Problema de Roteirização de Veículos com Multi-Compartimentos

Carlos **Leonardo** Ramos Póvoa, D.Sc.

Lab. Engenharia de Produção - LEPROD

Universidade Estadual do Norte Fluminense - UENF

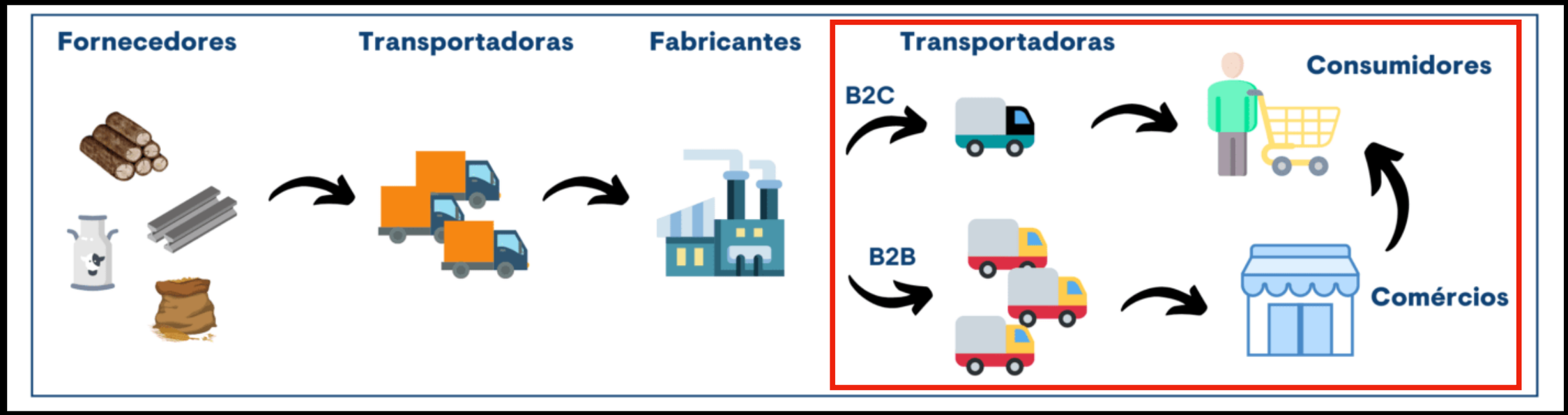
email: clrp@uenf.br

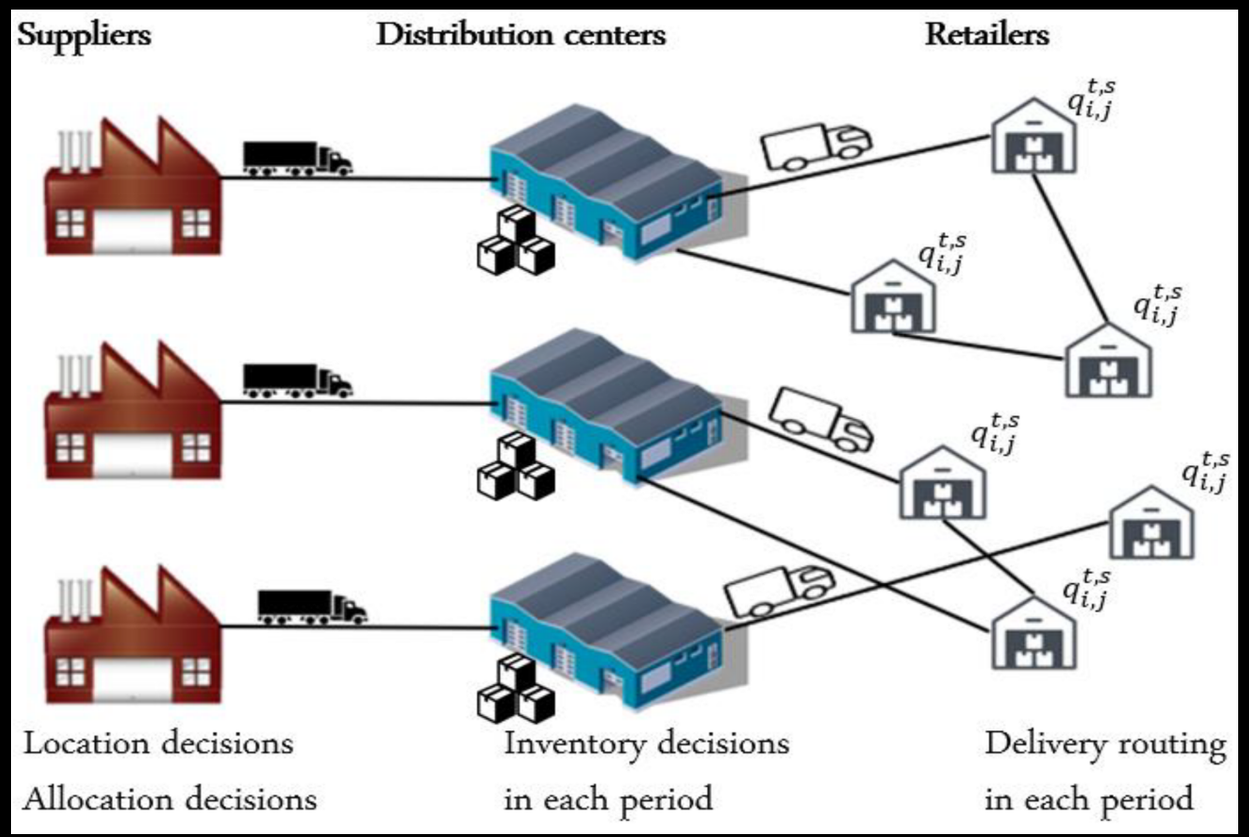
Cel: (22)99984-4899

http://www.loggeo.org

Seminário I

Supply Chain





Linha de Pesquisa

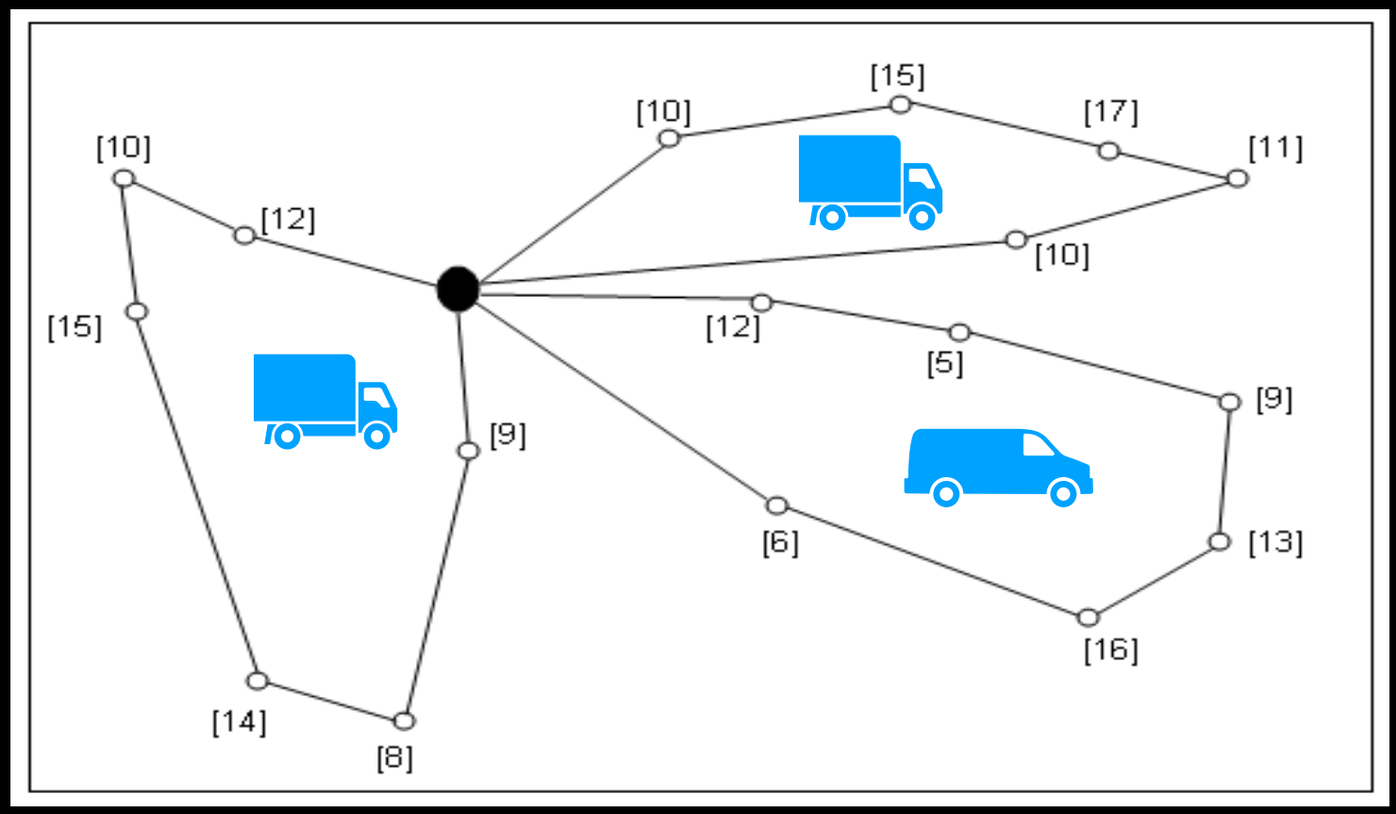
