ALGORITMOS HÍBRIDOS PARA UMA GENERALIZAÇÃO DO PROBLEMA DO CAIXEIRO VIAJANTE

Antônio Augusto Chaves
Instituto Nacional de Pesquisas Espaciais – INPE
São José dos Campos SP, Brasil
chaves@lac.inpe.br

Luiz Antônio Nogueira Lorena
Instituto Nacional de Pesquisas Espaciais – INPE
São José dos Campos SP, Brasil
lorena@lac.inpe.br

Resumo

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) é uma generalização do Problema do Caixeiro Viajante, podendo 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 objetivo é minimizar o somatório dos custos da viagem e penalidades pagas, incluindo na rota um número suficiente de cidades que permitam coletar um prêmio mínimo pré-estabelecido. Este trabalho aborda novas técnicas heurísticas para resolver o PCVCP, utilizando um algoritmo evolutivo híbrido, chamado *Evolutionary Clustering Search* (ECS) e uma adaptação deste, chamada *CS, onde o componente evolutivo do ECS será substituído pelas metaheurísticas GRASP e VNS. A validação das soluções obtidas se dará através da comparação com os resultados encontrados através de *solver* comercial, que consegue resolver de forma exata apenas problemas de pequeno porte.

Palavras-Chaves: Problema do Caixeiro Viajante; *Evolutionary Clustering Search*; GRASP; VNS.

Abstract

The Prize Collecting Travelling Salesman Problem (PCTSP) is a generalization of the Travelling Salesman Problem, it can be associated a salesman that collects a prize in each city visited and pays a penalty for each city no visited, with a cost among the cities. The objective is to minimize the sum of the costs of the trip and penalties, including in the route an enough number of cities that allow collecting a minimum prize. This paper approaches new heuristic techniques to solve the PCTSP, using a hybrid evolutionary algorithm, called Evolutionary Clustering Search (ECS) and an adaptation of this, called *CS, where the evolutionary component will be substituted by the metaheuristics GRASP and VNS. The validation of the obtained solutions will be through the comparison with the results found by a commercial solver that was able to solve only small size problem.

Keywords: Travelling Salesman Problem; Evolutionary Clustering Search; GRASP; VNS;

1. INTRODUÇÃO

Este trabalho tem como objetivo abordar novas técnicas heurísticas para resolver o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), referido na literatura como *Prize Collecting Traveling Salesman Problem*. Sendo uma generalização do Problema do Caixeiro Viajante, pode ser associado a um caixeiro que coleta um prêmio p_k, não negativo, em cada cidade k visitada e paga uma penalidade γ_t para cada cidade t não visitada, 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 pagas, enquanto inclui na sua rota um número suficiente de cidades que permitam coletar um prêmio mínimo, p_{min}, pré-estabelecido.

A dificuldade de solução do PCVCP está no número elevado de soluções existentes. Admitindo que o custo de deslocamento entre as cidades *i* e *j* seja simétrica, isto é, $c_{ij} = c_{ji}$, o número total de soluções possíveis é $(n - 1)! / 2$, sendo classificado na literatura como *NP-difícil*, isto é, não são conhecidos algoritmos que o resolvam em tempo polinomial. Mesmo com os rápidos avanços tecnológicos dos computadores, uma enumeração completa de todas essas soluções é inconcebível para valores elevados de *n*. Para se resolver o PCVCP por enumeração, para $n = 20$, tem-se 6×10^{16} rotas possíveis, desta forma, um computador que avalia uma rota em cerca de 10^{-8} segundos, gastaria 19 anos para examinar todas as rotas.

Problemas desta natureza são comumente abordados através de heurísticas. Define-se heurística como sendo uma técnica que procura boas soluções a um custo computacional razoável, sem, no entanto, estar capacitada a garantir se esta é o ótimo global, nem quão próxima uma determinada solução está do ótimo global. A grande desvantagem das heurísticas reside na dificuldade de fugir de ótimos locais, o que deu origem à outra metodologia, chamada de metaheurística, que possui ferramentas que possibilitam sair destes ótimos locais, permitindo a busca em regiões mais promissoras. O grande desafio é produzir, em tempo razoável, soluções tão próximas quanto possíveis do ótimo global.

Para a resolução do PCVCP, faz-se uso de conceitos de uma técnica heurística híbrida recente, proposta por Oliveira e Lorena (2004) [14], chamada *Evolutionary Clustering Search* (ECS) que consiste em detectar dinamicamente regiões promissoras de busca baseando-se na freqüência em que são amostrados nestas regiões os indivíduos gerados por um algoritmo evolutivo. Estas regiões promissoras devem ser exploradas tão logo sejam descobertas, através de métodos de busca local. Uma outra abordagem para esta técnica de busca através de agrupamentos é substituir o algoritmo evolutivo por outras metaheurísticas, tais como GRASP (Feo e Resende, 1995) [6] e VNS (Mladenovic e Hansen, 1997) [13], dando origem à outra abordagem, chamada *CS.

De forma a validar as soluções obtidas, propõe-se uma formulação matemática baseada em Balas (1989) [1] e Torres e Brito (2003) [17]. Utiliza-se o *software* CPLEX [10] para resolver esta formulação para instâncias de pequeno porte.

O restante do trabalho está organizado da seguinte forma. A seção 2 faz uma breve revisão bibliográfica sobre o PCVCP. Na seção 3 é mostrada uma formulação matemática para o PCVCP. Na seção 4 são descritas as técnicas que serão utilizadas neste trabalho. A seção 5 apresenta, em detalhes, o ECS e o *CS aplicados ao PCVCP. A seção 6 apresenta os resultados obtidos pelos métodos ECS e *CS, além dos resultados do CPLEX. Finalmente, na seção 7, são descritas algumas considerações a respeito desse trabalho.

