A Utilização do Algoritmo Genético na Solução de Problemas de Otimização de Rotas

Moacyr Franco de Moraes Neto, Mauro Faccioni Filho

Unisul Virtual, Grupo de Pesquisa SISPLEX – Universidade do Sul de Santa Catarina UNISUL, Palhoça, SC, Brasil

moacyr.moraes@unisul.br, mauro.faccioni@unisul.br

Resumo. *Mediante a utilização do algoritmo genético podem ser desenvolvidos mecanismos eficazes para a resolução de problemas de buscas e otimização de tarefas. Neste trabalho, após a apresentação teórica do modelo, foi realizada e apresentada a implementação de uma ferramenta para auxiliar o setor de logística de entrega industrial, simulando roteiros otimizados de entrega de mercadorias. Como exemplo de aplicação foi simulada a entrega na cidade de São Paulo, considerando várias possibilidades de caminhos e pontos diferentes. Realizaram-se simulações variando de 30 a 50 pontos de entrega, com resultados de otimização em processamento inferior a um minuto.*

Abstract. *By the use of genetic algorithm, effective mechanisms can be developed to solve problems of general search and optimization of tasks. In this work, after the theoretical presentation of the model, a tool was implemented to assist the industrial logistics sector, simulating optimized goods delivery. As an example of application, delivery in the city of São Paulo was simulated in this work, considering several possibilities of different paths and delivery points. Simulations ranging from 30 to 50 delivery points were performed, with optimization results processed in less than one minute.*

1. Introdução

O algoritmo genético (AG) é um dos ramos mais conhecidos da computação evolutiva. A computação evolutiva é um dos ramos da ciência da computação que utiliza paradigmas alternativos no processamento de dados convencional. O algoritmo genético teve origem nos anos 70 nos estudos sobre autômatos celulares, realizados na Universidade de Michigan pelo professor John HOLLAND (1992). A base teórica do algoritmo está centrada na teoria da evolução de Charles Darwin, que se baseia no processo de seleção natural fundamentada na sobrevivência dos indivíduos mais aptos. Este artigo não faz uma revisão da teoria e prática dos algoritmos genéticos, que pode ser vista em referências variadas, como MITCHELL (1996), ZBIGNIEW (1994)

O algoritmo genético na atualidade é largamente aplicado na resolução de problemas que envolvem situações de busca e otimização. Esta classe de problemas, em razão da grande dificuldade de solução, e de sua constante presença em grande parte dos

problemas da engenharia, têm atraído a atenção do mundo científico nas mais diversas áreas e gerado inúmeras pesquisas nesse sentido.

Este artigo aplica algoritmo genético no problema de otimização conhecido como "*traveling salesman problem*", ou simplesmente TSP, que pode ser traduzido como "problema do caixeiro viajante (PCV). Não trataremos das inúmeras abordagens existentes sobre soluções para o problema do TSP, mas simplesmente a aplicação de algoritmos genéticos como um modo de resolução.

A motivação desta aplicação partiu do setor de logística de entrega de mercadorias de uma grande indústria, que necessita diariamente definir, para cada um de seus entregadores, rotas de entrega. O problema pode ser descrito da seguinte forma: cada entregador, em média, realiza de trinta a quarenta entregas diárias, sendo necessário que cada entregador visite todos os pontos da rota para ele definida. Para tanto, a indústria dispõe da informação de distâncias entre os pontos, ou localidades, dessa rota a ser percorrida durante o dia.

A solução do problema se deu com o desenvolvimento de um protótipo de ferramenta que utiliza técnicas evolutivas, capazes de gerar as sequências de localidades a serem visitadas pelos entregadores. Para descrever a abordagem desse desenvolvimento, este artigo está dividido em seis seções distintas, organizadas na seguinte forma: na seção dois é descrito em detalhes a caracterização do problema TSP a ser modelado; na seção três é apresentada a abordagem do uso de AG para solução do TSP; na seção quatro descreve-se a concepção do AG utilizado para o desenvolvimento da solução; na seção cinco são apresentadas as tecnologias utilizadas no desenvolvimento da ferramenta e os resultados obtidos.

