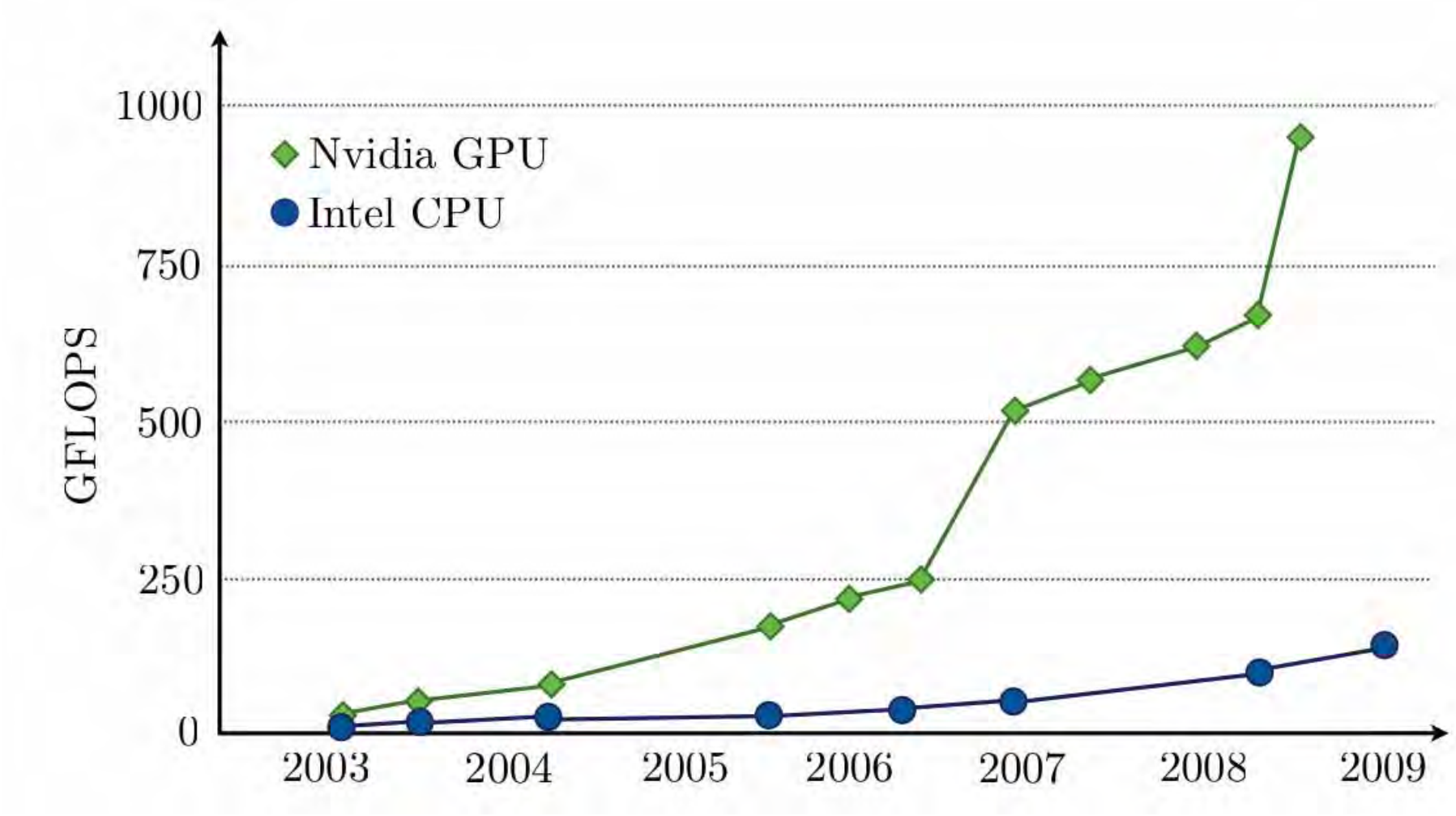
Computadores Paralelos

- A partir de 2003, a linha multicore ganhou força com processadores que priorizam manter a velocidade da execução sequencial dos programas, mas fornecendo múltiplos cores
- Processadores Intel Core i5, i7, ... e AMD Phenom
- A linha many-core investe fortemente no desempenho de execução de aplicações paralelas
- Exemplo: NVIDIA Titan X, **Unidade de Processamento Gráfico** (GPU) com 3072 cores

