CAP 254

Otimização Combinatória

Professor: **Dr. L.A.N. Lorena**

Assunto: Metaheurísticas

Antonio Augusto Chaves

Conteúdo

C01 – Simulated Annealing (20/11/07).

C02 – Busca Tabu (22/11/07).

C03 – Colônia de Formigas (27/11/07).

C04 - GRASP e VNS (29/11/07).

C05 – Metaheurísticas Híbridas – CS (04/12/07).

Metaheurísticas Híbridas

Introdução

- Metaheurísticas têm provado ser uma ferramenta poderosa para resolver de forma aproximada problemas de otimização combinatória na prática.

- O termo *metaheurística* se refere a uma larga classe de algoritmos para otimização e resolução de problemas.

- Existem várias definições para metaheurística:
  - *A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method. [Voß et al., 1999]*

  - *. . . these methods have over time also come to include any procedure for problem solving that employs a strategy for overcoming the trap of local optimality in complex solution spaces, especially those procedures that utilize one or more neighborhood structures as a means of defining admissible moves to transition from one solution to another, or to build or destroy solutions in constructive and destructive processes. [Glover e Kochenberger, 2003]*

Introdução

- Simulated annealing, tabu search, algoritmos evolutivos (algoritmo genético, estratégia evolutiva, etc), ant colony optimization, scatter search, path relinking, greedy randomized adaptive search procedure (GRASP), multi-start and iterated local search, variable neighborhood search (VNS) são geralmente listados como exemplos de metaheurísticas clássicas, possuindo seus próprios históricos e seguindo diferentes paradigmas e filosofias.

- Especialmente nos últimos anos um grande número de algoritmos reportados na literatura não seguem puramente os conceitos de uma única metaheurística clássica, mas combinam várias idéias, algumas vezes até fora do campo das metaheurísticas. Essas abordagens são tradicionalmente referenciadas como ***metaheurísticas híbridas***.

Motivação

- A motivação por trás da hibridização de diferentes metaheurísticas normalmente é obter sistemas com performance melhor que explorem e unam vantagens das estratégias puras aplicadas individualmente.

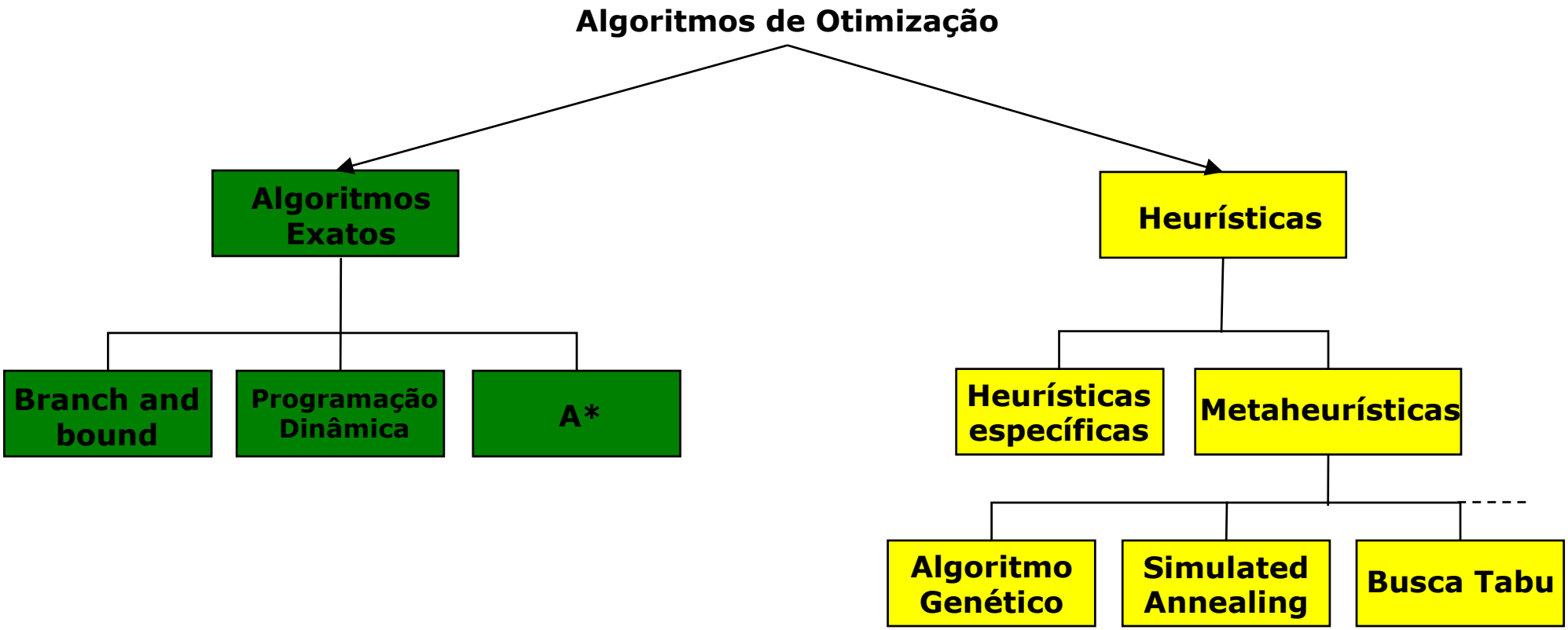
- Prova do sucesso das metaheurísticas híbridas são as inúmeras aplicações encontradas na literatura e também vários eventos específicos na área:
  - International Workshop on Hybrid Metaheuristics
  - Hybrid Metaheuristics: International Workshop
  - International Workshop on Hybrid Artificial Intelligence Systems

- Qual a melhor metaheurística?

  Não existe até hoje uma estratégia de otimização a qual seja globalmente melhor que todas as outras.

- Na realidade, hoje *parece* que escolher uma abordagem híbrida adequada é determinante para alcançar o melhor desempenho na resolução da maioria dos problemas difíceis.

Algoritmos de Otimização



- Metaheurísticas:
  - Solução única: Simulated Annealing, Busca Tabu (Tabu Search), GRASP, VNS...
  - População: Algoritmos Evolutivos, Scartter Search, Colônia de Formigas

Metaheurísticas Híbridas

- *O que* hibridizar, ou seja, quais tipos de algoritmos?

- Podemos combinar:
  (a) Diferentes metaheurísticas
  (b) Metaheurísticas com algoritmos específicos para o problema em foco
  (c) Metaheurísticas com outras técnicas de Pesquisa Operacional e/ou Inteligência Artificial

- As metaheurísticas híbridas podem ser classificadas de acordo com algumas características [Raidl, 2006]

- Nível de hibridização:
  - **High-level**: mantém a identidade individual dos algoritmos originais que cooperam por meio de uma interface bem definida;
  - **Low-level**: algoritmos dependem fortemente um do outro – os componentes individuais dos algoritmos são trocados;

Metaheurísticas Híbridas

- Ordem de execução:
  - **Seqüencial**: um algoritmo é executado depois o outro, e só é passada informação em uma direção;
  - **Intercalado**: a cada iteração de um algoritmo, o outro é executado;
  - **Paralelo**: os algoritmos são executados em paralelo e a informação pode ser passada em qualquer direção.

- Estratégia de Controle:
  - **Integrada**: um algoritmo é considerado um subordinado, ou seja, componente embutido de um outro algoritmo;
  - **Colaborada**: algoritmos trocam informação, mas não são partes um do outro.

- Importante: Quando aplicar um método de intensificação no processo de busca?

Clustering Search (CS)

Clustering Search (CS)

- Evolutionary Clustering Search (ECS)

  Proposto por **A.C.M. Oliveira** e **L.A.N. Lorena** em 2004

- Principais Referências:

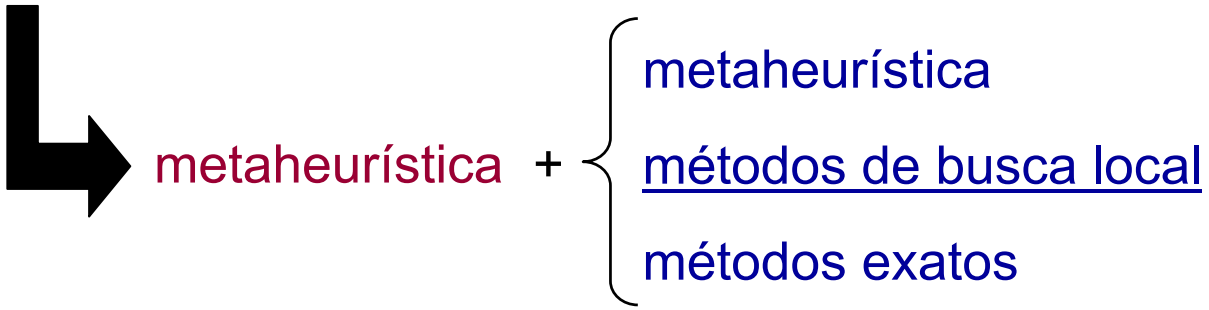
A. C. M. Oliveira and L. A. N. Lorena. Detecting promising areas by evolutionary clustering search. *Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series*, p. 385–394 (2004).

A. C. M. Oliveira. Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos. Tese de Doutorado, INPE (2004).

A.C.M. Oliveira and L.A.N. Lorena: Hybrid evolutionary algorithms and clustering search. In Grosan, C., Abraham, A., Ishibuchi, H., eds.: *Hybrid Evolutionary Systems - Studies in Computational Intelligence. Springer SCI Series* (2007) 81–102

Clustering Search (CS)

- Metaheurística Híbrida



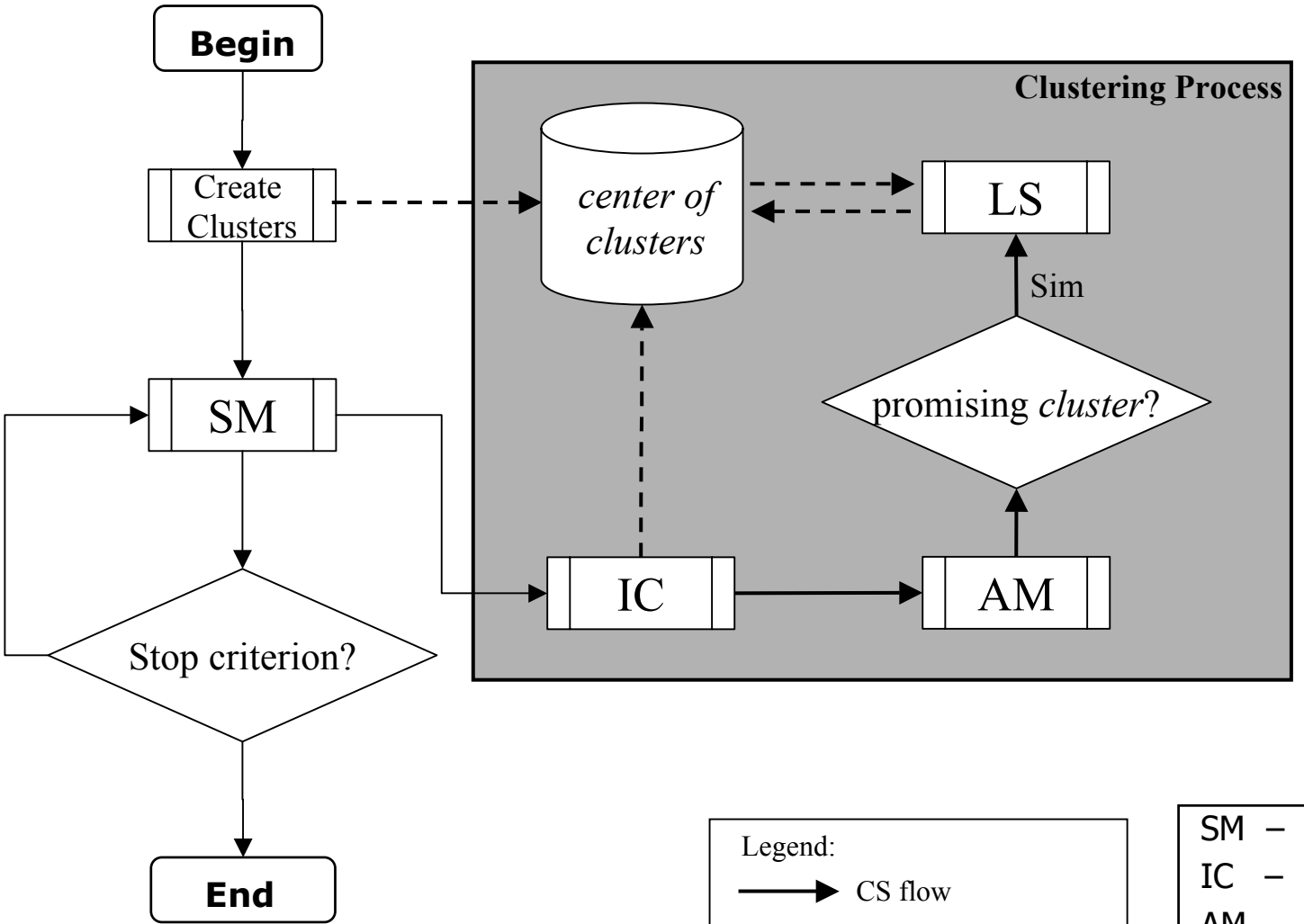
- **Diferencial do CS**:

  aplicar busca local somente em regiões supostamente promissoras, detectadas por meio de um processo de agrupamento de soluções.

