METAHEURÍSTICAS HÍBRIDAS PARA RESOLUÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE COM COLETA DE PRÊMIOS[1]

Resumo

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) pode ser associado a um caixeiro que coleta um prêmio em cada cidade visitada e paga uma penalidade para cada cidade não visitada, com um custo de deslocamento entre as cidades. O problema encontra-se em minimizar o somatório dos custos da viagem e penalidades, enquanto inclui na sua rota um número suficiente de cidades que o permita coletar um prêmio mínimo pré-estabelecido. Este trabalho contribui com o desenvolvimento de metaheurísticas híbridas para o PCVCP, baseadas em GRASP e métodos de busca em vizinhança variável (VNS/VND) para solucionar aproximadamente o PCVCP. De forma a validar as soluções obtidas, propõe-se uma formulação matemática a ser resolvida por um *solver* comercial objetivando encontrar a melhor solução para o problema, sendo este *solver* aplicado a problemas de pequeno porte. Resultados computacionais demonstram a eficiência da abordagem híbrida proposta, tanto em relação à qualidade da solução final obtida quanto em relação ao tempo de execução.

Palavras-chave: Problema do Caixeiro Viajante, Metaheurísticas, GRASP, VNS, VND.

HYBRID METAHEURISTICS FOR SOLVE THE PRIZE COLLECTING TRAVELING SALESMAN PROBLEM

Abstract

The Prize Collecting Traveling Salesman Problem (PCTSP) can be associated to a salesman that collects a prize in each city visited and pays a penalty for each city not visited, with travel costs among the cities. The objective is to minimize the sum of the travel costs and penalties, including in the tour an enough number of cities that allow collecting a minimum prize. This paper contributes with the development of a hybrid metaheuristic to PCTSP, based on GRASP and search methods in variable neighborhood (VNS/VND) to solve PCTSP approximately. In order to validate the obtained solutions, we proposed a mathematical formulation to be solved by a commercial solver to find the best solution for the problem, being this solver applied to small problems. Computational results demonstrate the efficiency of the proposed method, as much in relation to the quality of the obtained final solution as in

[1] Submetido à Revista Produção. Autores: Antônio Augusto Chaves, Fabrício Biajoli, Otávio Massashi Mine e Marcone Jamilson Freitas Souza.

relation to the time of execution.
***Key words**: Traveling Salesman problem, Metaheuristic, GRASP, VNS, VND.*

1. Introdução

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), referido na literatura inglesa como *Prize Collecting Traveling Salesman Problem* (PCTSP), é uma variante do Problema do Caixeiro Viajante. O PCVCP pode ser associado a um caixeiro viajante que coleta um prêmio p_k, não negativo, em cada cidade k que ele visita e paga uma penalidade γ_t para cada cidade t que não visita, com um custo c_{ij} de deslocamento entre as cidades i e j. O problema encontra-se em minimizar o somatório dos custos da viagem e penalidades, enquanto inclui na sua rota um número suficiente de cidades que o permita coletar um prêmio mínimo, $p_{\min}$, pré-estabelecido.

O PCVCP foi formulado inicialmente por Egon Balas (BALAS, 1989) como um modelo para a programação da operação diária de uma fábrica que produzia lâminas de aço. Por razões que tinham a ver com o desgaste dos rolos e também por outros fatores, a seqüência na ordem do processamento era essencial. A programação consistia na escolha de um número de lâminas associadas às suas ordens de execução, que satisfizessem o limite inferior do peso, e que ordenadas numa seqüência apropriada, minimizasse a função de seqüência. As tarefas de escolha das lâminas e das opções disponíveis para o seu sequenciamento necessitavam ser resolvidas em conjunto. Esse problema, chamado então de *Prize Collecting Traveling Salesman Problem*, serviu como base para o desenvolvimento de um software implementado por Balas e Martin, e foi resolvido aproximadamente por meio da combinação de várias heurísticas (BALAS, 2001).

As aplicações desenvolvidas para o PCVCP são relativamente poucas, apesar da grande aplicabilidade que este tem para o mundo real. GOEMANS E WILLIANSON (1992) desenvolveram um procedimento 2-aproximativo para uma versão do PCVCP no qual o prêmio mínimo a ser coletado foi removido e o objetivo passou a ser simplesmente minimizar o custo e as penalidades.

DELL'AMICO *et al.* (1998) exploraram uma relaxação Lagrangeana para o PCVCP, relaxando a restrição de capacidade. Um algoritmo baseado na heurística clássica para o PCV, conhecida como heurística de inserção mais barata, é utilizado para transformar a solução

relaxada em uma solução viável, através da inserção de vértices com melhores custos na rota até que o prêmio mínimo seja coletado. Propuseram ainda, utilizar uma heurística chamada de Extensão e Colapso para melhorar a solução viável obtida.

