UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

João Paulo Dias de Almeida

ALGORITMOS GENÉTICOS APLICADOS A PROBLEMAS DE ROTEAMENTO: CAMINHO MÍNIMO E CAIXEIRO VIAJANTE

Orientador: Profa. Ana Lúcia Lima Marreiros Maia

FEIRA DE SANTANA

2012

UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA – UEFS

João Paulo Dias de Almeida

Algoritmos Genéticos Aplicados a Problemas de Roteamento: Caminho Mínimo e Caixeiro viajante

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia de Computação da Universidade Estadual de Feira de Santana, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientadora:

Profa. Ana Lúcia Lima Marreiros Maia

FEIRA DE SANTANA
2012

RESUMO

Este trabalho apresenta o desenvolvimento de um Algoritmo Genético para auxiliar um robô autônomo no planejamento de rotas. O problema consiste em encontrar o menor caminho, dentre os possíveis, a ser percorrido por um robô. O ambiente onde o robô se encontraserá mapeado através de um grafo cujas arestas possuem pesos associados e representam o comprimento do caminho. Com base nestes pesos, um Algoritmo Genéticofoi usado para encontrar a melhor combinação de subcaminhos que gere o menor caminho de uma origem a um destino. Depois, restringiu-se o problema ao problema do caixeiro viajante e aplicou-se novamente o Algoritmo Genético desenvolvido. Para realizar os testes do problema do caminho mínimo foram gerados grafos aleatoriamente, para o problema do caixeiro viajante foram utilizadas as coordenadas do Oliver30 TSPe outros grafos disponíveis na TSPLib. Os experimentos foram realizados em grafos de variados tamanhos, com diferentes configurações de parâmetros iniciais e técnicas de recombinação e mutação, no intuito de encontrar a configuração mais precisa e eficiente. A partir destes experimentos foram geradas análises dos resultados obtidos para cada problema.

Palavras-chave: Algoritmo Genético.Caminho Mínimo.Caixeiro Viajante.Planejamento de Rotas.RobôsAutônomos.

ABSTRACT

This work presents the development of a Genetic Algorithm to assist in an autonomous robot route planning. The problem is to find the shortest path among the possible, to be followed by a robot. The environment where the robot is mapped is through graphs which edges have associated weights and represent the path length. Based on these weights, a Genetic Algorithm was used to find the best combination of sub-paths that generate the shortest path from a source to a destination. Then the problem was restricted to the Traveling Salesman Problem and applied again developed Genetic Algorithm. In order to testthe algorithm to solve the Shortest Path Problem were randomly generated graphs, and for the Traveling Salesman Problem, we used the coordinates of Oliver30 TSP and other graphs available in the TSPLIB. The experiments were performed on graphs of varying sizes, with different parameter settings and initial recombination and mutation techniques in order to find the most accurate and efficient configuration. From these experiments were generated analyzes of the results obtained for each problem.

Keywords: Genetic Algorithm. Shortest Path. Traveling Salesman. Planning Routes. Autonomous Robots.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A robótica é o estudo feito sobre qualquer tipo de equipamento mecânico, controlado e motorizado por computador. Esses dispositivos podem ser programados para fazer diversas tarefas de modo automático, ou seja, sem a supervisão humana (SALANT, 1990). A robótica móvel é um desafio para os pesquisadores, pois os robôs móveis que são autônomos requerem a integração do sensoriamento, planejamento e execução como um sistema único. A função do sistema de sensoriamento é traduzir a entrada dos sensores em um modelo global que represente o ambiente no qual o robô se encontra. O sistema de planejamento reproduz este modelo global, encontra um destino e gera um plano para chegar ao ponto desejado. Já o sistema de execução, parte do plano e produz as ações que ele executa (KORTENKAMP, 1998).