2. REVISÃO BIBLIOGRÁFICA

O PCVCP foi proposto por Balas (1989) [1], que apresentou algumas propriedades estruturais do problema e duas formulações matemáticas para este, sendo a segunda uma simplificação da primeira. O autor apresenta também um algoritmo para o problema, que combina a técnica de *cutting planes* com uma relaxação de programação linear a ser resolvida através do método *simplex*. Balas e Martin (1985) [2] desenvolveram um *software* para programação diária de uma fábrica de lâminas de aço, que utilizava a combinação de várias heurísticas para procurar boas soluções para o PCVCP.

Goemans et al. (1995) [8] desenvolveram um algoritmo 2-aproximativo para uma versão do PCVCP, onde o objetivo era minimizar o custo da rota e a penalidade paga nos vértices não visitados, não levando em consideração o prêmio mínimo a ser coletado.

Dell'Amico et al. (1998) [5] exploraram uma relaxação Lagrangeana para o PCVCP, relaxando a restrição de prêmio mínimo, e aplicando o método subgradientes para resolver o

problema dual. Um algoritmo baseado na heurística clássica para o PCV, conhecida como heurística de inserção mais barata, é utilizado para transformar a solução relaxada encontrada em uma solução viável, através da inserção de vértices com melhores economias 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) [9] apresentam uma metaheurística híbrida para solucionar o PCVCP, combinando a metaheurística GRASP com VND. Melo e Martinhon (2004) [12] também apresentam uma metaheurística híbrida para o PCVCP, combinando as metaheurísticas GRASP Progressivo e VNS.

Torres e Brito (2003) [17] apresentam uma nova formulação matemática para o PCVCP baseada na formulação apresentada em [1]. Nesta formulação é proposto um novo conjunto de restrições para evitar a formação de sub-rotas na solução.

Chaves et al. (2004) [3] desenvolveram trabalhos abordando duas modelagens. Uma modelagem matemática resolvendo o PCVCP de forma ótima para problemas de pequenas dimensões, e uma modelagem heurística, combinando as metaheurísticas GRASP e VNS.

3. FORMULAÇÃO MATEMÁTICA PARA O PCVCP

O PCVCP pode ser representado em um grafo completo não direcionado G = (V, E), onde existe associado a cada vértice $k \in V$ um prêmio p_k e uma penalidade γ_k, e cada aresta $(i,j) \in E$ possui um custo de deslocamento, c_{ij}. Sendo o vértice 0, sem perda de generalidade, assumido como vértice origem. Ressaltando que o vértice origem possui prêmio nulo ($p_0 = 0$) e penalidade infinita ($\gamma_0 = \infty$).

A formulação matemática apresentada neste trabalho é baseada nas formulações propostas por Balas (1989) [1] e Torres e Brito (2003) [17]. Utiliza-se o conjunto de restrições propostas em [17] para evitar a formação de sub-rotas. A vantagem de utilizar esse conjunto de restrições é que o número de restrições utilizadas é polinomial, ao contrário das restrições comumente usadas no PCV que exigem um número exponencial de restrições para o atendimento desta condição.

Considere x_{ij} ($i, j \in V, i \neq j$) sendo uma variável binária igual a 1 se a aresta (i, j) pertencer a solução, e x_{ij} igual a 0 caso contrário. A variável x_{ii}, $i \in V$, controla se o vértice i está presente na rota, assumindo valor 0 caso seja visitado e valor 1 caso contrário. A variável f_{ij} ($i, j \in V, i \neq j$) é a quantidade de fluxo (valor em prêmios) escoada na aresta (i, j), sendo utilizada para evitar que a solução contenha sub-rotas.

A formulação matemática é apresentada a seguir, onde:

$$\min \sum_{i \in V} \sum_{j \in V} b_{ij} x_{ij} \qquad 3.1$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \qquad 3.2$$

$$\sum_{j \in V} x_{ji} = 1 \qquad \forall i \in V \qquad 3.3$$

$$\sum_{i \in V} p_i x_{ii} \leq \left(\sum_{i \in V} p_i \right) - p_{\min} \qquad 3.4$$

$$\sum_{j \in V \setminus \{0\}} f_{0j} = 0 \qquad 3.5$$

$$\sum_{j \in V \setminus \{i\}} f_{ij} = \sum_{j \in V \setminus \{i\}} f_{ji} + p_i x_{ii} \qquad \forall i \in V \setminus \{0\} \tag{3.6}$$

$$\sum_{j \in V \setminus \{0\}} f_{j0} = \sum_{j \in V \setminus \{0\}} p_j x_{jj} \tag{3.7}$$

$$f_{ij} > x_{ij} - 1 \qquad \forall i \in V \setminus \{0\},\ \forall j \in V,\ i \neq j \tag{3.8}$$

$$\left(\sum_{j \in V} p_j \right) x_{ij} = f_{ij} \qquad \forall i \in V / \{0\},\ \forall j \in V \tag{3.9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{3.10}$$

$$f_{ij} \geq 0 \qquad \forall i, j \in V \tag{3.11}$$

onde,

$$b_{ij} = \begin{cases} c_{ij} & se\ \ i = j \\ \gamma_{ij} & se\ \ i \neq j \end{cases}$$