2. Caracterização do TSP

O TSP clássico consiste na seguinte proposição: dadas algumas cidades e a distância entre elas, o caixeiro viajante deve visitar todas as cidades com o menor custo possível. A solução deste problema consiste em encontrar a sequência de cidades, de forma que a distância percorrida na viagem do caixeiro seja a mínima possível. De acordo com LEWIS e CHRISTOS (2000), tal problema pode ser descrito matematicamente da seguinte forma: dado um conjunto $\{c_1, c_2, \dots, c_n\}$ de cidades e uma matriz $n\ x\ n$ de inteiros não negativos onde d_{ij} denota a distância entre a cidade c_i e a cidade c_j admitindo que $d_{ii} = 0$ e $d_{ij} = d_{ji}$ para todos os i, j , pode-se encontrar a trajetória mais curta pelas cidades onde $\pi(i)$ é a i-ésima cidade visitada nesta trajetória, tal que a distância total seja a menor possível:

$$c(\pi) = d_{\pi(1)\pi(2)} + d_{\pi(2)\pi(3)} + \cdots + d_{\pi(n-1)\pi(n)} + d_{\pi(n)\pi(1)}$$

O TSP é um problema de otimização pertencente à categoria dos problemas *NP-Hard* (*non-deterministic polynomial-time hard*), ou NP-Difíceis. Sua solução algorítmica é de complexidade exponencial, sendo que o esforço de processamento para

sua solução cresce exponencialmente com o tamanho do problema. Para o grupo de problemas *NP-Hard* não existem algoritmos exatos de solução em tempo polinomial.

Para se ter uma dimensão da complexidade computacional, tentar resolver o problema buscando todas as rotas possíveis, segundo ARAÚJO (2001), levará a um problema de otimização e a um de enumeração. Essa estratégia reducionista combinatória (força bruta), aparentemente viável, não é recomendável para a maioria dos casos.

Assim, supondo que $R(n)$ representa o número de rotas de n cidades, e considerando que o caixeiro sai de uma determinada cidade, a cidade de saída não afeta o cálculo e pode ser retirada do conjunto das cidades restantes. Ou seja, o caixeiro deve escolher sua melhor rota de um conjunto de $(n-1)!$ possibilidades. Considerando ainda que não importa o sentido em que se percorre o roteiro, o número de rotas é reduzido à metade, o que nos dá:

$$R(n) = (n-1)!/2$$

Julgando que se tem um computador capaz de fazer um bilhão de adições por segundo, para o caso de 20 cidades o computador precisa de 19 adições para retornar com o comprimento de uma rota. Logo, será capaz de calcular $10^{19}/19$, ou seja, 53 milhões de rotas por segundo. Entretanto, seria necessário analisar um total de 19!/2 rotas, totalizando 6,0 x 10^{16} possibilidades. Dividindo então o número de possibilidades pelo número de rotas por segundo, chegamos ao tempo total de 1,13 x 10^{9} segundos para completar sua tarefa, o que equivale a aproximadamente trinta e seis anos. A Tabela 1, desenvolvida por ARAÚJO (2001), apresenta um demonstrativo do esforço computacional exigido em relação a quantidade de localidades a serem percorridas, esforço esse que é inviável. Portanto a solução proposta com o uso de algoritmo genético, descrito na seção a seguir.

Tabela 1 – Esforço computacional

5	250 milhões	12	Insignificante
10	110 milhões	181.440	0,0015 seg.
15	71 milhões	$4{,}35 \times 10^{8}$	10 min.
20	53 milhões	$6{,}0 \times 10^{16}$	36 anos
25	42 milhões	$6{,}2 \times 10^{23}$	235×10^{16} anos

3. Uso de algoritmo genético na solução do TSP

O algoritmo genético tem como fundamento a teoria da evolução de Charles Darwin, descrita em "As Origens das Espécies", obra publicada em 1859. O algoritmo genético

faz uma analogia aos processos naturais da evolução das espécies: dada uma população de indivíduos, aqueles que contêm características genéticas melhores terão maiores chances de sobrevivência e maior probabilidade de gerar descendentes cada vez mais aptos e adaptáveis, enquanto indivíduos com características menos evoluídas tendem a desaparecer. As características genéticas dos descendentes são parcialmente herdadas de seus genitores, e tendem a se propagar para as novas gerações através da hereditariedade genética, de forma que as qualidades genéticas de uma população anterior poderão ser melhoradas a cada novo ciclo de novas gerações.

