Anais do IX Congresso Brasileiro de Redes Neurais / Inteligência Computacional (IX CBRN)
Ouro Preto 25-28 de Outubro de 2009

METAHEURÍSTICAS COM MEMÓRIA ADAPTATIVA PARA O PROBLEMA DE RECOBRIMENTO DE ROTAS

Luciene C. S. Motta*, Luiz S. Ochi*

**Instituto de Computação - Universidade Federal Fluminense*
Rua Passo da Pátria 156 - Bloco E - 3º andar - São Domingos. CEP: 24210-240
Niterói, RJ, Brasil

Emails: lmotta@ic.uff.br, satoru@ic.uff.br

Abstract— Given an undirected graph $G = (V \cup W, E)$, where $V \cup W$ is the vertex set and E is the edge set, the Covering Tour Problem (CTP) consists of determining a minimum length cycle over a subset of V which contains all vertices of $T \subseteq V$, and every vertex of W is covered by the tour. As a generalization of the Travelling Salesman Problem (TSP), the CTP is a $\mathcal{NP}$-Hard problem which, as a rule, restricts the optimal solutions obtained by using exact methods to small size instances. This paper proposes two new heuristic algorithms for the CTP, one based on the GRASP metaheuristic and the other based on the ILS metaheuristic. Computational experiments are reported, comparing the behavior of those proposed algorithms.

Keywords— Covering tour problem, Computational Inteligence, Metaheuristics

Resumo— Dado um grafo não direcionado $G = (V \cup W, E)$, onde $V \cup W$ é o conjunto de vértices e E é o conjunto de arestas, o Problema de Recobrimento de Rotas (PRR) consiste em determinar uma rota de comprimento mínimo sobre V, contendo todos os vértices de $T \subseteq V$ e com cada vértice de W coberto por esta rota. Sendo uma generalização do Problema do Caixeiro Viajante (PCV), o PRR é considerado um problema $\mathcal{NP}$-Difícil o que, em via de regra, restringe a obtenção de soluções ótimas através de métodos exatos às instâncias do problema de pequenas dimensões. Neste trabalho são propostos dois novos algoritmos heurísticos para o PRR, sendo um baseado na metaheurística GRASP e o outro baseado na metaheurística ILS. Experimentos computacionais são reportados, comparando comportamento dos algoritmos propostos.

Palavras-chave— Problema de Recobrimento de Rotas, Inteligência Computacional, Metaheurísticas

1 Introdução

O Problema de Recobrimento de Rotas (PRR), conhecido na literatura por Covering Tour Problem (CTP), é definido em um grafo completo e não direcionado $G = (V \cup W, E)$, onde $V \cup W = \{1, ..., n\}$ é o conjunto de vértices e $E = \{(i, j) \mid i, j \in V \cup W, i < j\}$ é o conjunto de arestas. O vértice $s = 1$ é considerado a origem, V é o conjunto dos vértices que podem ser visitados, $T \subseteq V$ é o conjunto dos vértices que devem ser visitados ($s \in T$), e W é o conjunto dos vértices que devem ser cobertos. Uma matriz de distância $C = (c_{ij})$ satisfazendo a desigualdade triangular é definida sobre E. O PRR consiste em determinar uma rota de comprimento mínimo sobre V, contendo todos os vértices de T e tendo cada vértice de W coberto por esta rota, isto é, cada vértice de W deve estar, no máximo, à uma distância $d \geq 0$ de um vértice da rota.

Aplicações para o PRR podem ser encontradas em diversas áreas, tais como logística, transporte, roteamento, rede de telecomunicações, entre outras. Uma aplicação é o roteamento de equipes para a entrega de medicamentos em países subdesenvolvidos, onde medicamentos somente podem ser entregues a um subconjunto de vilarejos e todos os usuários dos demais vilarejos devem ser capazes de atingir pelo menos um vilarejo da rota, visando suprir a necessidade de serviços e medicamentos.