- **Objetivo do CS**:

  melhorar o processo de convergência associado a uma diminuição no esforço computacional em virtude do emprego mais racional dos métodos busca local.

Clustering Search (CS)



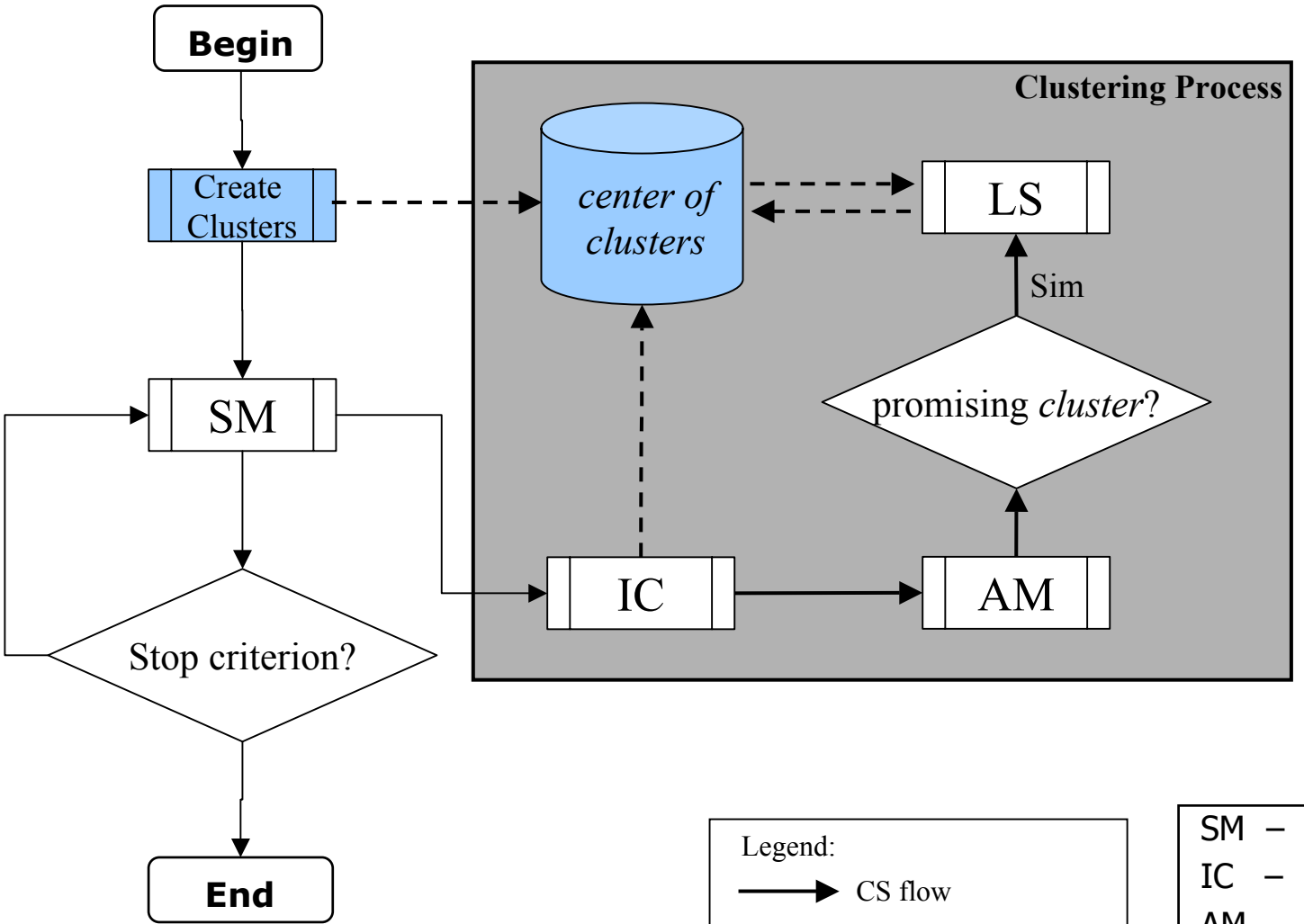
Legend:

CS flow

Clusters update flow

SM – search metaheuristic
IC – iterative clustering
AM – analyser module
LS – local search

Clustering Search (CS)



Legend:

CS flow

Clusters update flow

SM – search metaheuristic
IC – iterative clustering
AM – analyser module
LS – local search

Clustering Search (CS)

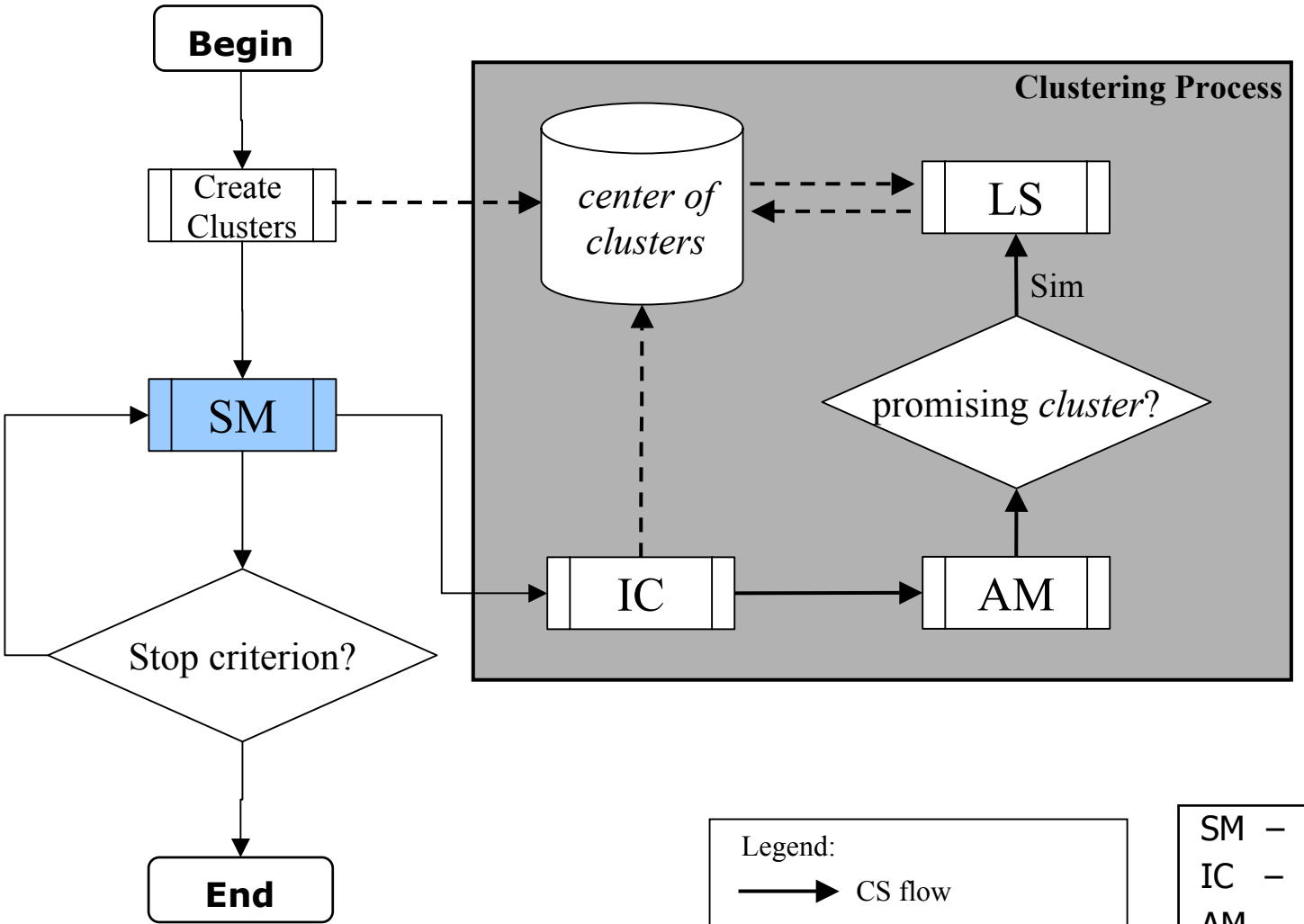
- Criar Clusters:
  - Aleatoriamente
  - Problema da Diversidade Máxima

    Dado um conjunto N com n elementos, selecionar um subconjunto M $\subset$ N de forma tal que os elementos de M possuam a maior diversidade possível entre eles

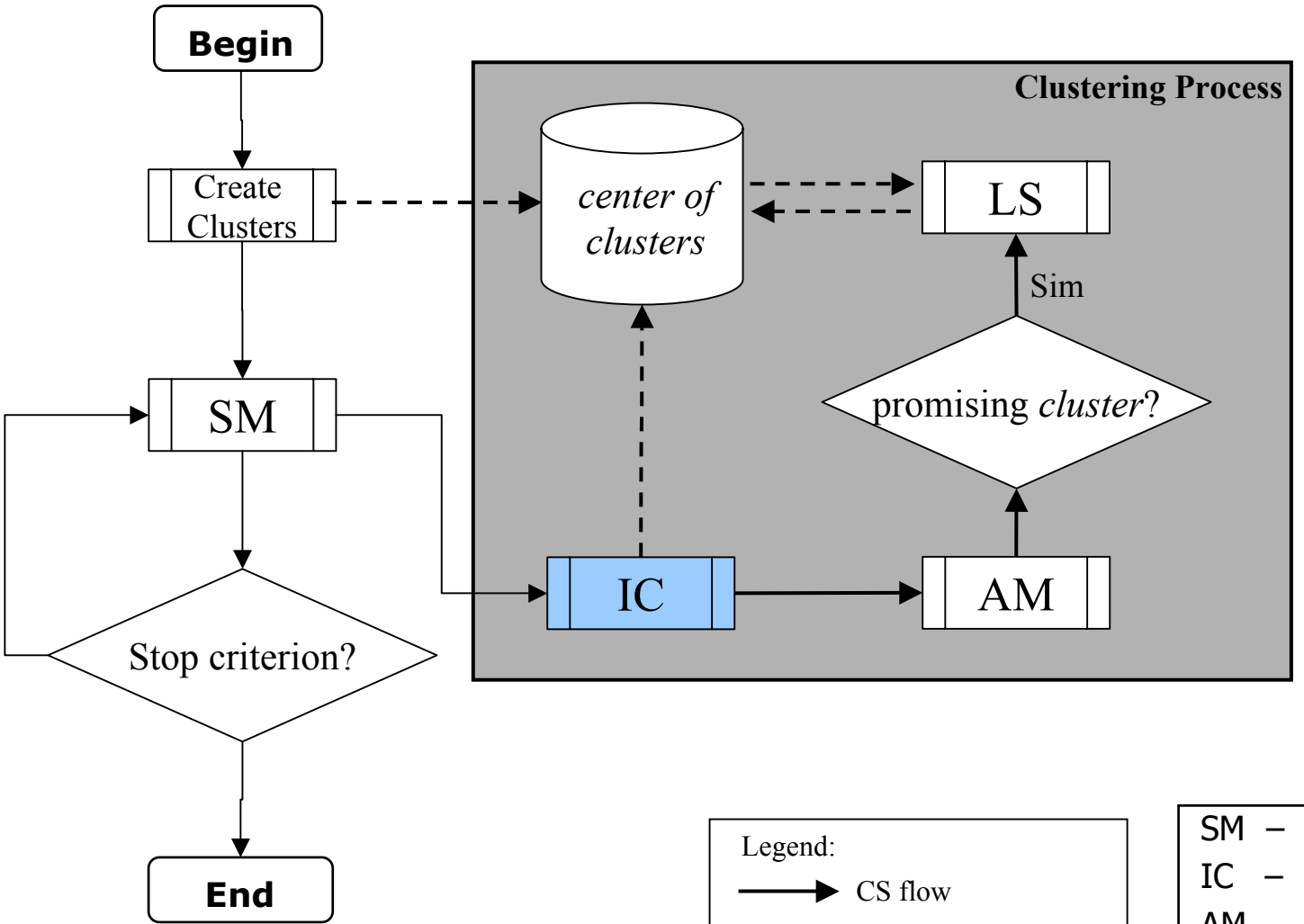
    1) Gerar um número grande de soluções iniciais aleatoriamente

    2) Determinar quais serão os $\mathcal{NC}$ centros de clusters do CS

Clustering Search (CS)

