MODELAGENS EXATA E HEURÍSTICA PARA RESOLUÇÃO DE UMA GENERALIZAÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE

Antônio Augusto Chaves
Instituto Nacional de Pesquisas Espaciais (INPE)
Rua Boulevar Vila Lobos, 41 ap. 23 G, Jardim das Colinas
CEP 12242-021 São José dos Campos, SP chaves@lac.inpe.br

Fabrício Lacerda Biajoli
Instituto Nacional de Pesquisas Espaciais (INPE)
Rua Boulevar Vila Lobos, 41 ap. 23 G, Jardim das Colinas
CEP 12242-021 São José dos Campos, SP lacerda@lac.inpe.br

Otávio Massashi Mine
Universidade Federal do Espirito Santo (UFES)
Rua Telmo Torres s/n, Cond. Mar Azul 2, apto 101, Itapoã
CEP 29101-931 Vila Velha, ES otavio@autovidros.com.br

Marcone Jamilson Freitas Souza
Departamento de Computação - Universidade Federal de Ouro Preto (UFOP)
Programa de Pós-Graduação em Engenharia Mineral
CEP 35.400-000 Ouro Preto, MG marcone@iceb.ufop.br

Resumo
O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) pode ser associado a um caixeiro viajante que coleta um prêmio w_k, não negativo, em cada cidade k que ele visita e paga uma penalidade p_l para cada cidade l 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, w_{min}, pré-estabelecido. O presente artigo contribui com o desenvolvimento de metaheurísticas híbridas para o PCVCP, baseadas em GRASP e métodos de pesquisa em vizinhança variável (VNS/VND) para solucionar aproximadamente o PCVCP. De forma a validar as soluções obtidas, implementa-se um modelo exato para encontrar a melhor solução para o problema, sendo este modelo aplicado a problemas de pequeno porte. Resultados computacionais demonstram a eficiência da metodologia híbrida proposta, tanto em relação à qualidade da solução final obtida quanto em relação ao tempo de execução.
Palavras chave: Problema do Caixeiro Viajante com Coleta de Prêmios, Método de Pesquisa em Vizinhança Variável, GRASP.

Abstract
The Prize Collecting Traveling Salesman Problem (PCTSP) can be associated a travelling salesman that collects a prize w_k, not negative, in each city k that he visits and he pays a penalty p_l for each city l that doesn't visit, with a cost displacement c_{ij} between the cities i and j. The problem is minimizing the total of cost of trip and penalties, while it includes in your route an enough number of cities that allow to collect a minimum prize, w_{min}. 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, an exact model is implemented to find the best solution for the problem, being this model 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 relation to the time of execution.

Key words: Prize Collecting Traveling Salesman problem, Variable Neighborhood Search, GRASP.

1. Introdução

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), referido na literatura como *Prize Collecting Traveling Salesman Problem* (PCTSP), é uma variante do Problema do Caixeiro Viajante, podendo ser associado a um caixeiro viajante que coleta um prêmio w_k, não negativo, em cada cidade k que ele visita e paga uma penalidade p_l para cada cidade l 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, $w_{\min}$, pré-estabelecido.

O PCVCP foi formulado inicialmente por Egon Balas (BALAS, 1986) 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 através 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. DELL' AMICO *et al*. (1995) propuseram o uso da relaxação lagrangeana para solucionar a ordem de visitas a clientes de uma empresa num determinado período, cujo objetivo era maximizar o total de valor das solicitações dos clientes menos o custo de deslocamento e venda, estando sujeitos a restrições de tempo.

GOMES *et al.* (2000) e MELO (2001) desenvolveram trabalhos que utilizam metaheurísticas híbridas para o PCVCP. Ambos utilizam a metodologia GRASP (FEO & RESENDE, 1995) associada a métodos de busca em vizinhança variáveis (VND e 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 frustada 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. Assumindo que a distância de uma cidade i a outra j seja simétrica, isto é, que $d_{ij} = d_{ji}$, o número total de soluções possíveis é $(n - 1)! / 2$. Desta forma, a resolução do PCVCP por enumeração completa é inviável para valores elevados de n.

Dado esse aspecto combinatorial, 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. Com o aparecimento das metaheurísticas, as quais possuem ferramentas que possibilitam escapar das armadilhas dos ótimos locais, permitindo a busca em outras regiões do espaço de busca,

muito progresso se tem conseguido com relação à qualidade das soluções finais obtidas com os procedimentos heurísticos.