Apesar de aplicável em diversas situações reais, o PRR ainda é pouco explorado na literatura (vide Seção 2), principalmente no que diz respeito às propostas heurísticas para resolver o problema de forma aproximada. O uso de heurísticas se justifica pelo fato do PRR ser classificado como $\mathcal{NP}$-Difícil, uma vez que este se reduz ao clássico PCV quando $d = 0$ e $W = V$. Para instâncias de dimensões elevadas, utiliza-se tradicionalmente métodos heurísticos, especialmente aqueles que possuem dispositivos que possibilitam escapar de ótimos locais ainda distantes de ótimos globais, o que justifica a proposta deste trabalho. Neste contexto, apresenta-se neste trabalho duas heurísticas para a solução do PRR, sendo uma baseada na metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*), Resende and Feo (1995) e outra baseada na metaheurística ILS (*Iterated Local Search*), Lourenço et al. (2002).

Este artigo está organizado da seguinte forma: na Seção 2 é apresentado uma breve descrição dos trabalhos da literatura relacionados ao PRR. A Seção 3 apresenta o detalhamento dos algoritmos propostos e a Seção 4 relata os experimentos e análises computacionais realizadas. A Seção 5 apresenta as conclusões e as referências bibliográficas.

2 Revisão da literatura

O PRR foi introduzido por Current (1981) na sua tese de doutorado. Anos mais tarde, Current and Rolland (1994) propuseram uma heurística para duas versões distintas do PRR: uma que visava

a minimização apenas do comprimento da rota e a outra que maximizava o número de vértices coberto pela rota gerada na primeira versão. Gendreau et al. (1997) apresentaram um modelo de Programação Linear Inteira (PLI), uma heurística, um algoritmo baseado na técnica *branch-and-cut* e um conjunto de quatro regras para reduzir os conjuntos W e V. Maniezzo et al. (1999) desenvolveram três algoritmos daseados na metaheurística *Scatter Search*, um novo modelo de PLI e uma nova regra para reduzir o conjunto W. Motta et al. (2001) apresentou uma versão generalizada do PRR, chamada de o Problema de Recobrimento de Rotas Generalizado (PRRG), onde é proposto um conjunto de regras de redução, algoritmos heurísticos e uma formulação baseada em PLI para o PRRG. Brito (2005) desenvolveu uma heurística lagrangeana, uma nova regra de redução e algoritmos heurísticos para o PRR. Uma abordagem multi-objetivo combinada com um algoritmo do tipo *branch-and-cut* foi proposta por Jozefowiez et al. (2007) para uma outra generalização do PRR, chamada originalmente na literatura de bi-Object Covering Tour Problem.

É desconhecida a existência de alguma outra abordagem, exata ou heurística, proposta na literatura para o PRR.

3 Heurísticas propostas para o PRR

Dentre as metaheurísticas existentes, o GRASP tem se mostrado uma das mais competitivas. No entanto, para obter um bom desempenho, algumas propostas baseadas nesta técnica tem feito o uso de memória adaptativa, assim como apresentaram Silva et al. (2007), Gonçalves et al. (2005) e Bastos et al. (2005).O algoritmo GRASP proposto neste trabalho para o PRR baseia-se em uma versão adaptativa da metaheurística GRASP, conhecida na literatura como GRASP Reativo (Prais and Ribeiro (2000)), que auto ajusta o valor do parâmetro α selecionando-o aleatoriamente em um conjunto discreto de valores $A = \{\alpha_0, \alpha_1, \alpha_2, ..., \alpha_m\}$, onde cada valor α_i é associado a uma probabilidade p_i de ser selecionado ($p_i = \sum_{i=1...m} p_i = 1/m$). Em linhas gerais, pode-se dizer que um GRASP Reativo incorpora mecanismos de memória na fase de construção, fato que na prática, faz com que se um determinado valor de α for mais adaptado à uma determinada instância, então este valor acabe por possuir uma probabilidade p_i de ser selecionado mais alta do que os demais valores de α. O Algoritmo 1 apresenta o procedimento proposto para o PRR baseado no GRASP Reativo.

A atualização das probabilidades no *GRASPadaptativoPRR* é feita de acordo com a regra denominada *qualificação absoluta*, definida por Prais and Ribeiro (2000) (linhas 12 a 17 do Alg. 1).

