Heurísticas Híbridas com Detecção de Regiões Promissoras para o Problema do Caixeiro Viajante com Coleta de Prêmios

Antonio Augusto Chaves
LAC/INPE
chaves@lac.inpe.br

Luiz Antonio Nogueira Lorena
LAC/INPE
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 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 um solver comercial, que consegue resolver de forma exata apenas problemas de pequeno porte.*

Palavras-chave: *Problema do Caixeiro Viajante, Evolutionary Clustering Search, GRASP, VNS*

1. Introdução

Este trabalho apresenta novas heurísticas híbridas para resolver o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP), referido na literatura como *Prize Collecting Traveling Salesman Problem* (PCTSP). O PCVCP é uma generalização do Problema do Caixeiro Viajante (PCV), podendo 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 (vértices) seja simétrico, o número de soluções possíveis é $(n-1)!/2$, sendo classificado na literatura como NP-*difícil*.

O PCVCP foi proposto por Balas [2] como um modelo para a programação diária de uma fábrica de lâminas de aço. O autor apresentou algumas propriedades estruturais do problema e duas formulações matemáticas para este.

Dell'Amico *et al.* [5] exploraram uma relaxação Lagrangeana para o PCVCP, relaxando a restrição de prêmio mínimo. A heurística *Adding-Nodes* também é proposta para transformar a solução relaxada em uma solução viável.

Gomes *et al.* [8] e Melo e Martinhon [10] apresentam metaheurísticas híbridas para solucionar o PCVCP. O primeiro combinando GRASP e VND e o segundo combinando GRASP Progressivo e VNS como método de busca local.

Torres e Brito [15] apresentam uma nova formulação matemática para o PCVCP baseada na formulação apresentada em [2]. Nesta formulação é proposto um novo conjunto de restrições para evitar a formação de sub-rotas na solução.

Chaves *et al.* [3] exploraram duas abordagens. Uma modelagem matemática para o PCVCP resolvendo-o de forma ótima para problemas de pequenas dimensões, e uma modelagem heurística, combinando as metaheurísticas GRASP e VNS/VND.

Neste trabalho, o PCVCP é resolvido usando os conceitos de uma heurística híbrida recente, proposta por Oliveira e Lorena [12], 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 alternativa para esta heurística de busca através de agrupamentos é substituir o algoritmo evolutivo por

outras metaheurísticas, tais como GRASP [6] e VNS [11] dando origem à outra abordagem, chamada *CS.

Este trabalho também explora uma formulação matemática para o PCVCP, para validar os resultados computacionais obtidos pelo ECS e *CS. O *software* CPLEX [1] é utilizado para resolver esta formulação para instâncias de pequeno porte.

O restante do trabalho está organizado da seguinte forma. A seção 2 apresenta uma formulação matemática para o PCVCP. A seção 3 descreve as idéias básicas e os componentes conceituais do ECS, e as seções 4 e 5 apresentam o ECS e o *CS aplicados ao PCVCP. A seção 6 apresenta os resultados computacionais e na seção 7 são descritas algumas conclusões a respeito desse trabalho.

2. Formulação

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}. Supondo que o vértice 0, sem perda de generalidade, seja o depósito ou a cidade de origem do caixeiro, este vértice deve ter prêmio nulo ($p_0 = 0$) e penalidade infinita ($\gamma_0 = \infty$).

A formulação matemática apresentada neste trabalho é baseada nas formulações propostas por Balas [2] e Torres e Brito [15]. Considerando $x_{ij}(i, j \in V, i \neq j)$ sendo uma variável binária igual a 1 se a aresta (i, j) pertencer à solução, e 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. O PCVCP pode ser formulado como um problema de programação linear inteira como a seguir.