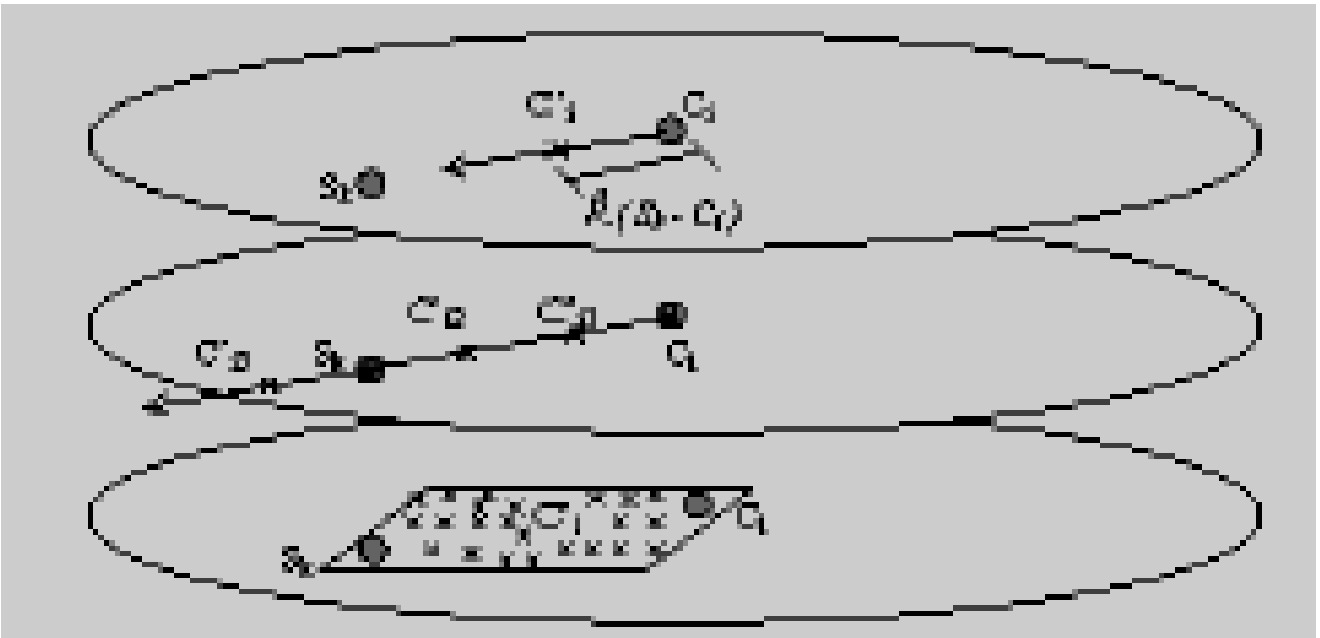


Clustering Search (CS)

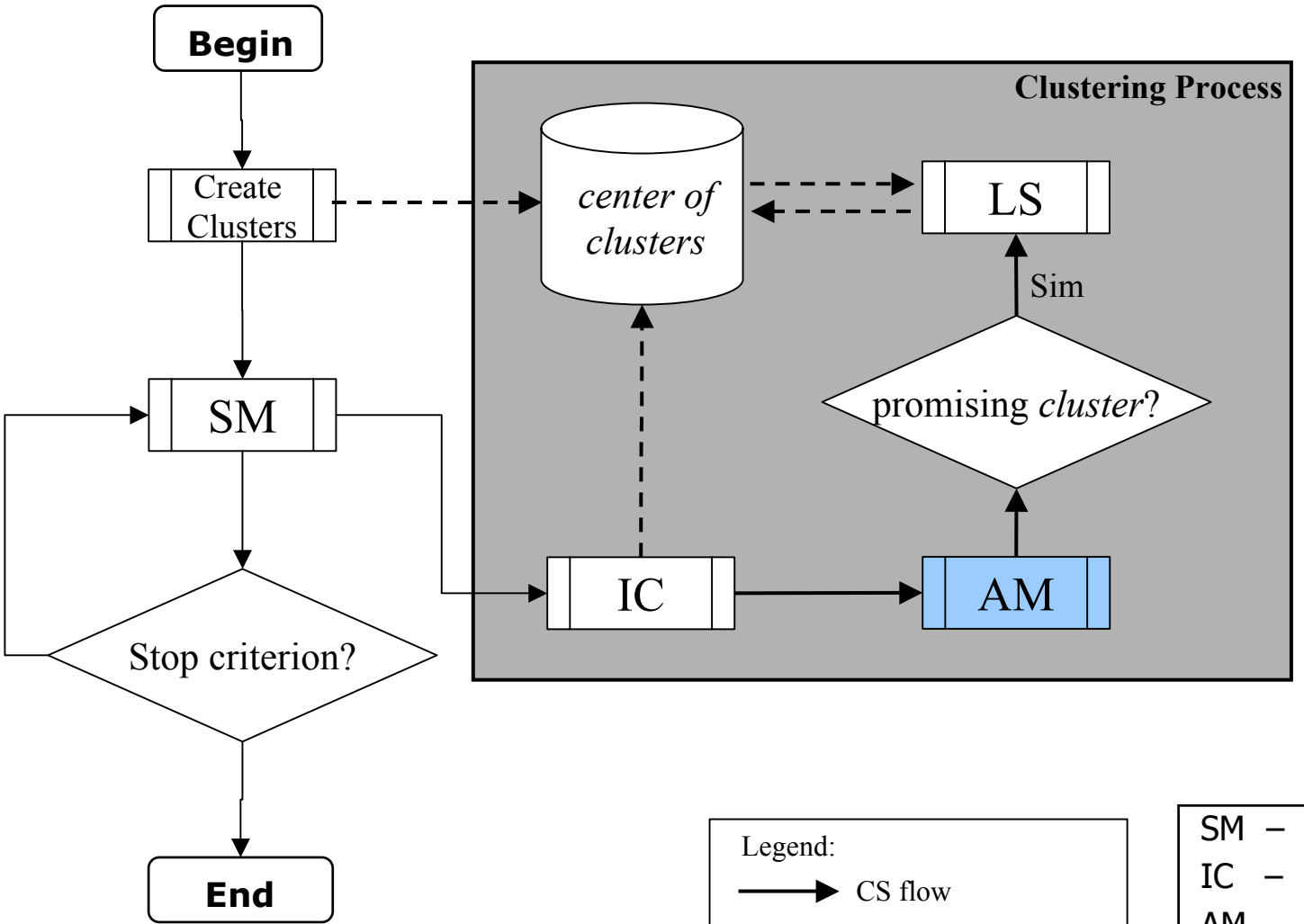


Clustering Search (CS)

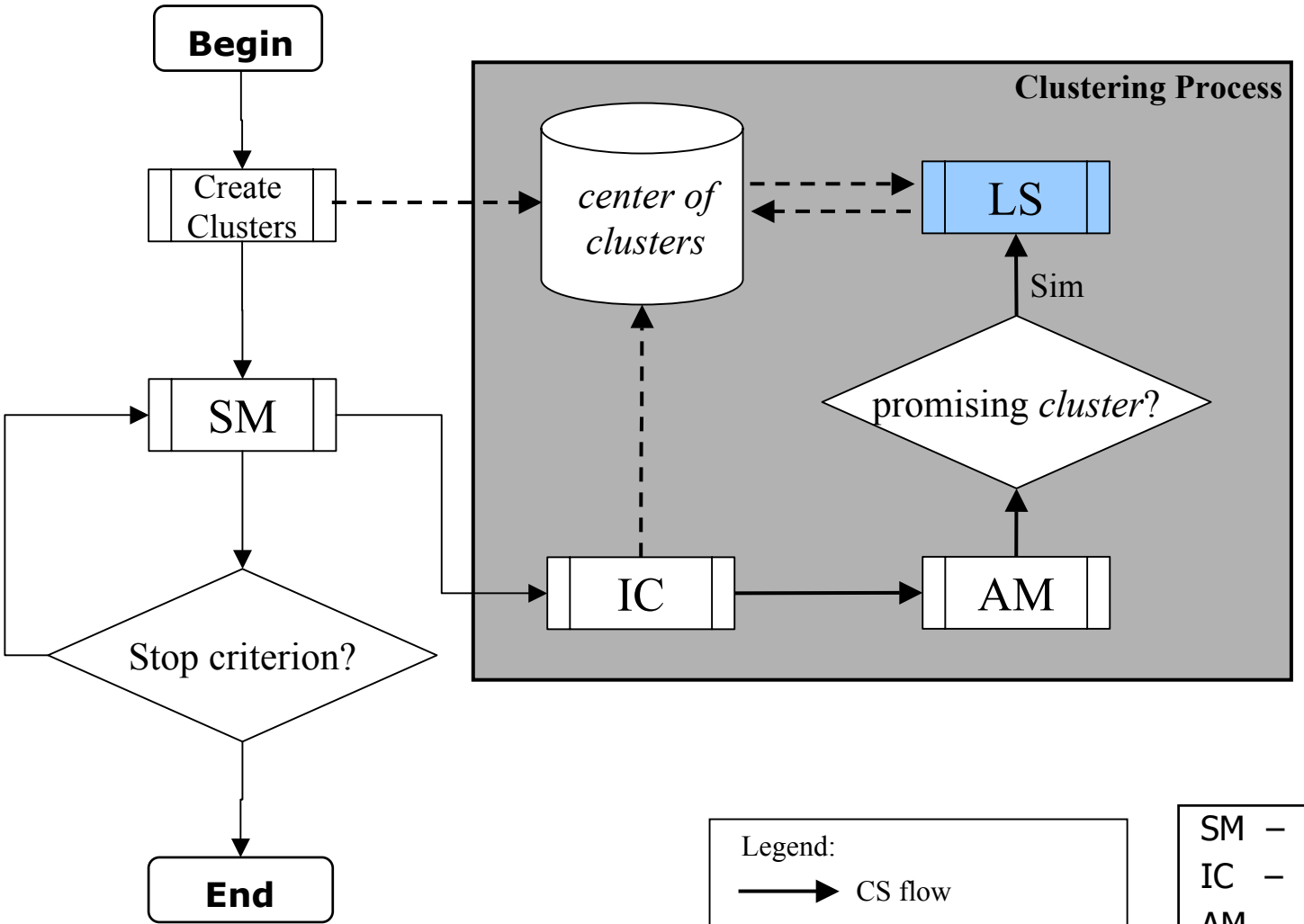
- Processo de Assimilação:
  - Assimilação Simples
  - Assimilação por Caminho
  - Assimilação por Recombinação



Clustering Search (CS)



Clustering Search (CS)



Clustering Search (CS)

procedimento CS
Criar os *clusters* iniciais aleatoriamente
//componente SM
enquanto (critério de parada não satisfeito) **faça**
gerar uma solução através da metaheurística escolhida
//componente IC
calcular a distância da solução para cada um dos *clusters*
se (solução NÃO for similar a algum *cluster* e número de *clusters* < $\mathcal{NC}$) **faça**
gerar um novo *cluster*
atribuir a solução ao centro do novo *cluster*
senão
inserir a solução no *cluster* similar mais recente
causar uma perturbação no centro do *cluster* ativado (assimilação)
//componente AM
analisar a densidade dos *clusters*, descobrindo possíveis *clusters* promissores
se (existir *clusters* que não estejam sendo ativados) **faça**
eliminar *clusters* não ativos
//componente LS
aplicar um método de busca local nos centros dos *clusters* promissores
fim-enquanto
fim-procedimento

