Submetido para TEMA

Um GRASP eficiente para Problemas de Roteamento de uma Frota de Veículos [1]

A. Tortelly Junior, L. Satoru Ochi[2], Instituto de Computação - Universidade Federal Fluminense, IC-UFF, 24210-240 Niterói, Rio de Janeiro, Brasil.

Resumo.

Apresentamos neste artigo, uma nova metaheurística híbrida baseada em conceitos de *Greedy Randomized Adaptive Search Procedure* (GRASP) e Busca Tabu (BT) para a solução do Problema de Roteamento Periódico de Veículos (PRPV). A busca local do algoritmo GRASP é efetuada através de um procedimento BT incorporando estratégias de intensificação e diversificação. Resultados computacionais obtidos de conjuntos de instâncias da literatura mostram que o algoritmo proposto é altamente competitivo quando comparado com as heurísticas e metaheuristicas existentes para o PRPV.

1. Introdução

Os Problemas de Roteamento e *Scheduling* de Veículos formam uma das áreas de maior sucesso nas áreas de Otimização Combinatória, Pesquisa Operacional e Computação Aplicada. Este sucesso em parte se deve a eficiência das técnicas existentes na literatura afim, que quando implementadas em situações reais, tem fornecido excelentes soluções do ponto de vista operacional. Mas devido a sua elevada complexidade, (em alguns modelos de Roteamento, o problema de gerar apenas uma solução viável já é classificado como NP-COMPLETO) o uso exclusivo de técnicas exatas tem se limitado apenas a pequenas instâncias do problema.A maioria das técnicas propostas na literatura utilizam heurísticas convencionais.

Somente nos últimos quinze anos surgiram muitos trabalhos utilizando heurísticas genéricas conhecidas como metaheurísticas. E dentre elas estão incluídas: os Algoritmos Evolutivos (AEs) e o seu representante mais popular; os Algoritmos Genéticos (AGs), as Redes Neurais Artificiais (RNs), *Simulated Annealing* (SA), Busca Tabu (BT), *Greedy Randomized Adaptive Search Procedure* (GRASP), entre outros. Neste artigo, propomos uma nova metaheurística híbrida utilizando conceitos de GRASP e Busca Tabu para a solução do Problema de Roteamento Periódico de Veículos (PRPV) que basicamente, consiste em resolver o Problema de Roteamento de Veículos diário (PRV) para um determinado horizonte de planejamento. Basicamente o método proposto possui a estrutura de um GRASP com a fase de busca

[1]Financiado parcialmente pelo CNPq.

[2]satoru@dcc.ic.uff.br

local sendo efetuada por um método de BT. Resultados computacionais mostram que o GRASP proposto apresenta um desempenho superior quando comparado com os algoritmos heurísticos da literatura em relação a qualidade das soluções geradas.

Este artigo apresenta nas próximas seções: a descrição e a literatura existente do PRPV, os algoritmos propostos, resultados computacionais e em seguida as conclusões e referências bibliográficas.

2. Descrição do PRPV

O PRPV básico consiste de uma frota homogênea de veículos que deve atender a um conjunto de clientes à partir de uma origem (depósito) de onde os veículos devem sair e retornar ao final da jornada. Cada veículo possui uma capacidade fixa que não pode ser excedida e cada cliente possui uma demanda conhecida que deve ser totalmente satisfeita numa única visita por um único veículo. O período de planejamento é de T dias. Quando T=1, o PRPV se restringe ao clássico Problema de Roteamento de Veículos (PRV). Cada cliente no PRPV deve ser visitado k vezes, onde k varia de 1 até T e no modelo clássico, a demanda diária de um cliente é sempre igual para cada dia de visita. O objetivo do PRPV pode ser visto como a de gerar um conjunto de rotas para cada dia de modo que as restrições envolvidas sejam atendidas e os custos globais minimizados.

