ITERATED LOCAL SEARCH PARA O PROBLEMA DO CAIXEIRO VIAJANTE COM GRUPAMENTOS

Mário Mestria
Coordenadoria da Área de Informática - Campus Colatina- Instituto Federal do Espírito Santo
Avenida Arino Gomes Leal - 1700 - Bairro Santa Margarida, Colatina, ES, Brasil
e-mails: {mmestria@ic.uff.br, mmestria@ifes.edu.br, mmestria@uol.com.br}

Luiz Satoru Ochi
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 3° andar - São Domingos
Niterói, RJ, Brasil, CEP: 24210-240
e-mail: satoru@ic.uff.br

Simone de Lima Martins
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 3° andar - São Domingos
Niterói, RJ, Brasil, CEP: 24210-240
e-mail: simone@ic.uff.br

Resumo: Nesse artigo, propomos novos algoritmos para resolver o Problema do Caixeiro Viajante com Grupamentos (PCVG). O PCVG é uma generalização do Problema do Caixeiro Viajante (PCV), onde os vértices são particionados em grupos disjuntos e o objetivo é encontrar um ciclo hamiltoniano de custo mínimo tal que os vértices de cada grupo são visitados de forma contígua. Foram propostos e implementados dois algoritmos heurísticos baseados no *Iterated Local Search* e no Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*) (ILS-VND) para resolver o PCVG. O desempenho dos algoritmos propostos é comparado entre si e com um algoritmo exato usando o *software* CPLEX Paralelo. Os algoritmos propostos foram testados em instâncias com até 2000 vértices e grupos variando entre quatro a 150 vértices. Resultados computacionais mostram que a heurística ILS-VND com um algoritmo construtivo no qual não aplica uma penalização nas arestas intergrupos foi capaz de obter soluções de melhor qualidade.
Palavras-chave: Otimização Combinatória, Problema do Caixeiro Viajante com Grupamentos, Algoritmos Heurísticos, Método de Descida em Vizinhança Variável, *Iterated Local Search*.

Abstract: In this paper, we propose new algorithms to solve the Clustered Traveling Salesman Problem (CTSP). The CTSP is a generalization of the Traveling Salesman Problem (TSP) in which the set of vertices is partitioned into clusters and the objective is to find a minimum cost Hamiltonian cycle such that the vertices of each cluster are visited continuously. It is proposed and implemented two heuristic algorithms based on Iterated Local Search and Variable Neighborhood Descent (ILS-VND) to solve the CTSP. The performance of the proposed algorithms is compared among them and then with an exact algorithm using the Parallel CPLEX software. The proposed algorithms were tested in instances with up to 2000 vertices and clusters varying between four to 150 vertices. Computational results show that ILS-VND heuristic with a constructive algorithm which not applies a penalization for inter-clusters edges was able to obtain the best quality solutions.
Keywords: Combinatorial Optimization, Clustered Traveling Salesman Problem, Heuristic Algorithm, Variable Neighborhood Descent, Iterated Local Search.

1. INTRODUÇÃO

O Problema do Caixeiro Viajante (PCV) é um dos problemas mais estudados na área de pesquisa operacional. Uma variante do PCV pouco estudada na literatura denominada de Problema do Caixeiro Viajante com Grupamentos (PCVG) é o foco deste artigo. O PCVG surgiu ao se estudar um problema de roteamento automático em sistemas de armazenagem (CHISMAN, 1975). O objetivo do PCVG é encontrar um ciclo hamiltoniano de custo mínimo que passe por todos os vértices do grafo associado e visita os vértices de cada grupo de forma contígua. No caso particular onde o número de grupos é igual a um, o PCVG se reduz ao PCV. Desta forma pode-se afirmar que o PCVG também pertence à classe *NP-Difícil* limitando com isso o uso exclusivo de métodos exatos. Neste contexto a solução proposta neste trabalho para o PCVG utilizará algoritmos heurísticos.

Na literatura existem várias aplicações que podem ser modeladas através do PCVG, como por exemplo: roteamento automático em armazéns (CHISMAN, 1975), planejamento da produção (LOKIN, 1979), despacho de veículos de emergência, (WEINTRAUB *et al*., 1999), sistemas de manufaturas (LAPORTE; PALEKAR, 2002), desfragmentação de discos rígidos e aplicações comerciais envolvendo supermercados, lojas e fornecedores de mercadorias (GHAZIRI; OSMAN, 2003). Apesar de várias aplicações serem encontradas na literatura, este tipo de problema é pouco explorado. Duas versões são utilizadas para resolver o PCVG, uma no qual se define *a priori* a sequência dos grupos a serem visitados (GENDREAU; LAPORTE; POTVIN, 1994), (POTVIN; GUERTIN, 1995), (LAPORTE; POTVIN; QUILLERET, 1996) e (ANILY; BRAMEL; HERTZ, 1999) e uma outra, cuja sequência de visita dos grupos deve ser definida pelo algoritmo. Assim, verifica-se que a segunda versão é mais complexa e geralmente mais próxima dos modelos reais, e foi selecionada para o estudo neste trabalho.

O PCVG proposto por Chisman (1975) foi utilizado para resolver um problema real de roteamento em sistemas de armazenagem, onde as ordens de despacho de produtos devem ser respeitadas e estas ordens contêm várias subordens. Cada subordem estabelece os produtos a serem despachados e quantidade de estoque dos produtos. Um veículo motorizado despachará através do armazém coletando os produtos estocados especificados em cada subordem. A restrição nos sistemas de armazenagem especifica que uma subordem deverá ser completamente despachada antes de a próxima ser iniciada. A ordem de coleta dos produtos estocados especificados em cada subordem e a ordenação das subordens é simultaneamente otimizada. Este tipo de logística foi modelado como um PCVG, onde os vértices são as localizações dos estoques dos produtos, as arestas são as distâncias a serem percorridas pelo veículo motorizado e os grupos são as subordens. O PCVG foi transformado num PCV adicionando-se uma penalidade M aos custos das arestas que ligam quaisquer dois vértices pertencentes a grupos diferentes e resolvido por um algoritmo exato *branch-and-bound*.