$$\min \sum_{i \in V} \sum_{j \in V} b_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{j \in V} x_{ji} = 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{i \in V} p_i x_{ii} \leq \left(\sum_{i \in V} p_i \right) - p_{\min} \tag{4}$$

$$\sum_{j \in V \backslash \{0\}} f_{0j} = 0 \tag{5}$$

$$\sum_{j \in V \backslash \{i\}} f_{ij} = \sum_{j \in V \backslash \{i\}} f_{ji} + p_i x_{ii} \qquad \forall i \in V \backslash \{0\} \tag{6}$$

$$\sum_{j \in V \backslash \{0\}} f_{j0} = \sum_{j \in V \backslash \{0\}} p_j x_{jj} \tag{7}$$

$$f_{ij} > x_{ij} - 1 \qquad \forall i \in V \backslash \{0\}, \ \forall j \in V, \ i \neq j \tag{8}$$

$$\left(\sum_{j \in V} p_j \right) x_{ij} \geq f_{ij} \qquad \forall i \in V \backslash \{0\}, \ \forall j \in V \tag{9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{10}$$

$$f_{ij} \geq 0 \qquad \forall i, j \in V \tag{11}$$

onde $b_{ij} = \begin{cases} c_{ij}, \text{ se } i \neq j \\ \gamma_i, \text{ se } i = j \end{cases}$

A função objetivo 1 procura minimizar o somatório dos custos de deslocamento e as penalidades pagas. As restrições 2 e 3 são conhecidas como restrições de atribuição. A restrição 4 assegura que a soma dos prêmios coletados será maior que o prêmio mínimo, p_{min}. As restrições 5 - 9 forçam que todos os vértices visitados sejam conectados ao vértice 0 (depósito), impedindo o aparecimento de sub-rotas na solução. As variáveis de fluxo f_{ij} são usadas para previnir estas sub-rotas. As restrições 10 asseguram a bivalência das variáveis x_{ij}. Por fim, as restrições 11 garantem que as variáveis f_{ij} sejam não-negativas.

Para resolução desta formulação, utilizou-se o software CPLEX versão 7.5 [1] buscando encontrar a solução ótima para o PCVCP. O CPLEX teve sucesso somente para instâncias com até 51 vértices.

3. Evolutionary Clustering Search

O Evolutionary Clustering Search (ECS) ou busca evolutiva através de agrupamentos é uma técnica evolutiva proposta por Oliveira e Lorena [12] que emprega conceitos de agrupamentos (*clustering*) de indivíduos para detectar regiões supostamente promissoras no espaço de busca.

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 G = *c*; *r*; β onde *c*, *r* e β são, respectivamente, o centro e o raio de uma região de busca, e uma estratégia de busca associada ao *cluster*.

O centro é um indivíduo que representa o *cluster*, identificando 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*. A estratégia de busca 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 conceitualmente independentes com diferentes atribuições:

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

A figura 1 ilustra os componentes, a população de indivíduos e os *clusters* que interagem entre si.

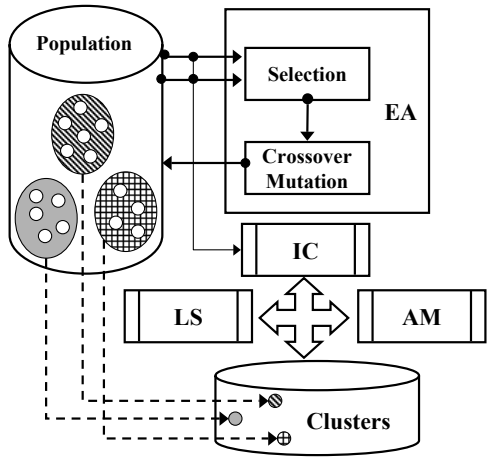

Figura 1. Diagrama conceitual do ECS.

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 informações (soluções representadas por indivíduos) que mantém no sistema apenas aquela que for relevante para o processo de intensificação de busca. 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 alta densidade 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. ECS aplicado ao PCVCP

Os detalhes da implementação do ECS para resolução do PCVCP serão descritos a seguir.

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 5 e 2 não são visitados.

1	3	-5	0	4	-2

Figura 2. Representaç^{a}o do indivíduo.