A função objetivo 3.1 procura minimizar o somatório dos custos de deslocamento e as penalidades pagas. As restrições 3.2 e 3.3 garantem que se o vértice *i* for visitado (x_{ii} = 0) uma aresta tem que sair deste vértice e uma aresta tem que chegar neste vértice, caso o vértice não seja visitado (x_{ii} = 1) nenhuma aresta pode chegar ou sair deste. A restrição 3.4 assegura que a soma dos prêmios coletados será maior que o prêmio mínimo, p_{min}. As restrições 3.5, 3.6 e 3.7 garantem a conectividade da solução, ou seja, evitam rotas desconexas da origem. As variáveis de fluxo f_{ij} impedem que sub-rotas sejam criadas, associando a quantidade de prêmio do vértice visitado à aresta que sai deste. A quantidade de fluxo que sai do vértice origem tem que ser igual a 0, e a quantidade de fluxo que volta para o vértice origem tem que ser igual à soma dos prêmios coletados na rota. As restrições 3.8 e 3.9 conectam as variáveis x_{ij} e f_{ij}, fazendo com que a rota gerada por ambas seja a mesma. Sendo que a restrição 3.8 garante que se uma aresta fizer parte da solução, a quantidade de fluxo escoada por esta tem que ser maior que 0, e a restrição 3.9 assegura que o valor do fluxo escoado por uma aresta não será maior que o total de prêmios de todos os vértices. As restrições 3.10 asseguram a bivalência das variáveis x_{ij}. Por fim, as restrições 3.11 garantem que as variáveis f_{ij} sejam não-negativas.

Para resolução desta formulação, utilizou-se o *software* CPLEX versão 7.5 [10] buscando encontrar a solução ótima para o PCVCP. Entretanto, pela natureza combinatória do PCVCP, tal modelagem só consegue resolver instâncias pequenas do problema. Através de testes, verifica-se que esta modelagem já se torna inviável computacionalmente para instâncias com 50 vértices.

4. MÉTODOS DE SOLUÇÃO

Para resolução do PCVCP são propostas duas abordagens. Na primeira abordagem utiliza-se o ECS e na segunda uma adaptação do ECS, onde o algoritmo evolutivo, neste caso o ATP – Algoritmo de Treinamento Populacional [15], será substituído pelas metaheurísticas GRASP [6] e VNS [13], dando origem à abordagem *CS. A seguir são apresentados os métodos utilizados nestas abordagens.

4.1. EVOLUTIONARY CLUSTERING SEARCH (ECS)

O Evolutionary Clustering Search (ECS) ou busca evolutiva através de agrupamentos é uma técnica evolutiva proposta por Oliveira e Lorena (2004) [14]. O ECS consiste num processo de agrupamento de indivíduos para guiar uma busca evolutiva, sendo usado para gerar um conjunto de soluções de referência para regiões supostamente promissoras.

No ECS um processo de agrupamento iterativo trabalha simultaneamente com um algoritmo evolutivo, identificando grupos de indivíduos que merecem especial interesse. As regiões destes grupos de indivíduos devem ser exploradas tão logo sejam detectadas, através de heurísticas de busca local específicas. Espera-se uma melhoria no processo de convergência associado a uma diminuição no esforço computacional em virtude do emprego mais racional dos métodos busca local.

O ECS procura localizar regiões promissoras através do enquadramento destas por *clusters*. Um *cluster* é definido pela tripla $\mathcal{G} = \{c; r; \beta\}$ onde c e r são, respectivamente, o centro e o raio de uma área de busca promissora. Existe também uma estratégia de busca β associada ao *cluster*.

O centro é um indivíduo (solução) representante do *cluster*, que identifica a sua localização dentro do espaço de busca. O raio estabelece a distância máxima, a partir do centro, até a qual um indivíduo pode ser associado ao *cluster*.

Em um problema de otimização combinatória, o raio pode ser definido em função de alguma distância métrica, como o número de movimentos necessários para transformar uma solução candidata em outra dentro de uma vizinhança. A vizinhança neste caso está relacionada com a estratégia de busca β. A estratégia β é uma sistemática de intensificação de busca, na qual indivíduos de um *cluster* interagem entre si, ao longo do processo de agrupamento, gerando novos indivíduos na mesma região.

O ECS consiste em quatro componentes com atribuições diferentes e conceitualmente independentes. São eles:

- um algoritmo evolutivo (AE);
- um agrupador iterativo (AI);
- um analisador de agrupamentos (AA);
- um algoritmo de otimização local (AO);

A figura 1 ilustra conceitualmente o ECS.

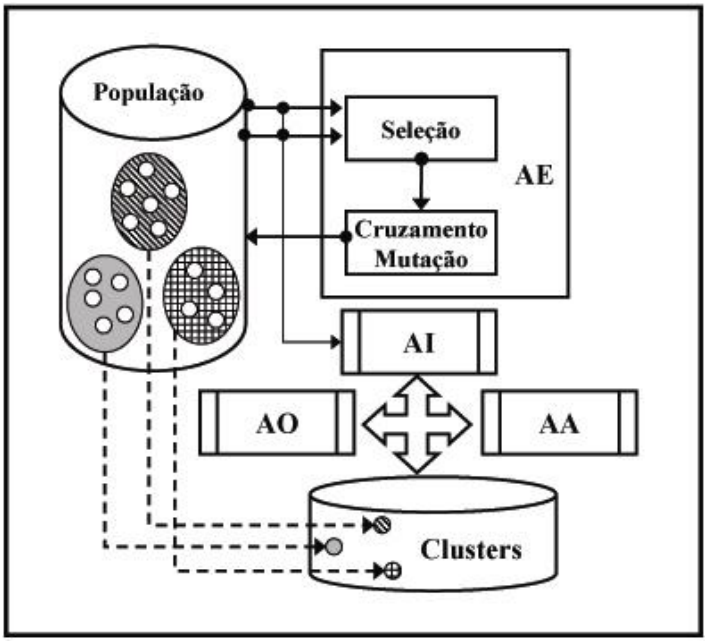

Figura 1 – Diagrama conceitual do ECS (adaptado [14])

O algoritmo evolutivo (AE) trabalha como um gerador de soluções de tempo integral. A população evolui independentemente dos componentes restantes. Indivíduos são selecionados, recombinados, e são atualizados para as próximas gerações. Simultaneamente, *clusters* são mantidos para representar estes indivíduos.