Para o PCVG, Jongens e Volgenant (1985) utilizaram *Relaxação Lagrangeana* para obter limites inferiores. O método para obtenção do limite inferior baseou-se na árvore geradora mínima desenvolvida primeiramente para o PCV. Foram criados vários conjuntos de instâncias variando o número de vértices de 80 a 150 com diferentes tamanhos de grupos. O algoritmo de Jongens e Volgenant (1985) obteve para instâncias testes, *gap* médio para os limites inferiores igual a 0,35%.

Uma Busca Tabu combinada com uma fase de diversificação usando um Algoritmo Genético foi proposta em Laporte, Potvin e Quilleret (1996) para o PCVG. Nesta Busca Tabu a sequência de visita dos grupos é definida *a priori* e os testes computacionais apresentados mostram que a Busca Tabu supera o Algoritmo Genético (AG) de Potvin e Guertin (1995) na qualidade da solução. A comparação dos tempos computacionais da Busca Tabu e AG não foi especificada. Os autores Potvin e Guertin (1996) desenvolveram um Algoritmo Genético para o PCVG sem a pré-fixação da sequência dos grupos, onde este algoritmo trata de forma

independente a sequência dos grupos e dos vértices em cada grupo, dando prioridade à solução intergrupos e em seguida a solução intragrupos.

No artigo proposto em Ding, Cheng e He (2007) um outro AG, sem incorporar módulos de busca local, usa o critério de visitação dos vértices de cada grupo de forma aleatória sem estabelecer uma relação com as distâncias entre os vértices. A escolha da sequência de visita de cada grupo também é aleatória. O AG foi dividido em dois níveis chamado de nível baixo e nível alto. No nível mais baixo, o AG acha um ciclo hamiltoniano para cada grupo. No nível mais alto, o algoritmo escolhe aleatoriamente uma aresta a ser excluída no ciclo de cada grupo e simultaneamente determina a sequência de visita de todos os grupos de forma aleatória.

No trabalho de Mestria, Ochi e Martins (2009a) foram desenvolvidas várias heurísticas para o PCVG. Um dos objetivos deste trabalho foi verificar o impacto ao se incluir na heurística GRASP, módulos de memória adaptativa e busca local mais eficiente utilizando conceitos da técnica de reconexão de caminhos e da heurística *Variable Neighborhood Search* (VNS). A versão híbrida que incorpora GRASP, reconexão de caminhos e o método de Descida em Vizinhança Variável (VND) obtiveram melhor desempenho em relação às outras versões GRASP. Uma única forma de construir soluções ao PCVG baseada na Inserção mais Próxima foi desenvolvida.

Em Mestria, Ochi e Martins (2009b) quatro heurísticas foram implementadas para o PCVG onde a sequência dos grupos não é estabelecida em princípio. A primeira versão corresponde ao GRASP tradicional e as outras três utilizaram memória adaptativa através do uso de uma técnica denominada reconexão de caminhos. Dois tipos diferentes de reconexão de caminhos foram desenvolvidos, uma (RC2) realizada durante as iterações do GRASP e outra (RC1) realizada depois das iterações do GRASP. Na etapa de busca local foi utilizado somente o método 2-Optimal. As versões que utilizaram reconexão de caminhos (RC1) foram aquelas que conseguiram o melhor desempenho.

A metaheurística *Iterated Local Search* (ILS) (LOURENÇO; MARTIN; STÜTZLE, 2002) mostrou-se promissora para diferentes problemas de natureza combinatória. Os fundamentos do ILS foram introduzidos por Martin, Otto e Felten (1992) e Martin e Otto (1996), que aplicaram ao PCV usando instâncias Euclidianas. Como busca local foi utilizada uma heurística *k*-Optimal e para o método de perturbação o movimento *double bridge*. ILS possui princípios simples, é fácil de implementar e possibilita o armazenamento de informações históricas da estrutura da solução ao longo do algoritmo. A metaheurística ILS será utilizada neste trabalho para solucionar o PCVG.

Uma outra metaheurística *Variable Neighborhood Descent* (VND) (HANSEN; MLADENOVIĆ, 2003), muitas vezes utilizadas como busca local, também se mostra como promissora para problemas de otimização combinatória. Em Hernández-Pérez, Rodríguez-Martín e Salazar-González (2009) foi apresentada uma heurística GRASP (RESENDE; RIBEIRO, 2002) combinada com a heurística VND para um problema de roteamento. A estrutura do VND não é complexa, de fácil implementação e seus componentes de vizinhanças podem ser alterados aumentando ou diminuindo sua complexidade para se atingir um compromisso entre qualidade da solução e tempo computacional exigido. Assim, propõe-se neste trabalho, utilizar o VND como módulo de busca local a ser incorporado nos algoritmos heurísticos para resolução do PCVG.

Neste trabalho foram desenvolvidos dois algoritmos heurísticos. O primeiro algoritmo (ILS-VND-P) implementa uma heurística ILS, acoplada à heurística VND para a busca local e um algoritmo construtivo baseado na Inserção mais Próxima que penaliza as arestas intergrupos (IMPP). O segundo algoritmo (ILS-VND-nP) é idêntico ao primeiro, exceto que o algoritmo construtivo não penaliza as arestas intergrupos (IMPnP). O artigo é estruturado conforme as seguintes seções. A segunda seção descreve os algoritmos heurísticos propostos para o PCVG. A terceira seção mostra os resultados computacionais e a finalmente na última seção são descritos conclusões e trabalhos futuros.

2. PROPOSTAS DOS ALGORITMOS HEURÍSTICOS PARA O PCVG

Os algoritmos heurísticos apresentados neste artigo diferem da maioria dos trabalhos da literatura, porque parte do princípio que não é pré-estabelecida nenhuma sequência de visita aos grupos e incorporam duas formas diferentes para construir soluções. A primeira forma de construir soluções penaliza as arestas intergrupos (IMPP) e neste caso converte o PCVG no PCV. A segunda forma não estabelece a penalização (IMPnP) e soluciona o PCVG na sua forma original. Ambas as formas de construir as soluções foram adaptadas do método de Inserção mais Próxima usando os conceitos do GRASP reativo que tem apresentado bons resultados (RESENDE; RIBEIRO, 2002). Na fase de busca local, os algoritmos heurísticos utilizam o VND que é caracterizado por trocas sistemáticas de vizinhanças. À medida que uma estrutura de vizinhança não consegue melhorar a solução corrente, permuta-se a estrutura de vizinhança. Caso haja melhoria na solução corrente, retorna-se a primeira estrutura de vizinhança.