O PRPV pode ser visto como um problema de otimização combinatória multi-nível. No primeiro nível, o objetivo é gerar um conjunto de alternativas (combinações) viáveis para cada cliente. Por exemplo, se o horizonte de planejamento for de $T = 3$ dias {d1, d2, d3} e se um cliente requer duas visitas, então este cliente possui as seguintes alternativas de visita: {d1,d2}, {d1,d3}, ou {d2,d3}(ou as opções: 6, 5, 3 da figura 2.1). No segundo nível, devemos selecionar uma alternativa de cada cliente de forma que as restrições diárias estejam satisfeitas. Desta forma, selecionamos para cada dia, os clientes a serem visitados. No terceiro nível, resolvemos um problema de roteamento de veículos para cada dia. Neste trabalho, o nosso algoritmo é aplicado ao modelo básico do PRPV incluindo a restrição adicional de que a cada dia existe uma quantidade de veículos (capacidade) limitada não necessariamente igual em todos os dias. Contudo enfatizamos que a técnica aqui proposta pode ser facilmente adaptada para outros problemas de otimização combinatória.

$0 \rightarrow 000$	$4 \rightarrow 100$
$1 \rightarrow 001$	$5 \rightarrow 101$
$2 \rightarrow 010$	$6 \rightarrow 110$
$3 \rightarrow 011$	$7 \rightarrow 111$

***Figura 2.1**. Alternativas de visitas possíveis para um horizonte de T=3 dias.*

3. Literatura do PRPV

Embora o PRPV não seja tão popular como o modelo básico de roteamento de veículos (PRV), ainda assim a literatura apresenta diversos trabalhos sobre PRPV e entre eles podemos citar os seguintes: Christofides e Beasley [2] propuseram duas heurísticas clássicas de construção para o PRPV. Tan e Beasley [10] propuseram um método que utiliza uma heurística de alocação generalizada. Outro trabalho sobre PRPV, foi proposto por Russel e Gribin [8]. Os autores desenvolveram um método de quatro fases; na primeira são geradas alternativas para cada cliente, na segunda fase a heurística de Christofides e Beasley [2] é utilizada para gerar cada rota. Na terceira fase, uma heurística de trocas é aplicada para as rotas atuais de cada veículo. Finalmente, na quarta fase, resolve-se um Modelo de Programação Inteira zero-um para tentar melhorar a solução atual. Golden, Chao e Wasil [5] desenvolveram uma heurística baseada no conceito de *"record-to-record"*.

De nosso conhecimento, ao contrário do Problema de Roteamento básico (PRV) já muito explorado, não existem muitos trabalhos utilizando Metaheurísticas para resolver o PRPV, e particularmente nenhum trabalho utilizando GRASP e Busca Tabu conjuntamente. Este trabalho apresenta um novo algoritmo metaheurístico híbrido reunindo conceitos de GRASP e Busca Tabu. A estrutura do algoritmo se baseia num GRASP, onde a fase de busca local é feita através de um método de Busca Tabu.

GRASP foi desenvolvido em 1995 por Tom Feo e Maurício Resende [7]. Sua metodologia consiste em um processo iterativo onde cada iteração possui duas fases: uma fase de construção, na qual uma solução viável é construída, seguida de uma fase de melhorias, cujo objetivo é encontrar uma solução ótima local. A fase de construção é iterativa, adaptativa, randomica e pode ser gulosa. Ela é iterativa porque a solução inicial é construída elemento a elemento. É adaptativa porque os benefícios associados a cada elemento são levados de uma iteração para outra, refletindo as mudanças ocasionadas pela seleção prévia de outros elementos e é gulosa porque a adição de cada elemento é feita a partir de uma lista restrita (LR) composta dos k melhores candidatos.

O componente probabilístico do GRASP é caracterizado pela escolha randômica de um dos elementos da lista LR, não necessariamente o melhor. A fase de melhorias consiste tipicamente em um procedimento de busca local, já que a solução gerada na fase de construção do GRASP pode não ser um ótimo local.