GOMES *et al.* (2000) e MELO (2001) desenvolveram trabalhos que utilizam metaheurísticas híbridas para o PCVCP. Ambos utilizam o método *Greedy Randomized Adaptative Search Procedure* – GRASP (FEO & RESENDE, 1995) associado a métodos de busca em vizinhança variáveis (*Variable Neighborhood Descent* – VND e *Variable Neighborhood Search* – VNS) (MLADENOVIC & HANSEN, 1997), sendo que MELO (2001) propôs a utilização de uma generalização do GRASP, chamada de GRASP-PSD.

A escolha deste problema para resolução foi feita em função de sua fácil adaptação a situações da vida real. Em linhas gerais, pode ser descrito como um universo de clientes em potencial, onde existe associado a cada cliente, quando não for atendido, uma penalidade pela expectativa frustrada de atendimento, e quando este for atendido, um ganho relativo. Deseja-se, então, partir de uma origem, montar um percurso contendo alguns clientes visitados uma única vez e retornar ao ponto de partida, minimizando o custo da distância total percorrida e a soma das penalidades, de forma a garantir um ganho mínimo que justifique o investimento.

A dificuldade de solução do PCVCP está no número elevado de soluções existentes. Considerando que o Problema do Caixeiro Viajante pode ser um caso particular do PCVCP, onde o prêmio mínimo (p_{min}) a ser coletado é igual ao somatório de todos os prêmios p_k para cada cidade k, o PCVCP pode ser classificado como NP-difícil. Sendo assim, a resolução deste por enumeração completa se torna inviável para problemas de dimensões mais elevadas.

Dado esse aspecto combinatório, a abordagem mais comum para problemas dessa natureza é através de heurísticas, as quais, no entanto, não estão capacitadas a garantir a otimalidade das soluções finais obtidas. Heurísticas clássicas têm a desvantagem de ficarem presas no primeiro ótimo local encontrado. Uma outra forma de resolver problemas combinatórios é por meio de metaheurísticas, as quais possuem ferramentas que possibilitam escapar das armadilhas dos ótimos locais, permitindo a busca em outras regiões do espaço de busca. Estas possibilitaram um grande avanço com relação à qualidade das soluções finais obtidas com os procedimentos heurísticos clássicos.

Dentre os procedimentos enquadrados como metaheurísticas que surgiram ao longo das últimas décadas, destacam-se: Algoritmos Genéticos (AGs) (GOLDBERG, 1989), *Simulated Annealing* (KIRKPATRICK *et al.*, 1983), Busca Tabu (BT) (GLOVER, 1986), *Greedy Randomized Adaptative Search Procedure* (GRASP) (FEO & RESENDE, 1995), Colônia de Formigas (DORIGO *et al.*, 1996), *Variable Neighborhood Search* (VNS) (MLADENOVIC & HANSEN, 1997), entre outros.

Para a resolução do PCVCP, fez-se uso de conceitos de técnicas mais recentes, no caso GRASP, VNS e VND, que têm se destacado na solução de problemas altamente combinatórios. Há, certamente, outras técnicas que poderiam ser empregadas, mas preferiu-se selecionar GRASP e VNS/VND tendo em vista a simplicidade dessas técnicas e sua eficiência na abordagem de diversos outros problemas combinatórios.

Este trabalho se diferencia do desenvolvido por GOMES *et al.* (2000) pelo fato de combinar as técnicas heurísticas VNS e VND, e se diferencia do trabalho de MELO (2001) por utilizar o GRASP básico com filtro, o qual consiste em refinar apenas a melhor solução construída, e também por trabalhar com um número mais elevado de vizinhanças para os métodos VNS e VND.

Outra particularidade deste trabalho é a utilização de uma formulação matemática para o PCVCP a ser resolvida por um *solver* comercial objetivando encontrar a melhor solução para o problema, sendo este *solver* aplicado a problemas de pequeno porte. A importância desta abordagem matemática está no fato desta permitir a validação das soluções obtidas pela abordagem heurística desenvolvida.

Este trabalho está organizado como segue. Na seção 2 apresenta-se uma formulação matemática para o problema. Na seção 3 são apresentados os principais procedimentos heurísticos tomados como base no desenvolvimento deste trabalho. A seção 4 descreve, em detalhes, o procedimento heurístico proposto para a solução do PCVCP. Os resultados computacionais obtidos pela aplicação do procedimento proposto são apresentados e discutidos na seção 5. A última seção conclui o trabalho.

2. Formulação Matemática

Seja $G' = (V, E)$ um grafo completo não direcionado, onde para cada aresta $(i,j) \in E$ é dado um custo c_{ij} e, para cada vértice $i \in V$, são associados um prêmio p_i caso o vértice seja visitado ou uma penalidade γ_i, caso contrário. Os vértices são numerados de 0 até $| V |$, sendo o vértice 0, sem perda de generalidade, assumido como depósito ou domicílio do caixeiro viajante. Para este vértice especial, $p_0 = 0$ e $\gamma_0 = \infty$. No PCVCP o objetivo é encontrar uma rota que permita coletar um prêmio mínimo p_{min}, previamente estabelecido, cuja soma dos custos e penalidades pagas pelos vértices que não compuserem a rota seja mínima.