Aplicações do CS

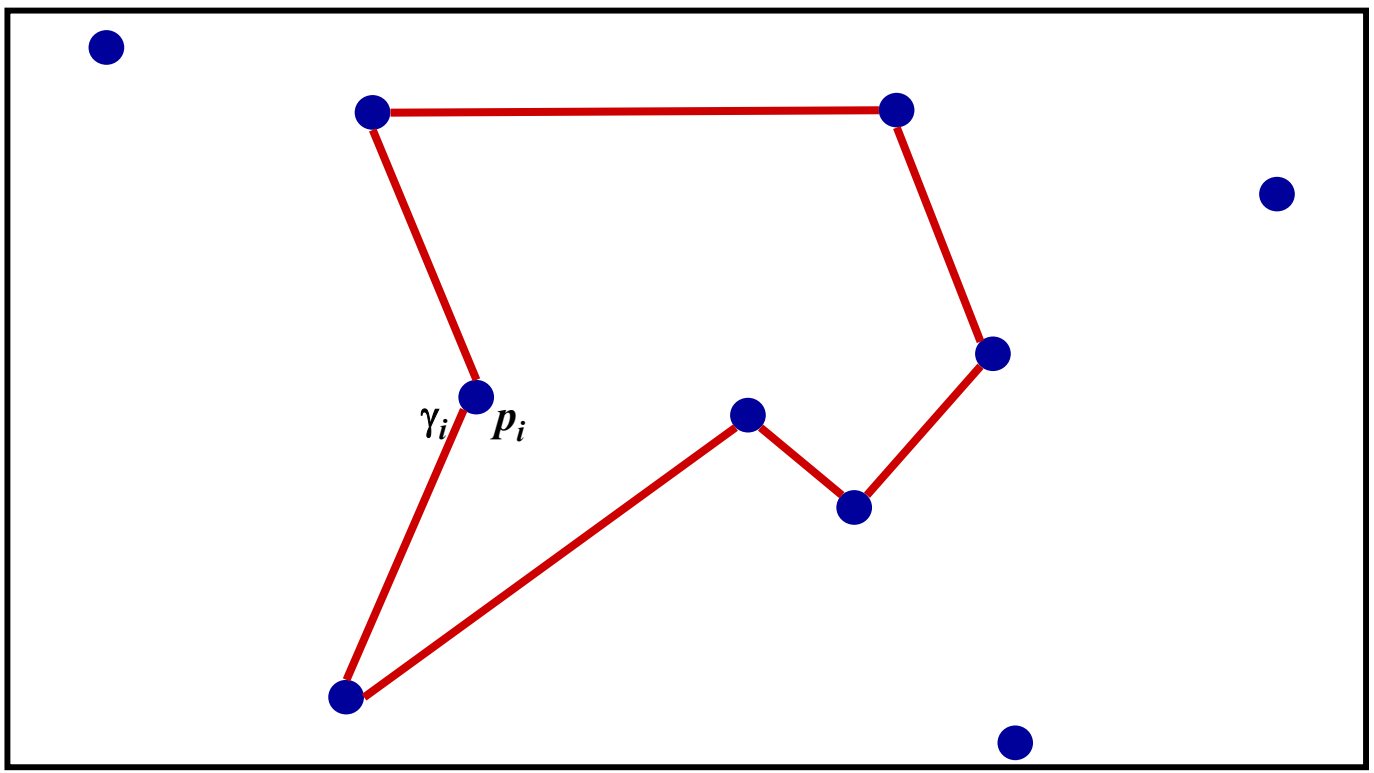
TSPP

- **Problema do Caixeiro Viajante (Traveling Salesman Problem – TSP)**
  - otimizar a seqüência de visitas a clientes
  - todos clientes precisam ser visitados
- **Problemas do Caixeiro Viajante com Lucros (Traveling Salesman Problem with Profits – TSPP)**
  - para cada vértice existe associado um **prêmio** quando a visita acontecer
  - cada aresta possui um **custo de deslocamento**
  - **objetivo**: determinar um circuito elementar, levando em consideração o **prêmio coletado** e os **custos de deslocamento**
  - os diferentes problemas que formam os TSPP surgem das diferentes maneiras que existem para tratar os dois objetivos.
    - ✓ tratar separadamente
    - ✓ combinar na função objetivo
    - ✓ definir o custo de deslocamento como uma restrição
    - ✓ definir prêmio como uma restrição

TSPP

- PCV com Coleta de Prêmios (Prize-Collecting TSP – PCTSP)

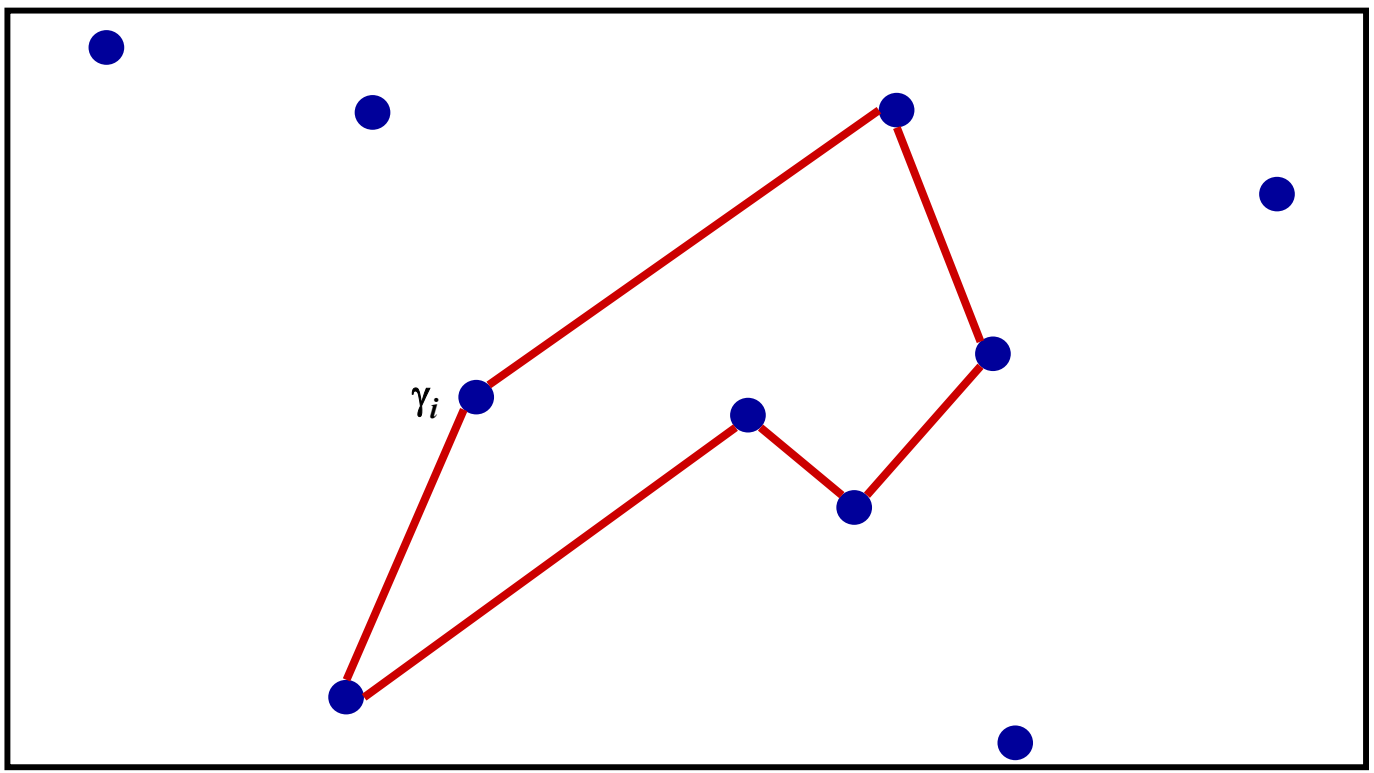
$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j} + \sum_{v_i \in V} \gamma_{v_i} (1 - y_{v_i}), \text{ sujeito à } \sum_{v_i \in V} p_{v_i} y_{v_i} \geq p_{\min}$$



TSPP

- Problema de Rotas com Lucro (Profitable Tour Problem – PTP)

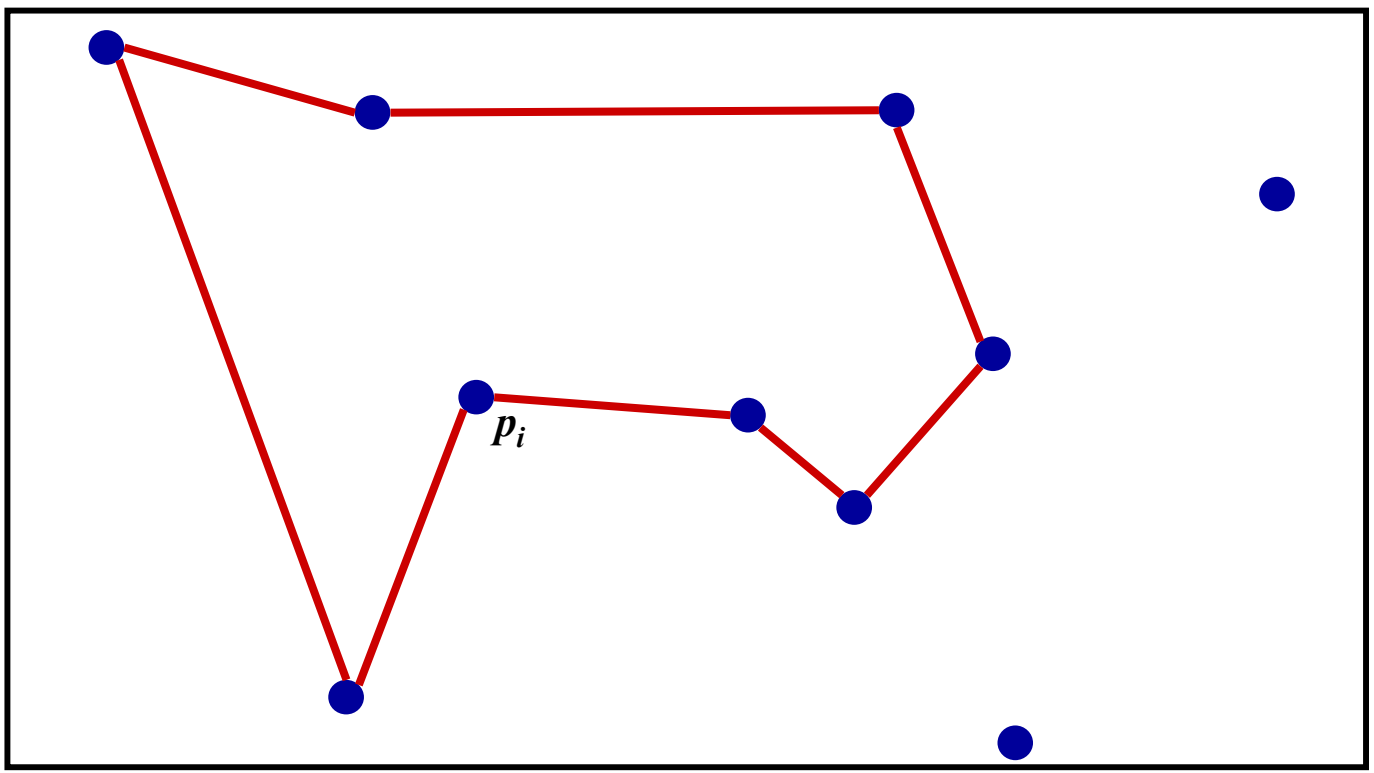
$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j} + \sum_{v_i \in V} \gamma_{v_i} (1 - y_{v_i})$$



TSPP

- Problema do Caixeiro Viajante com Quota (Quota TSP – QTSP)

$$\min \sum_{v_i \in V} \sum_{v_j \in V} c_{v_i v_j} x_{v_i v_j}, \text{ sujeito à } \sum_{v_i \in V} p_{v_i} y_{v_i} \geq p_{\min}$$



CS aplicado aos TSPP – componente SM

procedimento GRASP/VNS
  para (número de iterações não satisfeito) **faça**

Fase de Construção:

  $s = \varnothing$
  enquanto (solução não construída) **faça**
    produzir Lista de Candidatos (C) ⟶ $v(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$
    LCR = C * α
    e = selecionar elemento aleatoriamente(LCR)
    $s = s \cup \{e\}$
    atualizar lista de Candidatos(C)
  fim-enquanto

Fase de Busca Local (VNS):

  k_{max} = número de vizinhanças
  enquanto (critério de parada não satisfeito) **faça**
    $k \leftarrow 1$
    enquanto ($k \leq k_{max}$)
      gerar um vizinho s‘ qualquer na vizinhança $N^k(s)$
      s'' = Aplicar VND em s'
      se (s'' for melhor que s) **faça**
        $s \leftarrow s$''
        $k \leftarrow 1$
      senão
        $k \leftarrow k + 1$
    fim-enquanto
  fim-enquanto
  fim-para
fim-GRASP/VNS

CS aplicado aos TSPP – componente SM

procedimento GRASP/VNS

para (número de iterações não satisfeito) **faça**

Fase de Construção

$s = \varnothing$

enquanto (solução não construída) **faça**

produzir Lista de Candidatos (C) $\longrightarrow v(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$

LCR = C * α

e = selecionar elemento aleatoriamente(LCR)

$s = s \cup \{e\}$

atualizar lista de Candidatos(C)

fim-enquanto

Fase de Busca Local (VNS)

k_{max} = número de vizinhanças

enquanto (critério de parada não satisfeito) **faça**

$k \leftarrow 1$

enquanto ($k \leq k_{max}$)

gerar um vizinho *s*‘ qualquer na vizinhança $N^k(s)$

s'' = Aplicar VND em *s*'

se (*s*'' for melhor que *s*) **faça**

$s \leftarrow s''$

$k \leftarrow 1$

senão

$k \leftarrow k + 1$

fim-enquanto

fim-enquanto

fim-para

fim-GRASP/VNS

1. **Inserir um vértice**
2. **Retirar um vértice**
3. **Trocar dois vértices**
4. **Inserir dois vértice**
5. **Retirar dois vértice**
6. **Trocar quatro vértices**
7. **Inserir três vértice**
8. **Retirar três vértice**
9. **Trocar seis vértices**

CS aplicado aos TSPP – componente SM

procedimento GRASP/VNS
  para (número de iterações não satisfeito) **faça**

Fase de Construção:
  $s = \varnothing$
  enquanto (solução não construída) **faça**
    produzir Lista de Candidatos (C) $\longrightarrow v(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$
    LCR = C * α
    e = selecionar elemento aleatoriamente(LCR)
    $s = s \cup \{e\}$
    atualizar lista de Candidatos(C)
  fim-enquanto