4. GRASP com Busca Tabu para o PRPV

Fase de Construção do GRASP

Um procedimento simples foi utilizado para montar cada solução inicial do PRPV, ou seja para gerar um conjunto de rotas para cada dia do horizonte de planejamento, respeitando as restrições associadas. Inicialmente uma alternativa de visita de cada cliente é escolhida aleatoriamente. A seguir, para cada dia do

horizonte de planejamento, um método das pétalas [6] foi implementado para gerar as rotas diárias, ou seja para resolver um PRV.

Este método é um procedimento de cobrir todo o conjunto de vértices do grafo associado a cada PRV diário através de uma rotação com centro no vértice origem e iniciando com um vértice destino selecionado de forma aleatória. A seguir efetua-se uma varredura circular no sentido horário ou anti horário em todo o plano de localização dos clientes até que todos os vértices destinos alocados para um determinado dia estejam agrupados em rotas.

Os clientes são inseridos nas rotas um a um de acordo com a distância radial entre eles. Se a inclusão de um novo cliente violar a capacidade do veículo, a rota atual é fechada e outra é aberta. Uma segunda versão do GRASP proposto, utiliza na fase de construção um filtro para selecionar as melhores soluções iniciais.

Neste caso, a cada iteração do GRASP são geradas k soluções iniciais (onde k é um parâmetro de entrada) usando o método das pétalas e somente a melhor solução do PRPV é selecionado para a fase de busca local do GRASP. Esta versão com filtro denominamos (GRASPf).

Busca Local do GRASP

A cada iteração do GRASP, uma etapa de busca local é efetuada utilizando um método de Busca Tabu (BT). O método BT proposto incorpora etapas de busca intensiva e etapas de diversificação utilizando memória de curto e longo prazo [4].

A solução inicial da BT (semente) é a solução gerada na fase de construção do GRASP utilizando o método das pétalas. A partir desta semente, o método BT passa a explorar diferentes regiões do conjunto de soluções. Cada região está associada a uma busca local do BT, onde uma busca intensiva é efetuada. A troca de regiões de busca representa uma etapa de diversificação.

A busca local do método BT equivale a verificar as alternativas de cada vértice de cada PRV diário trocar de rota e/ou trocar de rota e dia. No método BT proposto por Cordeau, Gendreau e Laporte [3], *cada vértice do PRV diário tenta ser inserido (ou permutado) em (com vértices de) cada uma das demais rotas de todos os dias do horizonte de planejamento.* Isso torna o processo de busca extremamente oneroso. Pensando nisso, propomos para cada rota diária *rd* , uma vizinhança *Vrd* desta rota composto pelos vértices das k - rotas *mais próximas* da rota *rd* (onde k é um dado de entrada). A partir daí, efetuamos dois tipos de buscas: A primeira permutando vértices de *rd* com vértices de *Vrd* e na segunda, tentando inserir vértices de *rd* nas rotas que definem *Vrd*. A melhor solução resultante desta busca local, será considerada a nova semente para inicializar uma nova busca local no método BT. Este procedimento será repetido m vezes, onde m é um parâmetro de entrada.

Observe que no nosso método BT, um movimento da solução semente a uma solução vizinha equivale a permutar componentes da lista tripla (i , r , d) [cliente i, alocado na rota r, no dia d]. Quando uma alocação (i , r , d) é efetuada, este movimento é mantido TABU durante um certo número de iterações da BT. Ou seja, o vértice i não poderá ser retirado da rota r do dia d durante algum tempo. A etapa de diversificação do nosso método BT é ativada quando se atualiza uma

iteração do algoritmo GRASP. Ou seja, após m buscas locais do método BT, uma nova semente é construída de forma independente através do método de construção do GRASP.