Supply Chain: Desenvolver sistemas e métodos para otimizar as operações de: armazenagem, estoque e transporte.

Projetos

1) Roteirização de Veículos: MCVRP

2) Sistema de apoio a decisão (SAD): Roteirizador Geo-Rota web

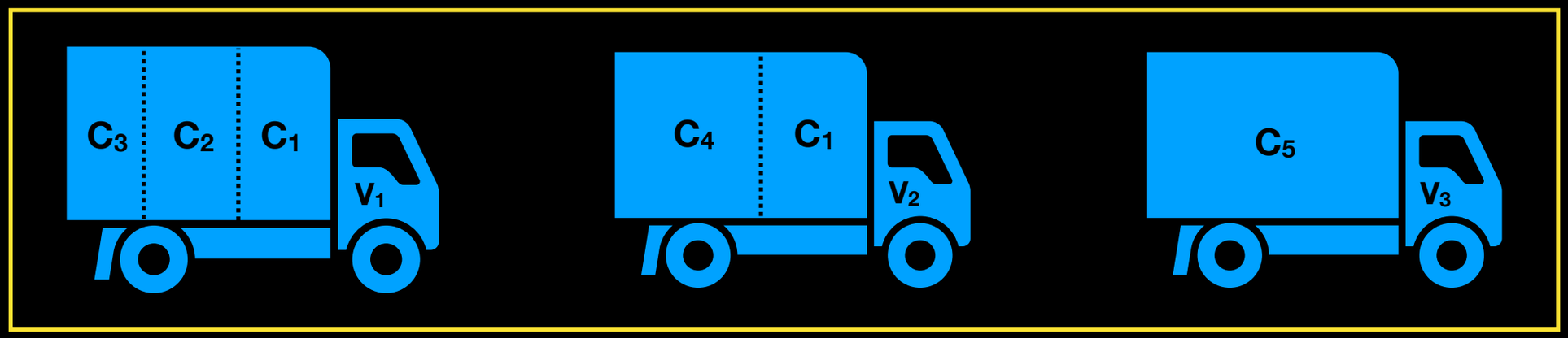
Problema do Roteirização de Veículos



* Problema *NP-hard* (Osman, 1993)
* Limitação para algoritmos exatos - *branch-and-cut-and-price (BCP)*.