Computadores Paralelos

Placa com GPU fabricada pela AMD

Motivação: computação GPU



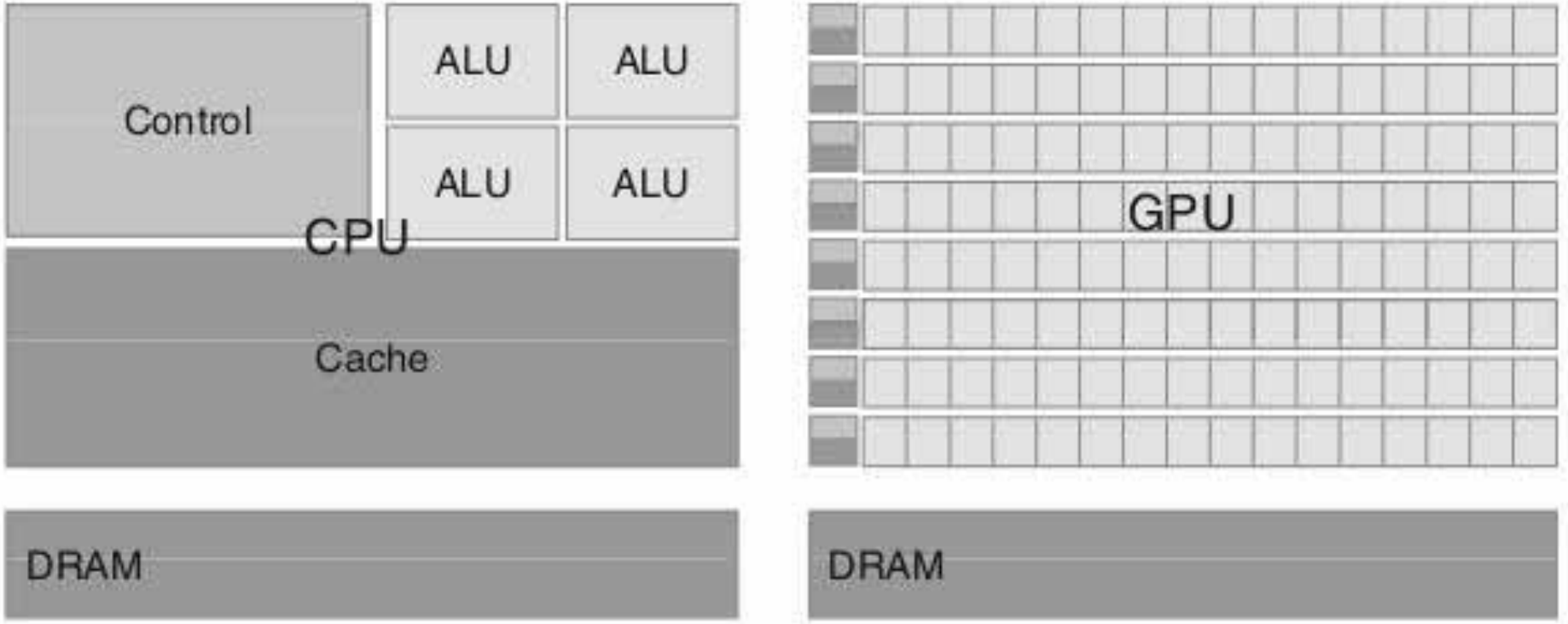
Evolução do poder computacional

Autor: Felipe A. Cruz – Bristol University

Motivação: computação GPU

- Massivamente paralela
- Centenas de *cores* de processamento
- Milhares de threads
- Baixo custo
- Altamente disponível
- Facilmente programável

Contraste de arquiteturas

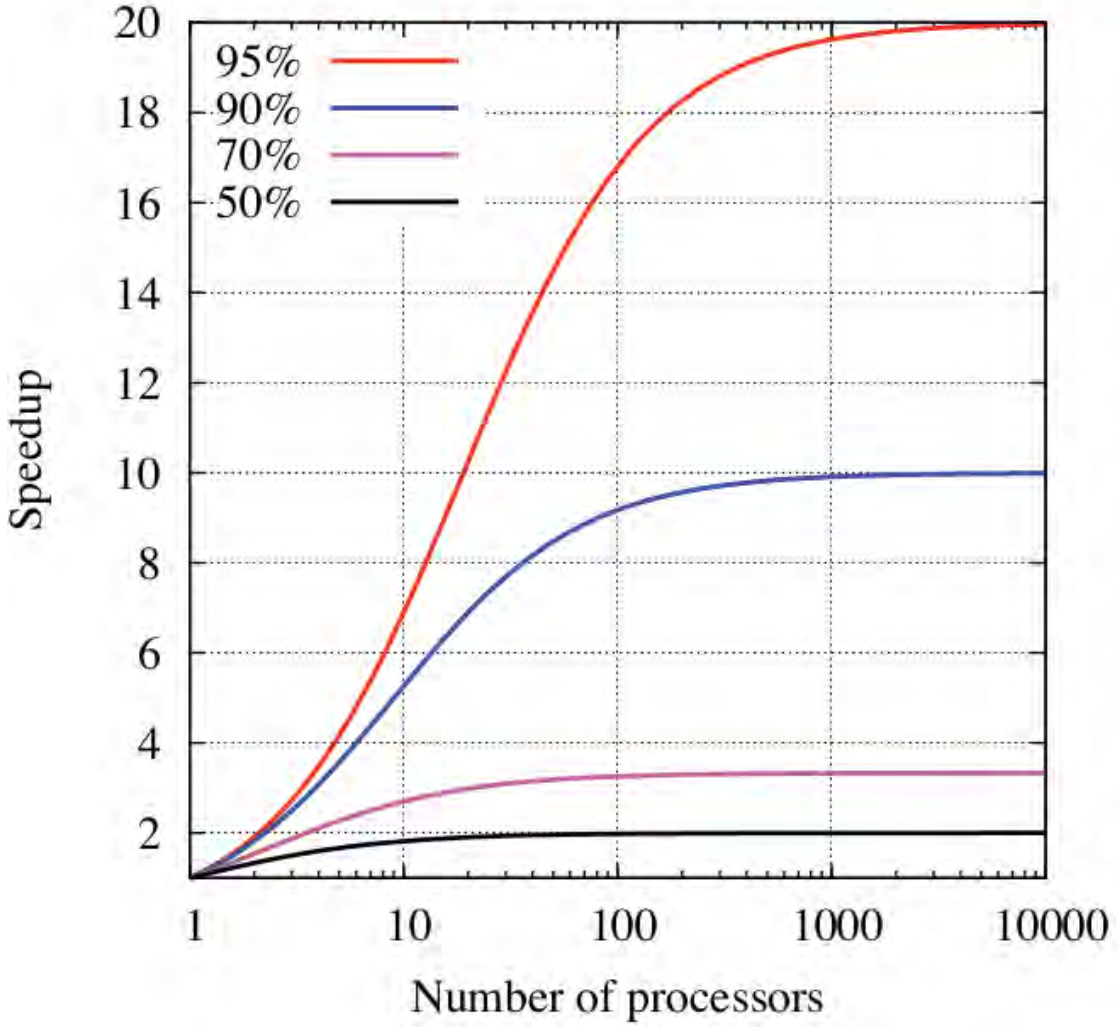


Fonte: Kirk & Hwu. Programming Massively Parallel Processors.

Motivação: computação GPU

- GPU's são mais adequadas para lidar com grandes porções de informação
- Transferência de memória mais rápida do que computação distribuída
- Hierarquia de memória sofisticada com memória constante, compartilhada, de textura e acesso global
- Relação de custo benefício atrativa
- Programação simplificada, porém não-trivial quando existem muitas dependências na informação tratada
- Possível alto ganho de aceleração no código: **speedup**

É possível ter 300x de speedup?