O agrupador iterativo (AI) é o núcleo do ECS, trabalhando como um classificador de informação que mantém no sistema apenas aquela que for relevante para o processo de intensificação de busca. Usa-se o termo informação, pois os indivíduos não se agrupam diretamente, e sim a informação semelhante que eles representam. Toda informação selecionada por AE é lida por AI que tenta agrupá-la como uma informação conhecida. Se a informação for considerada suficientemente nova, ela é mantida como um centro em um novo *cluster*. Caso contrário, a informação é considerada redundante, causando perturbação no centro do *cluster* mais similar. Tal perturbação é chamada de assimilação e consiste basicamente em atualizar o centro com a nova informação recebida.

O analisador de agrupamentos (AA) provê uma análise de cada cluster, em intervalos regulares de gerações, indicando um provável grupo promissor. Tipicamente, a densidade do *cluster* é usada nesta análise, ou seja, o número de seleções ou atualizações que aconteceram recentemente no *cluster*. Um *cluster* com densidade alta deve possuir um centro promissor. O AA também é responsável pela eliminação de *clusters* com baixas densidades.

Por fim, o algoritmo de otimização local (AO) é um módulo de pesquisa interno que provê a exploração de uma suposta região promissora. Este processo acontece depois que o componente AA tenha descoberto um *cluster* alvo. A busca local é realizada no indivíduo que esta no centro do *cluster*.

4.2. ALGORITMO DE TREINAMENTO POPULACIONAL (ATP)

Neste trabalho, utiliza-se como componente AE o Algoritmo de Treinamento Populacional (ATP), sendo este uma técnica evolutiva proposta por Oliveira e Lorena (2005) [15].

O ATP consiste em induzir uma população de indivíduos a assumir uma determinada característica, tornando-a adaptada a esta característica. A indução é realizada privilegiando os indivíduos mais adaptados e penalizando os menos. Heurísticas específicas sobre o problema tratado são utilizadas para determinar as características desejadas no treinamento. Estas heurísticas são utilizadas para buscar novas soluções viáveis melhores que as originais. Se a heurística encontrar uma solução melhor, diz-se que o indivíduo original não está bem-adaptado à heurística de treinamento empregada. Caso contrário, o indivíduo é o melhor dentro da vizinhança estabelecida pela heurística, devendo participar o máximo de gerações possíveis do processo de evolução.

O ATP trabalha com uma população dinâmica de indivíduos, sendo que, inicialmente a população é gerada aleatoriamente. Durante o processo evolutivo, a cada geração um novo indivíduo é gerado através do cruzamento entre dois indivíduos já existentes na população, privilegiando os indivíduos mais bem adaptados. O indivíduo gerado pode, eventualmente, sofrer alguma mutação, modificando suas características herdadas.

Cada novo indivíduo é avaliado por duas funções, f e g. A primeira avalia a qualidade do indivíduo e a segunda aplica a heurística de treinamento para avaliar a vizinhança do indivíduo, sendo a melhor solução encontrada atribuída como valor de g.

A adaptação de um indivíduo pode ser medida através do *ranking* δ, equação 4.1, que é composta:

- por um componente referente à adaptação do indivíduo em relação à heurística de treinamento (f - g);
- por um componente que privilegia a minimização da função g, calculando a distância, em termos de avaliação, entre o indivíduo e uma estimativa de um limite superior para todos os possíveis valores que as funções f e g podem

assumir, a constante G_{max}.

- e por uma constante d que tem o papel de equilibrar os componentes da equação.

$$\delta = d.[G_{\max} - g] - |\, f - g\,| \qquad 4.1$$

Em problemas de minimização, os indivíduos melhores adaptados são aqueles com baixos valores da heurística de treinamento e que estejam bem-adaptados a esta, produzindo valores maiores para os seus respectivos *rankings*.

A população é controlada dinamicamente por um limiar de rejeição, τ, que durante o processo evolutivo é atualizado através de incrementos adaptativos que consideram o tamanho atual da população, |P|, bem como a atual faixa de valores de *ranking* (do melhor, δ_1, ao pior, $\delta_{|P|}$), além de uma estimativa sobre o número de gerações que faltam para terminar o processo de evolução, *RG*.

O limiar de rejeição é dado por:

$$\tau_t = \tau_{t-1} + \varepsilon.|\, P\,|.\frac{(\delta_1 - \delta_{|P|})}{RG} \qquad 4.2$$

onde ε é uma constante que controla a velocidade do processo de esvaziamento da população, quanto menor mais lento é o processo evolutivo.

No fim de cada geração os indivíduos menos adaptados ($\delta \leq \tau$) são eliminados da população. No ATP a população tende a crescer, inicialmente, aceitando todos os novos indivíduos. Com o passar do tempo, o aumento do valor de τ determina uma quantidade cada vez maior de indivíduos a serem eliminados.

4.3. GRASP

GRASP (*Greedy Randomized Adaptive Search Procedure*) (Feo e Resende, 1995) [6] é uma metaheurística que objetiva encontrar soluções aproximadas para problemas de otimização combinatória. É composto de 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 pesquisado. 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 de candidatos (C), seguindo um critério de ordenação pré-determinado. Esse processo de seleção é baseado em uma função adaptativa gulosa h: $C \rightarrow \Re$, 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 promovidas pela 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. Este subconjunto recebe o nome de 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 quão gulosa será uma solução construída na 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.