Métodos Heurísticos

Multi-Compartment Vehicle Routing Problem (MCVRP)



- Complete undirected graph => $G = (V, E)$

- $V = \{0, 1, ..., n\}$ consists of a depot (node 0) and a set of customers $V^+ = \{1, ..., n\}$.

- Each edge $e \in E$ has a travel cost and a travel time denoted by respectively c_e and t_e.

- The demand of customer $j \in V^+$ for a product of the type $p \in P$ is given and it is denoted by d_{jp}.

- A set K of homogeneous vehicles is available where each one of them has |P| compartments each one reserved for a different product $p \in P$ with a given capacity denoted by Q_p.

---

* The MCVRP consists of designing at most K routes with minimal total cost that satisfies the following three constraints:

1) each customer is visited exactly once by a single vehicle and all demands of this customer are satisfied during that visit.

2) capacity of each vehicle compartment is respected.

3) total travel time spent on each route does not exceed a given upper limit T_{max}

MCVRP - Proposed Heuristic (1): GRASP

Algorithm 1 GRASP - MCVRP

procedure GRASP($\delta_1, \delta_2, lcr_i, lcr_f, iter$)
  $I \leftarrow \lceil (\sum_{i \in V} \sum_{p \in P} d_{ip}) / \sum_{p \in P} Q_p \rceil$
  ρ, $S \leftarrow Seed\,Selection(I)$
  $f^* \leftarrow +\infty$
  for $i = 1, \ldots, iter$ **do**
    $x \leftarrow Construct\,Greedy\,Randomized(\rho, S, \delta_1, \delta_2, lcr_i, lcr_f)$
    $\dot{x} \leftarrow Local\,Search(\mathrm{x})$
    if $(f(\dot{x}) < f^*)$ **then**
      $x^* \leftarrow \dot{x}$
      $f^* \leftarrow f(\dot{x})$
    end if
  end for
return x^*
end procedure

GRASP Seed Selection

Algorithm 2 Seed Selection

```
1: procedure Seed Selection(I)
2:     ρ ← {}
3:     S ← {}
4:     s ← the client c ∈ V⁺ that is farthest from the depot {0}
5:     while | ρ |< I do
6:         S ← S ∪ {s}
7:         r ← Route that only visits the customer s
8:         ρ ← ρ ∪ r
9:         s ← client c ∈ V⁺ \ S that maximizes the sum of distances
10:        to all clients in S
11:    end while
12: return ρ, S
13: end procedure
```

GRASP Construct Greedy Randomized

Algorithm 3 *Construct Greedy Randomized*

procedure *Construct Greedy Randomized*$(\rho, S, \delta_1, \delta_2, lcr_i, lcr_f)$
  $\lambda \leftarrow Random(lcr_i, lcr_f)$
  $V' \leftarrow V^+ \setminus S$
  while $V' \neq \emptyset$ **do**
    for $c = 1, \ldots, \mid V' \mid$ **do**
      for $r = 1, \ldots, \mid \rho \mid$ **do**
        $C_{c,r} \leftarrow \min\{C_{i,j,c,r}, \forall \text{ arc } (i,j) \in r\}$
      end for
      $C_c^* \leftarrow \min\{C_{c,r}, \forall r \in \rho\}$
    end for
    for $c = 1, \ldots, \mid V' \mid$ **do**
      $P_c \leftarrow \sum_{r \in \rho}(C_{c,r} - C_c^*)$
    end for
    $RCL \leftarrow$ Construct a list containing the λ highest Penalty costs (P_c)
    $\hat{c} \leftarrow$ Insert in ρ a randomly selected consumer from RCL
    $V' \leftarrow V' \setminus \{\hat{c}\}$
  end while
return solution ρ
end procedure

GRASP Construct Greedy Randomized

The customer k insertion cost, on route r, between nodes i and j is:

$$C_{i,j,k,r} = \delta_1 c^1_{i,j,k,r} + \delta_2 c^2_{i,j,k,r} \qquad \delta_1 + \delta_2 = 1$$

$$c^1_{i,j,k,r} = VFC_r - \sum_{\forall p \in P} d_{kp}$$

VFC_r is the free capacity of route r considering all the products.

$$c^2_{i,j,k,r} = c_{ik} + c_{jk} - c_{ij}$$

increase in the total travel cost when the customer k is inserted between the nodes i and j.

GRASP Local Search

Algorithm 4 *Local Search*

```
1: procedure Local Search(x)
2:     impro ← true
3:     ListMove ← ∅
4:     while impro = true do
5:         Find all feasible moves (Shift(1, 0); Swap(1, 1)) ∈ x
6:         ListMove ← Construct a list with the (up to) 5 best improving moves
7:         if ListMove = ∅ then
8:             impro ← false
9:         else
10:            Randomly select one move from MoveList and update the current
    solution
11:        end if
12:    end while
13: return x
14: end procedure
```