A heurística proposta para a fase de construção

Algoritmo 1: GRASPadaptativoPRR

```
 1: inicializar(melhorRota, A, Q, P, Qtde);
 2: fazer P[j] = 1/m, ∀ j=0,...,m;
 3: for i = 1,...,maxIteration do
 4:    selecionar α=α_i em A considerando P;
 5:    Qtde[i] = Qtde[i]+1;
 6:    rotaAtual = IMD-MaxMin(α_i, semente);
 7:    rotaAtual = BuscaVND(rotaAtual);
 8:    if f(rotaAtual) < f(melhorRota) then
 9:       melhorRota = rotaAtual;
10:    end if
11:    A[i] = A[i] + rotaAtual;
12:    if ((k mod blockIterations) == 0) then
13:       fazer A[j] = A[j] / Qtde[j], ∀ j=0,...,m;
14:       fazer Q[j] = f(melhorRota) / A[j], ∀ j=0,...,m;
15:       somaGer3alQ = soma de todos os valores em Q;
16:       fazer P[j] = Q[j] / somaGeralQ, ∀ j=0,...,m;
17:    end if
18: end for
19: return melhorRota;
```

do *GRASPadaptativoPRR* (linha 6) é uma versão randomizada da heurística construtiva Inserção Mais Distante proposta originalmente para o PCV por Rosenkrantz et al. (1977). O Algoritmo 2 apresenta o pseudo-código da heurística construtiva *IMD-MaxMin* proposta para o PRR.

Algoritmo 2: IMD-MaxMin

```
 1: rota=∅;
 2: if (T ≠ ∅) then
 3:    rota = iniciaRotaPorVerticeDeT();
 4: else
 5:    rota = iniciaRotaPorVerticeQueMaisCobreWs();
 6: end if
 7: LDispT = inicializaListaDisponiveisEmT();
 8: while (LdispT ≠ ∅) do
 9:    Selecionar k ∈ LdispT se o valor máximo de c_kl for
       mínimo, ∀ l ∈ rota;
10:    Encontrar (i,j) ∈ rota onde c_ik+c_kj-c_ij é mínimo;
11:    rota = insereVertice(k, rota, i, j);
12:    LdispT = LdispT - {k};
13: end while
14: LDisp = inicializaListaDisponiveis();
15: while (ehInviavel(rota)) do
16:    Selecionar z ∈ LdispT se o valor máximo de c_zl for
       mínimo, ∀ l ∈ rota;
17:    Encontrar (i,j) ∈ rota onde c_iz+c_zj-c_ij é mínimo;
18:    rota = insereVertice(z, rota, i, j);
19:    Ldisp = Ldisp - {z};
20: end while
21: removeVerticesSobressalentes(rota);
22: return rota;
```

A execução da heurística *IMD-MaxMin* parte da escolha aleatória de um vértice $v \in V$ para inicializar a rota. Quando $T \neq \emptyset$, a rota é inicializada por um vértice $v \in T$ escolhido aleatoriamente (linha 3). Quando $T = \emptyset$, a rota é inicializada pelo vértice optativo $v \in V \setminus T$ que cobrir o maior número de vértices de W (linha 5). Partindo da rota inicializada, escolhe-se a cada iteração o vértice obrigatório $k \in DispT$ cujo valor máximo de c_{kl}, considerando todo vértice l da rota, seja mínimo (linha 9). O vértice k é inserido na rota atual entre os vértices (i, j) que produzirem o menor acréscimo no custo desta rota (linhas 10 e 11). Quando não existirem vértices de T disponíveis para inserção,

isto é, quanto $LDispT = \emptyset$, verifica-se se a rota atual ainda é inviável (linha 15). Caso a inviabilidade seja comprovada, realiza-se um novo processo de escolha e inserção de vértices, porém desta vez considerando os vértices de $V \setminus T$ disponíveis (linhas 16 a 19). O processo encerra quando uma solução viável para o PRR é alcançada.

Antes de retornar a solução construída, é aplicado um procedimento de remoções sucessivas dos vértices $v \in V \setminus T$ que se tornaram redundantes durante o processo de construção da rota. Este processo inicia escolhendo aleatoriamente um vértice $t \in T$ e a partir deste t, percorre toda a rota tentando remover os vértices $v \in V \setminus T$ que se tornaram desnecessários à satisfação da restrição de cobertura dos vértices de W.

