GRASP COM MEMÓRIA ADAPTATIVA PARA O PROBLEMA DO CAIXEIRO VIAJANTE COM GRUPAMENTOS

Mário Mestria (IFES)
mmestria@ic.uff.br
Luiz Satoru Ochi (UFF)
satoru@ic.uff.br
Simone de Lima Martins (UFF)
simone@ic.uff.br

O Problema do Caixeiro Viajante com Grupamentos (PCVG) é uma variante do Problema do Caixeiro Viajante (PCV), onde o conjunto de vértices é particionado em grupos disjuntos, e o objetivo deste problema é encontrar um ciclo hamiltoniano de ccusto mínimo passando por todos os vértices uma única vez como no PCV, mas visitando os vértices de cada grupo contiguamente. Como o PCVG é uma extensão do PCV, ele também é classificado como NP-Difícil limitando com isso o uso exclusivo de métodos exatos para a sua solução. Neste contexto, este trabalho apresenta heurísticas híbridas incorporando memória adaptativa na versão tradicional do GRASP. Resultados computacionais mostram que as versões adaptativas apresentam um desempenho superior quando comparado com a versão tradicional do GRASP.

Palavras-chaves: Metaheurística com Memória Adaptativa, Inteligência Computacional, Heurísticas Híbridas

1. Introdução

O Problema do Caixeiro Viajante (PCV) é um dos principais problemas estudados na área de Otimização Combinatória e Pesquisa Operacional com aplicações nas áreas de sistemas de manufatura, escalonamento de tarefas, problemas de coletas e entregas de produtos, roteiros de veículos, redes de telecomunicações, dentre outros.

Uma generalização do PCV é conhecida como o Problema do Caixeiro Viajante com Grupamentos (PCVG) (*The Clustered Traveling Salesman Problem*). Embora este modelo seja aplicável em diferentes aplicações da vida real, não existem muitos trabalhos na literatura para sua resolução. Este problema surgiu quando foi estudado um problema prático envolvendo roteamento automático em um sistema de armazenagem, Chisman (1975).

Utilizando a mesma terminologia do PCV que representa cada vértice do grafo associado por uma cidade, podemos definir o PCVG da seguinte forma. O conjunto de cidades é particionado em *k* grupos disjuntos e não vazios e o caixeiro viajante deve sair de uma cidade origem, visitar todas as cidades uma única vez antes de retornar a sua cidade origem, e todas as cidades de cada grupo devem ser visitadas contiguamente. Obviamente quando o número de grupos for igual a um (*1*), o PCVG se reduz ao PCV. Desta forma podemos afirmar que o PCVG também pertence à classe NP-Difícil.

O PCVG aparece na literatura em duas versões. Na primeira (mais explorada), supõe-se que a ordem de visitas aos grupos já é previamente definida. No segundo caso, para o qual existe reduzido número de trabalhos, essa ordem de visitas não é definida previamente e esta escolha é deixada para o algoritmo. Obviamente a segunda versão é mais complexa, pois na busca da melhor sequência de visitas as cidades, deve-se também analisar a melhor ordem de visitas aos grupos. O objetivo deste trabalho é propor algoritmos heurísticos para este segundo caso do PCVG.

Algumas aplicações do PCVG podem ser encontradas em roteamento automático para armazéns e planejamento da produção (Lokin, 1979), sistemas de manufaturas, despacho de veículos de emergência, (Weintraub *et al*., 1999), desfragmentação de discos rígidos (Laporte & Palekar, 2002) e em transações comerciais evolvendo supermercados, lojas e fornecedores de mercadorias (Ghaziri & Osman, 2003).

A maioria dos trabalhos relacionados na literatura partem do princípio que já existe definida uma ordem de visita aos grupos (Gendreau *et al*.,1994), (Potvin & Guertin, 1995), (Laporte *et al*., 1996) e (Anily *et al*., 1999), fato que geralmente não ocorre em problemas reais. Um outro aspecto incentivador deste nosso trabalho é o fato das heurísticas existentes para o PCVG serem todos bastante simples, ou são heurísticas clássicas de construção e/ou busca local, ou metaheurísticas básicas sem incorporar módulos que tem se mostrado fundamentais no desempenho final destes algoritmos tais como a reconexão de caminhos e outras buscas locais mais sofisticadas (Resende e Ribeiro, 2002).

Dentre as metaheurísticas existentes na literatura, uma que tem se destacado é a "*Greedy Randomized Adaptive Search Procedures* – GRASP". GRASP é um algoritmo iterativo no qual cada iteração consiste de duas fases: uma de construção e outra de busca local. Na fase de construção, uma solução viável é construída, e na fase de busca local sua vizinhança é investigada até um mínimo local ser encontrado. A melhor solução obtida ao final de um determinado número de iterações é retornada como a solução do algoritmo.

