Trabalho apresentado no CNMAC, Gramado - RS, 2016.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Um Algoritmo Evolutivo para o Problema do Caixeiro Alugador

André Renato Villela da Silva[1]
Departamento de Computação, ICT, UFF, Rio das Ostras, RJ
Luiz Satoru Ochi[2]
Departamento de Ciência da Computação, IC, UFF, Niterói, RJ

Resumo.

Este artigo aborda uma variante do Problema do Caixeiro Viajante chamado Problema do Caixeiro Alugador, onde um cliente deseja viajar entre as cidades usando um veículo alugado. Basicamente, o cliente tem duas opções quando chega em uma cidade: devolver o veículo para a cidade de onde alugou e alugar outro para continuar a viagem ou manter o mesmo veículo. Cada vez que um carro é entregue em uma cidade, uma taxa de retorno deve ser paga. O custo de deslocamento entre qualquer par de cidades também depende do carro escolhido. O objetivo é estabelecer um ciclo hamiltoniano minimizando os custos de deslocamento e as taxas de retorno. Um novo algoritmo evolutivo é proposto para este problema e comparado com a melhor técnica conhecida da literatura.

Palavras-chave. Meta-heurísticas, Algoritmos Evolutivos, Problema do Caixeiro Alugador

1 Introdução

Frotas de veículos possuem características, comportamento e objetivos muito distintos, de acordo com segmento econômico no qual são empregadas. Algumas são muito conhecidas como sistemas de ônibus [5]. Outros tipos de frotas necessitam de cuidados especiais, seja no modo de operação [1], seja no tempo de resposta [4].

A locação de veículos é um dos segmentos mais interessantes devido ao seu crescimento significativo nos últimos anos [7]. Junto com o aumento dos lucros, há também uma crescente diversificação nos serviços prestados, em especial aqueles voltados para o setor de turismo. No entanto, poucos modelos abordam o ponto de vista do cliente (locatário). No setor de turismo, é comum um cliente desejar alugar um veículo para percorrer lugares específicos minimizando as despesas de viagem, que podem variar de acordo com o tipo de carro e taxas específicas.

Em [3] é apresentada uma variante do clássico problema do caixeiro viajante que modela aspectos centrais da locação de veículos pelo ponto de vista do cliente. Essa variante

[1]avillela@ic.uff.br
[2]satoru@ic.uff.br

DOI: 10.5540/03.2017.005.01.0460

é chamada Problema do Caixeiro Alugador. Neste problema, um cliente deseja visitar algumas cidades utilizando um veículo. Há um conjunto de tipos de carros disponíveis e o locatário pode escolher qual carro será usado para a primeira viagem. Ao chegar a outra cidade, o locatário pode continuar com o mesmo carro ou pode entregar o veículo, se for possível, e alugar outro para continuar a viagem. Ao entregar um carro em outra cidade, o cliente deve pagar uma taxa de retorno de acordo com o tipo de carro e a distância entre as cidades. Os custos de deslocamento entre duas cidades quaisquer também depende do tipo de carro escolhido. Em outras palavras, o cliente pode escolher diferentes tipos de carro para percorrer diferentes partes do percurso. No final do percurso, o cliente gostaria de ter o mínimo de despesas, incluindo custos de deslocamento e as taxas de retorno.

O restante deste artigo está organizado da seguinte forma. A Seção 2 define o problema e faz uma breve revisão da literatura. Um novo método heurístico proposto é descrito na Seção 3. Os resultados computacionais são apresentados na Seção 4. A Seção 5 traz as principais conclusões deste trabalho.

2 Definição do problema

O Problema do Caixeiro Alugador (CaRS) pode ser descrito da seguinte forma. Seja C um conjunto de diferentes tipos de carros e $G = (V, A)$ um grafo direcionado, onde V é o conjunto de cidades ($|V| = n$) e A o conjunto de arcos (estradas) entre as cidades de V. A cada carro c e arco $(i, j) \in A$, $i \neq j$, existe um custo de deslocamento D^c_{ij}, que indica o custo de viajar de i para j usando o carro c. Além disso, sempre que um carro c for alugado em uma cidade i e entregue em uma cidade j, um taxa de retorno F^c_{ij} deve ser paga pelo cliente. O objetivo é elaborar um ciclo Hamiltoniano de custo total mínimo, incluindo os custos de deslocamento e as taxas de retorno. O cliente deve começar e terminar o trajeto sempre pela cidade 1. Este trabalho estuda uma versão do problema que apresenta as seguintes características: **Completo** - o grafo G tem um arco entre cada par de cidades i e j; **Total** - qualquer tipo de carro pode ser alugado em qualquer cidade; **Irrestrito** - qualquer tipo de carro pode ser devolvido em qualquer cidade; **Sem repetição** - qualquer tipo de carro só pode ser alugado no máximo uma vez; **Livre** - as taxas de retorno não são dependentes da topologia do grafo; **Simétrico** - os custos de deslocamento $D^c_{ij} = D^c_{ji}$.