Em aplicações de robôs móveis existe um grande grupo que possui um problema em comum: uma vez que se possui um conjunto de tarefas a serem cumpridas, o sistema autônomo deve gerar uma rota que passe por diversos pontos de um mapa, como por exemplo, robôs com o objetivo de entrega de medicamentos em um hospital, robôs com o objetivo de inspeção de um sistema de encanamentos. Um fator importante a se observar em um sistema autônomo, que objetiva a obtenção de uma flexibilidade, para ter sua lista de tarefas reconfigurada a qualquer instante, é que o planejamento do comportamento do robô móvel deve ser realizado com relativa rapidez e maior otimização possível, com menor custo computacional (NETO; COELHO, 2005). Neste contexto, o primeiro problema a ser resolvido é o planejamento da rota a ser seguida pelo robô móvel. Nesta etapa busca-se determinar o trajeto que o robô deve percorrer, evitando ao máximo a redundância e o desperdício de energia, ou seja, otimizando-se a rota com a melhor relação custo-benefício.

Existem várias aplicações comerciais de robôs autônomos, dentre elas destacam-seo robô móvel autônomo para aplicação em agricultura (MANDOW, 1996), substituindo o trabalho humano em atividades inóspitas, como a pulverização de inseticidas em estufas, um veículo autônomo especializado em detectar e socorrer pessoas soterradas em escombros de estruturas danificadas (MURPHY, 2000), sendo que tal veículo poderia buscar por sobreviventes com um nível de rigor que normalmente é fatigante para os humanos, e um sistema mais otimizado para a inspeção de tubulações do que a inspeção manual, notada por Lockheed (1997), que afirma a “necessidade no Departamento de Energia dos Estados Unidos

em inspecionar as condições internas de tubulações e outros ambientes inacessíveis, perigosos ou restritos durante a descontaminação e desativação de instalações sem uso".

Este trabalho tem o objetivo de desenvolver uma abordagem baseada em algoritmos genéticos que possa ser utilizada para o planejamento de rotas de um robô autônomo, a qual pode ser utilizada em diversas aplicações comerciais, industriais e tecnológicas.

Algoritmos Genéticos são algoritmos de busca baseados no mecanismo da seleção natural e da genética natural, os quais devem sua popularidade à possibilidade de percorrer espaços de busca não-lineares e extensos, explorando-os de forma paralela (PIRES, 2009). Tal abordagem visa encontrar uma solução ótima para o problema de movimentação de um robô móvel em um ambiente desconhecido, iniciando o movimento a partir de uma coordenada, até outra posição de destino conhecida, resolvendo o problema do planejamento da rota de forma autônoma.

O planejamento de rotas foi explorado através de dois problemas: o problema do caminho mínimo e o problema do caixeiro viajante. O problema do caminho mínimo é um problema simples de busca pelo menor caminho entre dois pontos, com extensa aplicação no contexto de planejamento de rotas para robôs. O problema do caixeiro viajante consiste na busca por um ciclo hamiltoniano[1], também com conhecidas aplicações no planejamento de rotas para robôs, como na fabricação de placas de circuitos eletrônicos ou no sequenciamento de tarefas onde uma sequência de execução das tarefas ideal pode minimizar o tempo total de processamento (ARAÚJO, 2006). Além disso, a busca por um ciclo hamiltoniano é uma boa maneira de demonstrar a eficiência do algoritmo em um ambiente mais restritivo, por ser um problema NP-completo.

Este trabalho será dividido em quatro seções: Revisão Bibliográfica, Materiais e Métodos, Resultados e Discussão, e Conclusão. Na Revisão Bibliográfica serão apresentados os assuntos abordados neste trabalho de forma ordenada e pormenorizada. Em Materiais e Métodos será apresentada a metodologia utilizada para desenvolver o Algoritmo Genético proposto, enquanto na seção Resultados e Discussão serão apresentados alguns dos testes realizados, acompanhados de uma discussão do comportamento encontrado em cada experimento. Para finalizar, a Conclusão apresentará os objetivos alcançados e os objetivos futuros deste trabalho.