5. Implementação e Resultados Computacionais

Neste trabalho além do GRASP sem e com filtro propostos, (GRASP e GRASPf) implementamos uma versão do algoritmo de Busca Tabu de Cordeau, Gendreau e Laporte [3] aqui denominada (VCGL). A diferença entre a versão original (BTCGL) e a aqui implementada (VCGL) se refere a forma de construir uma solução inicial que não está clara no artigo [3]. Tanto na versão original como na nossa versão, inicialmente é selecionada de forma aleatória uma alternativa de visitas para cada cliente (vértice). Na solução de cada PRV diário, na nossa versão é utilizado uma heurística de inserção mais barata.

A segunda diferença está na busca local onde em [3] é usado o módulo da heurística GENIUS mas analisando todas as trocas entre todas as rotas de todos os dias enquanto na nossa versão (VCGL), utilizamos o mesmo procedimento usado na etapa de busca local do GRASP proposto, mas sem o uso do filtro que limita as permutações permitidas para cada vértice analisado (isto porque no GRASP, a cada iteração é ativado o módulo de busca local baseado num método BT). A justificativa para implementar uma variante da BT de CGL [3] ao invés da versão original são as seguintes. Primeiro, como já salientado, em [3] não está claro como a solução inicial é obtida. Segundo, desenvolver uma versão da BT de CGL que utilizasse o mesmo tipo de construção e busca local proposto ao GRASP.

Os algoritmos GRASP, GRASPf, e VCGL foram implementados na linguagem C e compilados com o Borland C++ Compiler 5.5 (BCC55). Foram utilizados instâncias retiradas da literatura e outros gerados aleatoriamente. As instâncias da literatura possuem a particularidade de algumas combinações de visitas serem fixas. Por exemplo, se um cliente necessita ser visitado duas vezes em um período de cinco dias, somente poderá ser visitado no primeiro e no terceiro dias, no segundo e no quarto ou no terceiro e no quinto. As outras combinações, como primeiro e segundo dias, não são permitidas. Também foram realizados testes com GRASP, GRASPf e VCGL, utilizando as mesmas instâncias, onde todas as combinações são permitidas, podendo o cliente ser visitado com qualquer combinação dentro de sua exigência de visitas no período (veja tabela 5.5). As Tabelas 5.1 e 5.2 mostram, respectivamente, as especificações dos dados da literatura e as combinações de visitas permitidas de acordo com o número de visitas exigido por cada cliente. Outras informações sobre estas instâncias, tais como as demandas e as coordenadas de cada cliente, podem ser vistas em [11]. O número de visitas de cada cliente e as combinações de visitas estão disponíveis em [3].

Propusemos, ainda, uma segunda bateria de testes com instâncias geradas aleatoriamente. Nesta bateria, variamos o número de clientes e, para cada grupo de dados com os mesmos clientes e mesma matriz de distâncias, alteramos o universo de dias do período e as capacidades dos veículos que compõem a frota diária. A Tabela

5.3 mostra as especificações dos dados gerados aleatoriamente. Para o VCGL [3] utilizamos, para os parâmetros do algoritmo, os mesmos valores sugeridos pelos autores. Os tempos (em minutos) contidos na Tabela 5.4 são os tempos totais dos testes com 10 iterações para o GRASP e GRASPf. O pequeno número de iterações GRASP igual a 10 se deve ao fato de o aumento deste número implicar, também, em um aumento considerável do tempo computacional total, devido ao tipo de busca local (Busca Tabu) utilizado. Além disso, nos testes foi verificado que realizar busca local em 10 soluções já foi suficiente para obter soluções satisfatórias. As Tabelas 5.4, 5.5 e 5.6 mostram as soluções obtidas para cada uma das instâncias descritas acima e os tempos computacionais de cada execução. Na Tabela 5.7, é mostrada a comparação entre os resultados obtidos pelo GRASPf, VCGL, CGL [3] e outras heurísticas da literatura: Christofides e Beasley (CB) [2], Tan e Beasley (TB) [10], Russel e Gribbin (RG) [8] e Golden, Chao e Wasil. (GCW) [5].