As instâncias para o problema podem ter distâncias euclidianas entre as cidades ou usar outro esquema para definir os custos de viagem. Consideramos instâncias não-euclidianas mais desafiadoras porque as distâncias euclidianas não se encaixam bem para aplicações reais. A maioria das estradas que ligam as cidades têm curvas ou desviam de obstáculos naturais. Nem sequer é possível ter uma conexão direta entre qualquer par de cidades.

2.1 Revisão da Literatura

O primeiro trabalho sobre o CaRS foi feito por [3]. Quatro meta-heurísticas foram apresentadas e após vários testes e comparações, o Algoritmo Memético MA2 superou os outros métodos. O ponto forte de MA2 foi a utilização de uma versão da heurística construtiva baseada no critério de escolha do vizinho mais próximo.

DOI: 10.5540/03.2017.005.01.0460

Uma abordagem transgenética foi aplicada para o problema por [2]. Algoritmos transgenéticos são meta-heurísticas inspirados na evolução endossimbiótica intracelular [6, 8], um processo evolutivo em que uma célula hospedeira e um endossimbionte cooperam para a otimização do problema. O Algoritmo Transgenético TA foi comparado com MA2 e obteve melhores resultados em muitos casos. O algoritmo TA é bastante complexo, com alguns operadores pouco comuns na literatura sobre algoritmos evolutivos, embora seus princípios não sejam novos. O algoritmo TA é o melhor algoritmo conhecido para o CaRS.

A representação de uma solução nas heurísticas da literatura consiste em dois vetores de N inteiros cada. O primeiro indica a sequência das cidades visitadas e o segundo indica o carro escolhido para visitar cada cidade. Esta representação é muito simples e funciona muito bem para heurísticas de construção. No entanto, para algoritmos com cruzamento ou qualquer outro operador de recombinação, inconsistências podem acontecer com bastante frequência como mostrado em [3]. Tais questões exigem um algoritmo de correção que pode demorar muito tempo ou limitar o potencial da solução, a fim de torná-la viável.

3 Métodos propostos

Este trabalho propõe um novo Algoritmo Evolutivo (EA) onde os indivíduos são representados por vetores de números reais que definem a prioridade de escolha das cidades e dos tipos de carros. Desta forma, se as cidades x e y têm prioridades respectivamente 0,45 e 1,23, a cidade y deve ser visitada antes da cidade x. O mesmo vale para os tipos de carros, cada um tem uma prioridade indicando o quão antes ele deve ser utilizado em relação aos demais.

No entanto, para que uma solução não apresente ambiguidade, é necessário que ela indique também quais cidades serão visitadas por quais carros. Os vetores possuem valores extras que representam em termos relativos a quantidade de cidades que deve ser visitada por cada carro. Um valor normalizado de 0,4 significa que este carro deve visitar 40% das cidades, por exemplo.

A Figura 1 mostra um exemplo de representação de um indivíduo (solução). A instância em questão possui 8 cidades e 3 carros. A cidade 1 é sempre o ponto de partida, logo sua prioridade é irrelevante. O carro 2 tem a maior prioridade, logo será utilizado primeiro. Como os valores normalizados são, respectivamente, 0,25 (carro 1), 0,5 (carro 2) e 0,25 (carro 3), este primeiro trecho visitará 4 (=0,5*8) cidades. Partindo da cidade 1 com o carro 2, o cliente visita as cidades 7, 4, 5 e 3 nesta ordem. O carro 2 é devolvido para a cidade 1 e o carro 1 passa a ser utilizado nas próximas 2 cidades: 2 e 8. O carro 1 é devolvido para a cidade 3. Finalmente, o carro 3 é alugado, o cliente visita a cidade 6 e retorna até a cidade 1 onde o carro 3 é devolvido para que volte até a cidade 8 de onde partiu. Esta solução pode ser transcrita por $(1 \overset{2}{\rightarrow} 7 \overset{2}{\rightarrow} 4 \overset{2}{\rightarrow} 5 \overset{2}{\rightarrow} 3 \overset{1}{\rightarrow} 2 \overset{1}{\rightarrow} 8 \overset{3}{\rightarrow} 6 \overset{3}{\rightarrow} 1$, onde $i \overset{c}{\rightarrow} j$ indica que o carro c é usado no deslocamento de i para j). Este escalonamento é realizado sempre que se precisar calcular a aptidão de um indivíduo, que é dada pela soma dos custos de deslocamento com as taxas de retorno dos carros.