O componente AE do ECS, responsável por gerar soluções para alimentar o processo de agrupamento, será o Algoritmo de Treinamento Populacional (ATP) [13] empregando operadores genéticos conhecidos, como a seleção base-guia [9], o cruzamento BOX [14] e operador de mutação 2-*Opt*.

O ATP trabalha com uma população dinâmica de indivíduos, e, inicialmente a população é gerada aleatoriamente. Todos os indivíduos são avaliados por duas funções, f e g. A primeira avalia a qualidade do indivíduo e a segunda aplica uma heurística espcífica para o problema (chamada heurística de treinamento) para avaliar a vizinhança do indivíduo, sendo a melhor solução encontrada atribuída como valor de g.

Neste trabalho, a heurística de treinamento usada para determinar as características desejadas no treinamento ao longo do processo evolutivo do ATP é o método SeqDrop-SeqAdd [8], que consiste em realizar sucessivas tentativas de troca da solução corrente, através de remoções e inserções de vértices na rota.

A cada geração um número constante de indivíduos (NS) são selecionados. Para cada cruzamento BOX são selecionados dois indivíduos, resultando na geração de um único filho. Este filho pode, eventualmente, sofrer mutação.

A adaptação de um indivíduo pode ser medida através do *ranking* δ,

$$\delta = d.\left[G_{\max} - g\right] - |f - g| \tag{12}$$

que é composta por:

- um componente referente à adaptação do indivíduo em relação à heurística de treinamento $(f-g)$;
- um componente que privilegia a minimização da função g, através da maximização da distância 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 uma constante d que tem o papel de equilibrar os componentes da equação.

Os indivíduos melhores adaptados são aqueles com maiores valores de *ranking*.

A população é controlada dinamicamente por um limiar de rejeição, τ,

$$\tau_{i+1} = \tau_i + \xi.\left|P\right|.\frac{\left(\delta_1 - \delta_{|P|}\right)}{RG} \tag{13}$$

que é atualizado durante o processo evolutivo. A expressão 13 utiliza 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, e uma constante ξ que controla a velocidade do processo evolutivo. No fim de cada geração os indivíduos menos adaptados ($\delta \leq \tau$) são eliminados da população.

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 nova e esta é armazenada em 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* [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*.

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,

$$\lambda_t \geq PD.\frac{NS}{|C_t|} \tag{14}$$

onde, PD é a densidade desejável além da densidade normal, obtida se NS for igualmente dividido por todos *clusters* e $|C_t|$ é o número de *clusters* existentes na geração t. O centro deste *cluster* é explorado através do componente AO.

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 destes *clusters*. Outra função do AA é eliminar, a cada geração, os *clusters* com baixa densidade.

No componente AO foi implementado a heurística 2-Opt [4], objetivando melhorar a solução presente no centro de um *cluster* promissor. Este método examina todas as possíveis trocas entre pares de arestas não consecutivas, realizando a que fornecer o maior ganho em termos de melhora da função objetivo. Caso AO encontre uma solução que seja melhor que o centro do *cluster*, este é atualizado com a solução encontrada.

5. *CS aplicado ao PCVCP

Uma 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 [6] e VNS [11].

O GRASP é basicamente composto por duas fases: uma fase de construção, na qual uma solução viável é gerada e uma fase de busca local, na qual a solução construída é melhorada.

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 o prêmio coletado for menor que o prêmio mínimo.

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 seis 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 da rota;
- m_4: Inserir 1 vértice e retirar 1 vértice da rota;
- m_5: Retirar 3 vértices da rota;
- m_6: Retirar 1 vértice e trocar 4 vértices da rota;

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 definido anteriormente. Esse vizinho é então submetido a um procedimento de busca local. Se o ótimo local, s'', for melhor que a solução s corrente, a busca continua de s'' recomeçando da primeira estrutura de vizinhança. Caso contrário, a busca continua a partir da próxima vizinhança. Este procedimento é encerrado quando o tempo sem melhora na solução corrente for maior que 200 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) [11] e será utilizado para refinar o vizinho gerado no VNS.