Neste trabalho foram desenvolvidas quatro versões utilizando a metaheurística GRASP. A primeira versão é um GRASP clássico (G1). Nas demais versões, um módulo de busca intensiva que utiliza conceitos de memória adaptativa é incorporado ao GRASP. Na segunda versão (G2) é adicionada uma segunda busca local baseada em reconexão de caminhos (RC1) entre soluções de um conjunto elite ao final da execução do GRASP clássico. A terceira (G3) utiliza uma reconexão de caminhos menos intensa (RC2) durante as iterações do GRASP e a última (G4) adiciona a RC1 ao final de G3. O artigo é estruturado conforme as seguintes seções. A segunda seção descreve os trabalhos mais importantes já existentes para o PCVG. A terceira seção descreve as heurísticas propostas para o PCVG. A quarta seção mostra os resultados computacionais efetuando uma análise de desempenho e a finalmente a última seção descreve as conclusões.

2. Trabalhos Relacionados

O PCVG foi proposto em Chisman (1975) para resolver um problema real de roteamento automático em sistemas de armazenamento. O autor resolveu o problema transformando o PCVG em um PCV adicionando uma penalidade constante de valor *M* aos custos das arestas que ligam quaisquer dois vértices pertencentes a grupos diferentes. Usando um algoritmo *branch-and-bound* os autores resolveram o PCV resultante, mas somente para instâncias de pequeno porte.

Limites inferiores utilizando Relaxação Lagrangeana foram desenvolvidos em Jongens & Volgenant (1985) para o PCVG. 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 & Volgenant (1985) foi comparado com o de Jonker & Volgenant (1984) para um único conjunto de 20 instâncias, todas com 80 vértices. Na comparação entre as soluções aproximadas do algoritmo e a solução ótima medida em *gap*, o algoritmo de Jongens & Volgenant (1985) obteve limites inferiores médios iguais a 0,37% e o algoritmo de Jonker & Volgenant (1984) limites inferiores iguais a 0,94%.

Em Laporte *et al*. (1996) uma Busca Tabu combinada com uma fase de diversificação usando um Algoritmo Genético foi proposta para o PCVG, mas a ordem de visita dos grupos já era definida *a priori*. Os resultados mostram que a Busca Tabu é competitiva comparada com o Algoritmo Genético (AG) de Potvin & Guertin (1995). A comparação no tempo computacional entre a Busca Tabu e AG não é especificada. A Busca Tabu se mostrou superior a um procedimento de inserção com pós-otimização de Gendreau *et al*. (1994), mas em contrapartida requer um esforço computacional maior.

Um Algoritmo Genético (AG) básico foi proposto para o PCVG em Potvin & Guertin (1996). O algoritmo tratou de forma independente o roteamento inter-grupos e intra-gupos, dando prioridade a solução inter-grupos e em seguida a solução intra-grupos. Ainda que este algoritmo resolva o PCVG para o caso geral, os autores só comparam o AG com uma heurística clássica de construção e busca local denominada GENIUS desenvolvida para o PCV. Além disso, na comparação com a heurística GENI (etapa construtiva do GENIUS) utilizou-se uma única vizinhança de tamanho fixo igual a cinco, limitando o desempenho desta heurística .

No trabalho de Ding *et al*. (2007) um outro AG também básico 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 ordem de visitação de cada grupo também é aleatória.

Os algoritmos α-aproximados para o PCVG com diferentes variantes são encontrados em Gendreau *et al*. (1994), Arkin *et al*. (1997), Gendreau *et al*. (1997), Anily *et al*. (1999) e (Guttmann-Beck *et al*., 2000).

O que observamos é que algumas metaheurísticas como GRASP e ILS e alguns módulos mais recentes que tem melhorado significativamente o desempenho de metaheurísticas como, por exemplo, a reconexão de caminhos (RC) (Silva *et al.*, 2007); (Bastos *et al*., 2005); (Trindade at al., 2005); (Subramanian et *al*., 2009) ainda não foram explorados para a resolução do PCVG.

3. Heurísticas Propostas para o PCVG

Neste trabalho quatro heurísticas foram desenvolvidas para o PCVG genérico onde a ordem de visitas aos grupos de vértices não é previamente estabelecida. A primeira versão (G1) utiliza a estrutura do GRASP clássico. Na etapa de construção, utilizamos uma adaptação do método de Inserção mais Próxima com penalização nas arestas cujos nós extremos pertençam a subconjuntos V_i distintos e na busca local, é utilizado o método 2-*optimal* desenvolvido originalmente para o PCV.