Foram utilizadas quatro estruturas de vizinhanças para a implementação da busca local VND. A primeira (*N1*) é obtida através de um movimento de deslocamento do vértice pertencente ao ciclo de uma posição para outra, denominada de 1-*Shift*. Para definir a segunda estrutura (*N2*) precisam-se definir dois tipos de procedimentos: *Drop* e *Cheap*. O procedimento *Drop* varre o ciclo de uma solução e retira deste, o vértice que proporcionará a maior economia com sua retirada. O procedimento *Cheap* insere um vértice não pertencente ao ciclo parcial, entre dois vértices adjacentes pertencentes ao ciclo, cuja inserção permita um custo adicional mínimo à solução. O ciclo construído após a inserção do vértice deverá permanecer viável. Diante destas duas definições de procedimentos pode-se estabelecer a segunda estrutura de vizinhança (*N2*), 1-*DropCheap*, cuja definição é a seguinte: uma sequência de procedimentos *Drop* e *Cheap* é realizada sobre uma solução até que nenhum melhoramento seja encontrado. A terceira (*N3*) é obtida através de permutas entre dois vértices de um mesmo grupo (intragrupos), 2-Swap. A quarta (*N4*) utiliza a clássica heurística 2-Optimal para trocar as arestas intragrupos e as intergrupos. Quando são retiradas às arestas intragrupos é reconstruída uma nova solução e somente a sequência dos vértices dentro do grupo é alterada. Ao trocar as arestas intergrupos, somente a sequência dos grupos é alterada.

Neste trabalho adotam-se as seguintes estratégias para reduzir o esforço computacional decorrente do uso das estruturas de vizinhanças. Quando são utilizadas as vizinhanças *N1* e *N3*, adota-se a estratégia *primeiro aprimorante*, ou seja, o primeiro movimento que melhora a solução corrente é atualizado, enquanto para *N2* e a *N4*, adota-se o método *melhor aprimorante* (analisam-se todas as soluções vizinhas e seleciona-se a melhor).

```
1: S^0 ← SolucaoInicial;
2: S ← BuscaLocal(S^0);
3: repita
4:     S' ← Perturbacao(S, história);
5:     S'' ← BuscaLocal(S');
6:     S ← CriterioAceitacao(S, S'', história);
7: até critério de parada não satisfeito
8: retornar S;
```

Figura 1 - Pseudocódigo para heurística *Iterated Local Search*

A metaheurística ILS vem se mostrando eficiente para obter soluções aproximadas de boa qualidade para problemas de otimização combinatória (LOURENÇO; MARTIN; STÜTZLE, 2002) e (SUBRAMANIAN *et al*., 2010). Sua estrutura canônica é formada por quatro módulos principais: construção de uma solução, busca local, perturbação e critério de aceitação. ILS explora o espaço de soluções de ótimos locais de modo descrito a seguir. Seja a

solução corrente *S* a qual aplica-se uma perturbação que permite um estado intermediário *S*'. Sobre *S*', obtém-se *S*'' através de uma busca local. Se a solução *S*'' for considerada pelo critério de aceitação, esta se tornará a próxima solução a sofrer a perturbação, caso contrário é retornada a solução *S*. O espaço continua a ser explorado através da perturbação até que um critério de parada escolhido, geralmente número máximo de iterações, seja encontrado. Na Figura 1 é mostrado o pseudocódigo para a heurística ILS adaptado de (LOURENÇO; MARTIN; STÜTZLE, 2002).

O primeiro algoritmo proposto (ILS-VND-P) utiliza os conceitos de ILS, adotando a heurística VND na fase de busca local e na fase de construção a IMPP e o segundo algoritmo (ILS-VND-nP) combina ILS, VND e IMPnP. A etapa de perturbação (*Pert*) consiste na realização do movimento *double-bridge* aplicado às arestas de um grupo escolhido aleatório. A sequência dos grupos não é alterada, somente a sequência dos vértices do grupo escolhido é modificada. Este tipo de perturbação foi utilizado no PCV com sucesso, mas aplicada em todo o ciclo sem restringir quaisquer arestas (MARTIN; OTTO; FELTEN, 1992). Uma segunda perturbação (*Pertdg*) também foi desenvolvida na qual se realiza uma troca aleatória no percurso de *dois grupos* sem alterar os vértices internos dos grupos. Cabe ressaltar que a *Pertdg* será utilizada, caso não seja possível aplicar a *Pert*.

```
1: iterprob ←1;
2: inialfas( ); {incializar a distribuição de probabilidade dos alfas}
3: S^0 ← Insercao_Mais_Proxima_X;
4: S* ← S^0;
5: para i=1 até maxext faça
6:       S^0 ← Insercao_Mais_Proxima_X;
7:       S ← VND(S^0);
8:       para i=1 até maxint faça
9:          S' ← Pert(S);
10:         S'' ← VND(S');
11:         S ← CritAceit(S, S'', S*);
12:         se S for melhor que S* então
13:              S* ← S;
14:         fim se
15:      fim para
16:      se iterprob = maxprob então
17:         atualalfas( ); {atualizar a distribuição de probabilidade dos alfas}
18:         iterprob ← 0;
19:      fim se
20:      iterprob ← iterprob + 1;
21: fim para
22: retornar S*;
```

Figura 2 - Pseudocódigo para os algoritmos com ILS (ILS-VND-X)

O critério de aceitação (*CritAceit*), utilizado na etapa seguinte da busca local, considera três soluções: duas soluções ótimas locais, *S* e *S*'', descritas anteriormente e a solução S^*, a melhor solução até a etapa corrente do algoritmo. Para este critério de aceitação, a solução que tiver o menor custo entre as três soluções (S, S'', S^*) será escolhida como solução para a etapa seguinte da perturbação. Nos algoritmos heurísticos propostos utilizando a metaheurística ILS foram verificadas que a partir de um número fixo de iterações, denominado de iterações internas, os métodos não conseguem atualizar a melhor solução. Então a partir de certo número de iterações internas sem melhorar a solução corrente, constrói-se uma nova solução e realizam-se todas as etapas de perturbação, busca local e

critério de aceitação. O número de vezes que o algoritmo reinicia a construção de soluções novas, depois de um número máximo de iterações internas (*maxint*), é considerado para o algoritmo como as iterações externas. Estabeleceu-se um número máximo de iterações externas (*maxext*) para os algoritmos ILS-VND-P e ILS-VND-nP.