O VND implementado faz uso de três procedimentos heurísticos de refinamento, sendo eles: (1) SeqDrop-SeqAdd [8]; (2) 2-Opt [4] e (3) AddDrop [8]. Cada iteração do método consiste na determinação de um ótimo local tendo por base o i-ésimo procedimento heurístico. Sempre que se obtém uma solução de melhora, retorna-se ao primeiro procedimento heurístico.

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.

6. Resultados computacionais

As abordagens ECS e *CS para o PCVCP foram codificadas em C++ e os experimentos foram conduzidos em uma máquina Pentium 4 3.02 GHz com 512 MB de mémoria. 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, Chaves *et al.* [3] geram aleatoriamente um conjunto de problemas testes para o PCVCP com $n \in \{11, 21, 31, 51, 101, 251, 501\}$ e os 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: $\gamma_i \in [1, 750]$. O valor do prêmio mínimo, p_{mim}, a ser coletado representa 75% do somatório de todos os prêmios associados aos vértices. Este conjunto de problemas testes estão disponíveis em *http//:www.lac.inpe.br/ chaves/intancias.html*.

Os valores para os parâmetros de desempenho das abordagens ECS e *CS foram ajustados através de várias execuções e também baseados em [12].

- 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 da função f do pior indivíduo existente na população inicial;
- incremento do limiar de rejeição e = 0,001;
- parâmetro $\alpha = 0{,}2$;

A formulação matemática apresentada na seção 2 foi resolvida utilizando o *software* CPLEX versão 7.5 [1], 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 31 vértices em um tempo computacional razoável. Entretanto, para as instâncias maiores, o problema já se torna inviável computacionalmente. Para as instâncias com 51 vértices, por exemplo, o CPLEX levou vários dias executando para poder encontrar o ótimo global. As instância com 101 vértices (*v100a* e *v100b*) foram executadas utilizando o resultados encontrados em [3] como valores *upper bound*, permitindo acelerar o processo de busca do CPLEX. No entanto, está foi executada por vários dias e não conseguiu fechar o *gap* entre *lower* e *upper bounds*.

A Tabela 1 também apresenta os melhores resultados computacionais encontrados pelas abordagens ECS e *CS. A melhor solução encontrada (MS) e o tempo de execução (TE), em segundos, necessário para encontrar a melhor solução são considerados para comparar as performances das abordagens. Os valores em negrito indicam qual abordagem encontrou os melhores valores de função objetivo e tempo de execução para cada instância. Note que, *CS obteve os melhores resultados em todas as instâncias testadas, encontrando o ótimo global para n com até 51 vértices e soluções melhores que as soluções obtidas pelo CPLEX para as instâncias *v100a* e *v100b*.

Tabela 1. Resultados computacionais.

		CPLEX			ECS		*CS	
Instância	n	MS	TE(s)	gap	MS	TE(s)	MS	TE(s)
v10	11	1765	0,06	0	1765	0,01	**1765**	**0,01**
v20	21	2302	3,73	0	2302	1,48	**2302**	**0,37**
v30a	31	3582	34,06	0	3582	19,40	**3582**	**2,48**
v30b	31	2515	45,59	0	2515	20,82	**2515**	**0,57**
v30c	31	3236	164,58	0	3236	16,03	**3236**	**0,71**
v50a	51	4328	433439,97	0	4368	146,53	**4328**	**65,21**
v50b	51	3872	241307,43	0	3928	78,31	**3872**	**17,10**
v100a	101	6879	153059,09	2,46	7200	489,34	**6832**	**560,72**
v100b	101	6784	246610,21	0,51	7420	679,51	**6782**	**681,74**
v250a	251	-	-	-	15935	2002,85	**15162**	**1066,70**
v250b	251	-	-	-	15615	2540,60	**14421**	**1056,09**
v500a	501	-	-	-	29018	2000,86	**28213**	**2258,37**
v500b	501	-	-	-	29391	5532,68	**28502**	**2228,75**

Na Tabela 2, o ECS e o *CS são comparados com uma outra metaheurística híbrida que também combina GRASP e VNS/VND, mas não usa busca por agrupamentos (Chaves *et al.* [3]). Novamente o *CS obteve os melhores resultados na resolução do PCVCP em todas as instâncias mostradas na Tabela 2.