Problema	# de Clientes	# de Dias	# Veículos	Capac.Veíc.
1	50	3	3	160
2	50	5	3	160
3	50	5	1	160
4	75	2	5	140
5	75	5	6	140
6	75	10	1	140
7	100	2	4	200
8	100	5	5	200
9	100	8	1	200
10	100	5	4	200

Tabela 5.1*. Instâncias da literatura, onde # = número.*

A Tabela 5.5, possui apenas resultados das instâncias 2, 5, 8 e 10. Isto porque, segundo a tabela 5.2, apenas para estas instâncias, existe clientes que necessitam de visitas em mais de um dia no período.

Análise dos Resultados

A tabela 5.4 mostra os resultados obtidos pelas duas versões GRASP (sem e com filtro) e a versão da Busca Tabu de Cordeau, Gendreau e Laporte (VCGL) aqui implementado aplicados às instância da literatura. Estes testes mostram uma superioridade evidente da versão GRASPf na qualidade da solução obtida. Outro aspecto importante, é que a introdução do módulo de filtro no GRASPf, não onera o tempo final quando comparado com a versão GRASP sem filtro. Isso se explica pelo fato da geração de soluções não ocupar tempos significativos quando comparados com os tempos gastos na busca local. Além disso, como o GRASPf possui escolhas aleatórias, este pode ter tempos até menores que o GRASP.

Problema	Demandas	Combinações
1	qualquer demanda	1 visita no período
2	menor ou igual a 10	1 visita no período
	entre 11 e 25 (inclusive)	2 visitas (10100), (01010) ou (00101)
	menor ou igual a 26	visitas todos os dias
3	qualquer demanda	1 visita no período
4	qualquer demanda	1 visita no período
5	menor ou igual a 15	1 visita no período
	entre 16 e 27 (inclusive)	2 visitas (10100), (01010) ou (00101)
	menor ou igual a 28	visitas todos os dias
6	qualquer demanda	1 visita no período
7	qualquer demanda	1 visita no período
8	menor ou igual a 10	1 visita no período
	entre 11 e 25 (inclusive)	2 visitas (10100), (01010) ou (00101)
	menor ou igual a 26	visitas todos os dias
9	qualquer demanda	1 visita no período
10	menor ou igual a 10	1 visita no período
	entre 11 e 25 (inclusive)	2 visitas (10100), (01010) ou (00101)
	menor ou igual a 26	3 visitas (10101), (01011) ou (11010)

***Tabela 5.2**. Especificação das combinações dos dados da literatura.*

Problema	# clientes	# dias	# Veíc.	Capac. Veic.
11	60	3	4	140
12	60	6	2	190
13	60	3	3	160
14	70	3	4	120
15	70	3	6	140
16	70	5	1	240
17	80	3	2	230
18	80	4	3	230
19	80	6	1	230
20	90	2	5	170
21	90	4	2	400
22	90	6	1	290
23	120	4	3	310
24	120	3	5	310
25	120	3	4	310
26	150	5	3	310
27	150	5	2	390
28	200	7	2	270
29	200	3	4	420
30	300	3	2	900

***Tabela 5.3**. Especificação das instâncias geradas aleatóriamente.*

	GRASPf		GRASP		VCGL	
Problema	Custo	Tempo	Custo	Tempo	Custo	Tempo
1	**524,61**	4,75	527,67	4,97	526,71	7,08
2	**1325,77**	39,30	1340,04	39,57	1331,29	43,8
3	**524,61**	2,87	529,34	3,8	526,09	11,8
4	**839,90**	13,70	857,21	11,17	850,09	31,43
5	**2075,42**	50,36	2106,58	51,40	2085,42	220,75
6	**840,04**	12,60	850,68	11,98	846,02	77,7
7	**830,86**	21,43	832,01	19,20	831,06	44,4
8	**2046,45**	311,1	2100,50	296,95	2068,17	380,57
9	**828,68**	24,73	829,60	18,40	830,89	84,17
10	**1626,27**	76,62	1671,56	78,73	1647,03	176,75