Fase de Busca Local (VNS):
  k_{max} = número de vizinhanças
  enquanto (critério de parada não satisfeito) **faça**
    $k \leftarrow 1$
    enquanto ($k \leq k_{max}$)
      gerar um vizinho s‘ qualquer na vizinhança $N^k(s)$
      s'' = Aplicar VND em s'
      se (s'' for melhor que s) **faça**
        $s \leftarrow s$''
        $k \leftarrow 1$
      senão
        $k \leftarrow k + 1$
    fim-enquanto
  fim-enquanto
  fim-para
fim-GRASP/VNS

1. **SeqAdd**
2. **AddDrop**
3. **SeqDrop**

CS aplicado aos TSPP – componente SM

procedimento GRASP/VNS

para (número de iterações não satisfeito) **faça**

Fase de Construção

$s = \varnothing$

enquanto (solução não construída) **faça**

produzir Lista de Candidatos (C) $\longrightarrow v(k) = c_{ij} + \gamma_k - c_{ik} - c_{kj}$

LCR = C * α

e = selecionar elemento aleatoriamente(LCR)

$s = s \cup \{e\}$

atualizar lista de Candidatos(C)

fim-enquanto

Fase de Busca Local (VNS)

k_{max} = número de vizinhanças

enquanto (critério de parada não satisfeito) **faça**

$k \leftarrow 1$

enquanto ($k \leq k_{max}$)

gerar um vizinho *s*‘ qualquer na vizinhança $N^k(s)$

s'' = Aplicar VND em *s*'

se (*s*'' for melhor que *s*) **faça**

$s \leftarrow s''$

$k \leftarrow 1$

senão

$k \leftarrow k + 1$

fim-enquanto

fim-enquanto

fim-para

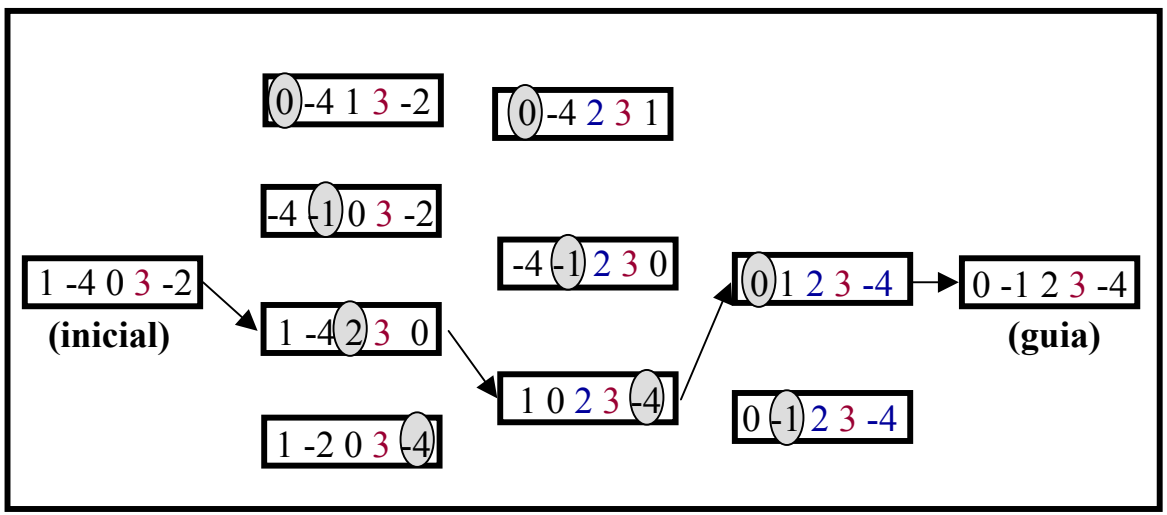
fim-GRASP/VNS

Solução s’’ é enviada para o componente AI

CS aplicado aos TSPP – componente IC

- Medida de distância entre a solução gerada pelo GRASP/VNS e o centro do cluster:
  - **Número de arestas diferentes**

- Processo de assimilação por caminho: **path-relinking**

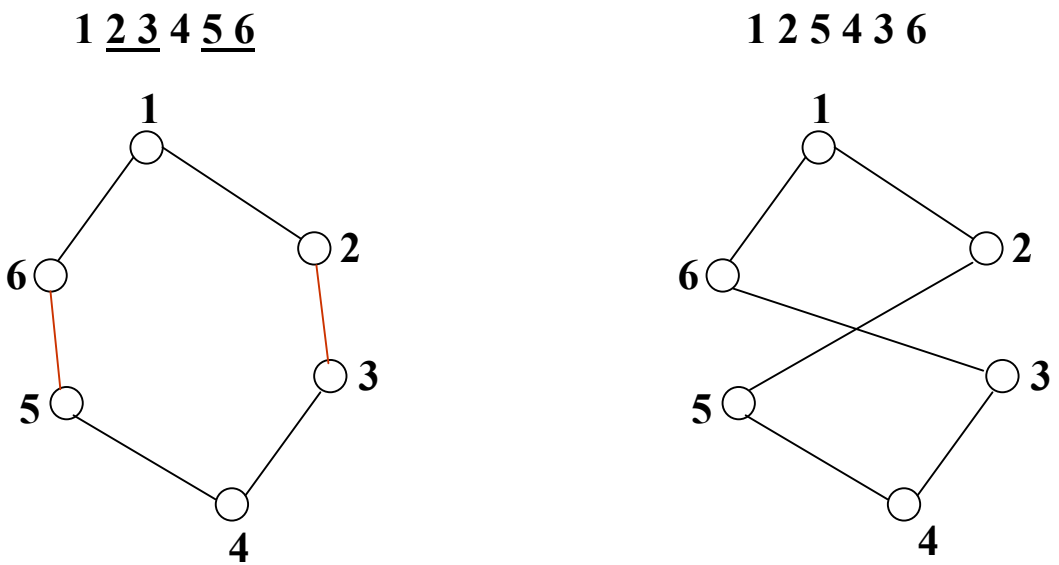


CS aplicado aos TSPP – componente AM

- É executado toda vez que uma solução for atribuída a um cluster
- **Função**: verificar se o cluster já pode ser considerado promissor
- Cluster Populoso: $$\lambda_t \geq PD.\frac{NS}{|C_t|}$$
- Outras funções:
  - Esfriamento de todos os clusters que foram ativados
  - Eliminar os clusters pouco populosos

CS aplicado aos TSPP – componente LS

- Heurística 2-Opt
- **Objetivo**: realizar uma busca local no centro dos clusters considerados promissores



Resultados Computacionais TSPP

- Ambiente computacional
  - Linguagem C++ (C++ Builder 6.0)
  - PC Pentium 4 3.02 GHZ e memória de 512 MB
- Não existe uma biblioteca pública de instâncias para os *TSPP*.
- Instâncias geradas aleatoriamente com distribuição uniforme:
  - distância entre os vértices: $c_{ij} \in [50, 1000]$
  - prêmio associado à cada vértice: $p_i \in [1, 100]$
  - penalidade associada à cada vértice: $\gamma_i \in [1,750]$
  - prêmio mínimo a ser coletado: $0,75 * \sum_{i \in V} p_i$

Resultados Computacionais (PCTSP)

		CPLEX		CS			GRASP/VNS		
Prob.	***n***	**MS**	**TE (s)**	**MS**	**SM**	**TE (s)**	**MS**	**SM**	**TE (s)**
v10	11	1765	0,11	**1765**	**1765**	**0,02**	1765	1765	0,05
v20	21	2302	2,65	**2302**	**2302**	**0,35**	2355	2403	0,27
v30a	31	3582	4,89	**3582**	**3582**	**0,82**	3734	3899	6,00
v30b	31	2515	6,18	**2515**	**2515**	**3,14**	2647	2771	5,93
v30c	31	3236	20,24	**3236**	**3242**	**1,54**	3301	3356	6,04
v50a	51	4328	1032,79	**4328**	**4335**	**9,78**	4752	4900	13,41
v50b	51	3872	634,01	**3872**	**3878**	**6,40**	4097	4383	35,90
v100a	101	**6762**	**86390,66**	6785	6822	108,55	8302	8476	52,69
v100b	101	**6760**	**85002,51**	6782	6844	216,16	7890	8367	87,78
v250a	251	-	-	**14483**	**14483**	**713,12**	18306	18510	364,16
v250b	251	-	-	**14078**	**14078**	**701,58**	17871	18191	393,47
v500a	501	-	-	**27189**	**27230**	**2880,03**	34437	34771	463,02
v500b	501	-	-	**28133**	**28334**	**2080,89**	34841	35296	529,72

Resultados Computacionais (QTSP)

		CPLEX		CS			GRASP/VNS		
Prob.	***n***	**MS**	**TE (s)**	**MS**	**SM**	**TE (s)**	**MS**	**SM**	**TE (s)**
v10	11	1755	0,96	**1755**	**1755**	**0,01**	1755	1755	0,07
v20	21	1439	5,13	**1439**	**1439**	**1,33**	1439	1469	1,24
v30a	31	2074	29,56	**2074**	**2075**	**3,09**	2182	2268	5,65
v30b	31	1552	15,40	**1552**	**1552**	**6,62**	1553	1772	4,01
v30c	31	1966	30,23	**1966**	**2004**	**2,49**	2086	2177	6,20
v50a	51	2438	372,09	**2438**	**2046**	**17,76**	2912	2956	11,31
v50b	51	2192	1231,20	**2192**	**2218**	**35,90**	2548	2600	24,06
v100a	101	**3770**	**46331,41**	3862	3930	84,94	4777	4811	68,15
v100b	101	**3935**	**45615,01**	4030	4092	126,07	5092	5191	44,58
v250a	251	-	-	**8253**	**8342**	**518,82**	10309	10626	295,34
v250b	251	-	-	**8336**	**8508**	**549,76**	10701	10865	313,50
v500a	501	-	-	**15367**	**15780**	**2207,11**	19510	19717	799,72
v500b	501	-	-	**15211**	**15757**	**1526,38**	19494	19567	750,24

Resultados Computacionais (PTP)

		CPLEX		CS			GRASP/VNS		
Prob.	***n***	**MS**	**TE (s)**	**MS**	**SM**	**TE (s)**	**MS**	**SM**	**TE (s)**
v10	11	1558	0,29	**1558**	**1558**	**0,001**	1558	1558	0,00
v20	21	2302	2,22	**2302**	**2302**	**0,94**	2302	2369	2,24
v30a	31	3582	3,94	**3582**	**3582**	**1,89**	3695	3871	9,48
v30b	31	2515	6,08	**2515**	**2515**	**1,92**	2571	2617	10,09
v30c	31	3236	16,30	**3236**	**3242**	**2,22**	3345	3408	6,55
v50a	51	4328	1261,03	**4328**	**4338**	**22,97**	4739	4852	19,33
v50b	51	3872	910,37	**3872**	**3877**	**12,62**	4219	4371	18,51
v100a	101	**6762**	**197783,19**	6784	6799	125,96	8191	8344	69,78
v100b	101	**6760**	**55953,18**	6826	6864	150,05	8212	8342	55,91
v250a	251	-	-	**14493**	**14584**	**1124,62**	17970	18432	733,19
v250b	251	-	-	**14055**	**14119**	**901,94**	17302	17786	637,00
v500a	501	-	-	**27316**	**27564**	**2387,02**	33727	34157	1401,86
v500b	501	-	-	**27890**	**27944**	**2345,02**	34757	35044	1220,97