A formulação apresentada a seguir foi proposta por BALAS (1989). As restrições 2.5 e 2.6, não contidas na formulação original de BALAS (1989), são inseridas para eliminar a existência de sub-rotas. Assume-se que y_i é a variável que controla se o vértice i é ou não visitado (recebendo o valor 1 caso seja visitado), que x é o vetor de incidência associado à rota (recebendo valor 1 se a aresta (i,j) estiver na rota) e que f_{ij} é o fluxo que passa pela aresta (i,j).

As restrições 2.2 garantem que se o vértice i estiver na rota, a quantidade de arcos que saem dele tem que ser igual a 1, e 0 caso contrário. As restrições 2.3 garantem que se o vértice j estiver na rota, o somatório das arestas que chegam nele tem que ser igual a 1, e 0 caso contrário. As restrições 2.4 asseguram que o prêmio coletado na rota será maior ou igual ao prêmio mínimo pré-estabelecido. As restrições 2.5 e 2.6 asseguram a eliminação de sub-rotas. As restrições 2.5 garantem que o fluxo que chega e que sai do vértice i é igual a 1 caso o vértice seja visitado, e 0 caso contrario. As restrições 2.6 asseguram que o fluxo máximo que pode passar por uma aresta é o número de vértices menos 1, ou seja, $|V|$-1. As restrições 2.7 a 2.9 estabelecem, respectivamente, os domínios das variáveis x, y e f.

$$\text{(PCVCP) minimizar} \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} + \sum_{i \in V} \gamma_i (1 - y_i) \tag{2.1}$$

sujeito à

$$\sum_{j \in V \setminus \{i\}} x_{ij} = y_i \qquad \forall\, i \in V \setminus \{0\} \tag{2.2}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = y_j \qquad \forall\, j \in V \setminus \{0\} \tag{2.3}$$

$$\sum_{i \in V} p_i y_i \geq p_{\min} \tag{2.4}$$

$$\sum_{j \in V} f_{ij} - \sum_{j \in V} f_{ji} = y_i \quad \forall i \in V \setminus \{0\} \tag{2.5}$$

$$f_{ij} \leq (|V| - 1) x_{ij} \qquad \forall (i, j) \in E \tag{2.6}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i, j) \in E \tag{2.7}$$

$$y_j \in \{0,1\} \qquad \forall i \in V \tag{2.8}$$

$$f_{ij} \geq 0 \qquad \forall (i, j) \in E \tag{2.9}$$

Para a resolução desta formulação matemática, fez-se uso do *software* LINGO (SHRAGE, 1991) versão 7.0, objetivando encontrar a solução ótima para o PCVCP. Observa-se, entretanto, pela natureza combinatorial do problema abordado, que tal método só consegue resolver problemas de pequenas dimensões. Através dos testes mostrados na seção 5, nota-se que para um número de vértices próximo a 50, a obtenção da solução por este método exato já se torna inviável computacionalmente.

3. Heurísticas Utilizadas

A seguir são apresentados os procedimentos heurísticos referenciados neste trabalho para a resolução do PCVCP.

3.1 ADD_Step

Uma solução inicial de boa qualidade é muito importante, uma vez que bons pontos de partida permitem acelerar a busca local. Este procedimento heurístico construtivo desenvolvido em GOMES *et al.* (2000) procura gerar uma solução inicial de qualidade.

Inicialmente tem-se uma rota R partindo e retornando à origem. Em seguida, para cada vértice k não pertencente à rota R, verifica-se sua economia atual em relação a todas as arestas (i, j) que formam a rota. A função que calcula a economia para cada inserção de k é definida da seguinte forma:

$$economia\ (k)\ =\ g_k\ =\ \max\ (i,j)\ \{\ c_{ij} + \gamma_k - c_{ik} - c_{kj}\ \},\ \forall\ k \notin R \qquad (3.1)$$

sendo composta pelo custo da aresta (i,j), a penalidade do vértice k e os custos das arestas (i,k) e (k,j), respectivamente.

Após isto, seleciona-se dentre todos os vértices não pertencentes à rota parcial atual o que apresentar a maior economia positiva, sendo este inserido em R. Feito isto, para cada vértice k não pertencente à rota R, faz-se o recálculo da economia em relação a todas as arestas (i,j) que pertencem à rota R. Note que se $g_k > 0$, após a inserção do vértice k, obtêm-se uma redução no valor da função objetivo.

O método termina quando o somatório de todos os prêmios dos vértices pertencentes a R for igual ou maior que o prêmio mínimo e não haja mais vértice k com economia positiva.