Lei de Amdahl: speedup máximo

Autor: Felipe A. Cruz – Bristol University

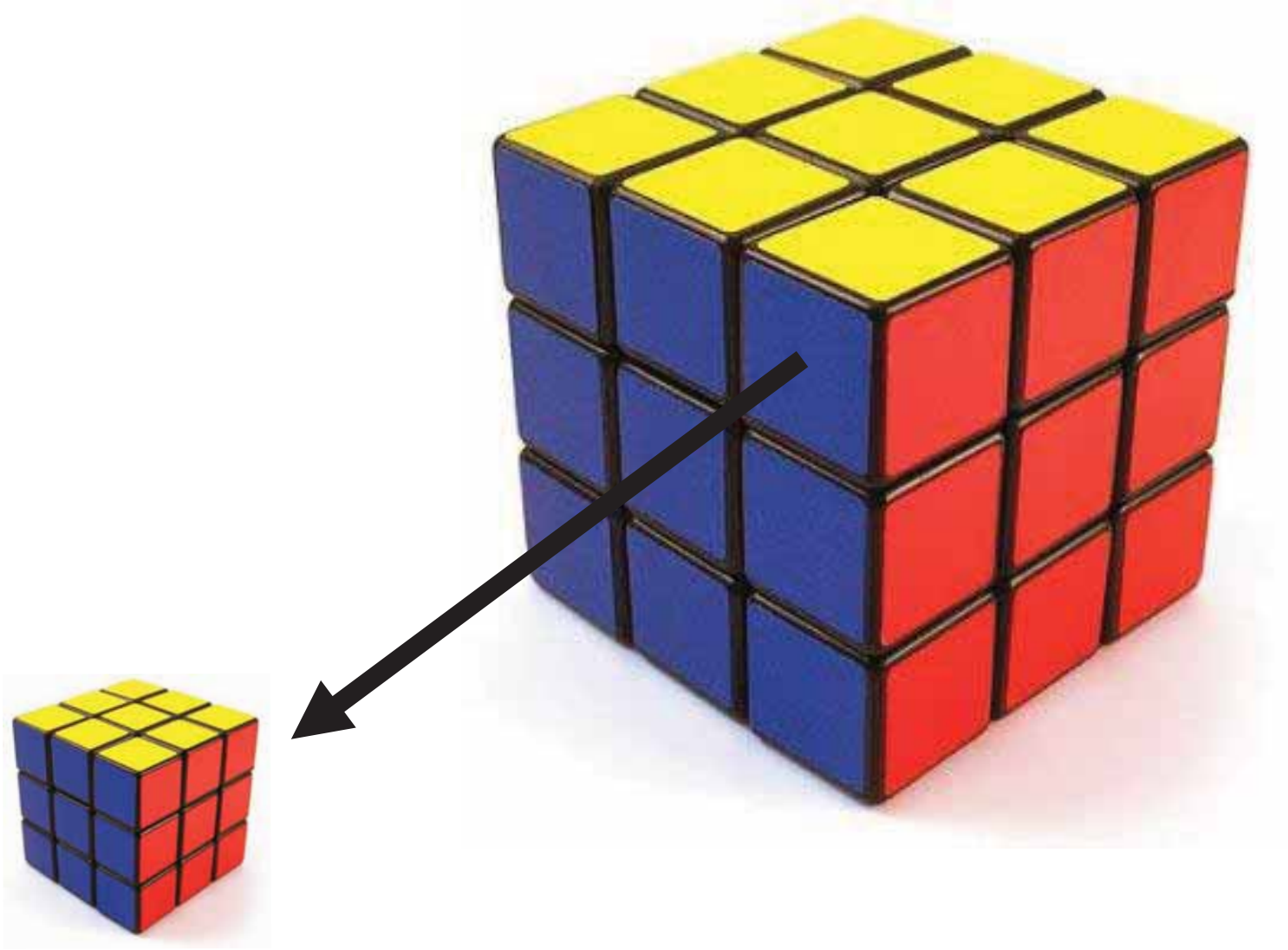
Lei de Amdahl

- É possível conseguir aceleração de 100x em **algumas** aplicações
- Esta aceleração depende da parte não-paralela
- Aplicações complexas normalmente fazem uso de muitos algoritmos
- Para maiores ganhos, é necessária a re-estruturação da forma como as computações são feitas
- **Significado da Lei de Amdahl**: um programa acelerado poderá ser tão rápido quanto a sua porção sequencial.

Compute Unified Device Architecture (CUDA)

- Linguagem desenvolvida pela NVIDIA (a partir de 2006)
- Extensão de C/C++
- Fácil de programar e especializada ao hardware
- Mais madura que sua concorrente OpenCL (padrão industrial para arquiteturas heterogêneas)
- OpenCL é bastante complexa e ainda menos madura, quando comparada a CUDA

CUDA – Grids e Blocos



CUDA – Adição de Vetores

Tempo

Alocar Memória

Cópia Host → GPU

Chamada de Kernel

Cópia GPU → Host

Liberar memória

```
void vecAddCPU(int* A, int* B, int* C, int N)
{
    int i;
    for (i = 0; i < N; ++i)
        C[i] = A[i] + B[i];
}

__global__ void vecAddGPU(int* A, int* B, int* C, int N)
{
    int i = threadIdx.x;
    if (i < N)
        C[i] = A[i] + B[i];
}
```