4.4. VARIABLE NEIGHBORHOOD SEARCH (VNS)

O método de Pesquisa em Vizinhança Variável (*Variable Neighborhood Search*, VNS) inicialmente proposto por Mladenovic e Hansen (1997) [13], combina busca local com mudanças sistemáticas de vizinhança, procurando escapar de ótimos locais. Ao contrário da maioria dos outros métodos de busca local, o VNS não segue uma trajetória, mas explora vizinhanças cada vez mais distantes da solução corrente, pelo fato de gerar vizinhos aleatoriamente, movendo-se para a nova solução se e somente se uma melhora for produzida. Além disso, um método de busca local é aplicado repetidamente para transformar a solução vizinha em um ótimo local.

Considerando um conjunto finito pré-definido de estruturas de vizinhança N^k e com $N^k(s)$ sendo o conjunto de soluções da *k*-ésima vizinhança da solução corrente *s*. O método inicializa com uma solução inicial qualquer e a cada iteração é gerado aleatoriamente um vizinho, *s'*, dentro da vizinhança N^k da solução corrente. Aplica-se então um método de busca local no vizinho gerado, encontrando uma solução que representa um ótimo local, *s"*. Se este ótimo local for melhor que a solução corrente a busca continua a partir deste, recomeçando da primeira estrutura de vizinhança. Caso contrário, continua-se a busca a partir da próxima estrutura de 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 melhora da solução corrente.

5. IMPLEMENTAÇÃO

5.1. ECS APLICADO AO PCVCP

Um dos maiores desafios dos métodos de otimização é definir estratégias eficientes para cobrir todo o espaço de busca, possibilitando a aplicação de busca local somente em regiões realmente promissoras. No ECS o objetivo é detectar regiões promissoras através de agrupamento de genótipos. Explorando estas regiões, através de heurísticas de busca local, tão logo elas sejam encontradas.

A representação de um indivíduo será realizada através de um vetor que contém os vértices do problema na ordem em que são visitados, observando que o sinal negativo indica que o vértice não será visitado. A figura 2 ilustra a representação de um indivíduo, onde a seqüência de visitas é {1,3,0,4} e os vértices 2 e 5 não foram visitados.

1	3	-5	0	4	-2

Figura 2 – Representação de um indivíduo

Como componente AE do ECS, responsável por gerar soluções para alimentar os agrupamentos, será utilizado o Algoritmo de Treinamento Populacional (ATP) [15] empregando operadores genéticos conhecidos, como a seleção base-guia [11], o cruzamento BOX [16] e a mutação 2-Opt.

A cada geração um número constante de indivíduos (*NS*) é selecionado. A seleção é realizada privilegiando os indivíduos com maior *ranking*, selecionando dois indivíduos para recombinação, o primeiro indivíduo (s_{base}) é selecionado aleatoriamente entre os melhores indivíduos da população, e o segundo indivíduo (s_{guia}) é selecionado aleatoriamente dentro da população. Os indivíduos s_{base} e s_{guia} são combinados, através da cópia aleatória de blocos de ambos indivíduos, o que resulta na geração de um único filho. Este filho pode, eventualmente, sofrer mutação.

A heurística de treinamento utilizada para determinar as características desejadas no treinamento ao longo do processo evolutivo do ATP será o método *SeqAdd-SeqDrop* (Melo,

2001) [12], que consiste em realizar sucessivas tentativas de troca da solução corrente por uma solução melhor na sua vizinhança. O que pode ser conseguido através de inserções de vértices com economia de inserção negativa, seguida de remoções de vértices com economia de remoção negativa, não permitindo que o prêmio mínimo deixe de ser coletado.

O componente AI realiza um agrupamento iterativo de cada indivíduo selecionado para recombinação. Define-se inicialmente um número máximo de *clusters MC*, objetivando evitar que o processo de agrupamento se torne muito lento. O *i-ésimo cluster* tem o seu próprio centro c_i e um raio *r* que é comum aos demais *clusters*.

Se a distância métrica entre o indivíduo selecionado e o centro de algum *cluster* for menor que o raio deste, a informação do indivíduo deve causar uma perturbação (assimilação) no centro do *cluster* mais similar. Caso contrário, a informação do indivíduo é considerada suficientemente nova e esta é armazenada no centro de um novo *cluster*. Neste trabalho utilizou-se a métrica 2-Troca para calcular a distância entre duas soluções, esta métrica computa a quantidade de trocas necessárias para transformar o indivíduo selecionado no centro do *cluster*.

No processo de assimilação será utilizado o método *path-relinking* (Glover, 1996) [7], o qual realiza movimentos exploratórios na trajetória que interconecta o indivíduo selecionado e o centro do *cluster*. Assim sendo, o próprio processo de assimilação já é uma forma de busca local dentro do cluster, pois o centro vai ser deslocado para a melhor solução avaliada nessa trajetória, caso ela seja melhor que o centro do *cluster*.

O componente AA é executado toda vez que um indivíduo for atribuído a um *cluster*. A função do AA é verificar se o *cluster* já pode ser considerado promissor. Um *cluster* se torna promissor quando atinge uma certa densidade λ_t dada pela equação 5.1, significando que um certo padrão de informação se tornou predominante no processo de evolução, sendo o centro deste *cluster* refinado através do componente AO.

$$\lambda_t = PD.\frac{NS}{|C_t|} \qquad 5.1$$

onde PD é o número de vezes que a densidade deve estar acima do normal para um *cluster* ser considerado promissor, *NS* é o número de indivíduos selecionados no intervalo entre a análise dos clusters não ativados e $|C_t|$ é o número de *clusters* existentes na geração *t*.