O pseudocódigo dos algoritmos com ILS desenvolvidos neste artigo para o PCVG é ilustrado na Figura 2 para o caso geral (ILS-VND-X), onde X representa a penalização (P) ou não (nP). No passo 2, a distribuição de probabilidade dos alfas *inialfas*( ) é obtida e a melhor solução é inicializada no passo 4. Para cada iteração externa do ILS-VND-X a solução é construída no passo 6 pelo procedimento *Inserção_Mais_Proxima_X* (IMPP ou IMPnP) e, no passo 7, a busca local é executada pelo *VND*. Para cada iteração interna do ILS-VND-X, a perturbação é aplicada no passo 9 e a busca local é executada novamente pelo *VND* no passo 10. No passo 11 aplica-se o critério de aceitação e a melhor solução é atualizada no passo 13. No passo 17, a distribuição de probabilidade dos alfas *atualalfas*( ) é atualizada quando o parâmetro *iterprob* atingir o valor máximo igual a *maxprob*. Finalmente, no passo 22 a melhor solução é retornada.

3. RESULTADOS COMPUTACIONAIS

Na literatura existem métodos exatos e heurísticos para o PCVG, mas de nosso conhecimento não se encontra nenhuma biblioteca pública com instâncias para este problema na sua versão genérica, sem pré-fixação da sequência de visitas aos grupos. Assim, tornou-se necessária a criação de um conjunto de instâncias para o PCVG genérico para avaliar e validar os algoritmos propostos. Foram gerados seis tipos diferentes de instâncias nos quais os vértices foram agrupados em diversas configurações (MESTRIA; OCHI; MARTINS, 2010). Todas as instâncias geradas nestas configurações são Euclidianas e podem ser acessadas em ftp://ftp.cefetes.br/Teses_Dissertacoes/MarioMestria/instances, onde constam os valores ótimos utilizando a formulação exata da literatura e os melhores valores alcançados pelas heurísticas.

O desempenho empírico dos algoritmos heurísticos propostos (ILS-VND-P e ILS-VND-nP) foram comparados às soluções ótimas ou aos limites inferiores obtidos pelo método exato usando a formulação matemática da literatura proposta em Chisman (1975) e o software CPLEX11.2 Paralelo (CPLEX, 2009). O CPLEX foi executado num computador Intel Core 2 Quad, 2.83 GHz com 8 GB de RAM com 4 núcleos. O sistema operacional utilizado foi Linux Ubuntu versão 4.3.2-1. A implementação dos algoritmos heurísticos foi codificada em linguagem de programação *C* e executada no mesmo computador descrito anteriormente, mas utilizando somente um núcleo.

Os principais parâmetros utilizados nos algoritmos heurísticos são descritos a seguir. O valor de penalização dos custos das arestas *M* foi definido como $10*max\{c_{ij}\}$. A expressão $max\{c_{ij}\}$ significa o maior custo entre todos os custos das arestas. Os valores de alfa (α) na etapa de construção da solução estão no intervalo de [0,1] utilizados através da estratégia reativa. O número máximo de iterações internas (*maxint*) foi definido como 35, o número máximo de iterações externas (*maxext*) como 40 e o número de iterações para atualizar as probabilidades do parâmetro alfa (*maxprob*) como 10. Devido à natureza aleatória dos algoritmos heurísticos, estes foram executados dez vezes para cada instância.

Para avaliar os algoritmos heurísticos foram realizados testes em primeiro lugar para instâncias de pequeno porte que são mostrados na Tabela 1. A comparação entre os algoritmos propostos foi efetuada com método exato CPLEX sem limitação de tempo para o CPLEX. Para este conjunto de instâncias, o CPLEX encontrou valores ótimos para todas as instâncias. Para os algoritmos heurísticos o critério de parada foi estabelecido pelo número de iterações (*maxint* com o valor de 35 e *maxext* igual a 40). Na primeira coluna da Tabela 1 encontra-se a instância, na segunda o valor ótimo obtido pelo CPLEX e na terceira o tempo demandado pelo CPLEX em segundos t_{CPLEX}(s). Na quarta coluna o gap_{hc} de ILS-VND-P, representado por ILS-P(%), entre a solução alcançada e o valor ótimo e na quinta o tempo

médio de execução $t_{\text{ILS-P}}$(s) de ILS-P medido em segundos. Na sexta e sétima colunas são mostradas o gap_{hc} do melhor valor obtido por ILS-VND-nP (ILS-nP(%)) e o tempo médio demandado pelo ILS-nP em segundos $t_{\text{ILS-P}}$(s), respectivamente. Na última coluna o tipo de perturbação realizada pelos algoritmos ILS-VND.

Os resultados apresentados na Tabela 1 mostram que o gap_{hc} médio dos melhores valores alcançados de ***ILS-VND-P ficou em 0,10% com o tempo médio total de 11,37s***. O algoritmo ***ILS-VND-P encontrou 21 soluções ótimas***. O algoritmo ILS-VND-nP conseguiu alcançar 17 soluções ótimas com gap_{hc} médio dos melhores valores igual a 0,27% e tempo médio total igual a 12,05s. Estes resultados mostram o potencial dos algoritmos heurísticos em encontrar soluções de boa qualidade em tempo computacional viável. A razão para executar a perturbação *Pertdg* em algumas instâncias deve-se ao fato que estas instâncias não têm vértices suficientes nos grupos para realizar a perturbação *Pert*, pois devem existir no mínimo oito vértices em um grupo para realizá-la. A *Pertdg* foi realizada em 11 instâncias em um total de 27.