CUDA – Adição de Vetores

Alocar Memória

Cópia Host → GPU

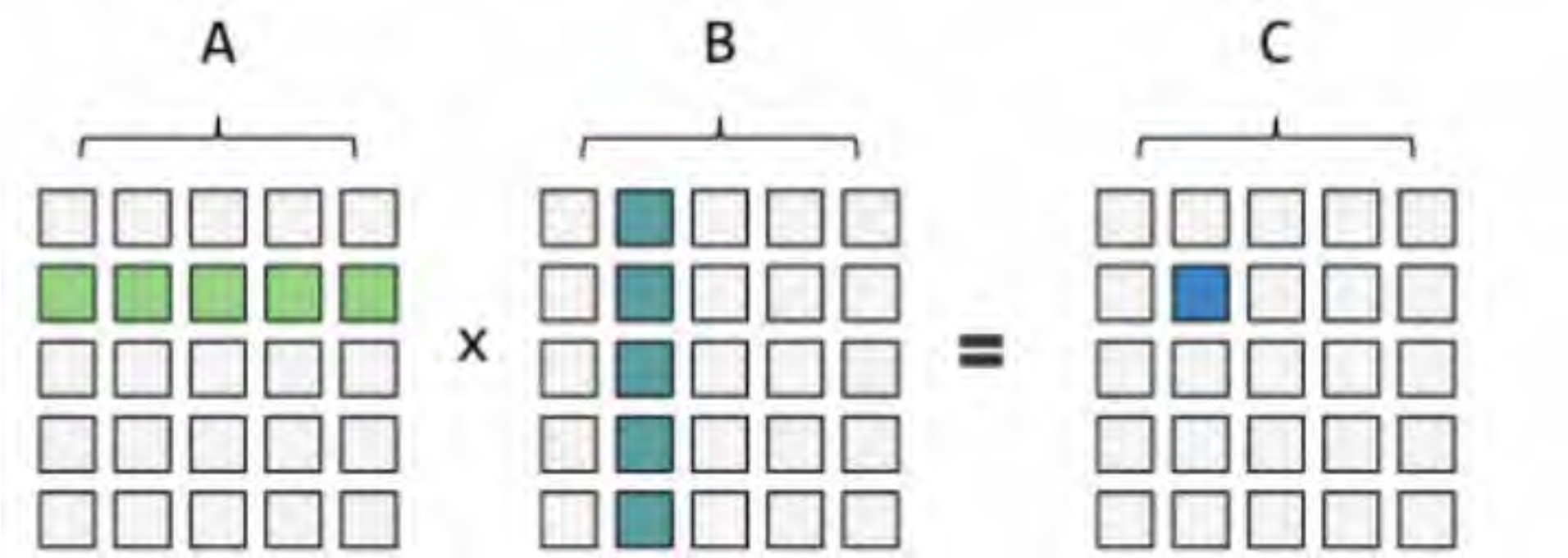
Chamada de Kernel

Cópia GPU → Host

Liberar memória

```
void somaVetores(int* A, int* B, int* C, int N)
{
    size_t size = N * sizeof(int);
    int *d_A, *d_B, *d_C;
    cudaMalloc((void**) &d_A, size);
    cudaMalloc((void**) &d_B, size);
    cudaMalloc((void**) &d_C, size);

    cudaMemcpy(d_A, A, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_B, B, size, cudaMemcpyHostToDevice);

    vecAddGPU<<<1, N>>>(d_A, d_B, d_C, N);

    cudaMemcpy(C, d_C, size, cudaMemcpyDeviceToHost);

    cudaFree(d_A);
    cudaFree(d_B);
    cudaFree(d_C);
}
```

CUDA – Multiplicação de Matrizes




```
C[i][j] = sum(A[i][k] * B[k][j]) for k = 0 ... n
```

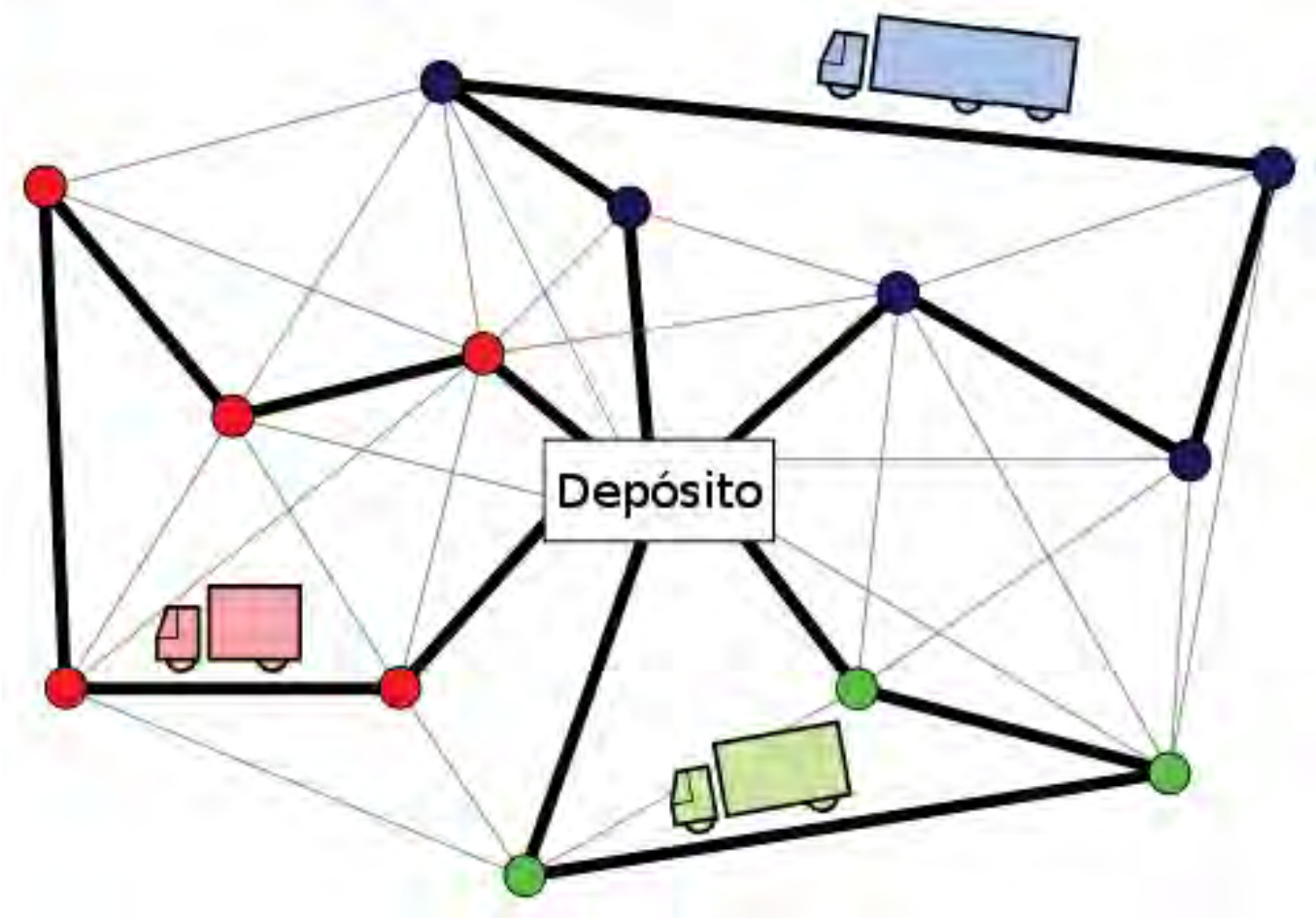
Cada célula recebe um produto linha x coluna

Autor: Felipe A. Cruz – Bristol University

```
void matrixSumCPU(int** A, int** B, int **C, int N)
{
    int i, j, k;
    for (i = 0; i < N; ++i)
        for (j = 0; j < N; ++j)
        {
            C[i][j] = 0;
            for (k = 0; k < N; ++k)
                C[i][j] += A[i][k] * B[k][j];
        }
}
```

```
__global__ void matrixSumGPU(int** A, int** B, int** C, int N)
{
    int i = threadIdx.y;
    int j = threadIdx.x;

    int k, acc = 0;
    for (k = 0; k < N; ++k)
        acc += A[i][k] * B[k][j];

    C[i][j] = acc;
}
```

GPU para Roteamento de Veículos e Scheduling



GPU para Roteamento de Veículos e Scheduling

- Como desenvolver um algoritmo eficiente utilizando tecnologia GPU?

- Evitar transferências de memória desnecessárias entre CPU e GPU

- Acelerar grandes porções do algoritmo (Lei de Amdahl)

- Paralelizar tarefas semelhantes (paralelismo de dados)

GPU para Roteamento de Veículos e Scheduling

- Como desenvolver um algoritmo eficiente utilizando tecnologia GPU?