3.2 DROP_Step

Este procedimento, também desenvolvido em GOMES *et al.*(2000), parte de uma solução que contém todos os vértices e utiliza uma abordagem oposta à dos algoritmos de inserção, isto é, a cada iteração retira-se um vértice segundo o cálculo de uma função gulosa.

Definida uma solução inicial, com uma rota R que contenha todos os vértices, para cada vértice k pertencente à rota, calcula-se a economia associada à remoção deste vértice. A função do cálculo para remoção do vértice k é definida por:

$$economia\ (k)\ =\ g_k = max\{\ c_{ak,k} + c_{k,sk} - c_{ak,sk} - \gamma_k\ \},\ \forall\ k \in R \qquad (3.2)$$

onde a_k e s_k representam, respectivamente, o antecessor e o sucessor do vértice k na rota R.

Após isto, seleciona-se dentre todos os vértices pertencentes à rota atual o que apresentar a maior economia positiva, sendo este removido de R. Feito isto, para cada vértice k pertencente à rota R, faz-se um recálculo da economia de remoção. Observa-se que se $g_k > 0$, a remoção do vértice k proporcionará uma redução no valor da função objetivo.

O método continua enquanto existir algum vértice com economia positiva e a retirada

desse vértice não acarretar uma solução inviável, ou seja, o somatório dos prêmios dos vértices pertencentes à rota for menor que o prêmio mínimo pré-estabelecido.

3.3 Método de Descida k-Optimal

Após realizar a fase de construção, tem-se uma rota com alguns (ou todos) vértices, e sabe-se qual a seqüência em que estes vértices são visitados. O desejo agora é conseguir uma possível melhora no valor da função de avaliação através da tentativa de mudança na ordem de visitas, o que pode ser conseguido realizando-se possíveis trocas de suas arestas.

As heurísticas do tipo *k-Optimal* (CROES, 1958) são estratégias de melhoria. Partindo-se de um ciclo hamiltoniano *H*, excluem-se *k* arestas de *H*, produzindo *k* caminhos desconectados. Reconectam-se esses *k* caminhos de alguma maneira para produzir outro ciclo *H'*, usando diferentes arestas daquelas que foram removidas de *H*. Desta maneira, *H* e *H'* serão diferentes entre si por exatamente *k* arestas. São verificadas todas as soluções viáveis contendo *H'*, escolhendo-se a melhor dentre todas, chamada de solução *k-Opt*.

À medida que o valor de *k* aumenta, em geral, aumenta também a probabilidade de se alcançar a solução ótima. Entretanto, o custo computacional também cresce rapidamente com o valor de *k*, pois a complexidade deste algoritmo é $O(n^k)$ (GOLDBARG & LUNA, 2000). Neste trabalho optou-se por utilizar a heurística 2-*Optimal*, por esta ser eficiente e exigir menor esforço computacional.

3.4 GRASP

A metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), (FEO & RESENDE, 1995), é um procedimento iterativo, no qual cada iteração consiste em duas fases distintas: a fase de construção, onde uma solução viável é construída, e a fase de busca local, onde um ótimo local na vizinhança da solução construída é encontrado. A melhor solução encontrada ao longo de todas as iterações GRASP realizadas é retornada como resultado.

Na fase de construção, uma solução é iterativamente construída, elemento por elemento. A cada iteração dessa fase, os próximos elementos candidatos a serem incluídos na

solução são colocados em uma lista *C* de candidatos, seguindo um critério de ordenação pré-determinado. Esse processo de seleção é baseado em uma função adaptativa $f: C \rightarrow R$, que estima o benefício da seleção de cada um dos elementos. A heurística é adaptativa porque os benefícios associados com a escolha de cada elemento são atualizados em cada iteração da fase de construção para refletir as mudanças oriundas da seleção do elemento anterior. O componente probabilístico do procedimento reside no fato de que cada elemento é selecionado de forma aleatória a partir de um subconjunto restrito formado pelos melhores elementos que compõem a lista de candidatos restrita (LCR) segundo a função *f*. Esta técnica de escolha permite que diferentes soluções sejam geradas em cada iteração GRASP, diversificando o espaço de busca. Um parâmetro $\alpha \in [0,1]$ determina o tamanho da lista de candidatos restrita. Valores de α próximos a zero conduzem a soluções muito próximas àquela obtida escolhendo-se a cada passo o melhor elemento segundo a função *f*, enquanto que valores de α próximo a 1 conduzem a soluções praticamente aleatórias, o que pode tornar o processo de busca local mais lento.

Assim como em muitas técnicas determinísticas, as soluções geradas pela fase de construção GRASP provavelmente não são localmente ótimas com respeito à definição de vizinhança adotada. Daí a importância da fase de busca local, a qual objetiva melhorar a solução construída.

A eficiência da busca local depende da qualidade da solução construída. A fase de construção tem então um papel importante na busca local, uma vez que as soluções construídas constituem bons pontos de partida para a busca local, permitindo assim acelerá-la.