O componente AA também tem como função executar um esfriamento de todos os *clusters* que foram ativados a cada geração, ou seja, diminuir a densidade dos clusters que foram ativados. Outra função do AA é eliminar, a cada geração, os *clusters* com baixa densidade.

No componente AO utiliza-se a heurística 2-Opt (Croes, 1958) [4], objetivando refinar a solução presente no centro de um *cluster* promissor. O método implementado no AO é executado enquanto a heurística 2-Opt estiver conseguindo encontrar soluções melhores para o centro c_i. Caso AO encontre uma solução que seja melhor que o centro do *cluster*, este é atualizado com a solução encontrada.

5.2. *CS APLICADO AO PCVCP

Outra abordagem proposta para resolver o PCVCP é o *CS, que propõe substituir o componente AE por uma outra metaheurística, sendo que esta precisa ser capaz de gerar um grande número de soluções diferentes para o processo de agrupamento. Neste trabalho propõe-se utilizar uma metaheurística híbrida, combinando GRASP e VNS.

Na fase de construção do GRASP utilizou-se a função de economia da heurística *Adding-Nodes* [5] para construir a lista dos elementos candidatos (C) a serem inseridos na solução. Cada elemento é selecionado de forma aleatória a partir de uma parte da lista C, contendo os melhores candidatos, chamada lista de candidatos restrita (LCR), sendo este elemento inserido na solução e a lista de candidatos atualizada. A fase de construção é executada enquanto existir candidatos com economia negativa ou a soma dos prêmios

coletados for menor que o prêmio mínimo pré-estabelecido.

Na fase de busca local do GRASP será utilizado o VNS procurando refinar a solução construída. Sendo que este explora o espaço de busca através de trocas sistemáticas de vizinhanças, aplicando um método de busca local sobre o vizinho gerado em uma determinada vizinhança.

As estruturas de vizinhanças são definidas através de movimentos aleatórios. No VNS proposto definem-se cinco estruturas de vizinhança, através dos seguintes movimentos:

- m_1: Inserir 2 vértices na rota;
- m_2: Retirar 2 vértices da rota;
- m_3: Trocar 4 vértices de posição;
- m_4: Inserir 1 vértice e retirar 1 vértice;
- m_5: Retirar 3 vértices;
- m_6: Retirar 1 vértice e trocar 4 vértices de posição;

A partir da solução construída, a cada iteração seleciona-se aleatoriamente um vizinho *s*' na *k-ésima* vizinhança da solução corrente *s*, realizando-se o movimento m_k, onde $k=\{1,2,3,4,5,6\}$ definido anteriormente. Esse vizinho é então submetido a um procedimento de busca local. Se ótimo local, *s*", for melhor que a solução *s* corrente, a busca continua de *s*" recomeçando da primeira estrutura de vizinhança. Caso contrário, continua-se a busca a partir da próxima vizinhança. Este procedimento é encerrado quando o tempo sem melhora for maior que 100 segundos.

Como um ótimo local dentro de uma vizinhança não é necessariamente o mesmo dentro de outra vizinhança, mudanças de vizinhanças também podem ser executadas na fase de busca local do VNS. Este método de busca local é chamado *Variable Neighborhood Descent* (VND) [13] e será utilizado neste trabalho para refinar o vizinho gerado.

O VND implementado faz uso de três procedimentos heurísticos de refinamento, sendo eles: (1) *SeqDrop-SeqAdd* [12], que consiste em retirar vértices enquanto existir algum vértice com custo de remoção negativo e adicionar vértices enquanto existir custo de inserção negativo; (2) 2-Opt [4], que examina todas as possíveis trocas de 2 arestas, realizando a que fornecer o maior ganho na função objetivo e (3) *AddDrop* [12], que consiste em inserir o vértice que possuir o menor custo de inserção e retirar o vértice que possuir o menor custo de remoção. Cada iteração do método consiste na determinação de um ótimo local tendo por base o *i-ésimo* procedimento heurístico de refinamento. Sempre que se obtém uma solução de melhora, retorna-se ao primeiro procedimento heurístico de refinamento.

A abordagem *CS possui os mesmos componentes que o ECS: agrupador iterativo (AI), analisador de agrupamentos (AA) e algoritmo de busca local (AO). Sendo que estes foram implementados de forma idêntica aos implementados no ECS, por isso não serão novamente descritos.

6. RESULTADOS

As abordagens ECS e *CS para o PCVCP foram codificados em C++ e os experimentos foram conduzidos em uma máquina AMD *Athlon XP 1.53 GHz e memória de 256 MB*. Os experimentos foram realizados com objetivo de evidenciar a flexibilidade do método em relação ao algoritmo utilizado para alimentar o processo de agrupamento, e também para validar as abordagens propostas, mostrando que algoritmos de busca por agrupamentos podem ser competitivos para resolução do PCVCP.

O PCVCP não possui uma biblioteca pública de problemas testes, sendo assim, um conjunto de dez instâncias para o PCVCP encontradas em [3] serão utilizadas nestes experimentos. Essas instâncias têm diferentes números de vértices ($n \in \{10, 20, 30, 50, 100, 250, 500\}$), sendo todas geradas aleatoriamente dentro dos seguintes intervalos:

- custo de deslocamento entre os vértices: $c_{ij} \in [50,1000]$;

- prêmio associado à cada vértice: $p_i \in [1,100]$;
- penalidade associada à cada vértice: $\gamma_i \in [1, 750]$;

Os valores para os parâmetros de desempenho das abordagens ECS e *CS foram ajustados através de várias execuções e também baseados no trabalho de Oliveira e Lorena (2004) [14].