Tabela 1 - Comparação do CPLEX com os algoritmos heurísticos para instâncias do tipo 1 de pequeno porte

Instâncias	**CPLEX**	t_{CPLEX} **(s)**	**ILS-P (%)**	$t_{\text{ILS-P}}$ **(s)**	**ILS-nP (%)**	$t_{\text{ILS-nP}}$ **(s)**	**tipo de perturbação**
5-eil51	437	12,31	**0,00**	5,20	**0,00**	3,80	*Pert*
10-eil51	440	74,38	**0,00**	4,40	**0,00**	5,50	*Pertdg*
15-eil51	437	2,04	**0,00**	4,90	**0,00**	5,00	*Pertdg*
5-berlin52	7991	201,80	**0,00**	2,90	0,11	4,50	*Pert*
10-berlin52	7896	89,17	**0,00**	2,50	0,52	5,10	*Pert*
15-berlin52	8049	75,93	**0,00**	4,30	**0,00**	4,00	*Pert*
5-st70	695	13790,11	**0,00**	5,40	**0,00**	6,90	*Pert*
10-st70	691	4581,00	**0,00**	6,60	0,43	3,70	*Pert*
15-st70	692	883,50	**0,00**	5,40	0,87	6,40	*Pert*
5-eil76	559	83,70	**0,00**	6,70	**0,00**	5,10	*Pert*
10-eil76	561	254,30	0,36	5,50	1,07	8,60	*Pert*
15-eil76	565	49,66	**0,00**	6,90	0,88	6,80	*Pert*
5-pr76	108590	99,29	**0,00**	8,20	**0,00**	6,30	*Pert*
10-pr76	109538	238,13	**0,00**	8,40	**0,00**	8,30	*Pert*
15-pr76	110678	261,94	0,49	10,00	**0,00**	9,70	*Pert*
10-rat99	1238	650,67	**0,00**	13,40	**0,00**	12,60	*Pert*
25-rat99	1269	351,15	**0,00**	23,80	**0,00**	21,40	*Pertdg*
50-rat99	1249	2797,58	**0,00**	23,40	**0,00**	18,60	*Pertdg*
25-kroA100	21917	3513,57	**0,00**	18,40	**0,00**	21,70	*Pertdg*
50-kroA100	21453	947,55	**0,00**	19,90	**0,00**	21,70	*Pertdg*
10-kroB100	22440	4991,44	0,16	9,40	0,91	12,40	*Pert*
50-kroB100	22355	2579,22	**0,00**	17,70	**0,00**	22,40	*Pertdg*
25-eil101	663	709,45	0,45	21,20	1,06	21,10	*Pertdg*
50-eil101	644	275,33	1,09	21,10	0,16	20,80	*Pertdg*
25-lin105	14438	6224,55	**0,00**	8,10	1,38	16,70	*Pert*
50-lin105	14379	1577,21	**0,00**	21,50	**0,00**	22,60	*Pertdg*
75-lin105	14521	15886,77	0,15	21,90	**0,00**	23,70	*Pertdg*
Val. médios	-	2266,73	***0,10***	***11,37***	0,27	12,05	-

Observa-se que o tempo médio de 12,05s do ILS-VND-nP foi um pouco maior do que o tempo médio de 11,37s do ILS-VND-P. Isto ocorre porque na heurística construtiva IMPnP, a Lista de Candidatos que auxilia a elaboração da solução parcial considera em cada etapa

todos os vértices pertencentes ao mesmo grupo dos vértices já contidos na solução parcial. Em oposição, IMPP considera somente os k=10 vértices mais próximos independentes que pertençam ou não ao mesmo grupo dos vértices da solução parcial já construída.

Um segundo teste foi realizado para instâncias de *tipos* diferentes mostradas na Tabela 2. A primeira coluna da Tabela 2 mostra as instâncias, seguida com os seus identificadores (coluna 2), número de vértices (nós), número de grupos e tipo. Mostram-se ainda as soluções e os limites inferiores gerados pelo software CPLEX para as instâncias relacionadas. Na última coluna é mostrado o gap_{li} (%) entre valor da solução e o limite inferior. Devido às longas iterações ocorridas no CPLEX para a maioria das instâncias, determinamos um tempo limite de 7200s (duas horas) para sua execução. Para a instância J_{18}, a solução ótima foi encontrada pelo CPLEX em 442s. Todas as outras execuções utilizaram 7200s sendo, portanto abortadas antes de se encontrar ou confirmar uma solução ótima.

Para as instâncias da Tabela 2, os testes computacionais foram realizados utilizando-se como critério de parada um tempo máximo idêntico para os algoritmos heurísticos. Os resultados destes testes são mostrados na Tabela 3. Foi utilizado um tempo limite de 720 segundos para as heurísticas, porque para cada instância, as heurísticas são executadas 10 vezes e a melhor solução encontrada nestas execuções por duas horas é retornada, assim como o valor médio. Somente a execução da instância J_{18} não foi limitada a duas horas, devido à solução ótima ser encontrada antes.

Tabela 2 - Instâncias com seus identificadores, número de nós, número de grupos, tipo e valores encontrados pelo CPLEX

Instâncias	**Id.**	**# nós**	**# Vi**	**Tipo**	**CPLEX**		
					Valor	**Lim. Inf.**	**(%)**
50-gil262	J_1	262	50	1	135529	135374,68	0,11
10-lin318	J_2	318	10	1	534640	526412,07	1,54
10-pcb442	J_3	442	10	1	547152	536478,33	1,95
C1k.0	J_4	1000	10	2	134025123	131354923,50	1,99
C1k.1	J_5	1000	10	2	130750874	128540131,50	1,69
C1k.2	J_6	1000	10	2	144341485	141501445	1,97
300-6	J_7	300	6	3	8969	8915,18	0,60
400-6	J_8	400	6	3	9117	9021,51	1,05
700-20	J_9	700	20	3	41638	41274,00	0,87
200-4-h	J_{10}	200	4	4	63429	62244,84	1,87
200-4-x1	J_{11}	200	4	4	60797	60242,96	0,91
600-8-z	J_{12}	600	8	4	132897	127901,75	3,76
600-8-x2	J_{13}	600	8	4	132228	127901,75	3,27
300-5-108	J_{14}	300	5	5	68361	67128,93	1,80
300-20-111	J_{15}	300	20	5	311286	308595,45	0,86
500-15-306	J_{16}	500	15	5	196001	193522,8	1,26
500-25-308	J_{17}	500	25	5	367586	364108,13	0,95
25-eil101	J_{18}	101	25	6	23671	23668,63	**0,01**
42-a280	J_{19}	280	42	6	130043	129560,53	0,37
144-rat783	J_{20}	783	144	6	916174	913715,52	0,27
						gap_{li} médio	1,36