Apesar dos bons resultados obtidos pela metaheurística GRASP em diferentes problemas de otimização, um dos limitantes deste método na sua forma clássica é a ausência de memória entre suas iterações. Ou seja, a cada nova iteração uma nova solução é obtida sem utilizar informação das soluções obtidas em iterações anteriores. Neste trabalho propomos a utilização de uma memória adaptativa no GRASP através do uso de uma técnica denominada reconexão de caminhos (RC) (Path Relinking). Esta técnica, proposta originalmente por Glover (1996), consiste basicamente em explorar as soluções intermediárias entre duas soluções extremas de boa qualidade. Para tanto, definidas as soluções origem s_0 e destino s_d, para gerar cada solução intermediária, movimentos são selecionados de forma a introduzir na solução corrente (inicialmente s_0), atributos presentes na solução destino (s_d). (RESENDE *et al*, 2008).

A segunda versão (G2) introduz em G1 a reconexão de caminhos (RC) ao final da execução de G1. Durante a execução do GRASP é gerado e atualizado um *conjunto elite* (CE) que armazena as *num_eli* melhores soluções distintas encontradas pelo GRASP. Tal procedimento equivale a usar uma memória que armazene em CE informações relevantes encontradas em iterações passadas. Como este conjunto sofre atualizações sempre que uma nova solução de boa qualidade é encontrada entre as iterações do GRASP, esta memória é também adaptativa. Após a finalização das iterações do GRASP, realiza-se a RC entre todos os pares de solução do conjunto elite. Vamos denotar esta RC como RC1.

A terceira versão (G3) incorpora ao G1 um módulo de reconexão de caminhos (RC) mais suave que a RC1 a cada iteração GRASP. A RC neste caso é realizada a cada iteração GRASP entre cada solução produzida pela busca local com uma solução extraída do conjunto de soluções elites - CE (conjunto das *num_eli* melhores soluções distintas geradas até o momento pelo GRASP). O conjunto elite tem um tamanho pré-definido e a estratégia para escolher as soluções que irão pertencer ao conjunto elite é a seguinte. Uma solução gerada em uma iteração GRASP é inserida no conjunto elite caso seu custo da função objetivo seja menor que o custo da solução de pior qualidade do conjunto elite e ela apresente uma diferença mínima na sua estrutura para cada solução presente no CE. Vamos denotar esta RC por RC2.

Na quarta heurística (G4) introduzimos a RC2 durante a execução das iterações GRASP e a RC1 ao final do GRASP. O parâmetro α para gerenciar a cardinalidade da Lista Restrita de Candidatos (LRC) na etapa de construção do GRASP utiliza a versão dinâmica ou reativa, onde os valores de α variam no intervalo [0,1]. A estratégia reativa tende a permitir uma melhor adequação do tamanho para a LRC a cada etapa de uma execução do GRASP (Resende & Ribeiro, 2002).

4. Resultados Computacionais

Apesar de existirem métodos exatos e heurísticos para o PCVG, não temos conhecimento de nenhuma biblioteca pública com instâncias do problema na sua versão genérica sem fixação da ordem de visitas aos grupos. Desta forma, tornou-se necessária a criação de um conjunto de instâncias para o PCVG genérico para avaliar os métodos aqui propostos. Foram geradas instâncias de seis tipos: ***(tipo1)***: Adaptação de instâncias do PCV (TSPLIB95) (2007) agrupadas usando o algoritmo de clusterização *k-means*; ***(tipo 2):*** Adaptação de instâncias disponíveis na literatura para o PCV, agrupando-se os vértices em torno de um centro geométrico Johnson & McGeoch (2002); ***(tipo 3)***: As instâncias são geradas através da interface Concorde Applegate *et al.* (2007); ***(tipo 4)***: As instâncias são geradas usando *layout* conforme proposto em (Laporte *et al*., 1996), ***tipo (5)***; Instâncias semelhantes aos do tipo 2 mas geradas com parâmetros diferentes (Johnson & McGeoch, 2002); ***(tipo 6):*** Instâncias do PCV (TSPLIB95, 2007) onde a área plana retangular que compõe a instância é dividida em diversos quadriláteros e cada quadrilátero corresponde a um grupo.

Para analisar o desempenho empírico das heurísticas propostas, os resultados de G1, G2, G3 e G4 foram comparados, quando possível, com a solução ótima (ou melhor solução) obtida pelo método exato usando a formulação matemática da literatura proposta em Chisman (1975) e o *software* CPLEX11.2. O CPLEX foi executado num computador com 4 processadores Intel Core 2 Quad, 2.83 GHz com 8 GB de RAM. O sistema operacional utilizado foi Linux Ubuntu versão 4.3.2-1. As heurísticas GRASP foram codificadas em linguagem de programação C e executadas no mesmo computador descrito anteriormente, mas utilizando somente um processador.