Segundo TAVARES NETO (2005) os algoritmos genéticos são estocásticos, baseados em população, e seu desempenho depende dos seguintes operadores:

- Reprodução/Seleção,
- Cruzamento,
- Mutação.

A reprodução é um processo iterativo com o objetivo de enfatizar as melhores soluções e penalizar as piores soluções de uma população. Durante a reprodução são geradas recombinações chamadas de "*crossover*" de genes, onde os genes dos pais se combinam para formar um novo cromossomo. Esse novo cromossomo poderá sofrer "mutação" devido a fatores ambientais ou por erro de cópias. A mutação é responsável pela geração de diversificação na população, pois gera alguma mudança na característica de um indivíduo.

Durante o processo iterativo de recombinação e mutação são gerados os indivíduos mais evoluídos, ou seja, os que melhor se adaptam ao meio. A aptidão "*fitness*" é responsável pelo processo de seleção dos indivíduos, em que são selecionados os que melhor se adaptam ao meio. Assim, os indivíduos que possuem maior aptidão terão maiores chances de reprodução.

Na engenharia os algoritmos genéticos são considerados métodos adaptativos muito utilizados para solucionar problemas de busca e otimização. Seu uso no problema do caixeiro viajante (TSP) é claramente definido, devido ao fato de possibilitar a exploração de pontos diferentes no espaço de soluções, de forma que não se prenda a uma solução ótima local.

A Tabela 2 apresenta uma relação entre os conceitos de algoritmo genético clássico (AG) e os problemas de otimização encontrados no TSP.

Tabela 2 – Relação entre AG e problema de otimização TSP

Indivíduo	Solução do problema, e a ordem em que deverão ser percorridas as rotas.
População	É o conjunto de soluções, ou seja, diferentes sequências possíveis de rotas a percorrer.
Cromossomo	Representação de uma solução. Um roteiro específico.
Gene	Parte da representação da solução, "sub-rotas".

Alelo	Valor que um gene pode assumir.
Crossover	Operadores de busca. Recombinação de genes de sequenciamento de rotas para geração da nova população.
Mutação	Operadores de busca. Mudanças aleatórias na ordem dos roteiros. Forma de sair de uma solução ótima local.
Aptidão "*fitness*"	Forma de medição de capacidade dos indivíduos. Rotas com menor custo são consideradas mais aptas.

Na Figura 1 é apresentado o fluxo básico a ser utilizado para implementação do algoritmo genético de forma genérica. Na seção 4, a seguir, será descrita em detalhes a abordagem utilizada para implementação do algoritmo genético para a solução do problema TSP.

Figura 1 – Fluxograma genérico do Algoritmo Genético

4. Abordagem utilizada no desenvolvimento da ferramenta

Para a implementação do algoritmo genético em ferramenta para solução do problema TSP, percorremos as seguintes etapas do fluxo detalhado na Figura 2: geração da população inicial, seleção das melhores rotas, recombinação de rotas, mutação e critério de convergência. Tais etapas estão descritas a seguir.

4.1 - Geração da população inicial

O primeiro passo é a codificação dos cromossomos. Na codificação dos cromossomos foi utilizada a codificação por permutação, ou *path*, onde cada cromossomo representa uma série de localidades, ou "vetor de localidades", e a posição de cada localidade dentro do vetor define a ordem em que o roteiro terá de ser percorrido. Conforme demonstrado no cromossomo apresentado na Figura 3, o entregador deverá percorrer o roteiro na seguinte sequência: primeiro a localidade um, a seguir a localidade três, depois a sete, e assim por diante, sempre respeitando a ordem de posição definida no vetor.