3.5 VNS

O Método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS), (MLADENOVIC & HANSEN, 1997), é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de vizinhanças. Contrariamente a outras metaheurísticas, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais "distantes" da solução corrente e focaliza a busca em torno de uma nova solução, se e somente se, um movimento de melhora é realizado. O método inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. Esta rotina de busca

local também pode usar diferentes estruturas de vizinhança.

Mais especificamente, esta heurística parte de uma solução inicial qualquer e a cada iteração seleciona aleatoriamente um vizinho dentro da vizinhança N^k da solução corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local for melhor que a solução corrente, a busca continua desta recomeçando da primeira estrutura de vizinhança. Caso contrário, continua-se a busca a partir da próxima vizinhança, N^{k+1}. Esta heurística é encerrada quando uma condição de parada for atingida, tal como o tempo máximo de processamento ou o número máximo de iterações consecutivas sem melhoramento. Os vizinhos da solução corrente são gerados aleatoriamente de forma a evitar ciclagem, situação que pode ocorrer se alguma regra determinística for usada.

3.6 VND

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND), (MLADENOVIC E HANSEN, 1997), também é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de vizinhanças. Nesse método, exploram-se seqüencialmente as vizinhanças, sendo que em cada uma delas procura-se o melhor vizinho. Quando esse vizinho não for de melhora em relação à solução corrente, passa-se para a vizinhança subseqüente. Em caso de melhora nessa última situação, retorna-se novamente à primeira vizinhança da seqüência. O método é interrompido quando não houver melhora em nenhuma das vizinhanças pesquisadas.

4. Procedimento heurístico híbrido para o PCVCP

Para se resolver o PCVCP desenvolveu-se o método GRASP+VNS, o qual combina as metaheurísticas GRASP, Método de Pesquisa em Vizinhança Variável (VNS) e Método de Descida em Vizinhança Variável (VND), descritas anteriormente.

A função que avalia uma solução do PCVCP e que deve ser minimizada é expressa pela seguinte fórmula:

$$f(x) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \; + \; \sum_{i \in V} \gamma_i (1 - y_i) \; + \alpha \; \times \max \{ 0, - \sum_{i \in V} p_i y_i \; + \; p_{\min} \} \qquad (4.1)$$

Como se observa, esta função é composta pelo somatório dos custos de deslocamento na rota, pelo somatório das penalidades pagas nos vértices que não foram visitados, e ainda, por uma parcela que penaliza, através de um peso α, uma solução na qual o somatório dos prêmios coletados seja menor que o prêmio mínimo pré-estabelecido. As demais restrições foram contempladas através da representação adotada, que impede que um vértice seja visitado mais de uma vez e que não haja a formação de ciclos na solução.

Para definir as vizinhanças de uma solução, foram implementados cinco tipos de movimentos, a saber:

m_1: Retirar vértice de maior economia: Utilizando-se o método *DROP-Step* (GOMES *et al.*, 2000) escolhe-se um dos vértice que compõe a rota e possui a maior economia, retirando-o da solução, sem levar em consideração se o prêmio deixará de ser coletado;

m_2: Inserir vértice de maior economia: Utilizando-se o método *ADD-Step* (GOMES *et al.*, 2000), verifica-se todos os vértices que ainda não estão na rota, escolhendo aquele que possuir a maior economia de inserção, inserindo-o na solução;

m_3: Trocar 2 vértices da solução: Escolhe-se 2 vértices que fazem parte da solução, trocando-os de posição;

m_4: Retirar vértice: Escolhe-se aleatoriamente um vértice que faça parte da solução, removendo-o da solução;

m_5: Inserir vértice: Escolhe-se aleatoriamente um vértice que não faça parte da solução, inserindo-o na solução.

Esses movimentos, m_k, definem as vizinhanças $N^k(s)$, k = {1, 2, 3, 4, 5}, de uma solução s.

O pseudo-código do algoritmo híbrido GRASP+VNS proposto é descrito na Figura 1. Nesse algoritmo, executa-se a fase de construção *MaxIter* vezes, retornando-se a melhor solução construída, a qual é submetida a uma busca local. Este procedimento, denominado de GRASP com filtro, permite ainda na fase de construção eliminar soluções iniciais de baixa qualidade. Durante a construção, uma lista de candidatos restrita é criada a cada iteração, com

os vértices que oferecem as maiores economias de inserção. A seguir, um vértice desta lista é escolhido aleatoriamente e inserido na solução. Este procedimento de construção termina quando não mais existir economia positiva e o prêmio coletado for maior que o prêmio mínimo pré-estabelecido.

```
Procedimento GRASP+VNS
        f* ← ∞;
        // Fase de Construção
        para j = 1, ..., MaxIter faça
                s ← ∅;
                Inserir a origem em s;
                para todo k não pertencente a s faça
                        Calcule a economia de inserção;
                fim-para;
                enquanto prêmio < p_min ou existir economia positiva faça
                        LCR ← Lista dos vértices com maior economia;
                        Selecione aleatoriamente um vértice v ∈ LCR;
                        s ← s ∪ {v};
                        Atualizar Lista de Candidatos;
                fim-enquanto;
                se f(s) < f* então
                        s* ← s;
                        f* ← f(s);
                fim-se
        fim-para;
        s ← s*;
        // Fase de Busca Local
        Aplicar VNS(s);
        Retorne s;
fim GRASP+VNS
```