A heurística proposta para a fase de busca local do *GRASPadaptativoPRR* (linha 7 do Algoritmo 1) é baseada na metaheurística VND (*Variable Neighborhood Descent*), Hansen et al. (2001). O Algoritmo 3 apresenta o pseudo-código da heurística de busca local *BuscaVND-PRR* proposta para o PRR.

Algoritmo 3: BuscaVND-PRR(rotaConstrutivo)

```
 1: Seja k_max o número de estruturas de vizinhanças;
 2: rota = rotaConstrutivo;
 3: k = 1;
 4: while (k ≤ k_max) do
 5:    rotaAtual = realizaBuscaNaVizinhaca(k, rota);
 6:    if (f(rotaAtual) < f(rota)) then
 7:       rota = rotaAtual;
 8:       k = 1;
 9:    else
10:       k = k + 1;
11:    end if
12: end while
13: rota = 2Opt(rota);
14: return rota;
```

A seguinte estrutura de vizinhança é considerada na heurística *BuscaVND-PRR*:
1DropAdd ($\mathcal{N}_1$) - Um vértice $i \in V$ é removido da solução. Se esta remoção implicar na inviabilidade da solução, um processo de inserções sucessivas é iniciado considerando os seguinte casos:
(a) Se $i \in T$, então i é reinserido na solução entre os vértices (k,h) que produzirem o menor acréscimo no custo total da rota;
(b) Se $i \in V \setminus T$, um vértice $j \in V \setminus T$ é aleatoriamente selecionado em $S_l - \{i\}$, onde $l \in W$ é o vértice que deixou de ser coberto após a remoção do vértice i e $S_l = \{u \in V \mid c_{lu} <= d\}$. Se $S_l - \{i\} = \emptyset$ o vértice i é reinserido na solução entre os vértices (k,h) que produzirem o menor acréscimo no custo total da rota. Este processo é repetido até que a viabilidade da solução seja reestabelecida.
2DropAdd ($\mathcal{N}_2$) - Esta vizinhança é similar à ($\mathcal{N}_1$), exceto que 2 vértices são removidos antes de eventualmente iniciar o processo de inserções.
3DropAdd ($\mathcal{N}_3$) - Neste caso, 3 vértices são removidos antes de eventualmente iniciar o processo de inserções.
4DropAdd ($\mathcal{N}_3$) - Neta vizinhança, 4 vértices são removidos antes de eventualmente iniciar o processo de inserções.

A estratégia *first-improving* é adotada na exploração da k-ésima vizinhança (linha 5). Um procedimento do tipo 2-*optimal* é aplicado no final do algoritmo *BuscaVND-PRR* (linha 13) com o objetivo de tentar melhorar a qualidade da solução obtida pela busca local, alterando a ordem de visita de alguns vértices da rota.

A segunda heurística proposta neste trabalho para o PRR baseia-se na metaheurística *Iterated Local Search* (ILS), Lourenço et al. (2002), que parte do pressuposto que os ótimos locais de um problema de otimização podem ser gerados a partir de perturbações na solução ótima local corrente. O Algoritmo 4 apresenta o pseudo-código da heurística *ILS-PRR* proposta para o PRR.

Algoritmo 4: ILS-PRR

```
 1: inicialRota = construcaoGulosaAleatorizada();
 2: rota = BuscaVNS-PRR(inicialRota);
 3: rota_1 = ∅;
 4: rota_2 = ∅;
 5: for i = 1,...,maxIteration do
 6:    rota_1 = pertubaRota(rota);
 7:    rota_2 = BuscaVND-PRR(rota_1);
 8:    if (f(rota_1) < f(rota_2)) then
 9:       rota = rota_1;
10:    else
11:       rota = rota_2;
12:    end if
13: end for
14: return rota;
```

Um procedimento guloso aleatorizado é usado na construção da solução inicial da heurística *ILS-PRR* (linha 1), que inicia a construção da solução pela escolha aleatória de γ vértices de V, inserindo-os na rota na seqüência em que são escolhidos de forma a produzir o menor acréscimo no custo total. Tendo a rota inicializada com os γ vértices, o procedimento inicia a escolha dos próximos vértices de acordo com a seguinte função gulosa $g : V \rightarrow \aleph$:

$$g(v) = \begin{cases} |V \cup W| - 2 \times (\text{(número de vértices de W descobertos que v cobre)} + 1), \text{ se } v \in T; \\ |V \cup W| - 2 \times \text{(número de vértices de W descobertos que v cobre)}, \text{ caso contrário.} \end{cases}$$