- Evitar transferências de memória desnecessárias entre CPU e GPU

- Acelerar grandes porções do algoritmo (Lei de Amdahl)

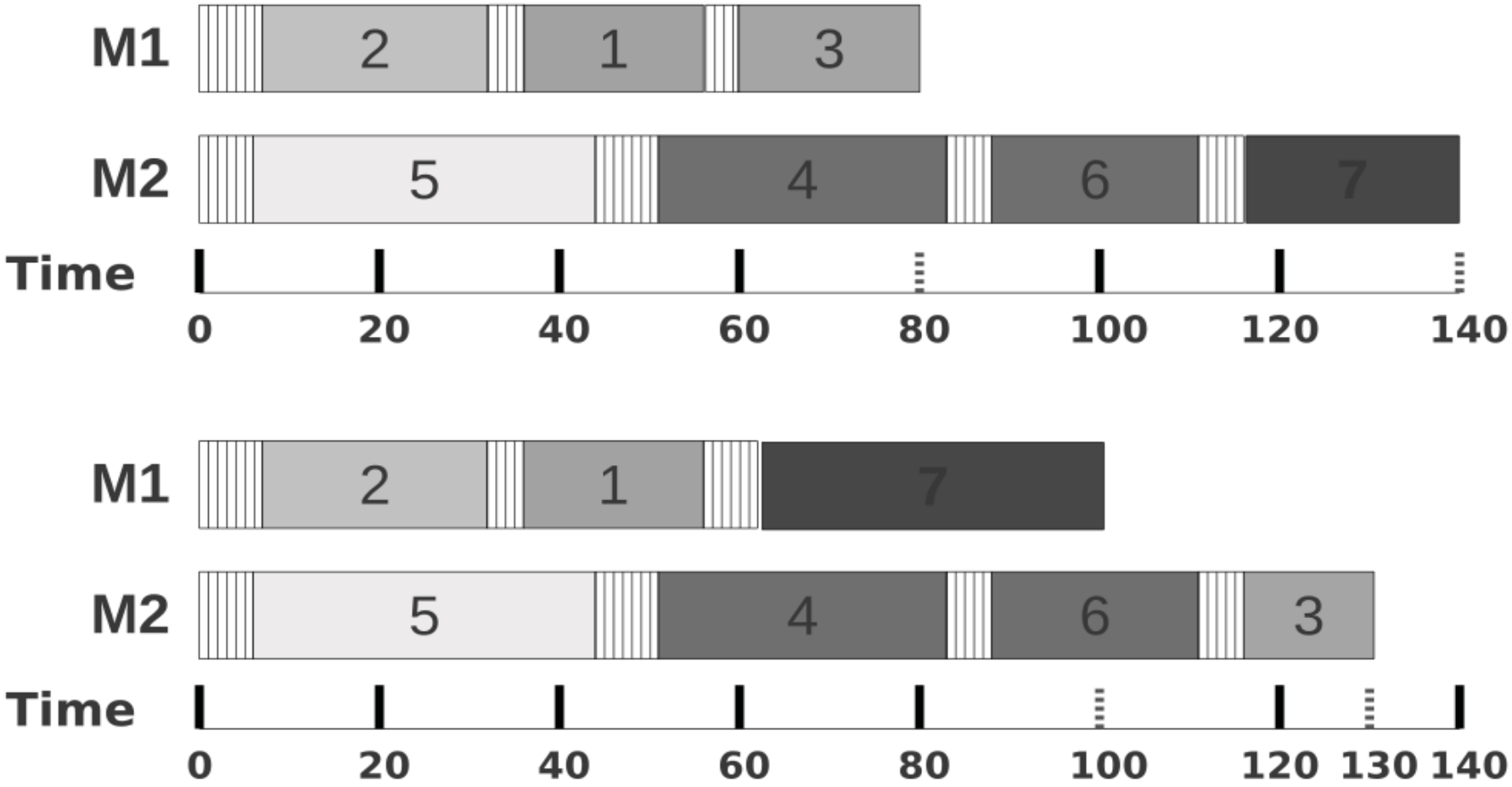
- Paralelizar tarefas semelhantes (paralelismo de dados)

- **BUSCA LOCAL!**

Problema de Scheduling com Máquinas Paralelas

- Conjunto de N tarefas
- Conjunto de M máquinas
- Atribuir cada tarefa a exatamente uma máquina
- Cada tarefa tem um tempo de processamento dependente da máquina
- Diferentes tempos de setup entre tarefas, dependentes da máquina
- Tempo de setup inicial, por máquina
- Minimizar o tempo máximo de processamento

Busca local inter-máquina



I.M.Coelho, M.N.Haddad, L.S.Ochi, M.J.Souza, R.Farias, *A hybrid CPU-GPU local search heuristic for the unrelated parallel machine scheduling problem*, 3rd Workshop on Applications Multi-Core Architecture (WAMCA), Nova York, 2012

Busca local inter-máquina

input : set of jobs N, set of machines M, solution s, linear time evaluation function $f(.)$, vector g_values

// get thread index information

1 $thread\_index \leftarrow$ get thread index using block sizes;

2 $(m_1, j_1, m_2, j_2) \leftarrow$ get swap information using the $thread\_index$ number;

// compute the makespan of new solution (same as CPU)

3 $makespan \leftarrow$ using $f(.)$, compute the new value of s with the swap of (m_1, j_1) and (m_2, j_2);

// store $makespan$ in vector g_values

4 $g\_values[thread\_index] \leftarrow makespan$;