Figura 1 – Algoritmo GRASP + VNS

Como dito anteriormente, as soluções obtidas pelos algoritmos de construção não garantem necessariamente a obtenção de um ótimo local. Por isso, é sempre benéfica a fase de busca local. Neste trabalho foram implementadas as heurísticas VNS e VND como métodos de refinamento da solução construída. Esses métodos consistem em explorar o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança.

O algoritmo VNS aplicado ao problema é descrito pela Figura 2. Neste método, parte-se de uma solução inicial gerada pela fase de construção do algoritmo da Figura 1 e, a cada iteração, um vizinho *s'* na *k*-ésima vizinhança da solução corrente *s* é selecionado aleatoriamente, realizando-se o movimento m_k, onde $k = \{1, 2, 3, 4, 5\}$, definido anteriormente. Esse vizinho é então submetido a um procedimento de busca local, no caso o VND. Se a solução ótima local, *s''*, for melhor que a solução *s* corrente, a busca continua de *s''* recomeçando da primeira vizinhança. Caso contrário, continua-se a busca a partir da próxima vizinhança. Este procedimento é encerrado quando o tempo de execução sem melhora ultrapassar *MaxTempo* segundos.

```
Procedimento VNS(s)
    r ← Número de vizinhanças;
    enquanto tempo sem melhora < MaxTempo faça
        k ← 1;
        enquanto k ≤ r faça
            Selecione um vizinho s' qualquer na vizinhança N^k(s);
            s'' ← VND(s');
            se f(s'') < f(s) então
                s ← s'';
                k ← 1;
            senão
                k ← k + 1;
        fim-enquanto;
    fim-enquanto;
    Retorne s;
fim VNS
```

Figura 2 – Algoritmo VNS aplicado ao PCVCP

O procedimento VND, apresentado na Figura 3, faz uso de três procedimentos heurísticos de refinamento, a saber: (1) SeqDropSeqAdd, que consiste em retirar os vértices enquanto existir algum vértice com economia de remoção positiva (DROP_*step*) e adicionar os vértices enquanto existir economia de inserção positiva (ADD_*step*); (2) 2-*Optimal*, que examina todas as possíveis trocas de 2 arestas, realizando a que fornecer o maior ganho na função de avaliação e (3) AddDrop, que consiste em inserir o vértice que possuir a maior

economia de inserção (ADD_*step*) e retirar o vértice que possuir a maior economia de remoção (DROP_*step*). Cada iteração do método consiste na determinação de um ótimo local tendo por base o *k*-ésimo procedimento heurístico de refinamento, $k = \{1, 2, 3\}$. Sempre que se obtém uma solução de melhora, retorna-se ao primeiro procedimento heurístico de refinamento.

```
Procedimento VND(s)
        r = Número de procedimentos de refinamento;
        k ← 1;
        enquanto k ≤ r faça
                Seja s' um ótimo local segundo o k-ésimo procedimento de refinamento;
                se f(s') < f(s) então
                        s ← s';
                        k ← 1;
                senão
                        k ← k + 1;
        fim-enquanto
        Retorne s;
fim VND
```

Figura 3 – Procedimento VND Aplicado ao PCVCP

5. Experimentos Computacionais

O Problema do Caixeiro Viajante com Coleta de Prêmios, embora seja um problema com inúmeras aplicações, ainda é pouco explorado pela literatura afim. Ao que é de nosso conhecimento, dos poucos trabalhos associados ao PCVCP, não existe nenhuma biblioteca pública de problemas-teste. Uma comparação com o trabalho de GOMES et al. (2000) e MELO (2001) não foi possível, pois os dados utilizados não estão disponíveis. Sendo assim, para avaliar o método heurístico proposto neste trabalho, problemas-teste foram obtidos da seguinte forma: as distâncias entre as cidades (vértices do grafo) foram geradas de uma distribuição uniforme no intervalo [50, 1000], o prêmio e a penalidade associados a cada uma dessas cidades também foram gerados de uma distribuição uniforme, considerando respectivamente os intervalos [1, 100] e [1, 750]. O prêmio mínimo, p_{mim}, a ser coletado representa 75% do somatório de todos os prêmios associados às cidades. Os intervalos para a determinação dos prêmios e penalidades foram os mesmos utilizados em outros trabalhos encontrados na literatura. Os problemas-teste utilizados neste artigo estão disponíveis em

http://www.lac.inpe.br/~chaves/instancias.html e em *http://www.decom.ufop.br/prof/marcone/Projetos/pcvcp/instancias.htm*.