Tabela 5.4*. Comparação dos melhores resultados obtidos com o GRASP (com e sem filtro) e o VCGL com as instâncias da literatura, onde os tempos são em minutos.*

Problema	GRASPf	GRASP	VCGL
2	**1325,52**	1333,14	1325,56
5	**2051,08**	2098,89	2065,98
8	**2034,94**	2046,38	2052,04
10	1605,05	**1598,47**	1611,23

Tabela 5.5*. Comparação dos melhores resultados obtidos com o GRASP e o BTCGL com os dados da literatura sem combinações fixas.*

A tabela 5.4 também nos mostra o porque de implementarmos uma versão do Tabu proposto em [3] ao invés da versão original. A meta aqui, foi implementar um Tabu nos moldes de [3], mas que utilizasse módulos similares aos utilizados pelos GRASP propostos. A tabela 5.5, ao contrário dos resultados da tabela 5.4, apresenta resultados de algumas das instâncias da literatura, mas relaxando as condições de visitas fixadas, como explicado anteriormente. Novamente verificamos que a performance do GRASPf foi superior que o GRASP e VCGL. A tabela 5.6, ilustra resultados do GRASP, GRASPf e VCGL agora para instância criadas aleatoriamente para o PRPV. Novamente percebemos uma ampla superioridade do GRASPf sobre o GRASP e deste sobre o VCGL. O objetivo neste caso, foi a de analisar o desempenho dos algoritmos em instâncias maiores. Finalmente a tabela 5.7, apresenta os resultados do GRASPf confrontando-os com os resultados das heurísticas existentes na literatura, incluindo aí, os resultados da versão original da Busca Tabu [3], aqui denominado CGL. Novamente o desempenho do GRASPf foi melhor que os demais, obtendo em 10 instâncias a melhor solução em 7 casos, seguido do CGL que obteve 5. Percebemos também que as alterações efetuadas no Tabu [3] prejudicou o seu desempenho, mas por outro lado a versão VCGL deve exigir um tempo computacional bem menor que CGL.

6. Conclusões

Este artigo apresenta uma nova metaheurística híbrida reunindo conceitos de GRASP e Busca Tabu para a solução do Problema de Roteamento Periódico de uma Frota de Veículos (PRPV). De nosso conhecimento não existe nenhuma contribuição para o PRPV usando conceitos de GRASP.

	GRASPf		GRASP		VCGL	
Problema	Custo	Tempo	Custo	Tempo	Custo	Tempo
11	**895,17**	2,57	900,28	2,38	900,15	24,33
12	**1358,08**	5,58	1362,38	7,83	1359,16	58,90
13	**894,69**	3,28	897,62	3,95	899,28	20,20
14	**837,77**	7,90	859,13	7,70	841,56	54,61
15	1375,42	14,30	**1365,53**	17,60	1409,90	76,68
16	708,54	4,83	919,50	4,30	**601,37**	64,22
17	329,61	9,60	345,63	8,27	**329,50**	28,20
18	773,84	23,57	**766,06**	26,58	794,05	84,08
19	**347,05**	12,43	363,40	10,13	348,54	49,83
20	1108,61	7,88	**1100,54**	9,13	1113,07	50,95
21	**1628,09**	20,37	1653,58	19,00	1636,60	55,08
22	**844,98**	18,52	846,50	19,00	856,85	50,47
23	954,64	40,03	**945,73**	50,43	967,85	115,83
24	**1658,70**	14,90	1680,02	20,60	1753,79	203,67
25	**948,15**	32,80	969,48	36,50	977,96	127,83
26	1426,64	50,83	**1423,35**	60,10	1472,56	72,50
27	**1616,46**	42,85	1635,16	44,30	1678,50	56,00
28	1613,95	91,90	**1604,27**	103,67	1619,27	129,51
29	**1630,95**	86,57	1748,97	86,63	1662,59	106,46
30	**2031,99**	186,90	2055,38	191,67	2442,94	374,83