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 *Greedy Randomized Adaptive Search Procedure* (GRASP), *Variable Neighborhood Search* (VNS) e *Variable Neighborhood Descent* (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 (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 implementação de um modelo exato para encontrar a melhor solução para o problema, sendo esse modelo aplicado a problemas de pequeno porte. A importância do modelo exato está no fato deste permitir a validação das soluções obtidas pela metodologia heurística desenvolvida.

Este trabalho está organizado como segue. Na seção 2 apresenta-se um modelo de programação 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. Modelagem Exata

O PCVCP pode ser definido como segue. Seja $G' = (V, E)$ um grafo completo não direcionado, onde para cada aresta (i,j) de E é dado um custo c_{ij}, e para cada vértice i de V, é associado uma penalidade p_i, a ser paga se o vértice i não compor a rota. Adicionalmente, para cada vértice i, existe um prêmio w_i associado. 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, será utilizado $w_0 = 0$ e $p_0 = \infty$.

A formulação aqui apresentada foi proposta por Egon Balas (BALAS, 2001). As restrições 2.5 e 2.6, não contidas nesse trabalho, são inseridas para eliminar a existência de subrotas. 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).

A restrição 2.2 diz 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. A restrição 2.3 diz 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. A restrição 2.4 assegura 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. A restrição 2.5 garante que o fluxo que chega e que sai do vértice *i* é igual a 1 caso o vértice seja visitado, e 0 caso contrario. A restrição 2.6 assegura 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 e 2.8 asseguram a bivalência das variáveis *x* e *y*, respectivamente.

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

sujeito à :

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

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

$$\sum_{i\in V} w_i y_i \geq w_{\min} \qquad (2.4)$$

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

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

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

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

Para a resolução desta formulação matemática, fez-se uso do *software* LINGO 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 esta metodologia exata 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:

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

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:

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

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 de substituição 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 computacionalmente barata.

3.4 GRASP (Greedy Randomized Adaptive Search Procedure)

GRASP (*Greedy Randomized Adaptive Search Procedure*), (FEO & RESENDE, 1995), pode ser vista como uma metaheurística que utiliza as boas características dos algoritmos puramente gulosos e dos procedimentos aleatórios na fase de construção de soluções viáveis.

GRASP é um processo 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 inicial 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 gulosa $f: C \rightarrow R$, 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). Esta técnica de escolha permite que diferentes soluções sejam geradas em cada iteração GRASP. Um parâmetro $\alpha \in [0,1]$ controla o nível de gulosidade e aleatoriedade da fase de construção. Um valor α igual a 0 faz gerar soluções puramente gulosas, enquanto que α igual a 1 faz produzir soluções totalmente aleatórias.

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 (Variable Neighborhood Search)

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 (Variable Neighborhood Descent)

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, explora-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 problema do Caixeiro Viajante com Coleta de Prêmios e que deve ser minimizada é expressa pela seguinte fórmula:

$$f(x) = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \; + \; \sum_{i \in V} p_i (1 - y_i) \; + \alpha \; * \max \{ 0, - \sum_{i \in V} w_i y_i \; + \; w_{\min} \}$$

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, implementou-se cinco tipos de movimentos, a saber:

- m_1: Retirar vértice de maior economia: Utilizando-se o método *DROP-Step* (GOMES *et al.*, 2000), descrito na seção 3.2, escolhe-se o vértice que compõe a rota e possui a maior economia positiva, retirando-o da solução;
- m_2: Inserir vértice de maior economia: Utilizando-se o método *ADD-Step* (GOMES *et al.*, 2000), descrito na seção 3.1, verifica-se para os vértices que ainda não estão na rota, aquele que possui 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 $N^k(s)$, $k = 1, ..., 5$, de uma solução s.

O pseudo-código do algoritmo híbrido GRASP+VNS proposto é descrito na Figura 4.1. 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 < w_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-para;
        s ← s*;
        // Fase de Busca Local
        Aplicar VNS(s);
        Retorne s;
fim GRASP+VNS
```

Figura 4.1 – Algoritmo GRASP + VNS

Como dito anteriormente, as soluções obtidas através dos algoritmos de construção não garantem necessariamente a obtenção de um ótimo local. Por isso, é sempre benéfica a fase de busca local. Neste trabalho implementou-se as heurísticas VNS e VND como métodos de refinamento da solução construída. Esses métodos consistem em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. O algoritmo VNS aplicado ao problema é descrito pela Figura 4.2.