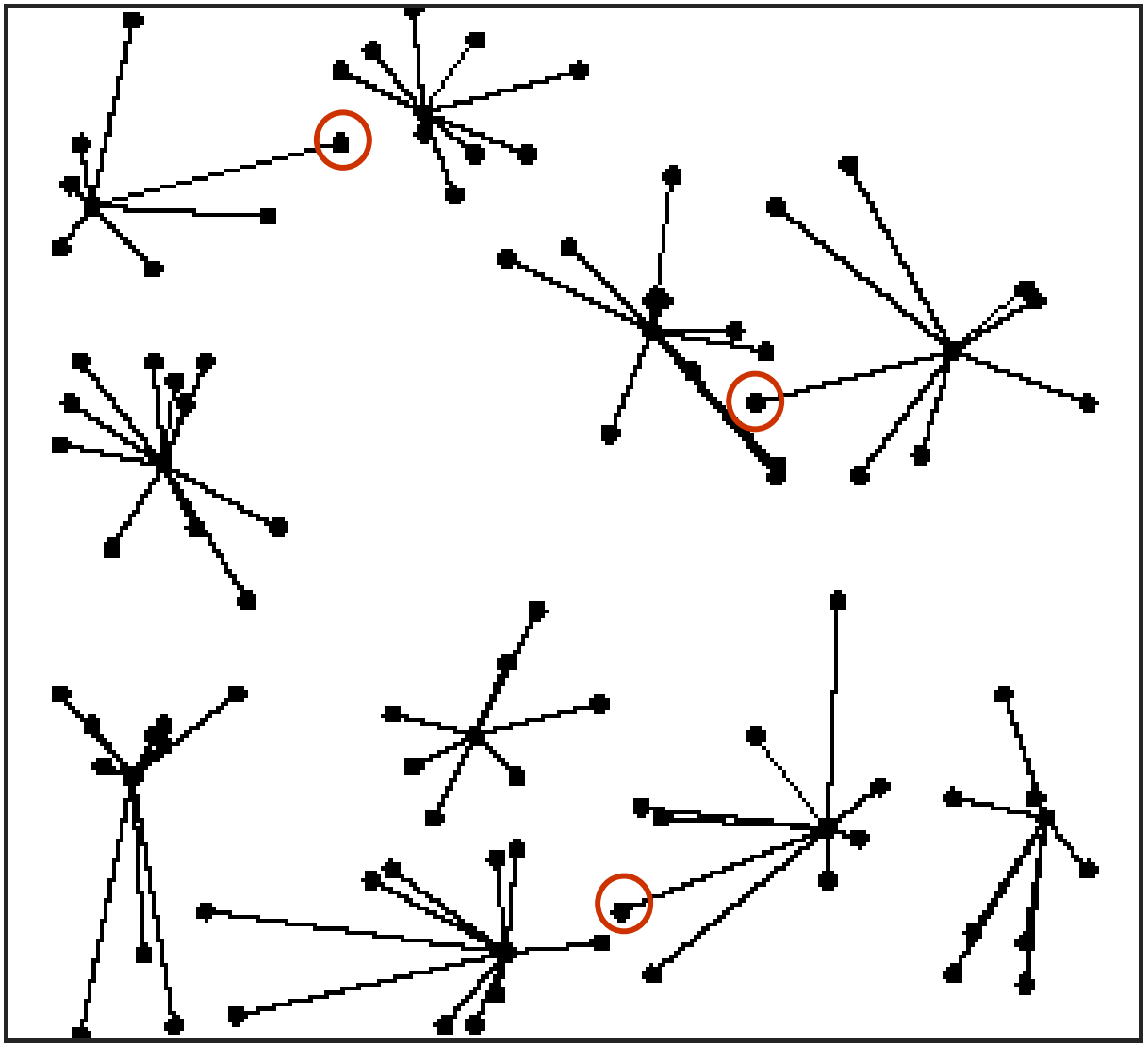
CPMP

- **Problemas de localização de facilidades:**
  - Problemas de cobertura
  - Problemas de localização de medianas

- **Problema de p-Medianas**
  - Localizar *p* medianas em um dado espaço, satisfazendo *n* pontos de demanda de tal forma que a soma total das distâncias entre cada ponto de demanda e sua mediana mais próxima seja minimizada

- **Problema de p-Medianas Capacitado (Capacitated p-Median Problem - CPMP)**
  - Generalização do Problema de p-Mediana
  - Define-se uma **capacidade** fixa para cada mediana candidata e uma **demanda** para cada ponto de atendimento, sendo que, a soma das demandas de todos os pontos alocados a uma mediana não pode exceder sua capacidade
  - Problema NP-difícil

CPMP



EXEMPLO DE UMA SOLUÇÃO DO CPMP

CS aplicado ao CPMP – componente SM

```
procedimento SA
    Criar Solução Inicial(sol)
    IterT = 0
    α = 0,95
    T = T0
    enquanto (T > 0.0001)
        enquanto (IterT < SAmax)
            IterT = IterT + 1
            Gerar um vizinho (sol') aleatoriamente na vizinhança Nk(sol)
            Δ = f(sol') – f(sol)
            se (Δ < 0)
                sol = sol'
            senão
                seja x ∈ [0,1]
                se (x < e-Δ/T)
                    sol = sol'
        fim-enquanto
        T = T x α
        IterT = 0
    fim-enquanto
fim-procedimento
```

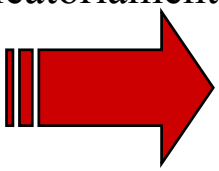
CS aplicado ao CPMP – componente SM

```
procedimento SA
    Criar Solução Inicial(sol)
    IterT = 0
    α = 0,95
    T = T0
    enquanto (T > 0.0001)
        enquanto (IterT < SAmax)
            IterT = IterT + 1
            Gerar um vizinho (sol') aleatoriamente na vizinhança Nk(sol)
            Δ = f(sol') – f(sol)
            se (Δ < 0)
                sol = sol'
            senão
                seja x ∈ [0,1]
                se (x < e-Δ/T)
                    sol = sol'
        fim-enquanto
        T = T x α
        IterT = 0
    fim-enquanto
fim-procedimento
```

1) Trocar a alocação de dois pontos
2) Trocar uma mediana por um ponto atendido por ela
3) Retirar um ponto de uma mediana e adicionar em outra mediana
4) Trocar uma mediana por um ponto qualquer

CS aplicado ao CPMP – componente SM

procedimento SA
  Criar Solução Inicial(sol)
  IterT = 0
  $\alpha = 0{,}95$
  $T = T_0$
  enquanto (T > 0.0001)
    enquanto (IterT < SAmax)
      IterT = IterT + 1
      Gerar um vizinho (sol') aleatoriamente na vizinhança N^k(sol)
      $\Delta = f(sol') - f(sol)$
      se ($\Delta < 0$)
        sol = sol'
      senão
        seja x $\in$ [0,1]
        se ($x < e^{-\Delta/T}$)
          sol = sol'
    fim-enquanto
    T = T x α
    IterT = 0
  fim-enquanto
fim-procedimento

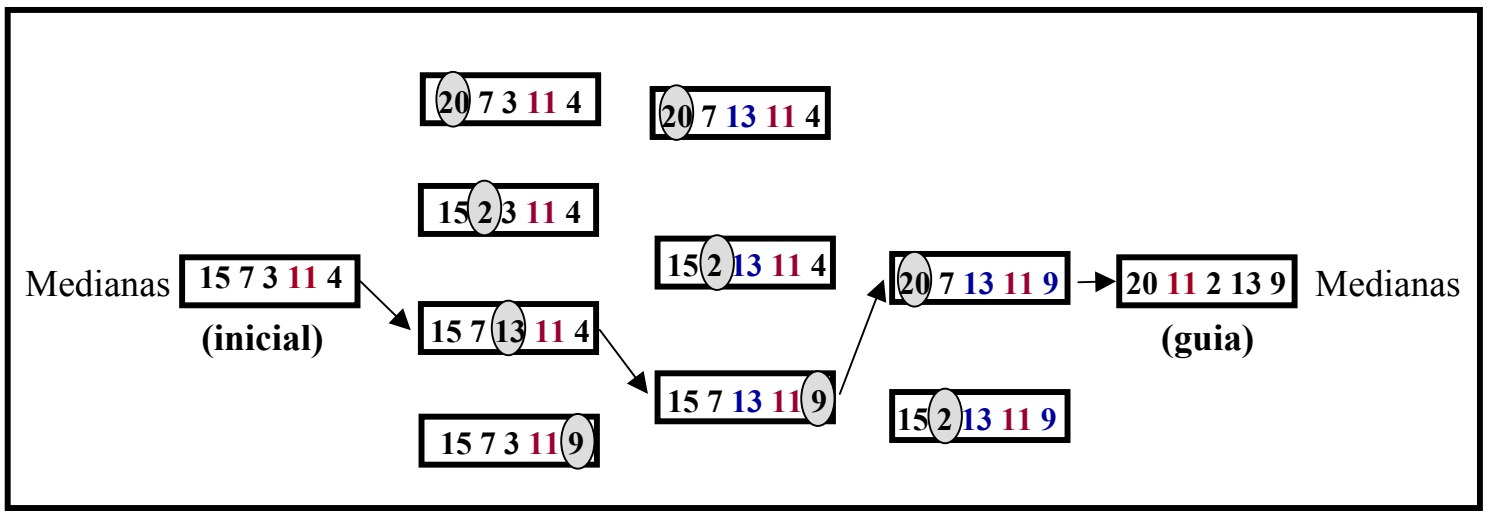


A cada 100 vizinhos gerados, a solução sol' é enviada para o componente AI

CS aplicado aos CPMP – componente IC

- Medida de distância entre a solução gerada pelo SA e o centro do cluster:
  - **número de pontos atribuídos para medianas diferentes**

- Processo de assimilação por caminho: **path-relinking**



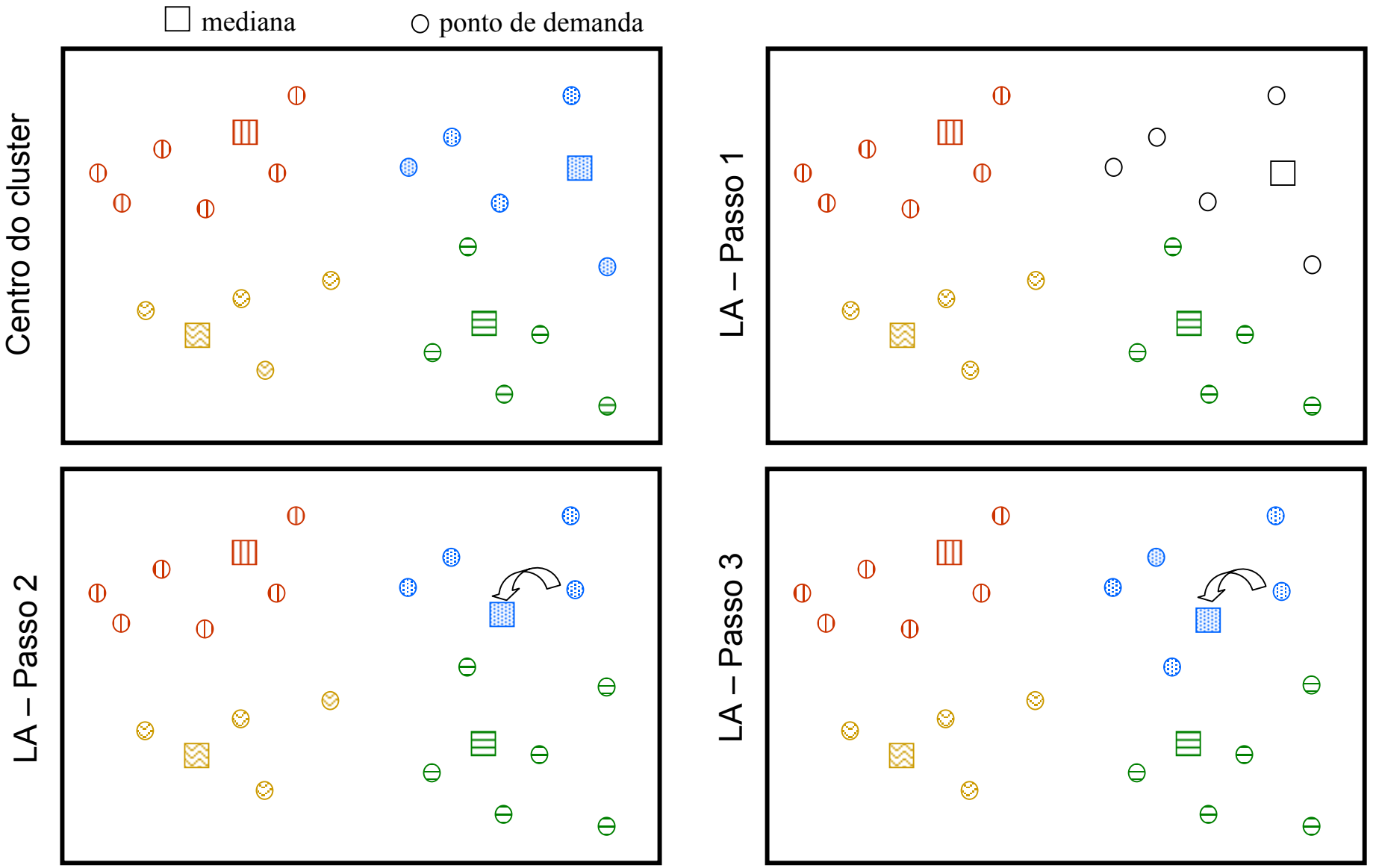
CS aplicado aos CPMP – componente AA

- É executado toda vez que uma solução for atribuída a um cluster
- **Função**: verificar se o cluster já pode ser considerado promissor
- Cluster Populoso: $\lambda_t \geq PD.\frac{NS}{|C_t|}$
- Outras funções:
  - Esfriamento de todos os clusters que foram ativados
  - Eliminar os clusters pouco populosos

CS aplicado aos CPMP – componente AO

- Heurística de Localização-Alocação (LA)
- **Objetivo**: realizar uma busca local no centro dos clusters considerados promissores

CS aplicado aos CPMP – componente AO



Resultados Computacionais CPMP

- Ambiente computacional
  - Linguagem C++ (C++ Builder 6.0)
  - PC Pentium 4 3.02 GHZ e memória de 512 MB