A Tabela 1 mostra as instâncias (coluna 1) com os seus identificadores (coluna 2), seguido do número de vértices (nós), número de grupos e tipo. Nesta Tabela 1 também é mostrada as soluções e os limites inferiores gerados pelo *software* CPLEX. Na última coluna mostramos o *gap (%)* entre valor da solução e limite inferior. Devido às longas iterações ocorridas no CPLEX para a maioria das instâncias, determinamos um tempo limite de 25200 s para sua execução. Para a instância I_3, a solução ótima foi encontrada pelo CPLEX em 442 s. Todas as outras execuções utilizaram 25200 s, sendo, portanto abortadas antes de se encontrar ou confirmar uma solução ótima.

					CPLEX		
Instâncias	**Id.**	**# nós**	**# V_i**	**Tipo**	**Valor**	**Lim. Inf.**	**(%)**
10-lin318	I_1	318	10	1	531931	526559,70	1,01
10-pcb442	I_2	442	10	1	546157	536478,37	1,77
25-eil101	I_3	101	25	6	**23671**	23668,63	**0,01**
144-rat783	I_4	783	144	6	916103	913715,52	0,26
300-20-111	I_5	300	20	5	310590	308627,83	0,63
500-25-308	I_6	500	25	5	367509	364114,62	0,92
300-6	I_7	300	6	3	**8956**	8916,41	**0,44**
700-20	I_8	700	20	3	41615	41274,00	0,82
C1k.0	I_9	1000	10	2	133638625	131360206,30	1,7

C1k.1	I_{10}	1000	10	2	130563491	128540131,50	1,55
200-4-h	I_{11}	200	4	4	**62835**	62391,08	**0,71**
600-8-z	I_{12}	600	8	4	132767	128083,87	3,53
						GAP médio =	1,11

Tabela 1: Instâncias com seus identificadores, número de nós, número de grupos, tipo e valores encontrados pelo CPLEX

Os principais parâmetros utilizados nas heurísticas são descritos a seguir. O valor de penalização dos custos das arestas *M* foi de $10*max\{c_{ij}\}$. A expressão $max\{c_{ij}\}$ significa o maior custo entre todos os custos das arestas (v_i,v_j). Os valores de alfa (α) na etapa de construção do GRASP estão no intervalo de [0,1]. O número de iterações do GRASP foi fixado em 200. A cardinalidade do conjunto de soluções elites, *num_eli*, foi igual a 10 (estabelecido após experimentos preliminares). Devido à natureza aleatória do GRASP, executamos dez (10) vezes cada heurística aqui proposta para cada instância.

Na avaliação entre as melhores soluções obtidas pelas heurísticas GRASP e o CPLEX, comparamos a melhor solução de cada método com a melhor solução conhecida (ótima ou não). Neste contexto o *gap* médio do CPLEX ficou em 1,11% (isso porque em muitas instancias o CPLEX não conseguiu convergir para um ótimo no tempo limite estabelecido), G1 em 1,01%, G2 em 0,74%, G3 em 0,95% e ***G4 em 0,72%.*** O CPLEX obteve a melhor solução em três num total de 12 instâncias, G2 obteve duas melhores soluções e ***G4 seis***. Em uma única instância, a melhor solução foi alcançada simultaneamente por G2 e G4. Na comparação entre a melhor solução conhecida com as soluções médias obtidas pelas heurísticas (10 execuções de cada instância), o *gap* de G1 ficou em 1,47%, de G2 em 1,13%, de G3 em 1,21% e de ***G4 igual a 1,04%***. Na Tabela 2 são mostrados os *gaps* dos melhores valores para o G1, G2, G3 e G4, assim como os gaps dos valores médios. Valores em negrito representam os melhores valores.

I.	**Melhores Valores**				**Valores Médios**			
	G1	**G2**	**G3**	**G4**	**G1**	**G2**	**G3**	**G4**
I_1	1,14	0,80	1,02	**0,79**	1,75	1,26	1,37	**1,13**
I_2	1,22	0,71	1,10	**0,70**	1,94	1,32	1,49	**1,19**
I_3	0,09	0,05	0,06	0,06	0,35	0,27	0,21	**0,18**
I_4	0,21	0,14	0,21	**0,14**	0,28	0,20	0,25	**0,19**
I_5	0,58	**0,45**	0,54	0,48	0,84	0,66	0,71	**0,62**
I_6	0,71	0,57	0,70	**0,56**	0,92	0,73	0,81	**0,70**
I_7	0,75	0,53	0,62	0,52	1,08	0,89	0,89	**0,79**
I_8	0,67	**0,59**	0,64	**0,59**	0,83	0,71	0,75	**0,68**
I_9	1,61	**1,34**	1,63	1,42	1,88	1,60	1,74	**1,59**
I_{10}	1,31	1,05	1,29	**1,03**	1,54	1,26	1,41	**1,24**
I_{11}	1,74	1,09	1,59	1,05	3,07	2,42	2,33	**2,03**