GRASP Resultado Parcial

GRASP: LCR: 2-10; 100/4; d1:0.2; d2:0.8

Instance	*n*	BKS	El Fallahi et al. (2008)						Mendoza et al. (2010)			GRASP				
			MA	t(s)	Gap %	TS	t(s)	Gap %	MA/ SCS	t(s)	Gap %	Best	Gap %	Avg	Gap %	t(s)
vrpnc1	50	**547.0**	558.8	6	2.2	556.1	5	1.7	550.2	5	0.6	550.2	0.6	550.7	0.7	1
vrpnc2	75	**856.7**	888.6	9	3.7	863.6	5	0.8	884.3	9	3.2	890.2	3.9	904.4	5.6	1
vrpnc3	100	**829.9**	878.4	7	5.8	837.6	13	0.9	852.7	13	2.7	852.0	2.7	874.9	5.4	2
vrpnc4	150	1070.7	1089.1	31	1.7	1070.7	37	0.0	1080	25	0.9	1122.8	4.9	1144.6	6.9	6
vrpnc5	199	**1343.9**	1408.5	39	4.8	1361.4	69	1.3	1403.5	46	4.4	1462.1	8.8	1494.4	11.2	13
vrpnc6	50	**557.5**	569.4	6	2.1	563.4	3	1.1	562.7	4	0.9	557.5	0.0	564.4	1.2	1
vrpnc7	75	**921.6**	955.1	13	3.6	949	7	3.0	946.3	8	2.7	950.8	3.2	959.4	4.1	1
vrpnc8	100	874.7	958.9	6	9.6	916.2	6	4.7	874.7	12	0.0	913.2	4.4	933.3	6.7	2
vrpnc9	150	1197.8	1262.7	33	5.4	1290.8	3	7.8	1240.5	26	3.6	1266.7	5.7	1284.2	7.2	7
vrpnc10	199	1490.2	1509.1	47	1.3	1490.2	63	0.0	1515.6	39	1.7	1530.3	2.7	1555.8	4.4	14
vrpnc11	120	1122.9	1122.9	16	0.0	1201.6	9	7.0	1162.4	24	3.5	1214.0	8.1	1235.1	10.0	2
vrpnc12	100	**899**	926.5	6	3.1	934.1	5	3.9	916.6	12	2.0	921.3	2.5	945.5	5.2	1
vrpnc13	120	1542.4	1542.4	25	0.0	1582.3	7	2.6	1607.4	19	4.2	1632.5	5.8	1668.3	8.2	3
vrpnc14	100	**930.4**	966.5	8	3.9	1141.6	12	22.7	937.6	12	0.8	951.8	2.3	957.4	2.9	2
e072-04f	72	262.3	263.6	4	0.5	262.3	2	0.0	262.5	11	0.1	270.5	3.1	272.8	4.0	1
e076-08s	76	**748.4**	793.5	5	6.0	772.2	5	3.2	778.2	7	4.0	769.9	2.9	786.2	5.1	0
e076-07u	76	**693.3**	702.2	5	1.3	697.8	6	0.6	704.1	8	1.6	707.2	2.0	713.3	2.9	1
e135-07f	135	**1214.4**	1233.2	16	1.5	1235.2	17	1.7	1293.8	24	6.5	1376.7	13.4	1429.4	17.7	5
e241-22k	241	787.8	796.7	168	1.1	787.8	68	0.0	810.7	58	2.9	817.2	3.7	826.9	5.0	28
e484-19k	484	1177.3	1240.9	548	5.4	1177.3	708	0.0	1269	391	7.8	1236.1	5.0	1249.2	6.1	172
Avg.		953.4	983.35	50	3.1	984.6	53	3.3	982.6	38	3.1	999.6	4.8	1017.5	6.7	13

Proposed Heuristic (2): Tabu Search

Algorithm 5 Tabu Search

procedure TABUSEARCH($x, iterMax, noimpMax$)
  $\alpha \leftarrow 1,\ \beta \leftarrow 1,\ noimprove \leftarrow 0,\ inf \leftarrow 0, iter \leftarrow 0$
  $x^* \leftarrow x,\ f^* \leftarrow f(x)$
  Choose $\delta \in [0,1],\ \zeta \in [0,1],\ \tau \in \left[1,\ \left\lceil \sqrt{|V^+| \times R(x)} \right\rceil\right]$ randomly, where $R(x)$ is the number of routes of the solution x
  repeat
    $iteration \leftarrow iteration + 1$
    $x \leftarrow best\,Move(x, \alpha, \beta, \zeta)$
    updatePenalties$(x, \alpha, \beta, \delta)$
    updateTabuList(τ)
    if x is feasible and $f(x) < f^*$ **then**
      $x^* \leftarrow x,\ f^* = f(x)$
      $noimprove \leftarrow 0$
    else
      $noimprove \leftarrow noimprove + 1$
    end if
    if x is feasible **then**
      $inf \leftarrow 0$
    else
      $inf \leftarrow inf + 1$
      if $inf = 5$ **then**
        $x \leftarrow$ perturb(x^*)
        $inf \leftarrow 0$
        $\alpha \leftarrow 1,\ \beta \leftarrow 1$
        Choose $\delta \in [0,1]$, $\zeta \in [0,1]$, $\tau \in [1,\ \lceil \sqrt{|V^+| \times R(x)} \rceil]$ randomly
      end if
    end if
  until $(iteration = iterMax)$ or $(noimprove = noimpMax)$
return x^*
end procedure

Neighborhood and Tabu list

Shift (1, 0) => $M(r, r', i, j)$

r is the origin route, r' is the destination route, i is the customer, and j is the position in r' where i should be inserted.