- número de indivíduos selecionados a cada geração *NS* = 200;
- número máximo de *clusters MC* = 20;
- pressão de densidade PD = 2,5;
- limite superior G_{max} é o valor do pior indivíduo existente na população inicial;
- incremento do limiar de rejeição $\varepsilon = 0,001$;
- parâmetro $\alpha = 0,2$;

A formulação matemática apresentada na seção 3 foi resolvida utilizando o *software* CPLEX versão 7.5, sendo os resultados obtidos apresentados na tabela 1. Nota-se que, através desta formulação matemática, o CPLEX consegue resolver o PCVCP para instâncias com até 30 vértices em um tempo computacional razoável. Entretanto, para as instâncias maiores, o problema já se torna inviável computacionalmente. Para as instâncias com 50 vértices, por exemplo, o CPLEX levou vários dias executando para poder encontrar o ótimo global. Uma instância com 100 vértices (*v100a*) foi executada utilizando o resultado encontrado em [3] como valor *upper bound*, permitindo acelerar o processo de busca do CPLEX. Mesmo assim, está foi executada por vários dias e não conseguiu fechar o *gap* entre *lower* e *upper bounds*.

Na tabela 1 são apresentados os resultados computacionais encontrados pelo CPLEX e pelas abordagens ECS e *CS. Na primeira coluna estão as instâncias do problema; na coluna 2, |V| representa a cardinalidade do conjunto de vértices que compõem a instância. As colunas 3, 4 e 5 dizem respeito aos resultados obtidos pelo CPLEX, e representam respectivamente o valor da melhor solução viável obtida (*upper bound*), o tempo de execução, em segundos, e o *gap* entre o *lower* e *upper bounds*. As colunas 6 e 7 referem-se à abordagem ECS e as colunas 8 e 9 referem-se à abordagem *CS. Na sexta e oitava colunas encontram-se os melhores resultados obtidos com as abordagens, e na sétima e nona colunas o tempo em segundos gasto para encontrar a melhor solução.

Técnicas de Solução								
		CPLEX			**ECS**		***CS**	
Instância	**|V|**	**solução viável**	**tempo (s)**	***gap***	**FO**	**tempo (s)**	**FO**	**tempo (s)**
v10	11	1765	0.06	0	1765	0.1	**1765**	**0.05**
v20	21	2302	3.73	0	2302	7.75	**2302**	**0.97**
v30a	31	3582	34.06	0	3582	44.36	**3582**	**3.30**
v30b	31	2515	45.59	0	2515	67.48	**2515**	**3.07**
v30c	31	3236	164.58	0	3236	58.43	**3236**	**6.19**
v50a	51	4328	433439.97	0	4450	467.66	**4328**	**238.40**
v50b	51	3872	241307.43	0	3928	475.96	**3872**	**266.78**
v100a	101	6879	153059.09	2.46	7200	2067.98	**6832**	**919.27**
v250a	251	-	-	-	16200	3807.45	**15273**	**1451.37**
v500a	501	-	-	-	32396	7658.23	**28460**	**1718.73**

Tabela 1 – Resultados dos experimentos computacionais

Segundo a tabela 1, ECS e *CS encontraram o ótimo global para as instâncias *v10, v20, v30a, v30b, v30c*, mas *CS encontra o ótimo em um tempo computacional menor. Para as instâncias maiores, *CS obteve resultados melhores que o ECS, encontrando o ótimo global para as instâncias com 50 vértices.

A tabela 2 apresenta uma comparação entre os resultados encontrados utilizando apenas os métodos ATP e GRASP/VNS e os resultados utilizando estes métodos como geradores de soluções para o processo de agrupamentos nas abordagens ECS e *CS. Observa-se que a inclusão do método de busca por agrupamentos provoca uma melhora nas soluções obtidas pelo ATP e pelo GRASP/VNS para as instâncias maiores. Para as instâncias menores, não há melhora na solução através da utilização do ECS e *CS, uma vez que os métodos ATP e GRASP/VNS também conseguem encontrar o ótimo global. Porém, com os métodos de busca por agrupamentos é possível encontrar estas soluções através de um número menor de soluções ou indivíduos gerados pelo ATP e GRASP/VNS.

Instância	ATP	ECS	melhora (%)	GRASP/VNS	*CS	melhora (%)
v10	1765	1765	0	1765	1765	0
v20	2302	2302	0	2302	2302	0
v30a	3582	3582	0	3582	3582	0
v30b	2515	2515	0	2515	2515	0
v30c	3236	3236	0	3236	3236	0
v50a	4484	4450	0.75	4328	4328	0
v50b	3992	3928	1.60	3872	3872	0
v100a	7462	7200	3.51	6884	6832	0.75
v250a	16369	16200	1.03	15472	15273	1.29
v500a	32962	32396	1.72	28564	28460	0.36

Tabela 2 – Comparação entre ATP e ECS e entre GRASP/VNS e *CS

A tabela 3 apresenta os resultados obtidos através da execução do método proposto em Chaves et al. (2004) [3] utilizando o conceito de metaheurística híbrida, combinando GRASP e VNS/VND. Mostra-se também uma comparação com os resultados obtidos neste trabalho, através das abordagens ECS e *CS. Observando esta tabela, a abordagem *CS obteve os melhores resultados na resolução do PCVCP, além dos valores de função objetivo para as instâncias maiores serem melhores, os tempos computacionais necessários para encontrar a melhor solução foram menores.