I_{12}	2,07	1,54	2,00	**1,38**	3,16	2,24	2,61	**2,15**
gap med	1,01	0,74	0,95	**0,72**	1,47	1,13	1,21	**1,04**

Tabela 2: Os melhores valores e valores médios obtidos pelas heurísticas GRASP

Para as instâncias (**I.**), na Tabela 3 são mostrados os tempos médios medidos em segundos. Observamos que G1 e G3 são aquelas que consomem o menor tempo médio, fato que era esperado, pois estas não utilizam a RC1 ao final do GRASP. Apesar de G2 e G4 apresentarem tempos médios maiores que G1 e G3, estes ainda são bem menores que o tempo limite de 25200 s utilizado como limite para o tempo de execução para o *software* CPLEX (CP).

	Tempos Médios (segundos)			
I.	**G1**	**G2**	**G3**	**G4**
I_1	57,05	158,87	87,48	184,75
I_2	151,01	451,11	227,67	495,25
I_3	2,48	3,72	3,95	6,43
I_4	750,30	3718,32	1133,96	4132,28
I_5	51,68	121,30	75,65	142,81
I_6	225,93	720,47	331,15	822,49
I_7	49,44	99,47	75,97	119,72
I_8	593,80	1618,11	881,57	1837,78
I_9	1762,68	7592,92	2608,65	8749,56
I_{10}	1765,84	9421,46	2616,59	10297,08
I_{11}	15,75	33,95	25,07	42,37
I_{12}	364,78	1533,07	554,93	1652,09

Tabela 3: Os tempos médios das heurísticas GRASP (em segundos)

De acordo com os resultados apresentados até o momento, onde o critério de parada é o número de iterações e utilizando o mesmo número para todas as heurísticas, pode-se observar que G2 e G4 obtiveram os melhores resultados, mas exigiram tempos computacionais maiores que G1 e G3. Realizamos novos testes computacionais colocando como critério de parada um tempo máximo idêntico para as quatro heurísticas. Foi utilizado um tempo limite de 2 horas para o CPLEX e 720 segundos para as heurísticas GRASP, porque, para cada instância, as heurísticas são executadas 10 vezes e a melhor solução encontrada nestas execuções é retornada. Somente a execução da instância I_3 não foi limitada por duas horas, devido à solução ótima ser encontrada antes. A comparação entre os melhores valores obtidos pelas heurísticas e pelo CPLEX com este novo critério de parada é mostrada na Tabela 4.

	CPLEX - Nova Execução			**Melhores Valores**				**Valores Médios**			
I.	**Valor**	**Lim. Inf.**	**(%)**	**G1**	**G2**	**G3**	**G4**	**G1**	**G2**	**G3**	**G4**
I_1	534640	526412,07	1,54	0,97	0,80	0,93	**0,76**	1,78	**1,18**	1,21	**1,18**
I_2	547152	536478,33	1,95	1,12	0,73	0,99	**0,66**	1,93	1,24	1,36	**1,22**
I_3	**23671**	23668,63	0,01	0,04	0,04	0,05	0,03	0,35	0,22	**0,14**	0,18

I_4	916174	913715,52	0,27	0,21	**0,19**	0,21	0,20	0,28	**0,23**	0,25	0,24
I_5	311286	308595,45	0,86	0,55	0,45	0,51	**0,43**	0,85	0,64	**0,63**	**0,63**
I_6	367586	364108,13	0,95	0,69	0,61	0,66	**0,58**	0,92	**0,73**	0,78	**0,73**
I_7	8969	8915,18	0,60	0,67	0,51	0,60	**0,49**	1,09	0,82	**0,78**	**0,78**
I_8	41638	41274,00	0,87	0,67	**0,63**	0,65	0,64	0,82	**0,71**	0,75	0,72
I_9	134025123	131354923,50	1,99	1,63	1,61	1,64	**1,60**	1,88	**1,76**	1,79	**1,76**
I_{10}	130750874	128540131,50	1,69	1,28	1,27	1,31	**1,27**	1,54	1,44	1,47	**1,42**
I_{11}	63429	62244,84	1,87	1,68	1,28	1,60	**1,19**	3,30	2,37	**2,01**	2,30
I_{12}	132897	127901,75	3,76	2,23	**1,80**	2,14	1,96	3,32	**2,43**	2,70	2,54
		gap medio =	1,36	0,98	0,83	0,94	**0,82**	1,51	1,15	1,16	**1,14**