A seguir, apresentam-se os resultados computacionais obtidos a partir de 30 execuções realizadas para cada problema-teste gerado. Todos os algoritmos foram implementados na linguagem C++ e os testes computacionais foram realizados sob o sistema operacional Windows, em um microcomputador com processador Athlon XP de 1,53 GHz e 256 MB de memória RAM. Os parâmetros adotados para o método GRASP+VNS, conforme seções 3 e 4, foram os seguintes: *MaxIter* = 4000, *MaxTempo* = 200 segundos e $\alpha = 0{,}2$.

Na Tabela 1 têm-se os resultados dos experimentos computacionais. Na primeira coluna estão os problemas-teste do PCVCP; na coluna 2, |V| representa a cardinalidade do conjunto de vértices que compõem o problema. A terceira e quarta colunas dizem respeito ao algoritmo exato, e representam respectivamente o tempo de execução, em segundos, e o valor do ótimo global. As colunas 5, 6 e 7 referem-se ao modelo heurístico. Na quinta e sexta colunas encontram-se o tempo médio de execução, em segundos, e os melhores valores da função de avaliação (*FOMelhor*) obtidos com o modelo heurístico. Na sétima coluna tem-se o desvio dos valores médios (*FOMedia*) encontrados em relação à melhor solução obtida em cada um dos problemas-teste, calculado conforme equação 5.1.

$$\text{Desvio} = \frac{FOMedia - FOMelhor}{FOMelhor} \tag{5.1}$$

		Método Exato		**GRASP+VNS**		
Problema-teste	**| V |**	**Ótimo Global**	**Tempo (s)**	**FOMelhor**	**Tempo (s)**	**Desvio (%)**
v10	11	1765	1	1765	0,10	0,00
v20	21	2302	65	2302	1,04	0,00
v30a	31	3582	86	3582	5,43	0,00
v30b	31	2515	100	2515	3,83	0,00
v30c	31	3236	1786	3236	7,83	0,05
v50a	51	-	10800	4328	132,45	0,42
v50b	51	-	10800	3872	43,76	0,31
v100a	101	-	-	6892	692,09	0,52
v100b	101	-	-	6814	446,81	0,12
v250a	251	-	-	15310	918,33	0,88

v250b	251	-	-	14678	996,72	0,76
v500a	501	-	-	28563	2145,79	0,67
v500b	501	-	-	28524	2410,21	0,82

Tabela 1 – Resultados dos experimentos computacionais

Pela aplicação do *solver* LINGO, versão 7.0, usando a formulação de programação matemática descrita na seção 2, foi possível encontrar o ótimo global somente para os problemas-teste com até 31 vértices. Observa-se que o modelo heurístico também conseguiu encontrar o ótimo global, com um desvio pequeno em relação às soluções geradas. Este fato contribui para a validação do método heurístico proposto neste trabalho. Observa-se, também, que nos problemas-teste *v50a* e *v50b*, o algoritmo exato não conseguiu encontrar nenhuma solução viável em 3 horas de execução. Entretanto, o método heurístico é capaz de produzir soluções viáveis em menos de 5 minutos.

Para os problemas-teste entre 101 e 501 vértices, as soluções foram geradas em um tempo relativamente pequeno, considerando a complexidade dos problemas, e o desvio também foi pequeno. Apesar de não se poder afirmar o quão perto essas soluções estão da solução ótima, o desempenho do método heurístico em problemas-teste de menores dimensões possibilita prever que essas devem ser boas soluções.

Na Tabela 2 procura-se mostrar a vantagem da utilização do filtro na fase de construção GRASP. Para isso, foram realizados testes com uma abordagem sem filtro na fase de construção. Sendo que, na terceira coluna é apresentado o valor da função de avaliação da solução obtida utilizando o GRASP com filtro, e na quarta coluna tem-se o valor da função de avaliação da solução obtida utilizando o GRASP sem filtro. Na quinta coluna tem-se o desvio em relação aos melhores resultados encontrados nas duas abordagens.

Observa-se, pela Tabela 2, que a abordagem GRASP com filtro encontra soluções melhores que a abordagem GRASP sem filtro para os problemas-teste de maior porte. Uma vez que as soluções encontradas pela fase de construção do GRASP com filtro são geralmente melhores do que as soluções encontradas com o GRASP sem filtro, isto acelera o processo de refinamento, proporcionando obter soluções finais melhores.