- Foram utilizados dois conjuntos de instâncias
  - Osman e Christofides: 10 problemas (n = 50 e p = 5)
    10 problemas (n = 100 e p = 10)

  - Lorena e Senne: problemas com dados reais coletados na cidade de São José dos Campos (6 problemas)

Resultados Computacionais CPMP

		CS				SA			
ID	Melhor	sol*	Tempo*	sol	Tempo	sol*	sol	Tempo	DR
p1	713	**713**	**0,02**	**713,00**	**2,24**	713	721,60	1,56	1,21
p2	740	**740**	**0,01**	**740,00**	**2,34**	740	740,00	1,59	0,00
p3	751	**751**	**0,08**	**751,00**	**2,29**	751	751,60	1,46	0,08
p4	651	**651**	**0,03**	**651,00**	**2,24**	651	651,20	1,52	0,03
p5	664	**664**	**0,03**	**664,00**	**2,35**	664	664,40	1,51	0,06
p6	778	**778**	**0,01**	**778,00**	**2,35**	778	778,00	1,57	0,00
p7	787	**787**	**0,01**	**787,00**	**2,34**	787	788,60	1,55	0,20
p8	820	**820**	**0,08**	**820,00**	**2,38**	821	825,80	1,60	0,71
p9	715	**715**	**0,02**	**715,00**	**2,36**	715	717,80	1,46	0,39
p10	829	**829**	**2,09**	**829,00**	**2,36**	829	833,80	1,58	0,58
p11	1006	**1006**	**7,97**	**1006,00**	**35,24**	1007	1015,40	10,11	0,93
p12	966	**966**	**0,05**	**966,00**	**49,39**	968	970,20	12,74	0,43
p13	1026	**1026**	**0,06**	**1026,00**	**41,05**	1026	1028,80	10,00	0,27
p14	982	**982**	**19,80**	**982,80**	**39,14**	985	997,20	10,35	1,47
p15	1091	**1091**	**1,30**	**1091,00**	**44,98**	1092	1102,60	12,15	1,06
p16	954	**954**	**15,94**	**954,00**	**36,03**	957	960,40	10,37	0,67
p17	1034	**1034**	**19,50**	**1034,20**	**48,18**	1037	1043,00	12,24	0,85
p18	1043	**1043**	**18,62**	**1044,60**	**46,39**	1045	1048,20	11,41	0,34
p19	1031	**1031**	**10,17**	**1031,80**	**38,56**	1034	1042,80	10,09	1,07
p20	1005	**1005**	**5,41**	**1005,40**	**48,77**	1009	1015,20	12,45	0,97
sjc1	17288,99	**17288,99**	**0,23**	**17288,99**	**22,80**	17288,99	17343,96	10,43	0,32
sjc2	33270,94	**33270,94**	**13,25**	**33275,43**	**120,40**	33372,98	33491,75	25,70	0,65
sjc3a	45335,16	**45335,16**	**405,50**	**45337,34**	**859,09**	46746,68	47110,93	52,51	3,91
sjc3b	40635,90	**40635,90**	**1626,52**	**40643,67**	**1649,41**	41551,34	41888,14	54,01	3,06
sjc4a	**61925,51**	61928,72	938,45	62017,51	2601,81	63710,71	64574,92	77,19	4,10
sjc4b	**52469,96**	52531,27	1402,25	52540,67	7233,59	53789,61	54716,58	92,05	4,14

Resultados Computacionais CPMP

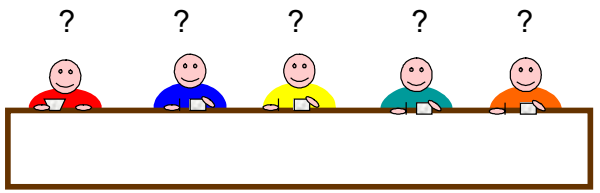
		SS-PR		VNS		CS	
ID	Melhor	sol*	Tempo	sol*	Tempo	sol*	Tempo
p1	713	713	6	**713**	**0,17**	713	2,24
p2	740	740	6	**740**	**0,05**	740	2,34
p3	751	751	6	**751**	**0,19**	751	2,29
p4	651	651	6	**651**	**0,11**	651	2,24
p5	664	664	6	**664**	**0,27**	664	2,35
p6	778	778	6	**778**	**0,11**	778	2,35
p7	787	787	6	**787**	**0,31**	787	2,34
p8	820	820	6	**820**	**0,92**	820	2,38
p9	715	715	6	**715**	**0,13**	715	2,36
p10	829	829	6	**829**	**0,75**	829	2,36
p11	1006	1006	60	**1006**	**7,91**	1006	35,24
p12	966	966	60	**966**	**4,81**	966	49,39
p13	1026	1026	60	**1026**	**2,17**	1026	41,05
p14	982	982	60	**982**	**10,33**	982	39,14
p15	1091	1091	60	**1091**	**10,23**	1091	44,98
p16	954	954	60	**954**	**4,20**	954	36,03
p17	1034	1034	60	**1034**	**5,50**	1034	48,18
p18	1043	1043	60	**1043**	**9,06**	1043	46,39
p19	1031	1031	60	**1031**	**8,64**	1031	38,56
p20	1005	1005	60	**1005**	**27,34**	1005	48,77
sjc1	17288,99	17288,99	60	17288,99	50,50	**17288,99**	**22,72**
sjc2	33270,94	33293,40	600	**33270,94**	**44,08**	33270,94	112,81
sjc3a	45335,16	45338,02	2307	45335,16	8580,30	**45335,16**	**940,75**
sjc3b	40635,90	40635,90	2308	40635,90	2292,86	**40635,90**	**1887,97**
sjc4a	61925,51	61925,52	6109	**61925,51**	**4221,47**	61928,72	2885,11
sjc4b	52469,96	52531,46	6106	**52469,96**	**3471,44**	52531,27	7626,33

ALWABP

- Centros de Trabalho para Deficientes (Sheltered Work Centres for Disabled – SWD)
- **Simple Assembly Line Balancing Problem (SALBP)**
  - Problema de balanceamento de uma linha de montagem
  - Conjunto finito de tarefas, cada uma tendo um tempo de execução fixo (independente do trabalhador que a executa)
  - Conjunto de relações de precedência, as quais especificam a ordem de execução das tarefas
  - Todo trabalhador é atribuído para somente uma estação de trabalho
  - Toda tarefa é atribuída para somente uma estação de trabalho
- **Assembly Line Worker Assigment and Balancing Problem (ALWABP)**

  (Problema de Atribuição de Trabalhadores e Balanceamento da Linha de Montagem)
  - Tempo de execução das tarefas pode ser diferente dependendo de qual dos trabalhadores executa a tarefa
  - Trabalhadores podem ser muito lentos, ou até incapazes, de executar algumas tarefas, mas muito eficientes quando executam outras tarefas
  - ALWABP-I: minimizar o número de estações de trabalho
  - ALWABP-II: maximizar a taxa de produção da linha (ou minimizar o tempo de ciclo) para um dado número de estações

ALWABP

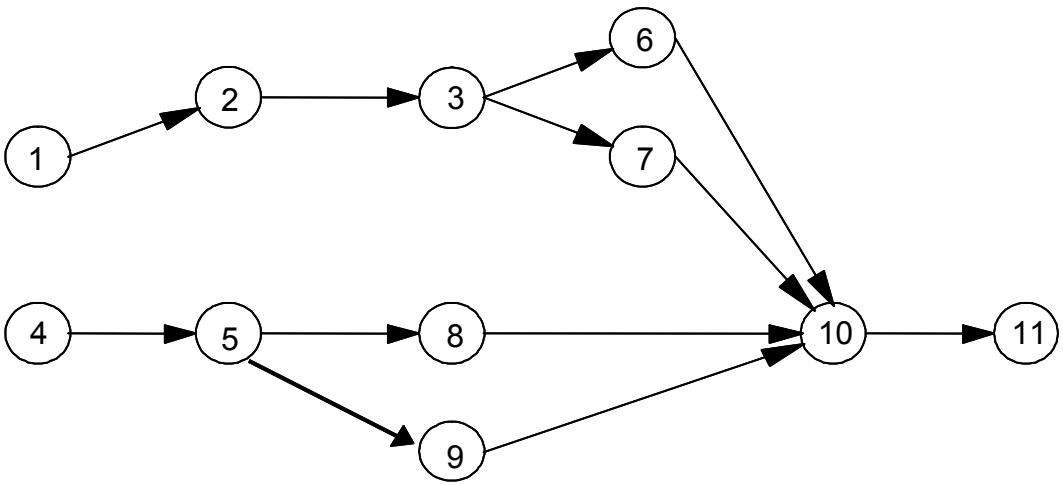


Estações

Tarefa	h1	h2	h3	h4	h5
1	45	-	57	45	38
2	12	10	11	11	8
3	7	10	12	9	15
4	44	50	40	52	50
5	23	23	15	15	15
6	14	18	10	12	12
7	12	25	12	15	20
8	12	-	12	12	12
9	-	12	12	12	-
10	8	-	20	8	14
11	30	24	29	35	35

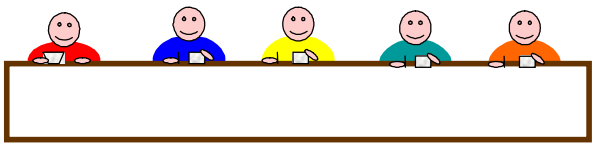
Tabela de Tempos tarefa/trabalhador

Rede de Precedência das Tarefas



ALWABP

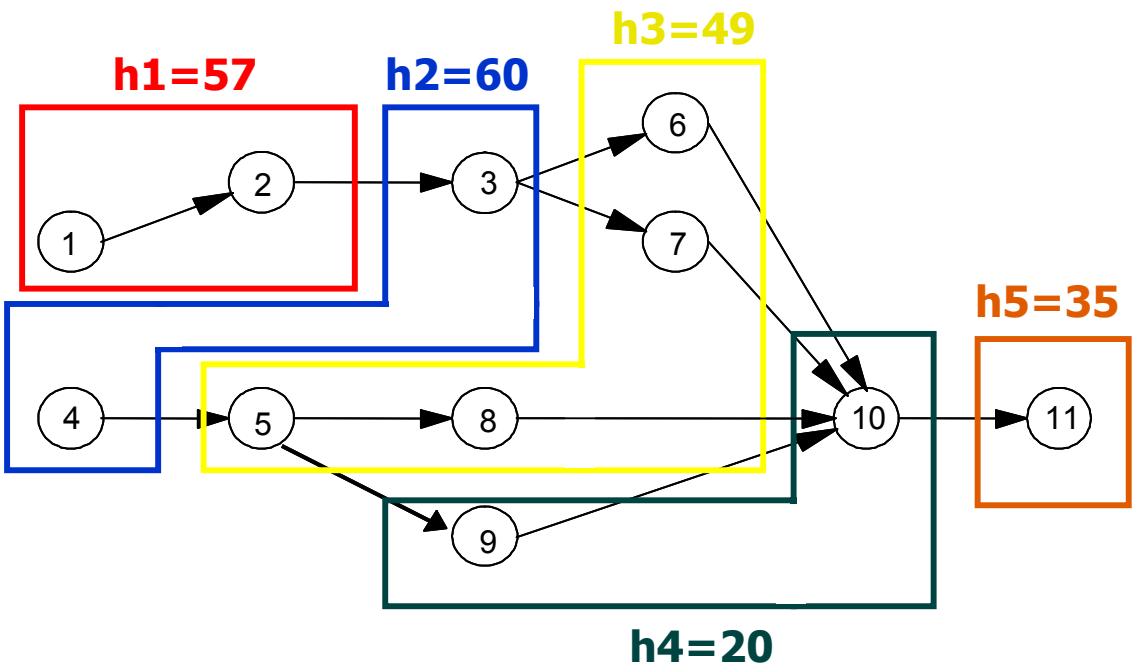
Estações



Tarefa	h1	h2	h3	h4	h5
1	45	-	57	45	38
2	12	10	11	11	8
3	7	10	12	9	15
4	44	50	40	52	50
5	23	23	15	15	15
6	14	18	10	12	12
7	12	25	12	15	20
8	12	-	12	12	12
9	-	12	12	12	-
10	8	-	20	8	14
11	30	24	29	35	35

Tabela de Tempos tarefa/trabalhador

Rede de Precedência das Tarefas



Tempo de Ciclo (C) = 221

CS aplicado ao ALWABP – componente SM

```
procedimento SA
    Criar Solução Inicial(sol)
    IterT = 0
    α = 0,95
    T = T0
    enquanto (T > 0.0001)
        enquanto (IterT < SAmax)
            IterT = IterT + 1
            Gerar um vizinho (sol') aleatoriamente na vizinhança Nk(sol)
            Δ = f(sol') – f(sol)
            se (Δ < 0)
                sol = sol'
            senão
                seja x ∈ [0,1]
                se (x < e-Δ/T)
                    sol = sol'
        fim-enquanto
        T = T x α
        IterT = 0
    fim-enquanto
fim-procedimento
```