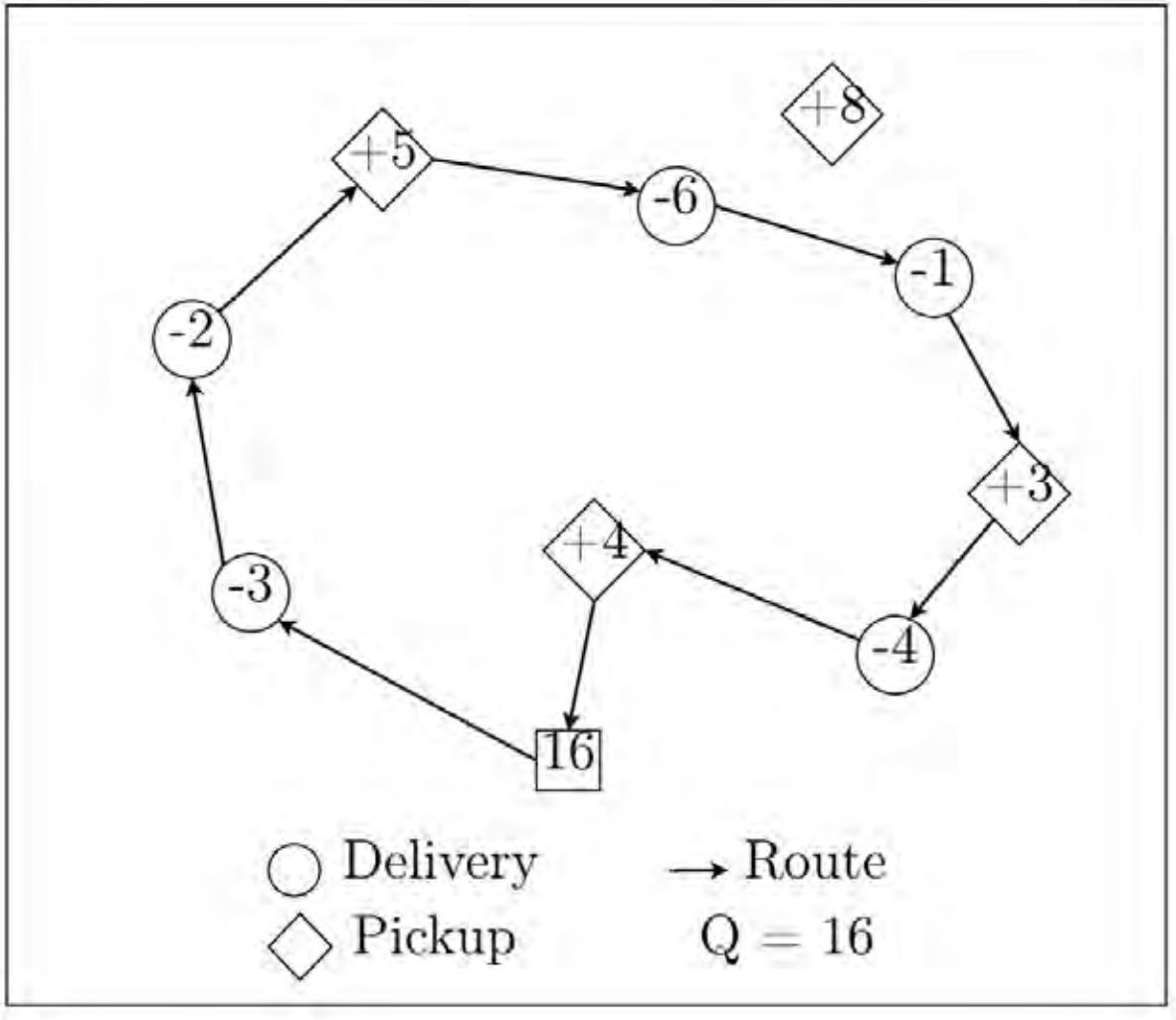
I.M.Coelho, M.N.Haddad, L.S.Ochi, M.J.Souza, R.Farias, *A hybrid CPU-GPU local search heuristic for the unrelated parallel machine scheduling problem*, 3rd Workshop on Applications Multi-Core Architecture (WAMCA), Nova York, 2012

Groups of Instances	Average		
	CPU (s)	Hybrid CPU-GPU (s)	Speedup
100x10	5.31	0.50	10.62
100x15	5.82	0.51	11.41
100x20	6.18	0.53	11.66
100x25	6.78	0.57	11.89
100x30	7.12	0.58	12.28
150x10	16.73	1.00	16.73
150x15	18.13	1.07	16.94
150x20	19.41	1.08	17.97
150x25	20.97	1.15	18.23
150x30	21.88	1.16	18.86
200x10	38.47	1.87	20.57
200x15	41.24	1.93	21.37
200x20	44.11	2.03	21.73
200x25	46.89	2.06	22.76
200x30	49.50	2.13	23.24
250x10	74.71	3.12	23.95
250x15	80.33	3.26	24.64
250x20	85.81	3.36	25.54
250x25	90.71	3.42	26.52
250x30	95.67	3.53	27.10
Average	**38.79**	**1.74**	**22.29**

Problema de Roteamento de Veículos com Entregas Obrigatórias e Coletas Seletivas

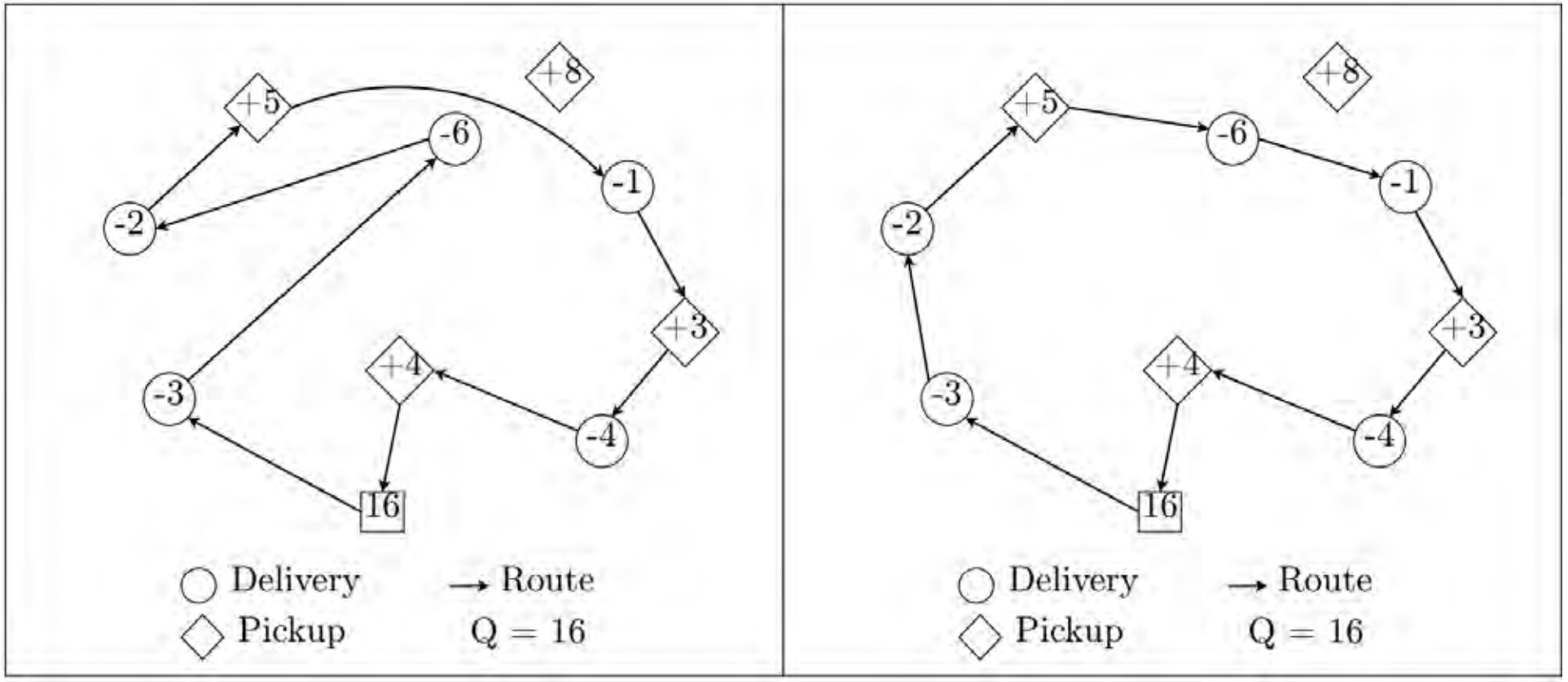
- Veículo de capacidade limitada
- Clientes de entrega obrigatória
- Clientes de coleta com benefício associado
- Matriz de distâncias
- Objetivo: minimizar a distância total percorrida, menos o total em benefícios adquiridos
- Aplicações: correios, reciclagem de garrafas de bebidas, etc.