```
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 4.2 – Algoritmo VNS aplicado ao PCVCP

A partir da solução inicial gerada, a cada iteração seleciona-se aleatoriamente um vizinho *s'* na *k*-ésima vizinhança da solução corrente *s*, realizando-se o movimento m_k, onde $k = 1,...,5$, 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.

O procedimento VND desenvolvido, apresentado na Figura 4.3, faz uso de $r=3$ 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 fonecer 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 4.3 – 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 testes. Sendo assim, para avaliar a modelagem heurística proposta neste trabalho, instâncias foram obtidas da seguinte forma: as distâncias entre as cidades (vértices do grafo) foram geradas aleatoriamente no intervalo [50, 1000], o prêmio e a penalidade associados a cada uma dessas cidades também foram gerados aleatoriamente, considerando respectivamente os intervalos [1, 100] e [1, 750]. O prêmio mínimo, w_{min}, a ser coletado representa 75% do somatório de todos os prêmios associados às cidades. A metodologia para a determinação dos prêmios e penalidades foi a utilizada no trabalho de MELO (2001). As instâncias utilizadas para teste neste artigo encontram-se disponíveis em http://www.decom.ufop.br/prof/marcone/Orientacoes/OrientacoesConcluidas.htm.

A seguir, apresentam-se os resultados computacionais obtidos a partir de 30 testes realizados em cada uma das instâncias geradas. 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 3.4 e 4, foram os seguintes: *MaxIter* = 4000, *MaxTempo* = 200 segundos e $\alpha = 0{,}2$.

Na Tabela 5.1 tem-se os resultados dos experimentos computacionais. Na primeira coluna estão as instâncias do problema; na coluna 2, |V| representa a cardinalidade do conjunto de vértices que compõem a instância. A terceira e quarta colunas dizem respeito ao modelo 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 encontrados em relação à melhor solução obtida em cada uma das instâncias, isto é:

$$\text{Desvio} = \left(\frac{FOMedia - FOMelhor}{FOMelhor} \right) \times 100$$

		Modelo Exato		GRASP+VNS		
Instância	**| V |**	**Tempo (s)**	**Ótimo Global**	**Tempo (s)**	**FOMelhor**	**Desvio**
v10	11	1	1765	1	1765	0,00 %
v20	21	65	2302	2	2302	0,00 %
v30a	31	86	3582	6	3582	0,00 %
v30b	31	100	2515	4	2515	0,00 %
v30c	31	1786	3236	11	3236	0,05 %
v50a	51	10800	-	326	4328	0,42 %
v50b	51	10800	-	292	3872	0,31 %
v100a	101	-	-	510	6892	0,52 %
v100b	101	-	-	446	6814	0,12 %
v250a	251	-	-	1033	15310	0,88 %
v250b	251	-	-	796	14378	0,76 %
v500a	501	-	-	2140	28842	0,67 %
v500b	501	-	-	2410	28524	0,82 %

Tabela 5.1 – Resultados dos experimentos computacionais

Apenas para as instâncias até 30 vértices foi possível encontrar o ótimo global através da formulação exata descrita na seção 2. 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 modelo heurístico proposto neste trabalho. Observa-se, também, que nas instâncias v50a e v50b, o modelo exato não conseguiu encontrar nenhuma solução viável em 3 horas de execução. Entretanto, o modelo heurístico é capaz de produzir soluções viáveis em menos de 5 minutos.

Para as instâncias entre 100 e 500 cidades, 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 procedimento heurístico proposto em instâncias menores possibilita prever que essas devem ser boas soluções.

6. Conclusão

Este artigo contribui com a apresentação de uma metodologia heurística híbrida, baseada em GRASP (*Greedy Randomized Adaptive Search Procedures*), VNS (*Variable Neighborhood Search*) e VND (*Variable Neighborhood Descent*), para resolver aproximadamente o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP).

Na comparação dos resultados encontrados nas modelagens exata e heurística, verifica-se que os algoritmos heurísticos sempre atingiram o ótimo global para as instâncias 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 desta metodologia para a resolução do Problema do Caixeiro Viajante com Coleta de Prêmios.

Referências

BALAS, E., (1986) - The prize collecting traveling salesman problem. ORSA/*Tims Meeting in Los Angeles*.

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

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.

DELL' AMICO, M., MAFFIOLI, F. & SCIOMACHEN, A. (1995) - A Lagrangian heuristic for the Prize CollectingTravelling Salesman Problem. *Annals of Operations Research*, 81: 297-306.

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, p. 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.