O algoritmo EA proposto possui alguns operadores explicados a seguir e um pseudocódigo pode ser visto no Algoritmo 1.

DOI: 10.5540/03.2017.005.01.0460

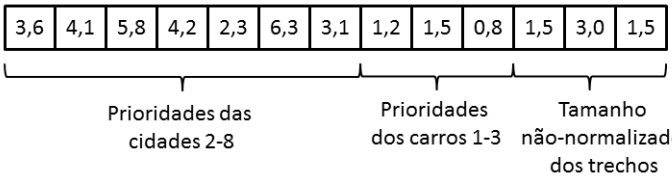

Figura 1: Representação do indivíduo

Algoritmo 1 Pseudo-código do Algoritmo EA

```
 1: pop ← PopulacaoInicial(totalIndiv)
 2: geracao ← 1
 3: melhorSol ← MelhorIndivIduo(pop)
 4: while tempo total não excedido do
 5:     filhos ← Recombinacao(pop)
 6:     pop ← SelecaoNatural(pop+filhos)
 7:     melhorSol ← MelhorIndividuo(pop)
 8:     geracao ← geracao + 1
 9:     if geracao mod mesmoLambda = 0 then
10:         lambda = lambda * incLambda
11:         if geracao mod intGer = 0 then
12:             Intensificacao(pop)
13:         end if
14:     end if
15: end while
16: return melhorSol
```

Primeiramente, é calculado o custo médio da viagem entre cada par de cidades i e j ($i \neq j$), considerando todos os carros. Os indivíduos iniciais são gerados com valores aleatórios para as prioridades dos carros e comprimento dos trechos. Gerar prioridades completamente aleatórias para as cidades não apresenta bons resultados. Por outro lado, uma abordagem puramente gulosa tendo em conta os custos médios gera indivíduos todos muito semelhantes entre si, o que afeta negativamente a diversificação da população. Para contornar este problema, foi aplicada uma estratégia mista, onde r cidades iniciais são escolhidas aleatoriamente. Em seguida, as cidades restantes são ordenadas em uma lista de acordo como os custos médios em relação à ultima cidade visitada até agora. Para dar preferência pelas cidades do início da lista, a seguinte fórmula é utilizada: $indice = \lceil x_1 * x_2 * n \rceil$, onde *indice* indica a posição da lista a ser escolhida, x_1 e x_2 são duas variáveis independentes com distribuição uniforme em (0,1] e n é o tamanho da lista.

Por fim, é necessário atribuir um valor de prioridade a cada cidade de acordo com sua ordem de visitação. A prioridade para a i-ésima cidade visitada é dada por $(1 - i/(n - 1)) * P_0$, onde P_0 é uma constante arbitrária. Ao todo *totalIndiv* indivíduos são gerados, mas apenas os *popTam* melhores indivíduos formam a população inicial (linha 1).

Em cada geração, a população é dividida em três classes: a classe A é composta pelos 10% melhores indivíduos; a classe C é composta pelos 60% piores indivíduos e a classe

DOI: 10.5540/03.2017.005.01.0460

B pelos 30% restantes. Um operador de recombinação (linha 5) pega dois indivíduos escolhidos aleatoriamente das classes A e B, respectivamente. A recombinação produz dois filhos calculando a média de cada um dos genes dos pais. A fim de produzir soluções diferentes, uma pequena porção da média é adicionada ao filho 1 e subtraída do filho 2. Esta porção é um número real aleatório entre 0 e λ, onde λ é um parâmetro de diversificação. Um valor próximo de zero significa que os filhos serão bastante semelhantes aos pais.