Problema de Roteamento de Veículos com Entregas Obrigatórias e Coletas Seletivas



Estruturas de Vizinhança: OrOpt-1

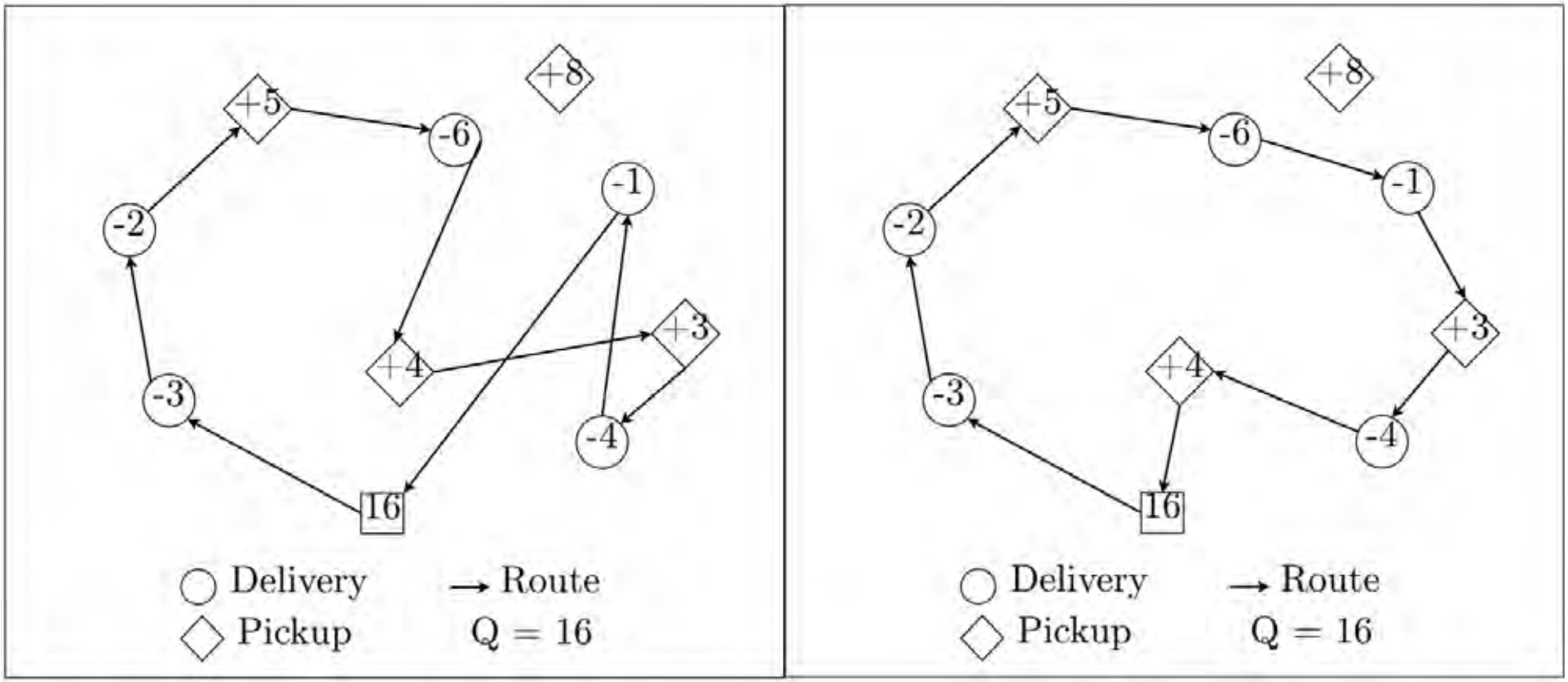
S : [−3, **−6,** −2, +5, −1, +3, −4, +4] → S' : [−3, −2, +5, **−6,** −1, +3, −4, +4]



I.M.Coelho, L.S.Ochi, P.L.A.Munhoz M.J.F.Souza, R.Farias, C.Bentes, *The Single Vehicle Routing Problem with Deliveries and Selective Pickups in a CPU-GPU Heterogeneous Environment*, 9th IEEE HPCC, Liverpool, 2012

Estruturas de Vizinhança: Swap

S : [−3, −2, +5, −6, **+4,** +3, −4, **−1**] → S 0 : [−3, −2, +5, −6, **−1,** +3, −4, **+4**]



I.M.Coelho, L.S.Ochi, P.L.A.Munhoz M.J.F.Souza, R.Farias, C.Bentes, *The Single Vehicle Routing Problem with Deliveries and Selective Pickups in a CPU-GPU Heterogeneous Environment*, 9th IEEE HPCC, Liverpool, 2012