A comparação entre os melhores valores obtidos pelos algoritmos heurísticos e pelo CPLEX com este critério de parada é mostrada na Tabela 3, assim como os valores médios. Na primeira coluna da Tabela 3 são mostrados os identificadores das instâncias, na segunda e terceira coluna o valor de gap_{li} para os melhores valores encontrados pelo ILS-P e ILS-nP e em seguida o valor gap_{li} relacionado aos valores médios. Na avaliação entre as melhores

soluções obtidas pelos algoritmos heurísticos e o CPLEX, compara-se a melhor solução de cada método com os limites inferiores ou com a solução ótima conhecida. Neste contexto o valor médio de gap_{li} para o CPLEX foi igual a 1,36%, para ***ILS-VND-P (ILS-P) igual a 0,65%*** e para ILS-VND-nP (ILS-nP) igual a 0,67%. Os valores em negrito na Tabela 3 mostram onde os algoritmos alcançaram melhor desempenho. O CPLEX obteve uma (1) melhor solução no total de 20 instâncias, ILS-P seis e ***ILS-nP oito***. Em cinco instâncias ambos os algoritmos (ILS-P e ILS-nP) alcançaram o mesmo resultado. Na comparação entre as soluções médias obtidas pelos algoritmos heurísticos (10 execuções de cada instância), o ***gap_{li} de ILS-P ficou em 0,80%*** e de ILS-nP igual a 0,86%.

Foram realizados novos testes em instâncias de grande porte para comprovar a eficiência dos algoritmos heurísticos propostos. Devido às instâncias serem de portes maiores, foi estabelecido um limite de 1080 segundos para cada execução das heurísticas ILS-P e ILS-nP. Como as execuções dos algoritmos heurísticos são realizadas 10 vezes, temos um total de três horas para cada instância. A Tabela 4 mostra as instâncias escolhidas para comparar ILS-P com ILS-nP, seus identificadores, número de nós, número de grupos, tipo e *gaps* encontrados. O percentual do valor de gap_h relativo aos melhores valores encontrados pelos algoritmos heurísticos é calculado em relação ao melhor valor encontrado por ILS-P ou ILS-nP. Para o percentual do valor de gap_h relacionado aos valores médios, o cálculo é realizado em relação ao melhor valor médio encontrado por ILS-P ou ILS-nP.

Tabela 3 - Os melhores valores e valores médios obtidos pelos algoritmos heurísticos

Id.	**Melhores Valores**		**Valores Médios**	
	ILS-P	**ILS-nP**	**ILS-P**	**ILS-nP**
J_1	0,11	**0,10**	**0,15**	**0,15**
J_2	**0,73**	**0,73**	**0,82**	0,83
J_3	0,56	**0,46**	0,79	**0,65**
J_4	**1,30**	1,45	**1,52**	1,81
J_5	**0,88**	0,99	**1,06**	1,28
J_6	1,29	**1,24**	1,46	**1,41**
J_7	0,24	**0,21**	0,35	**0,31**
J_8	**0,33**	**0,33**	0,52	**0,46**
J_9	0,51	**0,43**	0,55	**0,51**
J_{10}	**0,89**	**0,89**	**1,16**	1,29
J_{11}	**0,55**	1,13	**0,92**	1,85
J_{12}	**1,05**	1,17	**1,30**	1,56
J_{13}	1,43	**1,04**	1,79	**1,38**
J_{14}	1,03	**1,01**	1,23	**1,18**
J_{15}	**0,46**	0,52	**0,54**	0,59
J_{16}	**0,80**	0,86	1,02	**0,98**
J_{17}	0,51	**0,47**	0,59	**0,54**
J_{18}	0,02	0,04	**0,04**	0,09
J_{19}	**0,12**	**0,12**	**0,13**	0,21
J_{20}	**0,16**	**0,16**	**0,16**	0,18
gap_{li} médio	***0,65***	0,67	***0,80***	0,86

Nestes novos testes com instâncias e tempos maiores pode-se observar que o algoritmo heurístico ILS-nP encontrou as melhores soluções em número maior. ***ILS-nP alcançou 10 soluções melhores de um total de 15*** e ILS-P encontrou cinco. De fato na Tabela 4, observa-se que o valor médio de gap_h dos melhores valores encontrados por ILS-P é 0,58% e de ***ILS-nP igual a 0,45%***. Em relação aos valores médios encontrados pelas heurísticas, o valor médio de gap_h de ILS-P foi igual a 0,81% e de ***ILS-nP foi 0,55%***.

Comparações foram realizadas entre os algoritmos heurísticos aqui propostos (ILS-P e ILS-nP) com as heurísticas G1 (GRASP tradicional) e G4 (GRASP com Reconexão de Caminhos) (MESTRIA; OCHI; MARTINS, 2009b) e as heurísticas G-RC-VND-P e G-RC-VND-nP que utilizam GRASP, Reconexão de Caminhos e VND (MESTRIA; OCHI; MARTINS, 2010). O ambiente de execução descrito em (MESTRIA; OCHI; MARTINS, 2009b) e (MESTRIA; OCHI; MARTINS, 2010) é idêntico ao *ambiente de execução* utilizado neste trabalho. Para as instâncias da Tabela 1 os testes mostraram que G-RC-VND-P encontrou soluções ótimas em 18 de um total de 27 instâncias com valor médio de gap_{hc} relacionado aos melhores valores igual a 0,21% e ***tempo médio total de 6,05s***. O valor médio de gap_{hc} relacionado aos melhores valores de ***ILS-P foi 0,10%*** com o tempo médio de 11,37s. O algoritmo ILS-nP obteve valor médio de gap_{hc} relacionado aos melhores valores igual a 0,27% e tempo médio igual a 12,05s.