30% -> nearest routes

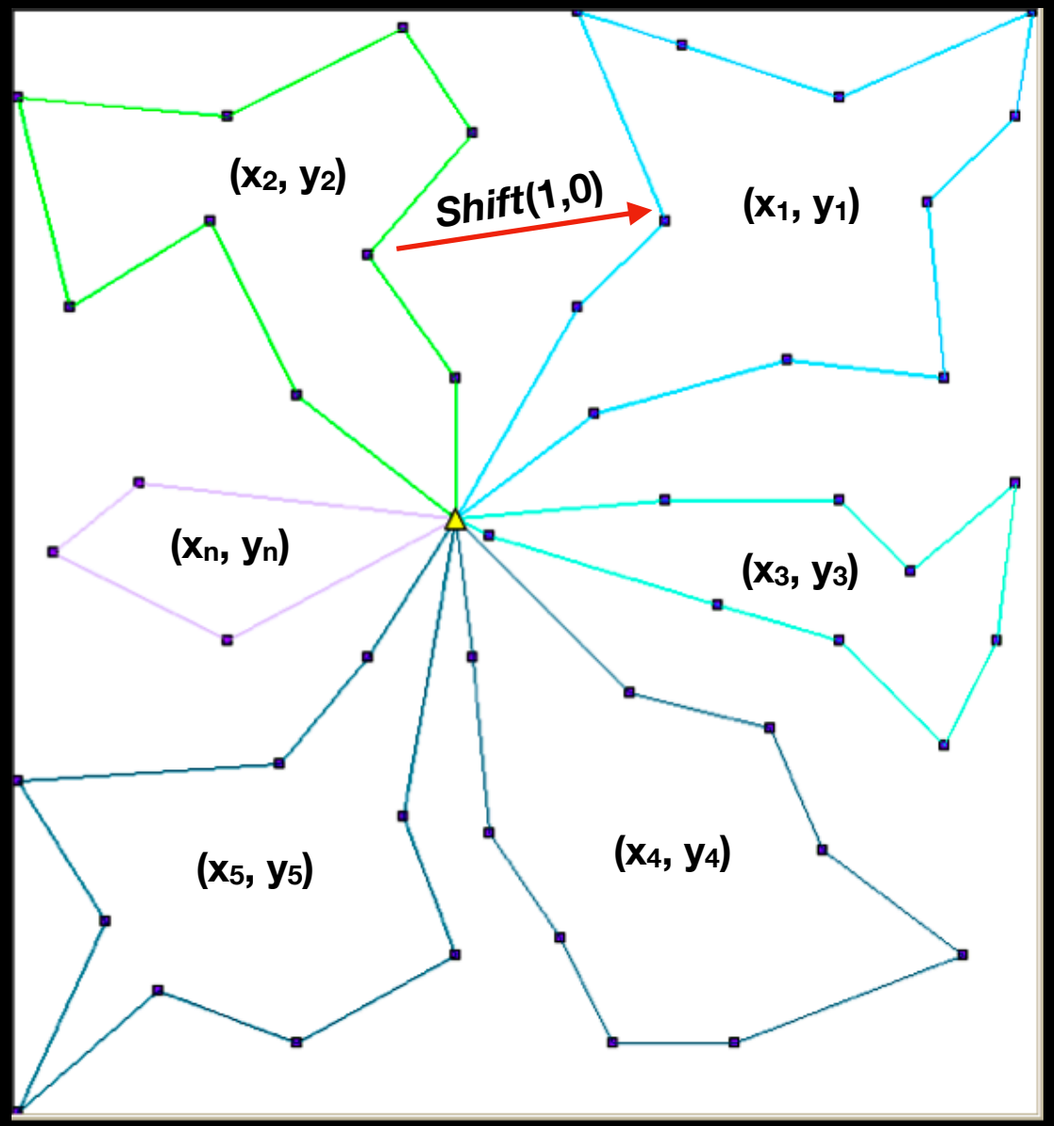
Route center of gravity:

$$\left(\sum_{j \in r'} x_j / |r'|, \sum_{j \in r'} y_j / |r'|\right)$$

Tabu list:

$M(r, r', i)$ **Tabu by** τ **iterations:**

$$[1, \lceil \sqrt{|V^+| \times R(x)} \rceil]$$



Função Objetivo

Possibilidade de soluções inviáveis

$$f(s) = \sum_{r \in R} c(r) + \alpha q(r) + \beta d(r)$$

$c(r)$: Custo total da rota r.

$q(r)$: Soma das violações dos compartimentos (tipos de produto).

$d(r)$: Violação do tempo máximo da rota.

α, β : Pesos autoajustáveis, inicialmente iguais a 1. Aumentam **1 +** δ quando as restrições são violadas, caso contrário são decrescidos de **1 -** δ.

δ : Escolhido de forma aleatória no intervalo [0, 1] no inicio do procedimento.

Diversificação

1) Possibilidade de soluções inviáveis:

$$f(s) = \sum_{r \in R} c(r) + \alpha q(r) + \beta d(r)$$

2) Penalizar Movimentos frequentes:

$$f'(s) = f(s)(1 + \zeta freq_M / iter)$$

$\zeta \in [0,1]$ *randomly*

$freq_M$ -> número de vezes que o movimento M foi realizado.

3) restart procedure:

- 5 soluções não viáveis consecutivas: S^* é perturbado -> $Swap(1, 1)$.
- Número de clientes escolhido de forma aleatória no intervalo $[1, 4]$.
- Soluções viáveis; Parâmetros $\alpha, \beta, \delta, \zeta, \tau$ são reiniciados.