Os vértices escolhidos de acordo com a função g são inseridos na rota de forma a produzir o menor acréscimo no seu custo total. A seleção e inserção dos vértices encerra quando uma solução viável para o PRR for alcançada. Um procedimento de busca local baseado na metaheurística VNS (*Variable Neighborhood Search*) (Hansen et al. (2001)) é aplicado na solução inical (linha 2). A *BuscaVNS-*

PRR utiliza a mesma estrutura de vizinhança definida para o procedimento *BuscaVND-PRR*.

A estratégia adotada na heurística *ILS-PRR* para perturbar uma solução (linha 6) é escolhida aleatoriamente no seguinte conjunto de movimentos: inserir e remover k vértices aleatoriamente (k variando de 1 a 5) ou inserir e remover k arestas aleatoriamente (k variando de 2 a 3). O procedimento de busca local utilizado após cada perturbação é o mesmo descrito no Algoritmo 3.

4 Resultados Computacionais

Como não existem instâncias para o PRR disponíveis na literatura, foram geradas neste trabalho 68 instâncias aleatórias da seguinte forma: o conjunto de vértices foi obtido gerando-se $|V| + |W|$ pontos no retângulo de dimensão $[0, 640] \times [0, 480]$, de acordo com uma distribuição uniforme. Os conjuntos T e W foram definidos tomando os primeiros $|T|$ e $|W|$ pontos respectivamente, e $V \setminus T$ foi definido com os demais pontos remanescentes. A matriz de distâncias foi computada calculando-se a distância Euclideana entre estes pontos. A distância de cobertura d foi determinada como sendo $d = max\{min\{c_{ij} \,|\, i \in W\, e\, j \in V, \forall i \neq j\}\}$. Desta forma, garante-se que cada vértice de W seja coberto por pelo menos um vértice de V.

Todos os algoritmos apresentados foram codificados em C++ (compilador g++, versão 4.1.2) e executados em uma máquina com processador Intel (R) Xeon(TM), CPU 3.40GHz, 8GB de RAM, utilizando como sistema operacional o SUSE Linux Enterprize Server 10 SP2 (X86-64), com kernel 2.6.16.60-0.21-smp.

Os testes foram executados para três grupos de instâncias: dois grupos gerados aleatoriamente como descrito no início desta seção e um grupo com instâncias do PCV adaptadas para o PRR. O primeiro grupo, aqui denominado GIII, era composto de instâncias de pequenas dimensões, sendo n={15, 20, 25, 30, 35, 40, 50, 60}. Para cada n, os conjuntos T, W e $V \setminus T$ foram criados considerando as seguintes combinações: $(|T|, |W|) = (\lfloor 0, 2n \rfloor, \lfloor 0, 2n \rfloor)$, $(\lfloor 0, 2n \rfloor, \lfloor 0, 6n \rfloor)$, $(\lfloor 0, 3n \rfloor, \lfloor 0, 4n \rfloor)$, $(\lfloor 0, 33n \rfloor, \lfloor 0, 33n \rfloor)$, $(\lfloor 0, 5n \rfloor, \lfloor 0, 5n \rfloor)$, $(\lfloor 0, 6n \rfloor, \lfloor 0, 2n \rfloor)$, e $|V \setminus T| = n - (|T| + |W|)$. O segundo grupo, aqui denominado GIV, era composto de instâncias de maiores dimensões, sendo n={100, 200, 300, 500, 700}, considerando a seguinte combinação: $(|T|, |W|) = (\lfloor 0, 2n \rfloor, \lfloor 0, 6n \rfloor)$, $(\lfloor 0, 33n \rfloor, \lfloor 0, 33n \rfloor)$, $(\lfloor 0, 5n \rfloor, \lfloor 0, 5n \rfloor)$, $(\lfloor 0, 6n \rfloor, \lfloor 0, 2n \rfloor)$, e $|V \setminus T| = n - (|T| + |W|)$. O terceiro grupo, aqui denominado GV, é composto da adaptação de 10 instâncias disponíveis no repositório TSPLIB[1]. Desta forma, foram selecionadas instâncias com até 1002 vértices, que foram adaptadas para o PRR da seguinte forma: sendo n o número total de vértices do grafo, os $\lfloor 0, 3n \rfloor$ primeiros vértices foram atribuídos ao conjunto T, os $\lfloor 0, 4n \rfloor$ seguintes foram atribuídos ao conjunto W e os vértices remanescentes atribuídos ao conjunto $V \setminus T$. A distância de cobertura foi calculada da mesma forma que nas instâncias dos grupos GIII e GIV. Todas as 78 instâncias utilizadas neste trabalho estão disponíveis em `http://labic.ic.uff.br/AutoIndex/`