Tabela 4: Valores da nova execução do CPLEX e os *gaps* das heurísticas GRASP com limite de tempo

O *gap* médio do CPLEX ficou em 1,36%, de G1 em 0,98%, de G2 em 0,83%, de G3 em 0,94% e de ***G4 igual a 0,82%.*** G4 obteve novamente o melhor desempenho obtendo a melhor solução em 8 de um total de 12 instâncias, G2 com 3 melhores soluções e o CPLEX obteve uma melhor solução. Observa-se desta forma a importância de se usar reconexão de caminhos intensiva como é o caso da RC1 ao final do GRASP (G2 e G4) e que a melhor opção foi usar a RC tanto durante as iterações (RC2) como após o GRASP (RC1) (G4). Nesta segunda bateria de testes, observa-se que apesar de G2 e G4 exigirem mais tempo por iteração, estas necessitam de menos iterações para atingir bons limites superiores de um valor ótimo.

A Tabela 4 ilustra adicionalmente o desempenho médio das heurísticas e do CPLEX quando utilizamos um tempo limite como critério de parada. O *gap* médio do G1 ficou em 1,51%, de *G2 em 1,15%*, de G3 com 1,16% e de ***G4 igual a 1,14%***. Novamente os resultados mesmo que empíricos mostram a superioridade das versões G2 e G4 em relação a G1 e G3. No entanto novamente ressaltamos o bom comportamento médio das quatro heurísticas aqui propostas mostrando que todas elas possuem uma robustez necessária para a sua confiabilidade prática.

Novos testes foram efetuados somente com G4 e o método exato CPLEX restritos as instâncias menores, mas sem limitação de tempo para o CPLEX. A heurística G4 foi executada com número de iterações fixo igual a 200. Na primeira coluna da Tabela 5 encontra-se o melhor valor obtido por G4, na segunda coluna o tempo médio de execução t_{G4}(s) de G4 medido em segundos e na terceira coluna o valor obtido pelo CPLEX. Na quarta e quinta coluna são mostrados o tempo demandado pelo CPLEX em segundos t_{CPLEX}(s) e o *gap* do melhor valor obtido por G4(%) em relação ao valor obtido pelo CPLEX.

Instâncias	**G4**	**t_{G4}(s)**	**CPLEX**	**t_{CPLEX} (s)**	**G4 (%)**
5-eil51	437	1,00	437	12,31	**0,00**
10-eil51	440	1,00	440	74,38	**0,00**
15-eil51	437	1,00	437	2,04	**0,00**
5-berlin52	7991	1,20	7991	201,80	**0,00**
10-berlin52	7896	1,10	7896	89,17	**0,00**
15-berlin52	8049	1,10	8049	75,93	**0,00**
5-st70	695	2,30	695	13790,11	**0,00**
10-st70	691	2,00	691	4581,00	**0,00**
15-st70	692	2,00	692	883,50	**0,00**
5-eil76	561	2,70	559	83,70	0,36

10-eil76	564	2,40	561	254,30	0,53
15-eil76	567	2,50	565	49,66	0,35
5-pr76	108590	2,70	108590	99,29	**0,00**
10-pr76	109538	2,20	109538	238,13	**0,00**
15-pr76	110843	2,30	110678	261,94	0,15
10-rat99	1238	4,90	1238	650,67	**0,00**
25-rat99	1277	4,70	1269	351,15	0,63
50-rat99	1258	4,90	1249	2797,58	0,72
25-kroA100	21917	4,70	21917	3513,57	**0,00**
50-kroA100	21453	5,20	21453	947,55	**0,00**
10-kroB100	22475	4,80	22440	4991,44	0,16
50-kroB100	22653	5,20	22355	2579,22	1,33
25-eil101	672	4,60	663	709,45	1,36
50-eil101	651	5,40	644	275,33	1,09
25-lin105	14438	5,10	14438	6224,55	**0,00**
50-lin105	14534	5,70	14379	1577,21	1,08
75-lin105	14607	6,40	14521	15886,77	0,59
				GAP Méd =	0,25

Tabela 5. Comparação do CPLEX com G4 para instâncias do tipo 1 de pequeno porte

Para todas as instâncias de pequeno porte, o CPLEX encontrou soluções ótimas. G4 encontrou soluções ótimas em 15 de um total de 27 instâncias e o *gap* médio dos valores obtidos pelo G4 em relação ao ótimo foi de 0,25%. Este último resultado reforça o potencial de G4 em encontrar uma solução de boa qualidade em tempo computacional viável.