Problema-teste	| V |	GRASP com Filtro + VNS	GRASP sem Filtro + VNS	Melhora %
v10	11	1765	1765	0,00
v20	21	2302	2302	0,00
v30a	31	3582	3582	0,00
v30b	31	2515	2515	0,00
v30c	31	3236	3236	0,00
v50a	51	4328	4378	1,14
v50b	51	3872	3945	1,85
v100a	101	6892	7345	6,17
v100b	101	6814	7193	5,27
v250a	251	15310	16036	4,53
v250b	251	14678	15376	4,54
v500a	501	28563	31000	7,86
v500b	501	28524	29858	4,47

Tabela 2 – Comparação entre GRASP com e sem filtro na fase de construção

6. Conclusões

Este artigo contribui com a apresentação de um método heurístico híbrido, baseada em GRASP, VNS e VND, para resolver aproximadamente o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP).

Na comparação dos resultados encontrados pelos métodos exato e heurístico, verifica-se que o algoritmo heurístico proposto sempre atingiu o ótimo global para os problemas-teste onde este é conhecido.

A metaheurística híbrida proposta mostrou-se também robusta, pois partindo-se de diferentes soluções iniciais chega-se a soluções finais que diferem da melhor solução encontrada com um pequeno desvio. Portanto, os resultados obtidos validam a utilização deste método para a resolução do Problema do Caixeiro Viajante com Coleta de Prêmios.

Tópicos interessantes para pesquisas futuras na área incluem estudos sobre a robustez do método aqui proposto, diante de mudanças nos parâmetros dos problemas-teste (por

exemplo, no grau de esparsidade dos grafos, na variabilidade dos valores aleatoriamente gerados, na percentagem de prêmio mínimo a ser coletado, etc) lidando com a conseqüente ampliação do número de experimentos via análise estatística dos resultados. Também é de interesse avaliar a qualidade das soluções obtidas em grafos realísticos (por exemplo, planares), bem como em grafos adaptados de problemas-teste criados para o caixeiro viajante original e suas variantes, disponíveis em bibliotecas consagradas (BEASLEY, 1990, TSPLIB).

Agradecimentos:

Os autores agradecem aos revisores anônimos que muito contribuíram para a melhoria deste trabalho por meio de suas correções e sugestões.

Referências

BALAS, E., (1989) - The prize collecting traveling salesman problem. Networks 19, 621-636.

BALAS, E., (2001) - The Prize Collecting Traveling Salesman Problem and Its Applications, *Management Science Research Report*, MSRR-664.

BEASLEY, J.E., (1990) – OR-Library: distributing test problems by eletronic mail, Journal of the Operational Research Society, 41, 1069-1072.

CHAVES, A. A., BIAJOLI, F. L., MINE, O. M., SOUZA, M. J. F. (2003) - Modelagens Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios. Relatório Técnico – DECOM, Universidade Federal de Ouro Preto, Ouro Preto. Disponível em http://www.decom.ufop.br/prof/marcone/Orientacoes/OrientacoesConcluidas.htm.

CROES, G. (1958) - A method for solving travelling salesman problems.” Operations Research, 6, 791–812.

DELL’AMICO, M.; MAFFIOLI, F. & SCIOMANCHEN, A. (1998) - A Lagrangian Heuristic

for the Prize Collecting Travelling Salesman Problem.” Annals of Operations Research, 81, 289–305.

DORIGO, M., MANIEZZO, V. & COLORNI, A., (1996) - The Ant System: Optimization by a colony of cooperating agents**.** *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26, 1, 29-41.

FEO, T.A. & RESENDE, M.G.C., (1995) - Greedy randomized adaptive search procedures, *Journal of Global Optimization*, 6:109-133.

GOEMANS, M. & Willianson, D. (1992) - A General Approximation Tchnique for Constrained Forest Problems, *In Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, 307-315.

GOLDBARG, M. C. & LUNA, H. P., (2000) - Otimização combinatória e programação linear: modelos e algoritmos. Rio de Janeiro: Editora Campus.

GOLDBERG, D. E., (1989) - Genetic Algorithms in Search, *Optimization and Machine Learning*, Addison-Wesley, Berkeley.

GOMES, L., DINIZ, V. & MARTINHON, C.A., (2000) - An Hybrid GRASP+VND Metaheuristic for the Prize-Collecting Traveling Salesman Problem, XXXII Simpósio Brasileiro de Pesquisa Operacional, 1657-1665.

GLOVER, F., (1986) - Future Paths for Integer Programming and links to Artificial Intelligence, *Computers and Operations Research*, 5:553-549.

KIRKPATRICK, S, GELLAT, D.C. & VECCHI, M.P., (1983) - Optimization by Simulated Annealing, *Science*, 220: 671-68.

MELO, V. A., (2001) - Metaheurísticas para o Problema do Caixeiro Viajante com Coleta de Prêmios, Dissertação de Mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói.

MLADENOVIC, N. & HANSEN, P., (1997) - Variable Neighborhood Search. *Computers and Operations Research*, 24:1097-1100.

SHRAGE, L., (1991) - User's Manual for LINGO, LINDO Systems Inc, Chicago, IL.

TSPLIB. Disponível em http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95. Acessado em 12/07/2006.