Baseado na formulação matemática proposta por Motta et al. (2001) para o PRRG, foi implementado um modelo para o PRR. Utilizando o software ILOG CPLEX 11.2 e os dois grupos de instâncias gerados aleatoriamente, foram obtidos todos os ótimos para as instâncias do grupo GIII e cinco ótimos para as instâncias do GIV. Nenhuma das instâncias do grupo GV foi testada com a formulação adaptada.

Em todos os experimentos realizados com as heurísticas propostas na Seção 3, utilizou-se um número máximo de iterações igual a 150, uma semente inicial para a geração de números pseudo-aleatórios igual a 1305 e foram realizadas cinco execuções para cada heurística. A atualização das probabilidades no *GRASPadaptativoPRR* ocorreu a cada 30 iterações.

Tanto a heurística *GRASPadaptativoPRR* quanto a *ILS-PRR* alcançaram as soluções ótimas conhecidas das 48 instâncias do grupo GIII. Entretando, a heurística *GRASPadaptativoPRR* se mostrou aproximadamente 7% mais rápida que a *ILS-PRR*. A Tabela 1 apresenta um comparativo dos resultados obtidos pelas heurísticas *GRASPadaptativoPRR* (2^a coluna) e *ILS-PRR* (3^a coluna) para um subconjunto das 30 instâncias dos grupos GIV e GV (1^a coluna). Para estes grupos, somente são conhecidas as soluções ótimas das instâncias ctp100_20_60, ctp100_33_33, ctp100_50_50, ctp100_60_20 e ctp200_20_60, das quais ambas heurísticas propostas neste trabalho conseguiram alcançar o ótimo, sendo o melhor desempenho de tempo observado para a heurística *GRASPadaptativoPRR*.

Os valores apresentados na Tabela 1 referem-se a diferença média percentual das soluções obtidas por cada heurística, sendo este valor calculado como $((avgSol - bestSol)/bestSol) \times 100$, onde *avgSol* é a média das cinco execuções da heurística em questão e *bestSol* é a melhor solução encontrada entre as execuções das duas heurísticas. Como pode-se observar, a heurística *ILS-PRR* superou a heurística *GRASPadaptativoPRR* em 16 instâncias e empatou em 9, sendo a média da diferença percentual respectivamente igual 0,13 e 0,41. Para as 16 instâncias onde não ocorreu empate, a melhor solução (*bestSol*) para 12 delas foi obtida pela heurís-

[1] www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95

Tabela 1: Resultados computacionais

Instâncias	GRASPadaptativoPRR	ILS-PRR
ctp200_33_33	0.01	0.00
ctp200_50_50	0.02	0.00
ctp200_60_20	0.00	0.00
ctp300_20_60	0.12	0.03
ctp300_33_33	0.05	0.00
ctp300_50_50	0.28	0.01
ctp300_60_20	0.00	0.00
ctp500_20_60	1.11	0.23
ctp500_33_33	0.38	0.02
ctp500_50_50	0.27	0.00
ctp500_60_20	0.81	0.15
ctp700_20_60	1.43	0.63
ctp700_33_33	1.25	0.51
ctp700_50_50	0.32	0.02
ctp700_60_20	0.68	0.04
eil51	0.00	0.00
eil76	0.00	0.00
pr76	0.00	0.00
kroA100	0.00	0.00
kroC100	0.00	0.00
kroD100	0.00	0.00
eil101	0.00	0.00
lin105	0.01	0.00
a280	1.29	0.73
pr1002	4.40	1.59

tica *ILS-PRR*. Em relação ao tempo computacional das duas heurística, pode-se dizer que o tempo consumido pelas iterações da *ILS-PRR* foi em média 9% superior quando comparado ao *GRASPadaptativoPRR*.