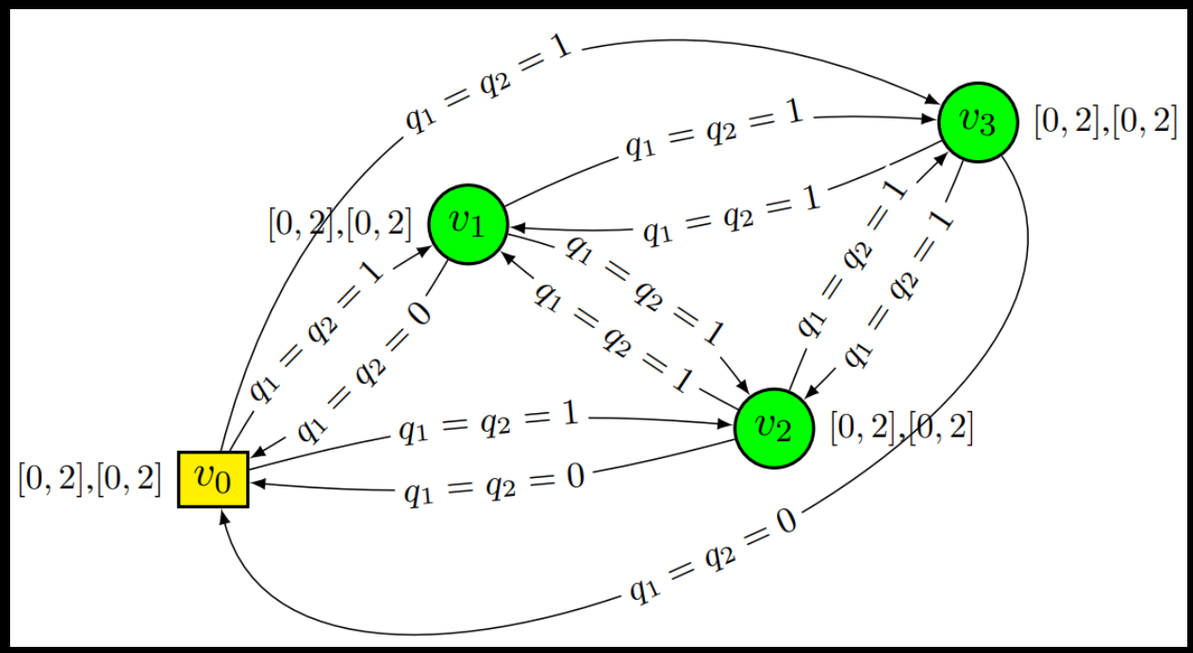
VRPSolver

- O VRPSolver facilita a construção de algoritmos BCP para problemas de roteamento de veículos e suas variantes.
- O modelo VRPSolver é um modelo de Programação Linear Inteira que contém variáveis de caminho restritas em recursos sobre um ou mais grafos direcionados.
- O usuário do VRPSolver define a formulação e os grafos adequados para se resolver um determinado problema.
- Com base na formulação e nos grafos definidos, o framework resolve a formulação através de um algoritmo BCP genérico que gera as variáveis de caminho dinamicamente através da resolução dos subproblemas de pricing, que são modelados como Problemas do Caminho mais Curto com Restrições de Recursos (RCSP).

$n = 3; K = 2, \left|P\right| = 2$

$Q_1 = Q_2 = 2$

Demandas unitárias



Computational Environment

Reference	Environment	Factor
Fallahi et al. (2008)	Pentium 4. 2.4 GHz	3
Mendoza et al. (2010)	Intel 2.4 Ghz Core 2 Duo Processor	2
Mirzaei and Wøhlk (2017)	Pentium Core i5 1.8 GHz	2
Abdulkader et al. (2015)	Server with 16 Core at 2.1 GHz	1.5
This paper (heuristic approach)	Intel Core i5-2400 at 3.1 GHz	1

Table 1: Computational environments for heuristic approaches.

- Configurações do experimentos (VRPSolver):
  - Processador Intel Core i7-10700 com 2.90 GHz e 16 GB of RAM
  - Sistema operacional é o Ubuntu 18.04.2 LTS.
  - VRPSolver v0.4.1 - https://vrpsolver.math.u-bordeaux.fr/.
  - Cada instância é resolvida usando uma única thread.
  - CPLEX v12.10 é usado como resolvedores PL e MIP.
  - Para cada experimento, nós usamos como solução inicial a melhor solução encontrada pela heurística (GRASP/TS). Isso aumenta a performance do algoritmo BCP.
  - Tempo limite: 18.000s

Experimentos Computacionais

1) Instâncias MCVRP [El Fallahi et al. 2008] -> 20 instâncias VRP de [Christofides et al. 1979] e [Christofides e Eilon 1969]

Instance	$\|V^+\|$	BKS	El Fallahi et al. (2008)						Mendoza et al. (2010)			GRASP/TS					
			MA	t(s)	Gap %	TS	t(s)	Gap %	MA/SCS	t(s)	Gap %	Best	Gap %	Avg	Gap %	t(s)	t(s) best
vrpnc 1	50	***547.0**	558.8	6	2.2	556.1	5	1.7	550.2	5	0.6	547.0	0.0	549.9	0.5	6	1
vrpnc 2	75	***856.7**	888.6	9	3.7	863.6	5	0.8	884.3	9	3.2	863.2	0.8	873.8	2.0	14	<1
vrpnc 3	100	***829.9**	878.4	7	5.8	837.6	13	0.9	852.7	13	2.7	833.6	0.4	842.1	1.5	18	<1
vrpnc 4	150	**1067.2**	1089.1	31	2.1	1070.7	37	0.3	1080	25	1.2	1067.2	0.0	1074.3	0.7	37	10
vrpnc 5	199	***1343.9**	1408.5	39	4.8	1361.4	69	1.3	1403.5	46	4.4	1351.9	0.6	1375.3	2.3	76	17
vrpnc 6	50	***557.5**	569.4	6	2.1	563.4	3	1.1	562.7	4	0.9	557.5	0.0	561.7	0.8	20	<1
vrpnc 7	75	***921.6**	955.1	13	3.6	949	7	3.0	946.3	8	2.7	922.4	0.1	935.1	1.5	18	8
vrpnc 8	100	**874.2**	958.9	6	9.7	916.2	6	4.8	874.7	12	0.1	874.2	0.0	889.6	1.8	25	<1
vrpnc 9	150	**1197.8**	1262.7	33	5.4	1290.8	3	7.8	1240.5	26	3.6	1200.4	0.2	1227.4	2.5	55	44
vrpnc 10	199	**1456.0**	1509.1	47	3.6	1490.2	63	2.4	1515.6	39	4.1	1456.0	0.0	1485.9	2.1	93	29
vrpnc 11	120	1122.9	1122.9	16	0.0	1201.6	9	7.0	1162.4	24	3.5	1148.3	2.3	1192.4	6.2	25	4
vrpnc 12	100	***899**	926.5	6	3.1	934.1	5	3.9	916.6	12	2.0	903.2	0.5	914.4	1.7	20	6
vrpnc 13	120	1542.4	1542.4	25	0.0	1582.3	7	2.6	1607.4	19	4.2	1557.4	1.0	1656.4	7.4	29	13
vrpnc 14	100	***930.4**	966.5	8	3.9	1141.6	12	22.7	937.6	12	0.8	936.1	0.6	943.0	1.3	19	1
e072-04f	72	**262.3**	263.6	4	0.5	262.3	2	0.0	262.5	11	0.1	262.3	0.0	263.0	0.3	11	1
e076-08s	76	***748.4**	793.5	5	6.0	772.2	5	3.2	778.2	7	4.0	748.4	0.0	758.8	1.4	12	1
e076-07u	76	***693.3**	702.2	5	1.3	697.8	6	0.6	704.1	8	1.6	693.3	0.0	696.1	0.4	12	1
e135-07f	135	***1214.4**	1233.2	16	1.5	1235.2	17	1.7	1293.8	24	6.5	1227.4	1.1	1273.4	4.9	32	7
e241-22k	241	**763.5**	796.7	168	4.4	787.8	68	3.2	810.7	58	6.2	763.5	0.0	775.4	1.6	127	8
e484-19k	484	**1164.2**	1240.9	548	6.6	1177.3	708	1.1	1269	391	9.0	1164.2	0.0	1174.0	0.8	416	115
Avg		949.6	983.35	50	3.6	984.6	53	3.7	982.6	38	3.5	953.9	0.4	973.1	2.5	53	17