Para comparar a eficiência de inserir reconexão de caminhos durante as iterações e o uso de reconexão de caminhos no final do GRASP, realizamos mais uma bateria de testes em instâncias de grande porte. A comparação foi realizada entre o GRASP clássico (G1) e G4. Como as instâncias são de portes maiores, estabelecemos um limite de 1080 segundos para cada execução de G1 e G4. Como as execuções das heurísticas GRASP são realizadas 10 vezes, temos um total de três horas para cada instância. A Tabela 6 mostra as instâncias escolhidas para comparar G1 com G4 com seus identificadores, número de nós, número de grupos, tipo e *gaps encontrados*.

					Melhores Valores		**Valores Médios**	
Instâncias	**Id.**	**# nós**	**# Vi**	**Tipo**	**G1**	**G4**	**G1**	**G4**
49-pcb1173	I_{13}	1173	49	6	0,40	**0,00**	2,94	**0,00**
100-pcb1173	I_{14}	1173	100	6	0,54	**0,00**	2,87	**0,00**
144-pcb1173	I_{15}	1173	144	6	0,08	**0,00**	2,68	**0,00**
10-nrw1379	I_{16}	1379	10	6	0,66	**0,00**	2,33	**0,00**
12-nrw1379	I_{17}	1379	12	6	0,26	**0,00**	2,96	**0,00**
1500-10-503	I_{18}	1500	10	5	0,32	**0,00**	1,03	**0,00**
1500-20-504	I_{19}	1500	20	5	0,06	**0,00**	1,36	**0,00**
1500-50-505	I_{20}	1500	50	5	0,03	**0,00**	0,65	**0,00**
1500-100-506	I_{21}	1500	100	5	0,52	**0,00**	1,19	**0,00**
1500-150-507	I_{22}	1500	150	5	0,01	**0,00**	0,43	**0,00**
				GAP médio =	0,29	**0,00**	1,84	**0,00**

Tabela 6: Comparação entre G1 e G4 para instâncias de grande porte

Nestes novos testes com instâncias e tempos maiores podemos observar novamente que a introdução dos módulos de reconexão de caminhos em G4 é de fundamental importância na busca de soluções de boa qualidade. De fato na tabela 6, observamos que o gap médio dos

melhores valores encontrados por G1 é 0,29% e por G4 igual a zero. Nos valores médios temos o *gap* médio de G1 igual a 1,84% e de G4 permanece igual a zero.

5. Conclusão

O trabalho desenvolvido neste artigo abordou o Problema do Caixeiro Viajante com Grupamentos (PCVG) que é uma generalização do clássico Problema do Caixeiro Viajante (PCV). Quatro métodos heurísticos foram desenvolvidos para o PCVG, todos baseados na metaheurística GRASP. De nosso conhecimento, esta é a primeira vez que o GRASP é utilizado para a solução do PCVG genérico onde não é previamente fixada a ordem de visitas aos grupos. As versões que utilizaram reconexão de caminhos (RC) foram aquelas que conseguiram o melhor desempenho (G2 e G4) e foi mostrado nos testes empíricos que o uso da RC durante e após o GRASP é a melhor opção dentre as propostas feitas neste trabalho. Trabalhos futuros sugeridos incluem o uso da metaheurística *Iterated Local Search* (ILS) bem como formas híbridas reunindo GRASP, RC e ILS.

Agradecimentos

1. Agradecemos a Jean-Yves Potvin, Département d'informatique et de recherche opérationnelle, Université de Montréal pelo apoio ao material bibliográfico e a David S. Johnson - AT&T Labs - Research, pelos códigos para gerar instâncias do tipo 2.

2. Os autores também agradecem ao apoio parcial dos seguintes órgãos de fomento: CNPq (CT-INFO & UNIVERSAL & Bolsa de Produtividade); CAPES (Pró-Engenharia e PROCAD-NF); 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,. 24(1-2), p.29-35, 1999.

APPLEGATE, D., BIXBY, R., CHVÁTAL, V., & COOK, W. *Concorde TSP Solver*. School of Industrial and Systems Engineering, Georgia Tech, http://www.tsp.gatech.edu/concorde/index.html, acesso em 22/12/2007, 2007.

ARKIN, E. M., HASSIN, R., & KLEIN, L. *Restricted Delivery Problems on a Network*. Networks. Vol 29, n.4, p.205-216, 1997.

BASTOS, L. O., OCHI, L. S., & MACAMBIRA, E. *GRASP with Path Relinking for the SONET Ring Assignment Problem*. Proc. of the 5th International Conference on Hybrid Intelligence System (HIS2005) in cooperation with IEEE Computational Intelligence Society. p. 239-244, 2005.