Tabela 4 - Comparação entre ILS-P e ILS-nP para instâncias de grande porte

Instâncias	**Id.**	**# nós**	**# Vi**	**Tipo**	**Melhores Valores**		**Valores Médios**	
					ILS-P	**ILS-nP**	**ILS-P**	**ILS-nP**
49-pcb1173	J_{21}	1173	49	6	1,05	**0,00**	1,58	**0,00**
100-pcb1173	J_{22}	1173	100	6	0,21	**0,00**	**0,00**	0,26
144-pcb1173	J_{23}	1173	144	6	0,22	**0,00**	0,87	**0,00**
10-nrw1379	J_{24}	1379	10	6	1,11	**0,00**	1,38	**0,00**
12-nrw1379	J_{25}	1379	12	6	0,49	**0,00**	1,07	**0,00**
1500-10-503	J_{26}	1500	10	5	0,05	**0,00**	0,10	**0,00**
1500-20-504	J_{27}	1500	20	5	**0,00**	1,21	**0,00**	0,97
1500-50-505	J_{28}	1500	50	5	**0,00**	3,67	**0,00**	2,22
1500-100-506	J_{29}	1500	100	5	**0,00**	1,32	**0,00**	4,79
1500-150-507	J_{30}	1500	150	5	**0,00**	0,31	0,30	**0,00**
2000-10-a	J_{31}	2000	10	4	**0,00**	0,25	0,07	**0,00**
2000-10-h	J_{32}	2000	10	4	2,66	**0,00**	2,04	**0,00**
2000-10-z	J_{33}	2000	10	4	0,89	**0,00**	2,48	**0,00**
2000-10-x1	J_{34}	2000	10	4	2,05	**0,00**	1,98	**0,00**
2000-10-x2	J_{35}	2000	10	4	0,02	**0,00**	0,28	**0,00**
				gap_h médio =	0,58	***0,45***	0,81	***0,55***

Para as instâncias da Tabela 2 com o critério de parada limitado pelo tempo de 720s em cada execução das heurísticas, o valor médio de gap_{li} relacionado aos melhores valores de G-RC-VND-P foi 0,85%, de G-RC-VND-nP 0,86%, de ***ILS-P 0,65%*** e de ILS-nP 0,67%. A Figura 3 mostra as comparações, para as instâncias da Tabela 4, entre os melhores valores encontrados por G1, G4, ILS-P e ILS-nP com o critério de parada de 1080s a cada execução.

A porcentagem do *gap* (%) (eixo *y*) das heurísticas mostrado na Figura 3 é calculada em relação ao melhor valor encontrado pelo ILS-P (***gap médio igual 0,00%***). O *gap* médio de G1 ficou em 4,39% e de G4 em 4,09%. Verifica-se que as heurísticas G1 e G4 obtiveram desempenho inferior em relação ao ILS-P. Observe que o ILP-nP foi melhor do que o ILS-P em seis instâncias de um total de dez. Na comparação entre os algoritmos ILS-P e ILS-nP com G1, G4 e G-RC-VND-P e G-RC-VND-nP, constatou-se que os algoritmos propostos neste trabalho obtiveram melhor desempenho.

Para algumas instâncias constata-se que o ILS-P permitiu alcançar resultados melhores do que o ILS-nP. Neste sentido, tratar o PCVG transformando-o em PCV é uma alternativa eficaz. Para as instâncias de pequeno porte, mostradas na Tabela 1, observa-se que o algoritmo heurístico ILS-P obteve melhores resultados do que o ILS-nP. Em contrapartida, os resultados mostrados nas Tabelas 3 e 4, para as instâncias de médio e grande porte, verifica-se que o ILS-nP alcançou melhor desempenho.

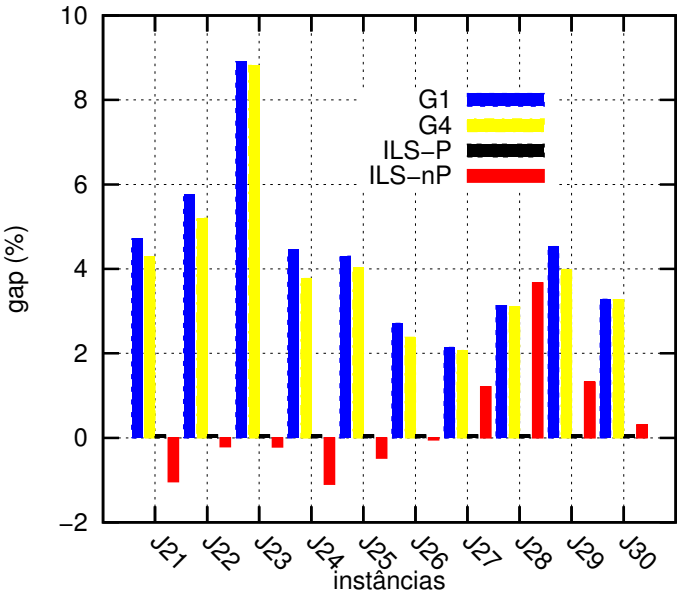

Figura 3 - Comparações entre os algoritmos ILS-P e ILS-nP com G1 e G4

4. CONCLUSÕES

Neste trabalho foram propostos e implementados dois algoritmos baseados na heurística *Iterated Local Search* e no Método de Descida em Vizinhança Variável (ILS-VND) para o PCVG. Foram desenvolvidos dois algoritmos construtivos diferentes (IMPP e IMPnP) usando os conceitos do GRASP reativo. O algoritmo ILS-nP obteve melhor desempenho do que o ILS-P na maioria dos casos. Em algumas instâncias os melhores resultados foram encontrados com o algoritmo ILS-P que utiliza a estratégia de penalizar as arestas intergrupos. Pode-se afirmar que a penalização é uma estratégia adequada para algumas instâncias. Nos testes computacionais foram constatados que os algoritmos heurísticos propostos estão aptos a encontrar soluções de boa qualidade para instâncias de grande porte. Trabalhos futuros incluem utilizar a metaheurística *Variable Neighborhood Search* (VNS) e Algoritmos Evolutivos, assim como novos algoritmos heurísticos utilizando ILS, VND e/ou Reconexão de Caminhos.