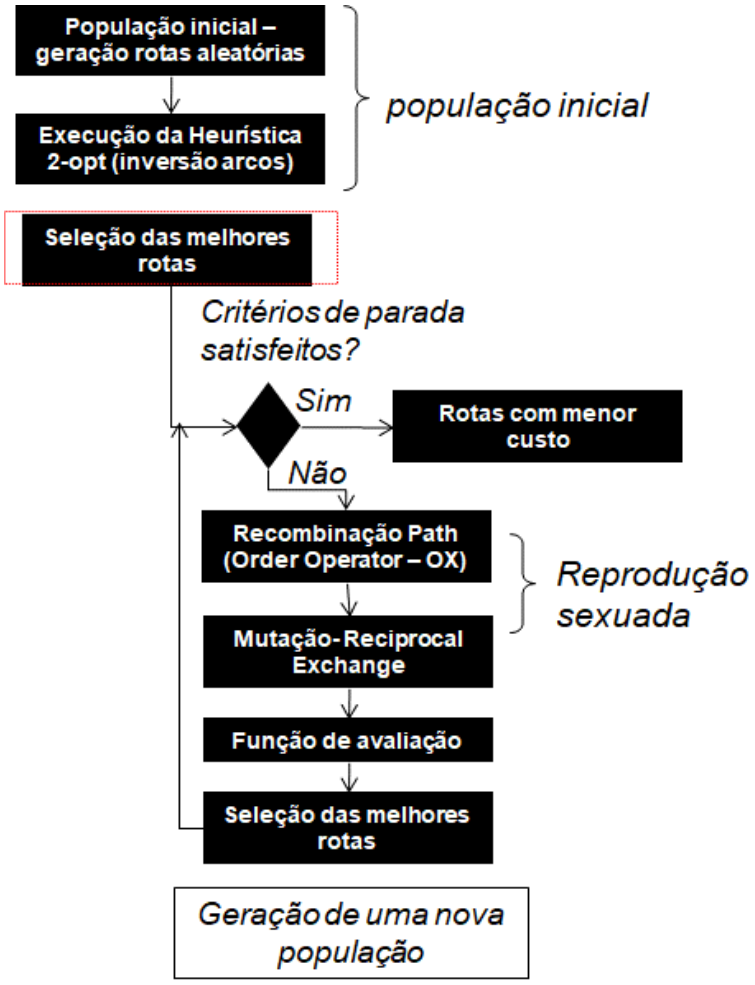

Figura 2.0 – Fluxograma AG no TSP

Na geração da população inicial foi utilizada uma solução do tipo aleatória, e aplicada uma solução heurística com o objetivo de minimizar o tempo de convergência do algoritmo.

Para o protótipo, foi gerada aleatoriamente uma população inicial P de dimensão N, em que o tamanho de P é proporcional ao número de rotas selecionadas, sendo o tamanho da população inicial definida por $P = S * 3$, em que S é o número de rotas a serem percorridas e 3 é uma constante.

Após a geração da população inicial aleatória, foi aplicado uma heurística gulosa utilizando o algoritmo 2-opt, conforme proposta originalmente por CROES (1958). Com o uso dessa heurística na população inicial é possível definir o melhor caminho de uma determinada rota, obtendo solução que corresponde a uma solução ótima local, mas não tendo garantia de que esta seja uma solução ótima global. O 2-opt é um algoritmo de busca local que, segundo NUNES (2006), baseia-se em modificações simples na rota realizando trocas entre as localizações do percurso, e tem como objetivo reduzir o

comprimento da rota, ou seja, minimizar a distância dos percursos pelas as n cidades; quando uma redução não é mais possível, tem-se um circuito localmente ótimo.

O algoritmo 2-opt trabalha com a ideia de vizinhança, sendo S o espaço da busca e f a função-objetivo a ser minimizada. A vizinhança da solução inicial S_0 corresponde ao conjunto $N(s_0) \sqsubseteq S$, que reúne um número determinado de soluções s. Cada solução $s \in N(s_0)$ é chamada de *vizinho* de S_0, e é obtida a partir de uma operação chamada de *movimento*.