Tabela 2. Confronto com outra abordagem.

Instância	Chaves et al.	TE(s)	ECS	TE(s)	*CS	TE(s)
v10	1765	0,1	1765	0,01	**1765**	**0,01**
v20	2302	1,04	2302	1,48	**2302**	**0,37**
v30a	3582	5,43	3582	19,40	**3582**	**2,48**
v30b	2515	3,83	2515	20,82	**2515**	**0,57**
v30c	3236	7,83	3236	16,03	**3236**	**0,71**
v50a	4328	132,45	4368	146,53	**4328**	**65,21**
v50b	3872	43,76	3928	78,31	**3872**	**17,10**
v100a	6892	692,09	7200	489,34	**6832**	**560,72**
v100b	6814	446,81	7420	679,51	**6782**	**681,74**
v250a	15310	918,33	15935	2002,85	**15162**	**1066,70**
v250b	14678	996,72	15615	2540,60	**14421**	**1056,09**
v500a	28563	2145,79	29018	2000,86	**28213**	**2258,37**
v500b	28524	2410,21	29391	5532,68	**28502**	**2228,75**

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.

Os resultados obtidos pelo ECS e *CS são competitivos para resolução do PCVCP, conseguindo encontrar o ótimo global para instâncias com até 51 vértices. Além disso, a abordagem *CS obteve resultados melhores que um trabalho encontrado na literatura que usa GRASP e VNS/VND, para o mesmo conjunto de problemas testes. Estes resultados validam a utilização destas abordagens para a resolução do PCVCP.

Referências

[1] *ILOG CPLEX 7.5 Reference Manual*. ©Copyright by ILOG, France, 2001.

[2] E. Balas. The prize collecting travelling salesman problem. *Networks*, 19:621–636, 1989.

[3] A. A. Chaves, F. L. Biajoli, O. M. Mine, and M. J. F. Souza. Modelagens exata e heurística para resolução de uma generalização do problema do caixeiro viajante. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 36:1367–1378, 2004.

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

[5] M. Dell'Amico, F. Maffioli, and A. Sciomanchen. A lagrangian heuristic for the prize collecting travelling salesman problem. *Operations Research*, 81:289–305, 1998.

[6] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[7] F. Glover. Tabu search and adaptive memory programing: Advances, applications and challenges. *Interfaces in Computer Science and Operations Research*, pages 1–75, 1996.

[8] L. M. Gomes, V. B. Diniz, and C. A. Martinhon. An hybrid grasp+vnd metaheuristic fo the prize collecting traveling salesman problem. *Simpósio Brasileiro de pesquisa Operacional (SBPO)*, 32:1657–1665, 2000.

[9] L. Lorena and J. Furtado. Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 9(3):309–327, 2001.

[10] V. A. Melo and C. A. Martinhon. Metaheurísticas híbridas para o problema do caixeiro viajante com coleta de prêmios. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 36:1295–1306, 2004.

[11] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[12] A. C. M. Oliveira and L. A. N. Lorena. Detecting promising areas by evolutionary clustering search. *Advances in Artificial Intelligence. Springer Lecture Notes in Artificial Intelligence Series*, pages 385–394, 2004.

[13] A. C. M. Oliveira and L. A. N. Lorena. Population training heuristics. *Lecture Notes in Computer Science*, 3448:166–176, 2005.

[14] G. Syswerda. Uniform crossover in genetic algorithms. *International Conference on Genetic Algorithms(ICGA)*, 3:2–9, 1989.

[15] R. D. Torres and J. A. M. Brito. Problemas de coleta de prêmios seletiva. *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 35:1359–1371, 2003.