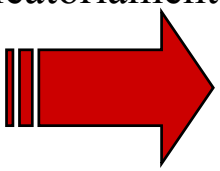
CS aplicado ao ALWABP – componente SM

```
procedimento SA
    Criar Solução Inicial(sol)
    IterT = 0
    α = 0,95
    T = T0
    enquanto (T > 0.0001)
        enquanto (IterT < SAmax)
            IterT = IterT + 1
            Gerar um vizinho (sol') aleatoriamente na vizinhança Nk(sol)
            Δ = f(sol') – f(sol)
            se (Δ < 0)
                sol = sol'
            senão
                seja x ∈ [0,1]
                se (x < e-Δ/T)
                    sol = sol'
        fim-enquanto
        T = T x α
        IterT = 0
    fim-enquanto
fim-procedimento
```

1) Trocar dois trabalhadores de estação
2) Trocar duas tarefas de estação
3) Retirar uma tarefa de uma estação e adicionar em outra estação

CS aplicado ao ALWABP – componente SM

```
procedimento SA
    Criar Solução Inicial(sol)
    IterT = 0
    α = 0,95
    T = T0
    enquanto (T > 0.0001)
        enquanto (IterT < SAmax)
            IterT = IterT + 1
            Gerar um vizinho (sol') aleatoriamente na vizinhança Nk(sol)
            Δ = f(sol') – f(sol)
            se (Δ < 0)
                sol = sol'
            senão
                seja x ∈ [0,1]
                se (x < e-Δ/T)
                    sol = sol'
        fim-enquanto
        T = T x α
        IterT = 0
    fim-enquanto
fim-procedimento
```

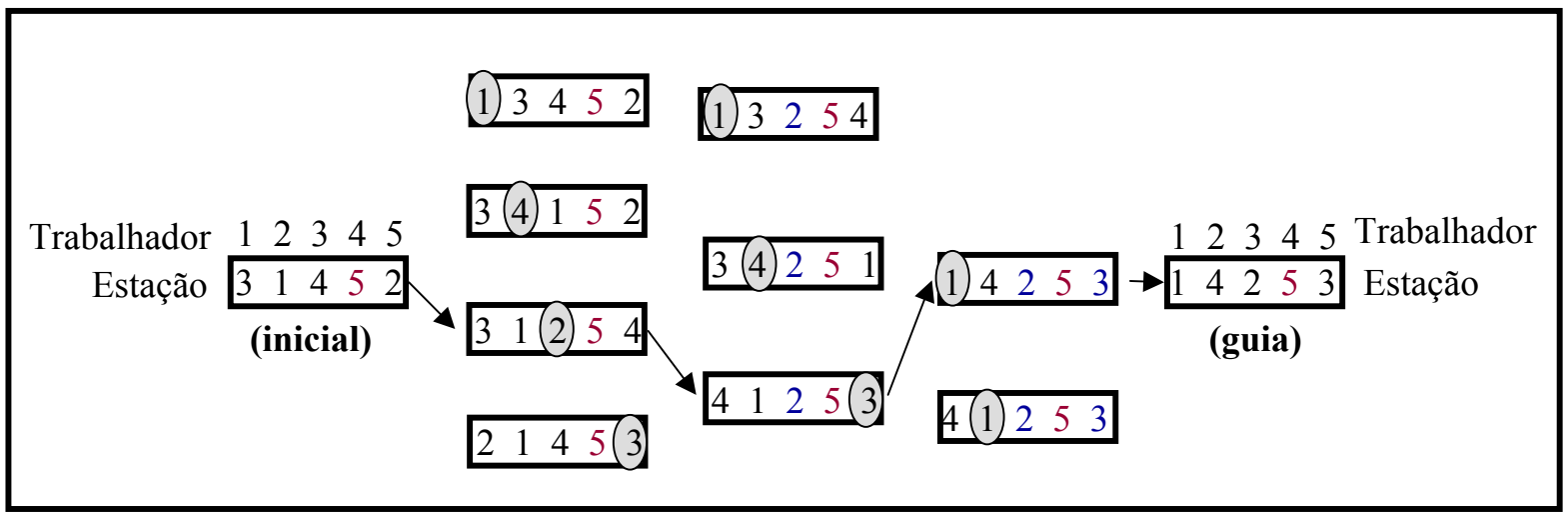


A cada 100 vizinhos gerados, a solução sol' é enviada para o componente AI

CS aplicado ao ALWABP – componente IC

- Medida de distância entre a solução gerada pelo SA e o centro do cluster:
  - **número de tarefas atribuídas para estações diferentes**

- Processo de assimilação por caminho: **path-relinking**



CS aplicado ao ALWABP – componente AM

- É executado toda vez que uma solução for atribuída a um cluster
- **Função**: verificar se o cluster já pode ser considerado promissor
- Cluster Populoso: $\lambda_t \geq PD.\frac{NS}{|C_t|}$
- Outras funções:
  - Esfriamento de todos os clusters que foram ativados
  - Eliminar os clusters pouco populosos

CS aplicado ao ALWABP – componente LS

- **Objetivo**: realizar uma busca local no centro dos clusters considerados promissores

- Heurística SWAP
  - examinar todos as possíveis trocas de duas tarefas que estão atribuídas à diferentes estações. A melhor troca é realizada e o processo é repetido novamente até que não ocorra mais melhora

- Heurística SHIFT
  - examinar todas as relações de tarefas e estações, removendo uma tarefa de uma estação e inserindo esta em uma outra estação. O melhor movimento é realizado e todo o processo é repetido novamente até não ocorrer mais melhora

Resultados Computacionais ALWABP

- Ambiente computacional
  - Linguagem C++ (C++ Builder 6.0)
  - PC Pentium 4 3.02 GHZ e memória de 512 MB
- Foram utilizados quatro conjuntos de instâncias

	Tamanho	Order Strength
Heskia	LOW (28 tarefas)	LOW (OS=22.49)
Roszieg	LOW (25 tarefas)	HIGH (OS=71.67)
Wee-Mag	HIGH (75 tarefas)	LOW (OS=22.67)
Tonge	HIGH (70 tarefas)	HIGH (OS=59.42)

- Para cada instância varia-se:
  - Relação entre número de tarefas e trabalhadores (tamanho da matriz tarefa-trabalhador)
  - Variabilidade dos tempos de execução das tarefas para cada trabalhador (L1 e H3)
  - Porcentagem de incompatibilidade tarefa-trabalhador (I10 e I20)

Resultados Computacionais ALWABP

				CPLEX		CS	
	NrT	Var	Inc	Sol	Tempo	Sol	Tempo
Roszieg	4	L1	I10	20,1	4,72	**20,1**	**3,76**
			I20	31,5	3,77	**31,5**	**3,76**
		H3	I10	28,1	5,46	**28,1**	**3,93**
			I20	28,0	4,27	**28,0**	**4,29**
	6	L1	I10	9,7	332,21	**9,7**	**4,68**
			I20	11,0	281,03	**11,0**	**4,83**
		H3	I10	16,0	390,07	**16,0**	**4,75**
			I20	15,1	686,75	**15,1**	**4,70**
Heskia	4	L1	I10	102,3	6,56	**102,3**	**4,63**
			I20	122,6	5,78	**122,6**	**4,67**
		H3	I10	172,5	6,70	**172,5**	**4,81**
			I20	171,2	7,30	**171,2**	**4,81**
	7	L1	I10	34,9	177,36	**34,9**	**5,55**
			I20	42,6	216,38	**42,6**	**5,29**
		H3	I10	75,2	260,93	**75,2**	**5,53**
			I20	67,2	341,83	**67,2**	**5,73**

Publicações

1) CORREA, F. A.; CHAVES, A. A.; LORENA, L. A. N. - **Hybrid Heuristics for the Probabilistic Maximal Covering Location-Allocation Problem**. Operational Research: An International Journal, 2007.

2) CHAVES, A. A.; CORREA, F. A.; LORENA, L. A. N. - **Clustering Search Heuristic for the Capacitated p-Median Problem**. Advances in Soft Computing, v. 44, p. 136-143, 2007.

3) CHAVES, A. A.; CORREA, F. A.; LORENA, L. A. N. - **Clustering Search Heuristic for the Capacitated p-Median Problem**. In: HAIS 07 2nd International Workshop on Hybrid Artificial Intelligence Systems, 2007, Salamanca. Proceedings of HAIS, 2007.

4) CHAVES, A. A.; Cristobal, M.; LORENA, L. A. N. - **Clustering Search Approach for the Assembly Line Worker Assignment and Balancing Problem**. In: ICC&IE'2007 37th International Conference on Computers and Industrial Engineering, 2007, Alexandria. Computers and Industrial Engineering, 2007.

5) CORREA, F. A.; CHAVES, A. A.; LORENA, L. A. N. - **Heurística Híbrida com Detecção de Regiões Promissoras Aplicada ao Problema Probabilístico de Localização-Alocação de Máxima Cobertura**. In: XXXIX SBPO Simpósio Brasileiro de Pesquisa Operacional, 2007, Fortaleza. Anais do SBPO, 2007.

6) CHAVES, A. A.; LORENA, L. A. N. - **Aplicação do Algoritmo Clustering Search aos Traveling Salesman Problems with Profits**. In: XXXIX SBPO Simpósio Brasileiro de Pesquisa Operacional, 2007, Fortaleza. Anais do SBPO, 2007.

7) CHAVES, A. A.; LORENA, L. A. N. - **A Preprocessing Phase for the Evolutionary Clustering Search**. In: I Workshop on Computational Intelligence (WCI), 2006, Ribeirão Preto. In International Joint Conference 2006, 2006.

8) CHAVES, A. A.; LORENA, L. A. N. - **Hybrid Algorithms with Detection of Promising Areas for the Prize Collecting Travelling Salesman Problem**. In: HIS'05 Fifth International Conference on Hybrid Intelligent Systems, 2005, Rio de Janeiro. HIS 2005. California : IEEE Computer Society, 2005. p. 49-54.

9) CHAVES, A. A.; LORENA, L. A. N. - **Algoritmos Híbridos para uma Generalização do Problema do Caixeiro Viajante**. In: Simpósio de Pesquisa Operacional e Logística da Marinha (SPOLM), 2005, Rio de Janeiro. Anais do SPOLM, 2005.

Outras Aplicações do CS

- Oliveira A. C. M. and Lorena, L. A. N.
  Pattern Sequencing Problems by Clustering Search. Jaime Simão Sichman, Helder Coelho and Solange Oliveira Rezende (Eds.). Springer Lecture Notes in Artificial Intelligence Series vol. 4140, pp. 218 - 227, 2006

- Ribeiro Filho, G.; Nagano, M. S. and Lorena, L. A. N.
  Evolutionary Clustering Search for Flowtime Minimization in Permutation Flow Shop. T. Bartz-Beielstein et al. (Eds.): HM 2007, LNCS-Lecture Notes in Computer Science. 4771, Springer, pp. 69–81, 2007

- Ribeiro Filho, G.; Nagano, M. S. and Lorena, L. A. N.
  Hybrid Evolutionary Algorithm for Flowtime Minimisation in No-Wait Flowshop Scheduling. A. Gelbukh and A.F. Kuri Morales (Eds.): MICAI 2007, LNAI 4827, pp. 1099–1109, 2007. Springer-Verlag Berlin Heidelberg 2007

- Biajoli, F. L. and Lorena, L. A. N.
  Clustering Search Approach for the TravelingTournament Problem. A. Gelbukh and A.F. Kuri Morales (Eds.): MICAI 2007, LNAI 4827, pp. 83-93, 2007. Springer-Verlag Berlin Heidelberg 2007

- Correa, F. A.; Chaves, A. A. and Lorena, L. A. N.
  Hybrid heuristics for the probabilistic maximal covering location-allocation problem. Operational Research Journal, 2007

Considerações Finais

- Clustering Search (CS): uma nova maneira de combinar metaheurísticas e métodos de busca local de forma eficiente
- Os resultados obtidos mostram o potencial do CS para resolução de problemas de Otimização Combinatória
- Objetivo:
  - Colocar a disposição uma nova técnica para resolução de problemas de otimização que necessitem ser resolvidos de forma aproximada e em um tempo computacional competitivo

Referências

- Voß S., Martello, S., Osman, I.H., Roucairo, C.: Meta-Heuristics: Andvances and Trends in Local Search Paradigms for Optimization. Kluwer, Boston (1999)

- Glover, F., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer (2003)

- Raidl, G.L.: A Unified View on Hybrid Metaheuristics. Lecture Notes in Computer Science, Book: Hybrid Metaheuristics, volume 4030, pages 1-12 (2006)

Conteúdo

C01 – Simulated Annealing (20/11/07).

C02 – Busca Tabu (22/11/07).

C03 – Colônia de Formigas (27/11/07).

C04 - GRASP e VNS (29/11/07).

C05 – Metaheurísticas Híbridas – CS (04/12/07).