Gerada a população inicial e aplicado o método 2-opt, é realizada a avaliação da população através do método de cálculo de aptidão, descrito a seguir.

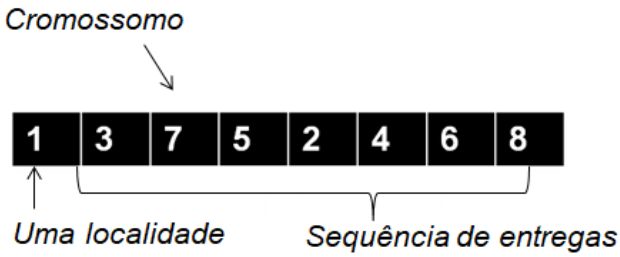

Figura 3 – Representação do cromossomo (vetor de localidades)

4.2 – Seleção das melhores rotas (cálculo de aptidão do indivíduo, "*fitness score*")

Na função utilizada para a avaliação dos indivíduos, foi realizado o cálculo do custo da sequência de rotas contido em cada indivíduo "cromossomo" (sendo o custo dado pela distância total da sequência considerada). Tal função retorna o valor específico do custo de cada cromossomo, sendo que foram considerados como indivíduos com maior aptidão aqueles que obtiveram menor distância entre as localidades a serem percorridas.

Esta função de aptidão recebe como parâmetro de entrada um vetor de cromossomos, que são os "indivíduos da população inicial", e tem como saída um vetor de cromossomos ordenados da forma que o primeiro elemento do vetor seja o cromossomo com menor custo, o segundo elemento tenha o segundo menor custo e assim sucessivamente. Objetivando maior desempenho para a ferramenta na ordenação dos cromossomos, foi utilizado o algoritmo recursivo de ordenação por intercalação, conhecido como "*merge sort*" (ordenação por mistura), onde seu tempo de execução é da ordem de $\Theta(n \log n)$ (CORMEN, 2001).

4.3 – Recombinação de rotas (*crossover*)

A forma de recombinação utilizada na reprodução dos cromossomos foi a recombinação do tipo sexuada, ou seja, o descendente "filho" foi gerado a partir da fusão de dois cromossomos "pais". Nesse caso, após obtido o retorno da função de aptidão contendo

os cromossomos com rotas com menor custo, foram escolhidos aleatoriamente dois cromossomos, "pai" e "mãe", para gerar o descendente.

O operador utilizado nesta recombinação foi o operador tipo *Order Operator* (OX), proposto por DAVIS (1985). Tal operador, conforme exemplificado na Figura 4, gera o descendente escolhendo uma subsequência de um dos pais e preservando a ordem relativa das localidades do outro pai. As seguintes etapas são realizadas na recombinação OX:

1. Seleciona aleatoriamente dois pontos de corte nos cromossomos. No exemplo apresentado na Figura 4, o intervalo de corte $S_1 = \{4,5,6\}$ se refere ao primeiro pai, o P_1, e o intervalo de corte $S_2 = \{7,8,2\}$ ao segundo pai, o P_2.
2. Uma cópia dos elementos do intervalo de corte selecionado é realizada. De S_1 para o F_2 , e S_2 para F_1. Em que F_1 e o F_2 serão os filhos resultantes do cruzamento.
3. F_1 recebe o restante dos elementos de P_1, iniciando a recombinação do ponto de corte S_1 , preservando os elementos já presentes no cromossomo. A recombinação é realizada de forma circular, e ao término do último elemento há o retorno para o primeiro, até que todos os elementos do cromossomo sejam preenchidos.
4. O mesmo procedimento é realizado com F_2.