CHISMAN, J. A. The Clustered Traveling Salesman Problem, *Computers & Operations Research*. Vol 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. Vol 12, n.4, p.459-465, 2007.

GENDREAU, M., LAPORTE, G. & POTVIN, J. Y. *Heuristics for the Clustered Traveling Salesman Problem*. Technical Report nº CRT-94-54, Centre de recherche sur les transports, Universidade de Montreal, Montreal, Canadá, 1994.

GENDREAU, M., LAPORTE, G., & HERTZ, A. *An Approximation Algorithm for the Traveling Salesman Problem with Backhauls*. Operational Research. Vol 45, n.4, p.639-641, 1997

GHAZIRI, H. & OSMAN, I. H. *A Neural Network for the Traveling Salesman Problem with Backhauls*. Computers & Industrial Engineering. Vol 44, n.2, p.267-281, 2003.

GLOVER, F. *Tabu Search and Adaptive Memory Programing - Advances, Applications and Challenges*. In Barr, R.S., Helgason, R.V., and Kennington, J.L., editors, Interfaces in Computer Science and Operations Research, p,1-75. Kluwer Academic Plublishres, Dordrecht, MA, EUA, 1996.

GUTTMANN-BECK, N., HASSIN, R., KHULLER, S. & RAGHAVACHARI, B. *Approximation Algorithms with Bounded Performance Guarantees for the Clustered Traveling Salesman Problem*. Algorithmica. Vol 28, n.4, p.422-437, 2000.

JOHNSON, D. S. & MCGEOCH, L. A. *Experimental Analysis of Heuristics for the STSP*. In G. Gutin and A. Punnen (eds), The Traveling Salesman Problem and its Variations, Kluwer Academic Plublishres, Dordrecht, Holanda, pp. 369-443., 2002.

JONGENS, K. & VOLGENANT, T. *The Symmetric Clustered Traveling Salesman Problem.* European Journal of Operational Research. Vol 19, n.1, p. 68-75, 1985.

JONKER, R. & VOLGENANT, T. *Non-optimal Edges for the Symmetric Traveling Salesman Problem*. Operations Research. Vol 32, n.4, p.837-846, 1984.

LAPORTE, G. & PALEKAR, U. *Some Applications of the Clustered Travelling Salesman Problem.* Journal of the Operational Research Society. Vol 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. Vol 2, n.3, p.187-200, 1996.

LOKIN, F. C. J. *Procedures for Travelling Salesman Problems with Additional Constraints*. European Journal of Operational Research, Vol 3, n.2, p.135-141, 1979.

POTVIN, J.-Y. & GUERTIN, F. *A Genetic Algorithm for the Clustered Traveling Salesman Problem with a Priori Order on the Clusters*. Technical Report nº 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 and J. Kelly (eds), Metaheuristics: Theory & Applications, Kluwer Academic Plublishres, Norwell, MA, EUA, cap. 37, p.619-631, 1996.

REINELT, G. *The Traveling Salesman: Computational Solutions for TSP Applications*. Lecture Notes in Computer Science. Vol 840, 213pp, Springer Verlag, Heidelberg, Berlim, 1994.

RESENDE, M. & RIBEIRO, C. *Greedy Randomized Adaptive Search Procedures*. In F. Glover and G. Kochenberger (eds), Handbook of Metaheuristics, Kluwer Academic Publishers, p. 219-249, 2002.

RESENDE, M. G., MARTÍ , R., GALLEGO, M. & DUARTE, A. *GRASP and Path Relinking for the Max-Min Diversity Problem*. To appear in Computers & Operations Research 2009.

SILVA, G. C., DE ANDRADE, M. R. Q., OCHI, L. S., MARTINS, S. L. & PLASTINO, A. *New heuristics for the maximum diversity problem*. Journal of Heuristics. Vol 13, p. 315-336, 2007.

SUBRAMANIAN, A., OCHI, L. S., DRUMMOND, L. M. A., BENTES, C. & FARIAS, R., A Parallel Adaptive Metaheuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery .Submitted to Special Issue: Logistics - Computers & Operations Research , 2009.

TRINDADE, V. A. & OCHI, L. S. Hybrid Adaptive Memory Programming using GRASP and Path Relinking for the Scheduling Workover Rigs for Onshore Oil Production. Proc. of the 5th International Conference on Hybrid Intelligence System (HIS2005) in cooperation with IEEE Computational Intelligence Society., 2005.

TSPLIB95. *Traveling Salesman Problem.* Gerhard Reinelt, Universität Heidelberg, Institut für Informatik, Heidelberg, Alemanha, acesso em 17/09/2007.
http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/index,

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. Vol 50, p. 690-696, 1999.