Instância	Chaves et al. (2004)	tempo (s)	ECS	tempo (s)	*CS	tempo (s)
v10	1765	0.1	1765	0.1	**1765**	**0.05**
v20	2302	1.04	2302	7.75	**2302**	**0.97**
v30a	3582	5.43	3582	40.28	**3582**	**3.30**
v30b	2515	3.83	2515	23.52	**2515**	**3.07**
v30c	3236	7.83	3236	59.56	**3236**	**6.19**
v50a	4328	332.45	4450	467.66	**4328**	**238.40**
v50b	3872	243.76	3928	475.96	**3872**	**266.78**
v100a	6892	892.09	7200	2067.98	**6832**	**919.27**
v250a	15310	1118.33	16200	3807.45	**15273**	**1451.37**
v500a	28563	2345.79	32396	7658.23	**28460**	**1718.73**

Tabela 3 – Comparação com resultados encontrados na literatura

7. CONCLUSÃO

Este trabalho propõe duas abordagens para a resolução do PCVCP, ECS e *CS. Estas utilizam o conceito de algoritmos híbridos, combinando metaheurísticas com um processo de agrupamento de soluções em subespaços de busca (*clusters*), visando detectar regiões promissoras. Sempre que uma região for considerada promissora é realizada uma intensificação da busca nesta região, objetivando uma aplicação mais racional do método de busca local.

Este trabalho se justifica segundo alguns aspectos. Primeiramente, pelo fato do ECS

ser um método novo e que obteve sucesso ao ser aplicado a problemas de otimização combinatória.

Além disso, a substituição do componente AE por uma outra metaheurística, neste caso GRASP e VNS, contribui para a criação de uma nova abordagem, o *CS, visando investigar a geração contínua de soluções diretamente para o processo de agrupamento.

Os resultados obtidos mostram que estas abordagens são competitivas para resolução do PCVCP, conseguindo encontrar o ótimo global para instâncias com até 50 vértices em um tempo computacional razoável. Além disso, a abordagem *CS obteve resultados melhores que os apresentados na literatura para as instâncias maiores. Portanto, estes resultados validam a utilização destas abordagens para a resolução do PCVCP.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Balas, E. The Prize Collecting Travelling Salesman Problem. Networks, v. 19, p. 621–636, 1989.

[2] Balas, E.; Martin, G. ROLL-A-ROUND: Software Package for Scheduling the Rounds of a Rolling Mill. 104 p. © Copyright by Balas and Martin Associates, Pittsburgh, 1985.

[3] Chaves, A. A.; Biajoli, F. L.; Mine, O. M.; Souza, M. J. F. Modelagens Exata e Heurística para Resolução de uma Generalização do Problema do Caixeiro Viajante. In: Simpósio Brasileiro de Pesquisa Operacional (SBPO), 36, 2004, São João Del Rei. Anais... Rio de Janeiro: SOBRAPO, 2004. p. 1367–1378.

[4] Croes, G. A method for solving travelling salesman problems. Operations Research, v. 6, p. 791–812, 1958.

[5] Dell'Amico, M.; Maffioli, F.; Sciomanchen, A. A Lagrangian Heuristic for the Prize Collecting Travelling Salesman Problem. Anals of Operations Research, v. 81, p. 289–305, 1998.

[6] Feo, T.; Resende, M. Greedy randomized adaptive search procedures. Journal of Global Optimization, v. 6, p. 109–133, 1995.

[7] Glover, F. Tabu search and adaptive memory programing: Advances, applications and challenges. In: Barr, R.; Helgason, R.; Kennington, J. (eds.) Interfaces in Computer Science and Operations Research. Kluwer, 1996. p. 1–75.

[8] Goemans, M. X.; Williamson, D. P. A General Aproximation Technique for Constrained Forest Problems. SIAM Journal on Computing, v. 24, n. 2, p. 296–317, 1995.

[9] Gomes, L. M.; Diniz, V. B.; Martinhon, C. A. An Hybrid GRASP+VND Metaheuristic fo the Prize Collecting Traveling Salesman Problem. In: Simpósio Brasileiro de pesquisa Operacional (SBPO), 32, 2000, Viçosa. Anais... Rio de Janeiro: SOBRAPO, 2000. p. 1657–1665.

[10] ILOG CPLEX 7.5 Reference Manual. 7.5v. 610 p. © Copyright by ILOG, France, 2001.

[11] Lorena, L.; Furtado, J. Constructive genetic algorithm for clustering problems. Evolutionary Computation, v. 9, n. 3, p. 309–327, 2001.

[12] Melo, V. A.; Martinhon, C. A. Metaheurísticas Híbridas para o Problema do Caixeiro Viajante com Coleta de Prêmios. In: Simpósio Brasileiro de Pesquisa Operacional (SBPO), 36, 2004, São João Del Rei. Anais... Rio de Janeiro: SOBRAPO, 2004. p. 1295–1306.

[13] Mladenovic, N.; Hansen, P. Variable Neighborhood Search. Computers and Operations Research, v. 24, p. 1097–1100, 1997.

[14] Oliveira, A. C. M.; Lorena, L. A. N. Detecting promising areas by evolutionary clustering search. In: Bazzan, A. L. C.; Labidi, S. (eds.) Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series, 2004. p. 385–394.

[15] Oliveira, A. C. M.; Lorena, L. A. N. Population training heuristics. Lecture Notes in Computer Science, v. 3448, p. 166–176, 2005.

[16] Syswerda, G. Uniform crossover in genetic algorithms. In: International Conference on Genetic Algorithms(ICGA), 3., 1989, Virginia. Fairfax: Morgan Kaufmann... George Mason University: Proceedings, 1989. p. 2–9.

[17] Torres, R. D.; Brito, J. A. M. Problemas de Coleta de Prêmios Seletiva. In: Simpósio Brasileiro de Pesquisa Operacional (SBPO), 35, 2003, Natal. Anais... Rio de Janeiro: SOBRAPO, 2003. p. 1359–1371.