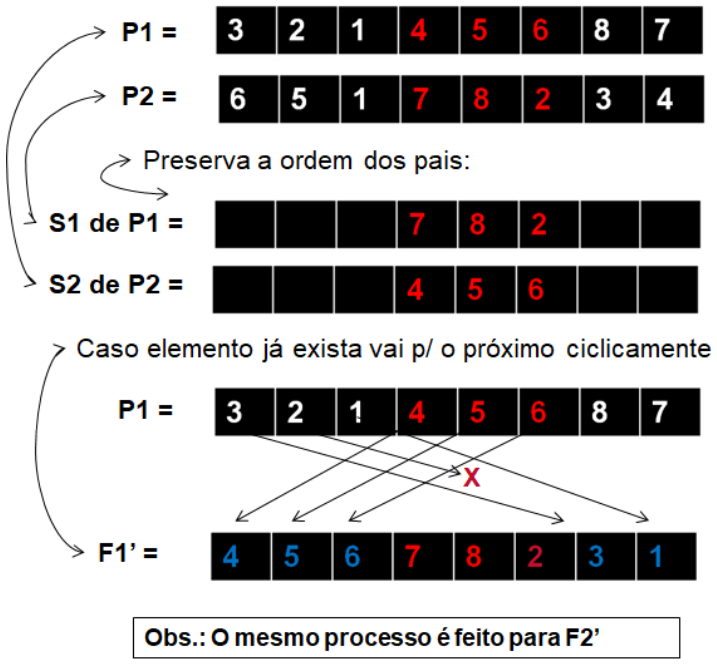

Figura 4 – Recombinação do tipo *Order Operator (OX)*

No processo de recombinação optou-se por manter, em cada nova população gerada, 10% da população atual. Tal abordagem foi adotada com o objetivo de manter junto aos novos indivíduos, os indivíduos mais aptos até então, mantendo assim um histórico dos indivíduos mais aptos da população anterior.

4.4 – Mutação

Depois de realizado o processo de recombinação dos cromossomos, gerando assim uma nova população, é realizado o processo de mutação. A mutação é aplicada a cada um dos cromossomos desta nova população, satisfazendo a regra Rn_i, em que a mutação se dá por troca recíproca, ou "*Reciprocal Exchange*" (RE), que sorteia aleatoriamente dois genes "localidade" e inverte sua posição.

A regra Rn_i é definida da seguinte forma: para cada 10 recombinações sucessivas em que o custo das rotas se mantiver, o algoritmo genético deverá aplicar mutação por troca recíproca a seus descendentes, como ilustrado no exemplo da Figura 5.

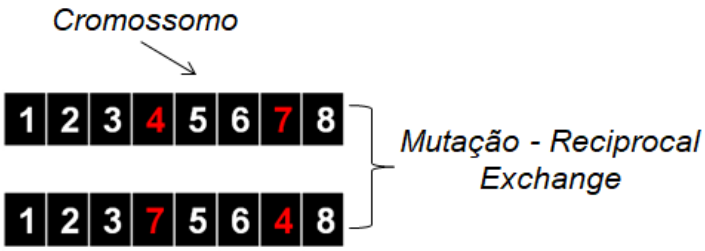

Figura 5 – Mutação por troca recíproca

4.5 – Critério de convergência

Para um algoritmo genético a convergência se caracteriza como sendo uma progressão até certo grau de uniformidade entre os indivíduos. No protótipo desenvolvido foram utilizadas duas premissas como critérios de convergência e permanência do processamento do algoritmo, sendo elas:

- Primeira premissa: todos os cromossomos devem possuir o mesmo custo para as sequências de visitas a serem realizadas, para que se obtenha uma convergência global;
- Segunda premissa: dada a rota R contendo N localidades a visitar, o tempo máximo de execução do algoritmo, caso não seja satisfeita a primeira premissa, é delimitado por certa quantidade de iterações de reprodução, definida por $N * C$, em que N é igual ao número de localidades que devem ser visitadas, e C é uma constante que define o número máximo de iterações que o algoritmo poderá realizar. No protótipo a constante está parametrizada para 800 iterações, e pode ser ajustada conforme o aumento do número de localidades a visitar. O objetivo da segunda premissa é obter um critério de parada para o algoritmo, caso o mesmo não tenha uma convergência global em até um determinado limite máximo de iterações.

5. Tecnologias adotadas e resultados obtidos

No desenvolvimento da ferramenta foram utilizadas as seguintes tecnologias *Open Source*: banco de dados MySQL versão 5.0, linguagem de programação Java versão 1.6 com Adobe Flex versão 3.0, e servidor Web Apache Tomcat versão 6.0.