Em alguns momentos, os indivíduos são submetidos a uma mutação chamada k-swap. Este procedimento troca as prioridades das cidades, k cidades de cada vez, desde que a troca produza um indivíduo melhor. O operador de mutação é aplicado quando um novo indivíduo é criado. Na população inicial, a mutação 3-swap é aplicada com *probMut*% de chance. Se a mutação não acontecer, é aplicada a este indivíduo uma mutação 2-swap compulsória. Depois da recombinação, cada filho também tem *probMut*% de chance de sofrer a mutação 2-swap. Tal esquema de mutação foi planejado para permitir a criação de melhores indivíduos iniciais e para não tornar o EA muito lento.

Após *popTam* filhos serem produzidos, todos os pais e filhos são misturados e um critério elitista forma uma nova população com os melhores *popTam* indivíduos (linha 6). Depois de algumas gerações, λ é aumentado em *incLambda* para permitir uma diversificação maior em futuros indivíduos.

Um mecanismo de intensificação (linha 12) é aplicado a cada *intGer* gerações: os melhores *topIndiv* indivíduos e alguns dos piores realizam uma mutação 3-swap obrigatória. Buscas locais sobre os melhores indivíduos tendem a melhorar os resultados globais do EA. Por outro lado, sem realizar buscas sobre os piores indivíduos, o EA produz populações desequilibradas contendo soluções muito boas e muito ruins. O operador de recombinação só funciona bem nas primeiras gerações, levando o EA à convergência prematura. O esquema escolhido tenta equilibrar desempenho e tempo computacional, calculando um valor de $p = (MP - MI)/MP$, onde MP é a aptidão média da população e MI é a aptidão do melhor indivíduo. Desta forma, os p% piores indivíduos executam um 3-swap. Com o passar das gerações, p tende a diminuir consideravelmente. O critério de parada do EA (linha 4) é um limite de tempo. A Tabela 1 resume todos os parâmetros do EA e os respectivos valores definidos após testes preliminares com diversos valores-candidatos, buscando o melhor equilíbrio entre desempenho do algoritmo e tempo de processamento.

Tabela 1: Valores e descrição dos parâmetros de EA.

Parâmetro	Valor	Descrição
r	20%	Porcentagem de cidades escolhidas aleatoriamente
totalIndiv	2000	Número total de indivíduos criados na População Inicial (linha 1)
popTam	150	Número de indivíduos em cada geração do EA
probMut	5%	Probabilidade de ocorrência de mutação em um indivíduo
mesmoLambda	50	Número de gerações consecutivas com o mesmo valor de λ
incLambda	1.04	Incremento de λ após *mesmoLambda* gerações
intGer	100	Número de gerações entre duas intensificações
topIndiv	3	Número dos melhores indivíduos que realizam intensificação

DOI: 10.5540/03.2017.005.01.0460

4 Resultados Computacionais

Todas as instâncias não-euclidianas de [2] com mais de 50 cidades foram testadas. Os testes foram feitos em um notebook i7 3630-QM 2.4GHz, 8GB RAM e Windows 8.0 64-bits. Nas instâncias euclidianas os algoritmos MA e TA, propostos em [2] obtiveram resultados muito bons. No entanto, como já foi mencionado, essas instâncias não se aplicam a situações reais, por não ser possível utilizar essa métrica de forma prática.

O principal experimento computacional deste trabalho comparou o melhor algoritmo conhecido para o problema, TA proposto em [2], com o EA. Devido à diferenças entre os sistemas computacionais utilizados nos trabalhos, utilizamos um limite de tempo 66% menor em relação ao tempo do TA. O conhecido software PassMark® calculou esse fator. Em instâncias maiores, o limite de tempo do EA foi ainda mais restrito. A Tabela 2 apresenta os resultados comparativos, sendo n o número de cidades e $|C|$ a quantidade de tipos diferentes de carros à disposição do cliente. Em itálico, foi utilizado um tempo computacional mais restrito para o EA. As melhores soluções estão em negrito.

Tabela 2: Resultados após 30 execuções dos algoritmos TA e EA.