Experimentos Computacionais

2) Instâncias MCVRP [Abdulkader et al, 2015] -> 28 instâncias VRP de [Christofides et al. 1979].

Instance	$\|V^+\|$	BKS	Abdulkader et al. (2015)						GRASP/TS					
			ACS	t(s)	Gap %	HAC	t(s)	Gap %	Best	Gap %	Avg	Gap %	t(s)	t(s) best
vrpnc1a	50	***550.7**	559.6	16	1.6	550.7	5	0.0	550.7	0.0	550.8	0.0	11	0
vrpnc1b	50	***551.9**	569.1	17	3.1	551.9	5	0.0	551.9	0.0	551.9	0.0	11	0
vrpnc2a	75	***868.6**	957.5	36	10.2	890.7	15	2.5	872.0	0.4	875.3	0.8	26	1
vrpnc2b	75	***878.7**	954.9	35	8.7	919.0	14	4.6	886.2	0.9	889.7	1.2	27	3
vrpnc3a	100	***860.4**	964.1	122	12.1	874.1	40	1.6	866.6	0.7	871.8	1.3	37	2
vrpnc3b	100	***859.4**	959.3	122	11.6	895.3	44	4.2	873.0	1.6	877.9	2.1	37	3
vrpnc4a	150	***1075.7**	1253.9	345	16.6	1126.1	146	4.7	1087.6	1.1	1094.3	1.7	75	7
vrpnc4b	150	***1092.1**	1254.5	336	14.9	1159.5	151	6.2	1108.8	1.5	1116.1	2.2	78	12
vrpnc5a	199	**1384.1**	1587.0	688	14.7	1444.3	257	4.3	1384.1	0.0	1395.0	0.8	152	19
vrpnc5b	199	***1373.3**	1640.6	676	19.5	1525.9	236	11.1	1427.8	4.0	1439.0	4.8	154	21
vrpnc6a	50	***557.5**	573.3	13	2.8	557.5	11	0.0	557.5	0.0	557.5	0.0	30	1
vrpnc6b	50	***557.5**	573.4	13	2.8	559.4	10	0.3	557.5	0.0	557.5	0.0	26	1
vrpnc7a	75	***920.5**	997.0	33	8.3	928.2	28	0.8	921.5	0.1	931.4	1.2	39	2
vrpnc7b	75	***930.7**	969.3	32	4.2	932.7	26	0.2	930.7	0.0	937.6	0.7	37	2
vrpnc8a	100	***876.7**	963.4	97	9.9	883.0	93	0.7	886.6	1.1	897.8	2.4	47	4
vrpnc8b	100	***875.3**	976.2	97	11.5	884.9	95	1.1	880.4	0.6	900.1	2.8	43	4
vrpnc9a	150	***1177.2**	1343.1	273	14.1	1228.9	326	4.4	1236.0	5.0	1252.3	6.4	110	18
vrpnc9b	150	***1175.7**	1346.6	274	14.5	1226.6	333	4.3	1219.4	3.7	1233.0	4.9	110	16
vrpnc10a	199	**1469.7**	1645.6	606	12.0	1511.7	624	2.9	1469.7	0.0	1492.0	1.5	183	31
vrpnc10b	199	**1470.8**	1659.9	608	12.9	1526.0	620	3.8	1470.8	0.0	1495.2	1.7	179	27
vrpnc11a	120	***1103.5**	1133.9	281	2.8	1110.5	75	0.6	1105.1	0.1	1107.7	0.4	47	6
vrpnc11b	120	**1199.0**	1247.5	280	4.0	1221.7	87	1.9	1201.8	0.2	1209.7	0.9	48	7
vrpnc12a	100	***899.6**	911.9	105	1.4	912.6	15	1.4	901.4	0.2	906.6	0.8	38	6
vrpnc12b	100	***950.8**	970.8	100	2.1	950.8	30	0.0	950.8	0.0	957.6	0.7	39	4
vrpnc13a	120	1556.5	1577.5	171	1.3	1556.5	117	0.0	1558.9	0.2	1619.9	4.1	57	10
vrpnc13b	120	1550.1	1572.1	168	1.4	1550.1	123	0.0	1560.7	0.7	1627.0	5.0	59	11
vrpnc14a	100	***911.0**	914.9	91	0.4	911.4	34	0.0	911.0	0.0	914.4	0.4	39	5
vrpnc14b	100	***964.7**	970.9	91	0.6	965.8	38	0.1	965.3	0.1	970.2	0.6	40	5
Avg		1022.9	1108.8	204.5	8.4	1048.4	128	2.5	1031.9	0.9	1043.9	1.8	64	8

Experimentos Computacionais

3) 189 instancias (MCVRP) com até 4 compartimentos propostas por Mirzaei and Wøhlk (2017) que são agrupadas em 5 conjuntos.