Considerando que as informações de rotas de entregas e distâncias são dados confidenciais da empresa, para a publicação deste artigo houve a necessidade de se utilizar dados fictícios para o desenvolvimento e realização dos testes. Tais informações fictícias de rotas foram extraídas do mapa dos bairros da cidade de São Paulo, disponível no website da Prefeitura da cidade de São Paulo. Para o cálculo das distâncias entre os bairros foi considerada uma escala medida em pixels, do ponto central de um bairro ao ponto central do bairro vizinho, em linha reta. Para que tivéssemos o resultado da distância entre todos os bairros foi utilizado o algoritmo de *Floyd-Warshall*, que é um algoritmo para cálculo de caminhos mínimos entre todos os pares de um grafo. O custo computacional desse algoritmo na implementação foi de ordem $O(n3)$.

Na Tabela 3 são apresentados os resultados obtidos nos testes da ferramenta, para diversas quantidades de localidades de entrega. Para os testes de desempenho foi utilizado computador com as seguintes configurações: Processador AMD turion Dual Core 2.0Ghz, 2Gb de memória Ram DDR, Hard Disk 250Gb com 5.400 *rpm*. O tempo médio computacional para a convergência do algoritmo genético foi bastante satisfatório em relação aos requisitos do problema.

A Figura 6 apresenta o caminho a ser percorrido num dia por um entregador em roteiro com 30 localidades, ou seja, $N = 30$. Para este caso, conforme descrito na Tabela 3, obtivemos uma convergência global num tempo computacional médio aproximado de 31 segundos.

Tabela 3 – Tempo médio computacional de convergência.

10	100%	07 segundos
20	100%	20 segundos
30	100%	31 segundos
50	100%	52 segundos
80	100%	83 segundos

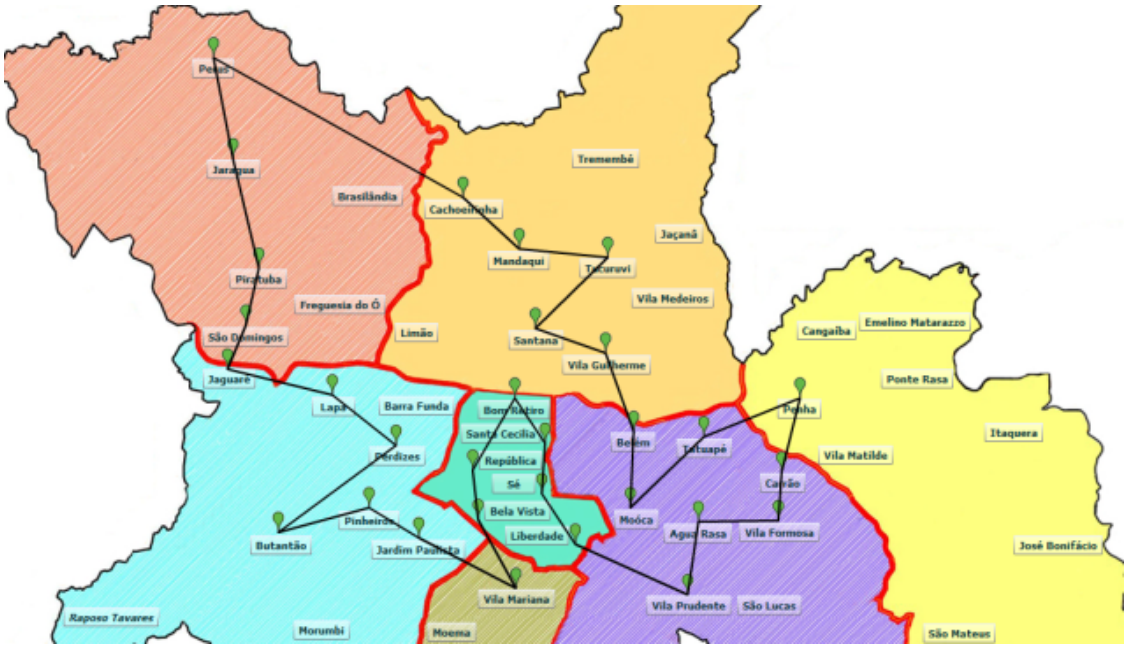

Figura 6 – Roteiro com 30 localidades de entrega.