Instância				TA		EA		Melhoria%	
Nome	n	$\|C\|$	Tempo(s)	média	melhor	média	melhor	média	melhor
BrasilNE50n	50	5	98,0	629	618	629,1	**611**	0,0	1,1
Santos50n	50	5	102,0	402	392	394,1	**384**	2,0	2,0
Berlin52nA	52	3	120,6	1350	1326	1324,8	**1311**	1,9	1,1
st70nB	70	4	276,6	934	910	905,9	**890**	3,0	2,2
Macapa80n	80	5	416,0	636	616	616,0	**605**	3,1	1,8
rat99nB	99	5	873,3	1432	1399	1400,7	**1385**	2,2	1,0
Londrina100n	100	3	738,0	1201	1186	1179,2	**1166**	1,8	1,7
w100nB	100	4	777,3	1703	1670	1667,5	**1650**	2,1	1,2
rd100nB	100	4	834,0	1442	1412	1410,4	**1399**	2,2	0,9
Osasco100n	100	4	666,0	1005	993	988,8	**978**	1,6	1,5
pr107n	107	5	1054,3	1740	1698	1701,1	**1676**	2,2	1,3
ch130n	130	5	*1200,0*	1737	1696	1687,0	**1673**	2,9	1,4
Cuiaba140n	140	4	*1200,0*	1364	1339	1329,2	**1315**	2,6	1,8
krob150n	150	3	*1200,0*	3018	2966	2947,9	**2923**	2,3	1,4
PortoVelho160n	160	3	*1200,0*	1446	1426	1415,7	**1405**	2,1	1,5
d198n	198	4	*2400,0*	3264	3188	3152,2	**3130**	3,4	1,8
Aracaju200n	200	3	*2400,0*	1984	1942	1897,3	**1885**	4,4	2,9
Teresina200n	200	5	*2400,0*	1467	1410	1384,3	**1373**	5,6	2,6
Curitiba300n	300	5	*3600,0*	2272	2222	2162,6	**2145**	4,8	3,5

O algoritmo EA obteve melhores resultados em todas as instâncias. Se apenas os melhores resultados das 30 execuções forem comparados, o EA é claramente melhor do que o TA, uma vez que nosso algoritmo encontrou novos limites inferiores para todas as instâncias. É importante destacar que o tempo computacional do TA para instâncias grandes foi ainda maior do que o gasto pelo EA. Em relação ao percentual de melhora, o EA se mostrou um pouco mais robusto em instâncias grandes, com melhorias médias de até 5,6% e índices de até 3,5% nos melhores resultados.

DOI: 10.5540/03.2017.005.01.0460

5 Conclusões

Este trabalho abordou uma variante do clássico Problema do Caixeiro Viajante, chamado Problema do Caixeiro Alugador (CaRS). Neste problema, o cliente é um locatário de carros que deseja passar por todas as cidades uma vez e, em seguida, retornar para a cidade original. O objetivo é percorrer uma rota minimizando os custos de deslocamento e as taxas de retorno, que dependem do tipo de carro utilizado e das respectivas cidades.

Foi proposto um novo Algoritmo Evolutivo para o problema, que se diferencia dos demais pela forma de representação da solução e pelos operadores evolutivos que fazem bom uso dela. Após alguns experimentos computacionais, o algoritmo proposto apresentou melhores resultados para quase todas as instâncias da literatura, com resultados até 5,6% melhores em média. Em todas as instâncias de maior porte, o algoritmo proposto superou os resultados da literatura.

Referências

[1] M. Beaulieu and M. Gamache. An enumeration algorithm for solving the fleet management problem in underground mines. *Comput. & ops. res.*, 33:1606–1624, 2006.

[2] M.C. Goldbarg, E.F.G. Goldbarg, P.H. Asconavieta, M. da S. Menezes, and H.P.L. Luna. A transgenetic algorithm applied to the traveling car renter problem. *Expert Systs. with Appl.*, 40:6298–6310, 2013.

[3] M.C. Goldbarg, P.H. Asconavieta, and E.F.G. Goldbarg. Memetic algorithm for the traveling car renter problem: An experimental investigation. *Memet. Comput.*, 4:89–108, 2012.

[4] A. Jaoua, D. Riopel, and M. Gamache. A simulation framework for real-time fleet management in internal transport systems. *Simul. model. pract. and theory*, 21:78–90, 2012.

[5] P. Misiurski. The impact of the quality of the bus fleet in the implementation of strategy of sustainable development of a region. *Manag. of Environ. Qual.: An Int. J.*, 26(4):471–484, 2015.

[6] P. Moscato and C. Cotta. A modern introduction to memetic algorithms. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 141–183. Springer US, 2010.

[7] S. Seay and A. Narsing. Transitioning to a lean paradigm: a model for sustainability in the leasing and rental industries. *Acad. of Strateg. Manag. J.*, 12:113–114, 2013.

[8] K.S. Shin, J.O. Park, and Y.K. Kim. Multi-objective fms process planning with various flexibilities using a symbiotic evolutionary algorithm. *Comput. & Ops. Res.*, 38:702–712, 2011.

DOI: 10.5540/03.2017.005.01.0460