5 Conclusões

Neste trabalho foram apresentadas duas novas heurísticas para o PRR, sendo uma baseada no GRASP Reativo e a outra baseada na metaheurística ILS.

Experimentos computacionais empíricos avaliaram o desempenho de cada heurística proposta para três grupos de instâncias, sendo dois grupos gerados aleatoriamente, do qual se conhecia 53 soluções ótimas e o outro grupo de instâncias gerado a partir da adaptação de instâncias do TSPLIB.

Constatou-se que, para as intâncias consideradas, a abordagem utilizada pela heurística *ILS-PRR*, conseguiu atingir os melhores resultados, apesar de exigir um tempo computacional um pouco maior.

Agradecimentos

Os autores agradecem ao apoio parcial dos seguintes órgãos de fomento: CNPq (CT-INFO, UNIVERSAL & Bolsa de Produtividade), CAPES (Pró-Engenharia & PROCAD-NF) e FAPERJ (PENSA RIO, Prioridade Rio & Cientista do Estado).

Referências

Bastos, L. O., Ochi, L. S. and Macambira, E. M. (2005). Grasp with path relinking for the sonet ring assignment problem, *5th Int. Conference on Hybrid Intelligence System (HIS2005) in cooperation with IEEE Comp. Intelligence Society*, Vol. 1, pp. 239–244.

Brito, L. R. (2005). *Novas propostas para o Problema de Recobrimento de Rotas*, PhD thesis, Universidade Federal do Rio de Janeiro, COPPE, Rio de Janeiro, Brasil.

Current, J. (1981). *Multiobjective Design of Transportation Networks*, PhD thesis, Department of Geography and Environmental Engineering, The Johns Hopkins University.

Current, J. and Rolland, E. (1994). Efficient algorithms for solving the shortest covering path problem, *Transp. Science* **28**: 317–327.

Gendreau, M., Laporte, G. and Semet, F. (1997). The covering tour problem, *Operations Research* **45**: 568–576.

Gonçalves, L. B., Martins, S. L. and Ochi, L. S. (2005). A grasp with adaptive memory for a period vehicle routing problem, *IEEE International Conference on Computational Intelligence for Modelling Control and Automation*, Vol. 1, pp. 721–727.

Hansen, P., Mladenovic, N. and Perez-Britos, D. (2001). Variable neighborhood decomposition search, *J. of Heuristics* **7**(4): 335–350.

Jozefowiez, N., Semet, F. and Talbi, E.-G. (2007). The bi-objective covering tour problem, *Comput. Oper. Res.* **34**(7): 1929–1942.

Lourenço, H. R., Martin, O. and Stützle, T. (2002). Iterated local search, *Handbook of Metaheuristics*, Vol. 57 of *Series in Operations Research & Management Science*, pp. 321–353.

Maniezzo, V., Baldacci, R., Boschetti, M. and Zamboni, M. (1999). Scatter search methods for the covering tour problem, *Scienze dell'Informazione*, University of Bologna.

Motta, L. C., Ochi, L. S. and Martinhon (2001). Reduction rules for the covering tour problem, *Eletronic Notes in Discrete Applied Mathematics* (7): 168–171.

Prais, M. and Ribeiro, C. (2000). Parameter variation in GRASP procedures, *Investigación Operativa* **9**: 1–20.

Resende, M. G. C. and Feo, T. A. (1995). Greedy randomized adaptative search procedures, *J. of Global Optimization*, pp. 1–27.

Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M. (1977). An analysis of several heuristics for the traveling salesman problem, *SIAM Journal on Computing* **6**(3): 563–581.

Silva, G. C., Andrade, M. R. Q., Ochi, L. S., Martins, S. L. and Plastino, A. (2007). New heuristics for the maximum diversity problem, *Journal of Heuristics*, Vol. 13, SPRINGER, pp. 315–336.