				Mirzaei and Wøhlk (2017)			This paper			
				Exact		SA	VRPSolver		GRASP/TS	
Set	Demand	Cap.	#Inst.	#Opt	Perc.(%)	Gap %	#Opt	Perc.(%)	Gap best %	Gap %
1	u(1, 5)	10	44	28	64	9.9	44	100	2.3	3.2
2	u(1, 5)	15	24	1	4	5.0	22	92	2.3	3.1
3	u(1, 5)	20	24	1	4	11.4	20	83	1.8	2.7
4	bin	3	73	48	66	8.0	73	100	0.5	1.1
5	bin	4	24	1	4	13.1	24	100	0.6	1.4
Total			189	79	42	9.5	183	97	1.5	2.3

Table 6: Results of proposed algorithms for the instances proposed by Mirzaei and Wøhlk (2017)

Experimentos Computacionais

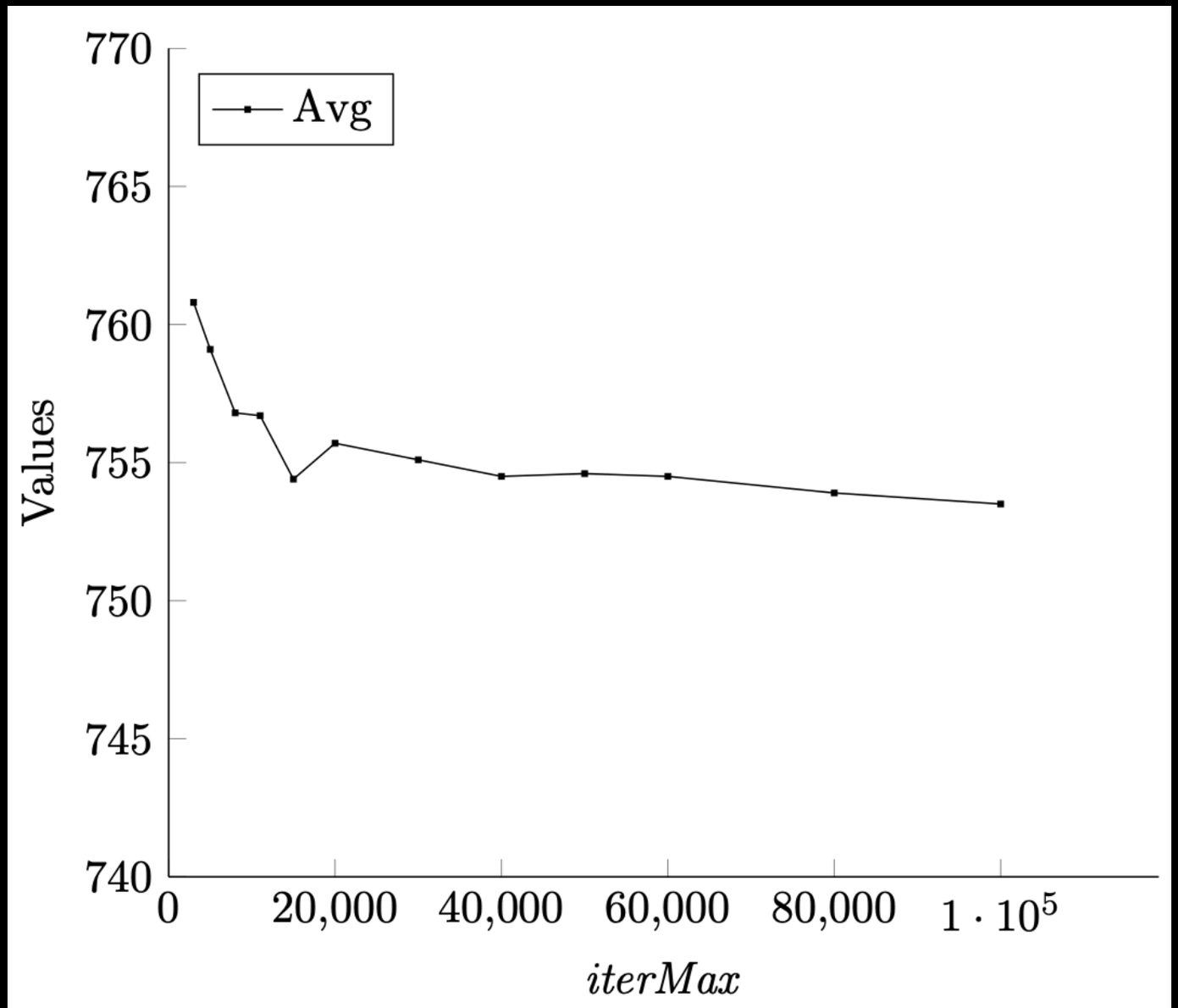


Figure 1: Effect of *iterMax* on the performance of GRASP/TS

OBRIGADO !!

Carlos Leonardo Ramos Póvoa D.Sc
Universidade Estadual do Norte Fluminense - UENF
Laboratório de Engenharia de Produção
clrp@uenf.br
(22)99984-4899

Lab55 Geo-Rota
http://www.loggeo.org