AGRADECIMENTOS

O primeiro autor agradece ao Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo. Os autores também agradecem ao apoio parcial dos seguintes órgãos de fomento: CNPq (CT-INFO & UNIVERSAL & Bolsa de Produtividade); CAPES (Pró-Engenharia, PROCAD-NF & PIQDTec); FAPERJ (PENSA RIO, Prioridade Rio & Cientista do Estado).

REFERÊNCIAS

ANILY, S.; BRAMEL, J.; HERTZ, A. A 5/3-approximation Algorithm for the Clustered Traveling Salesman Tour and Path Problems. *Operations Res. Letters*. v. 24 n.1-2, p.29-35, 1999.

CHISMAN, J. A. The Clustered Traveling Salesman Problem. *Computers & Operations Research*. v. 2, n.2, p.115-119, 1975.

CPLEX. ILOG CPLEX 11.2 User's Manual and Reference Manual. ILOG S.A, 2009.

DING, C.; CHENG, Y.; HE, M. Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs. *Tsinghua Sci. and Technology*. v. 12, n.4, p.459-465, 2007.

GENDREAU, M.; LAPORTE, G.; POTVIN, J. Y. Heuristics for the Clustered Traveling Salesman Problem. Relatório Técnico n.CRT-94-54, Centre de recherche sur les transports, Universidade de Montreal, Montreal, Canadá, 1994.

GHAZIRI, H.; OSMAN, I. H. A Neural Network for the Traveling Salesman Problem with Backhauls. *Computers & Industrial Engineering*. v. 44, n.2, p.267-281, 2003.
HANSEN, P.; MLADENOVIĆ, N. Variable Neighborhood Search. In: Glover, F. W.; Kochenberger, G. A., (eds), *Handbook of Metaheuristics*, p. 145-184, 2003.
HERNÁNDEZ-PÉREZ, H.; RODRÍGUEZ-MARTÍN, I.; SALAZAR-GONZÁLEZ, J. J. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research*. v. 36, n.5, p.1639-1645, 2009.
JONGENS, K.; VOLGENANT, T. The Symmetric Clustered Traveling Salesman Problem. *European Journal of Operational Research*. v. 19, n.1, p. 68-75, 1985.
LAPORTE, G.; PALEKAR, U. Some Applications of the Clustered Travelling Salesman Problem. *Journal of the Operational Research Society*. v. 53, n.9, p. 972-976, 2002.
LAPORTE, G.; POTVIN, J.-Y.; QUILLERET, F. A Tabu Search Heuristic using Genetic Diversification for the Clustered Traveling Salesman Problem. *Journal of Heuristics*. v. 2, n.3, p.187-200, 1996.
LOKIN, F. C. J. Procedures for Travelling Salesman Problems with Additional Constraints. *European Journal of Operational Research*, v. 3, n.2, p.135-141, 1979.
LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated local search. In: Glover, F.; Kochenberger, G., (eds), *Handbook of Metaheuristics*, Kluwer Academic Publishers, p.321-353. 2002.
MARTIN, O. C.; OTTO, S. W. Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research*, v. 63 n.1, p.57-75, 1996.
MARTIN, O.; OTTO, S. W.; FELTEN, E. W. Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics. *Operations Research Letters*, v. 11, n.4, p.219-224, 1992.
MESTRIA, M.; OCHI, L. S.; MARTINS, S. L. Desenvolvimento e Análise Experimental de Heurísticas GRASP para o Problema do Caixeiro Viajante com Grupamentos. In: *IX Congresso Brasileiro de Redes Neurais/Inteligência Computacional (IX CBRN)*. Ouro Preto, MG: Anais do IX CBRN/Inteligência Computacional, SBRN, RJ (CD-ROM), v. 1, ISSN 2177-1200, 2009a.
MESTRIA, M.; OCHI, L. S.; MARTINS, S. L. GRASP com Memória Adaptativa para o Problema do Caixeiro Viajante com Grupamentos. In: *XXIX Encontro Nacional de Engenharia de Produção*. Salvador, BA: Anais do XXIX ENEGEP, ABEPRO, RJ (CD-ROM), v. 1, 2009b.
MESTRIA, M.; OCHI, L. S.; MARTINS, S. L. Heurísticas Híbridas para o Problema do Caixeiro Viajante com Grupamentos. In: *XXX Encontro Nacional de Engenharia de Produção*. São Carlos, SP: Anais do XXX ENEGEP, ABEPRO, RJ (CD-ROM), v. 1, 2010.
POTVIN, J.-Y.; GUERTIN, F. A Genetic Algorithm for the Clustered Traveling Salesman Problem with a Priori Order on the Clusters. Relatório Técnico n.CRT-95-06. Centre de recherche sur les transports, Universidade de Montreal, Montreal, Canadá, 1995.
POTVIN, J.-Y.; GUERTIN, F. The Clustered Traveling Salesman Problem: A Genetic Approach. In: I. H. Osman; J. Kelly (eds), *Metaheuristics: Theory & Applications*, Kluwer Academic Plublishres, Norwell, MA, EUA, cap. 37, p.619-631, 1996.
RESENDE, M.; RIBEIRO, C. Greedy Randomized Adaptive Search Procedures. In: F. Glover; G. Kochenberger (eds), *Handbook of Metaheuristics*, Kluwer Academic Publishers, p. 219-249, 2002.
SUBRAMANIAN, A.; DRUMMOND, L. M. A.; BENTES, C.; OCHI, L. S.; FARIAS, R. A Parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research*. v. 37, p.1899-1911, 2010.
WEINTRAUB, A.; ABOUD, J.; FERNANDEZ, C.; LAPORTE, G.; RAMIREZ, E. An Emergency Vehicle Dispatching System for an Electric Utility in Chile. *Journal of the Operational Research Society*. v. 50, p. 690-696, 1999.