A Figura 7 apresenta o caminho a ser percorrido num dia por um entregador em roteiro com 50 localidades, ou seja, $N = 50$. Para este caso obtivemos uma convergência global num tempo computacional médio aproximado de 52 segundos.

Figura 7 – Roteiro com 50 localidades de entrega.

6. Conclusões

O trabalho apresenta uma abordagem de utilização de algoritmo genético como solução do problema de otimização conhecido como TSP. Embora não se possa afirmar que a solução gerada pela ferramenta seja ótima, verifica-se que os resultados apresentados são adequadas às necessidades. Nota-se, também, que uma das características mais importantes da utilização dos algoritmos genéticos, para a solução desse tipo de problema, se dá pela relativa facilidade de relacionar a abordagem genética ao problema do TSP, obtendo assim resultados significativos.

No entanto, quando tal abordagem é utilizada para casos que exigem otimização de número mais elevado de localidades, o tempo de convergência do algoritmo se prolonga de maneira polinomial em relação à dimensão do roteiro. Dependendo dos requisitos da aplicação, a estratégia adotada pode ser inadequada para a solução do problema em questão.

Um avanço da proposta pode se dar pela adoção de abordagem heurística mais sofisticada, visando melhorar a seleção das soluções "ótimas locais" no processo de geração da população inicial. A inclusão de uma heurística de busca local, no processo de reprodução dos indivíduos, pode tornar o algoritmo atual um algoritmo com características híbridas "meméticas", combinando as técnicas evolucionárias com as técnicas heurísticas de busca local. Concluindo, tais mudanças poderão otimizar o algoritmo atual, levando assim a convergência a um tempo computacional menor, mesmo para grande número de localidades.

Contribuição dos autores: MFMN realizou o projeto de desenvolvimento da ferramenta, experimentos e obtenção de resultados, bem como versão inicial do artigo. MFF colaborou na discussão, revisão e versão final do texto.

Referências

HOLLAND, J.H. **Adaptation in Natural and Artificial Systems.** MIT Press. USA. 232 pp. 1992.

LEWIS e CHRISTOS**. Elementos de teoria da computação / Harry R. e Chiistos H. Papadimitriu;** trad. Edson Furmankiewicz. -2ª. Ed.- Porto Alegre: Bookman, 2000.

ARAUJO, H.A. **Algoritmo Simulated Annealing**: Uma nova abordagem [Dissertação de Mestrado]. Florianópolis: UFSC, 2001. Disponível em https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/80386/225675.pdf?sequence=1 (Acesso: 04/08/2017).

TAVARES NETO, R. F. **Planejamento e Vistoria Usando Robôs Móveis Autônomos e Otimização pelo Algoritmo de Colônia de Formigas** [Dissertação de Mestrado]. Curitiba, PUC-PR, 2005.

CROES, G. A. (1958). **A method for solving traveling salesman problems**. Operations Res. 6 (1958), pp., 791-812

CORMEN, T. H., LEISERSON, C. E., RIVEST, R.L., STEIN, C. **Introduction to Algorithms**. MIT Press and McGraw-Hill. 2nd Edition. 2001.

DAVIS, L: **Applying Adaptive Algorithms to Epistatic Domains**. Proceedings of the International Joint Conference on Artificial Intelligence, 1985.

NUNES, ALVARO PRESTES: **Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante.** [Dissertação de Mestrado]. Rio Grande do Norte, UFRG, 2006.

MITCHELL, MELAINE. **An Introduction to Genetic Algorithms**. 1st Edition. The MIT Press. 1996.

ZBIGNIEW, MICHALEWICZ. **Genetic Algorithms + Data Structures = Evolution Programs**. Springer-Verlag.1994.