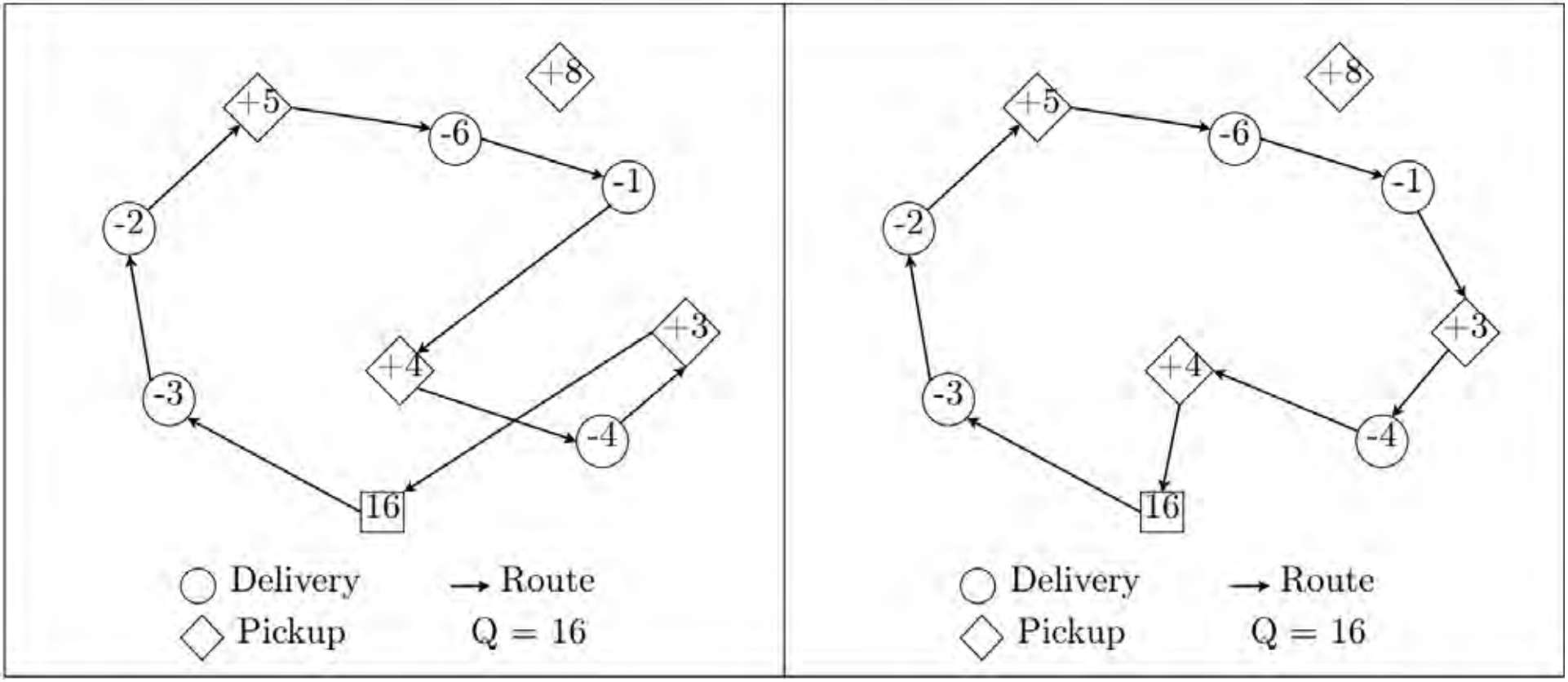
Buscas Locais em GPU

Group of Instances	Speedups of GPU over CPU		
	Swap	*1Or-opt*	*2Or-opt*
small	1.91	5.39	6.81
medium	6.70	18.34	21.43
large	14.36	36.11	43.75
All (Avg)	10.32	27.26	32.88

I.M.Coelho, L.S.Ochi, P.L.A.Munhoz M.J.F.Souza, R.Farias, C.Bentes, *The Single Vehicle Routing Problem with Deliveries and Selective Pickups in a CPU-GPU Heterogeneous Environment*, 9th IEEE HPCC, Liverpool, 2012

Estruturas de Vizinhança: 2-Opt

S : [−3, −2, +5, −6, −1, ***+4, −4, +3***] → S' : [−3, −2, +5, −6, −1, ***+3, −4, +4***]



I.M.Coelho, P.L.A.Munhoz, L.S.Ochi, M.J.F.Souza, C.Bentes, R.Farias, *An integrated CPU–GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups*, International Journal of Production Research, 2015

Buscas Locais em GPU

Grupo de Problemas	Número de Problemas	*Swap*		*2-Opt*		*1Or-opt*		*2Or-opt*	
		méd	máx	méd	máx	méd	máx	méd	máx
small	28	1.81	3.33	2.61	4.73	3.61	6.21	3.16	6.14
medium	28	7.63	16.41	10.36	20.49	12.81	24.69	11.28	21.00
large	12	17.42	19.21	24.47	29.45	26.50	28.97	24.49	26.42
huge	32	55.56	76.84	45.74	52.83	58.36	71.95	54.23	64.79
Todas	**100**	**22.51**	**76.84**	**21.21**	**52.83**	**26.45**	**71.95**	**24.33**	**64.79**

I.M.Coelho, P.L.A.Munhoz, L.S.Ochi, M.J.F.Souza, C.Bentes, R.Farias, *An integrated CPU–GPU heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups*, International Journal of Production Research, 2015

Algoritmo VNS integrado CPU-GPU

- Baixa transferência de dados entre CPU/GPU
- Solução inicial híbrida gerada na CPU via programação matemática
- Cópia da solução para a GPU
- Pertubar solução diretamente na GPU
- Efetuar busca local na GPU e retorna vetor com custo de cada movimento
- CPU escolhe melhor movimento e atualiza solução da GPU

Algoritmo VNS integrado CPU-GPU

- Aceleração máxima na busca Swap: 76x.
- Aceleração geral de 0,93x a 14x.

- Desafio: trabalhar com grandes porções de informação

- Utilizar tipo de memória correto (global, shared, const, texture)

- Número limitado de registradores

- Evitar divergência

Divergência

```
__global__ void kernel1(int* result)
{
    int i = threadIdx.x;
    int j = threadIdx.y;

    if (i < j)
        result[0] = 1000;
    else
        result[0] = 2000;
}
```

Divergência

```
__global__ void kernel1(int* result)
{
    int i = threadIdx.x;
    int j = threadIdx.y;

    if (i < j)
        result[0] = 1000;
    else
        result[0] = 2000;
}

__global__ void kernel2(int* result)
{
    int i = threadIdx.x;
    int j = threadIdx.y;

    result[0] = (i < j) * 1000 + (i >= j) * 2000;
}
```

Conclusões

- CUDA é uma linguagem relativamente fácil de se utilizar (extensão de C/C++)
- Maior esforço na escolha do tipo correto de memória e boa alocação de registradores
- Estudo das opções para lançamento de kernel
- Altos ganhos de velocidade
- **Muito interessante** aprender e explorar uma nova arquitetura de computadores!

Quem tiver interesse neste ou em outros problemas de Otimização Combinatória e/ou queira fazer uma Pós-Graduação (Mestrado ou Doutorado ou um Pós-Doutorado) no IC-UFF (Niterói/RJ)

Contatos:
Luiz Satoru Ochi, Igor Machado Coelho, Puca H. V. Penna
http://www2.ic.uff.br/~satoru/

e-mail:
satoru@ic.uff.br, ou, luiz.satoru@gmail.com
Igor Machado <igor.machado@gmail.com>, ou <igor.machado@ime.uerj.br>
Puca Huachi V. Penna pucahuachi@gmail.com, ou <ppenna@ic.uff.br>

SOBRAPO
Sociedade Brasileira de Pesquisa Operacional

Universidade Federal Fluminense

Instituto de Computação

UERJ
Universidade do Estado do Rio de Janeiro

Obrigado!!


Parque da Cidade – Niterói/RJ

Vista da trilha do Costão de Itacoatiara – Niterói/RJ