***Tabela 5.6**. Comparação dos resultados obtidos com o GRASP e o VCGL com dados geradas aleatoriamente, onde os tempos são em minutos.*

Problema	GRASPf	VCGL	CGL	CB	TB	RG	CGW
1	**524,61**	526,71	**524,61**	457,4	-	537,3	524,6
2	**1325,77**	1331,29	1330,09	1443,1	1481,3	1355,4	1337,2
3	**524,61**	526,09	**524,61**	546,7	-	-	524,6
4	839,90	850,09	**837,94**	843,9	-	867,8	860,9
5	2075,42	2085,42	**2061,36**	2187,3	2192,5	2141,3	2089,0
6	**840,04**	846,02	840,30	938,2	-	-	881,1
7	830,86	831,06	**829,37**	839,2	-	833,6	832,0
8	**2046,45**	2068,17	2054,90	2151,3	2281,8	2108,3	2075,1
9	**828,68**	830,89	829,45	875,0	-	-	829,9
10	**1626,27**	1647,03	1629,96	1674,0	1833,7	1638,5	1633,2

***Tabela 5.7**. Comparação entre os resultados (custos) obtidos com o GRASPf; o VCGL e outras heurísticas da literatura.*

Isso foi uma motivação para o desenvolvimento deste trabalho onde colocamos como objetivo, mostrar que um GRASP acoplado a módulos eficientes de busca local pode gerar soluções altamente competitivas para o PRPV. Desta forma, reunindo o bom desempenho de versões GRASP com filtro na fase de construção, e acoplada a uma forma eficiente de efetuar uma busca local através de um método BT, provamos que a versão híbrida aqui proposta possui potencial para obter bons limites superiores para estes tipos de problemas altamente combinatórias.

Abstract.

This paper presents a hybrid metaheuristic based in concepts of GRASP and Tabu Search (TS) to the solution of the PVRP. The local search of GRASP algorithm is a TS procedure including intensification and diversification strategies. Computational results for sets of benchmark instances are reported. The results indicate that the proposed algorithm is highly competitive with all existing heuristic and metaheuristics for the PVRP.

Referências

[1] Beltrami,E. and Bodin,L., Networks and Vehicle Routing for Municipal Waste Collection, Networks 4, pp: 65 - 94, 1974.

[2] Christofides,N. and Beasley,J., The PRP, Networks 14, 237 - 256, 1984.

[3] Cordeau, F., Gendreau, M., and Laporte, G., A Tabu Search heuristic for period reuting problems., Networks 30, 105-119, 1997.

[4] Glover, F. and Laguna, M., Busca Tabu . Kluwer Academic Pub. 1998.

[5] Golden,B.L.; Chao,I.M. and Wasil,E., An Improved Heuristic for the Period Vehicle Routing Problem. Networks, pp: 25 - 44, 1995.

[6] Ochi, L. S., Drummond, L. M. A., Vianna, D. S., Victor, A. O. , A Parallel Evolutionary Algorithm for the VRP, Future Generation Computer Systems Journal, Elsevier Science, Vol. 14, pag.: 285-292, 1988.

[7] Resende, M., Pitsoulis, L. S., GRASP, in Handobook of Applied Optimization, OXFORD Univ. Press, pp. 168-182, 2002.

[8] Russell,R.A. and Gribbin,D., A Multi-Phase Approach to the Period Routing Problem. Networks 21, pp: 747 - 765, 1991.

[9] Silva, M., Drummond, L., and Ochi, L.S., Metaheuristics based on GRASP and VNS for solving the Traveling Purchaser Problem, Proc. of the IV Metaheuristic International Conference, 12-17, Porto, Portugal, 2001.

[10] Tan, C. C. R. and Beasley,J. E., A Heuristic Algorithm for the Period Vehicle Routing Problem. Omega 12(5), pp: 497 - 504, 1984.

[11] Instâncias do PRPV: Disponível em http://mscmga.ms.ic.ac.uk/jeb/orlib/