[1] Ciclo Hamiltoniano é o circuito em um grafo, em que todos os vértices devem ser visitados uma única vez.

2. REVISÃO BIBLIOGRÁFICA

A seguir são apresentados os principais conceitos trabalhados,a iniciar pelo Planejamento de Rotas, e em seguida tratando sobre algoritmos genéticos e todos os operadores genéticos avaliados e utilizados para a resolução do problema do caminho mínimo do caixeiro viajante.

2.1 Planejamentode Rotas

A grande evolução da área de robótica móvel pode ser observada em diversas áreas do conhecimento humano, aplicando robôs dentro de reatores nucleares, linhas de manufatura, inspeção de dutos, fundo de oceanos, além da comercialização de brinquedos infantis e educacionais. Além disso, outro aspecto relevante é que os robôs cada vez mais têm sido dotados da capacidade de aprender, atuar autonomamente, e interagir com os humanos e seu ambiente.

Em muitas das áreas de aplicação de robótica móvel existe um problema em comum: uma vez que exista um conjunto de tarefas a serem cumpridas, o sistema autônomo deve gerar/planejar uma rota que passe por diversos pontos de um mapa. Em um sistema autônomo é importante a obtenção de uma boa flexibilização, para que a lista de tarefas a serem executadas possam ser reconfiguradas a qualquer instante, e que o planejamento do comportamento do robô móvel seja capaz de ser realizado com relativa rapidez, otimização e com o menor custo computacional possível (NETO; COELHO, 2004).

Outro aspecto de destaque é que o trajeto que o robô irá percorrer, de uma origem até um destino seja um trajeto ótimo, para os problemas abordados neste trabalho, este trajeto ótimo é o melhor caminho dentre os disponíveis. Geralmente, a trajetória a ser percorrida pelo robô é mapeada e implementada através da utilização de grafos (FRACASSO, 2008). Tipicamente, um grafo é representado como um conjunto de vértices ligados por arestas e podem possuir pesos (ou custos), sendo que o custo total em estudo será calculado a partir destes pesos. A Figura 1 mostra um exemplo de grafo com pesos nas arestas.

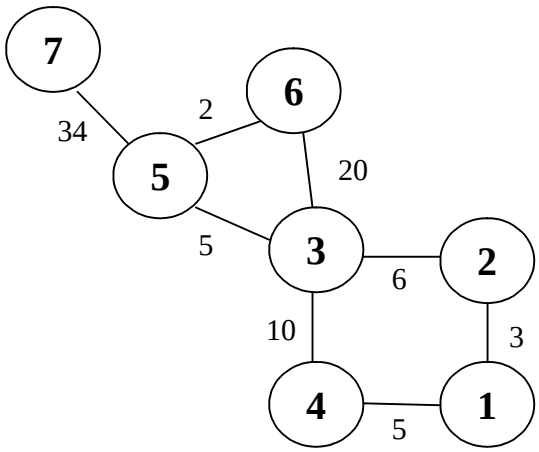

Figura 1 - Exemplo de grafo com os respectivos pesos de suas arestas

No problema de planejamento de rotas, os vértices representam locais que o robô pode visitar e as arestas representam os caminhos entre esses locais. Cada aresta possui um valor associado, que determina o comprimento do caminho (distância) entre os dois vértices adjacentes pela aresta.

Para construir o grafo, ou qualquer outra estrutura de dados, que represente os possíveis caminhos a serem percorridos é necessário construir o mapa do ambiente em que o robô se encontra. Para construir o mapa são utilizadas técnicas de representação e interpretação do espaço (FRACASSO, 2008) ou o mapa pode ser extraído diretamente do ambiente utilizando técnicas de processamento de imagens (ARAUJO; LIBRANTZ, 2006). Estas técnicas irão construir o mapa a ser percorrido pelo robô, indicando os obstáculos a serem desviados e os locais que podem ser percorridos livremente. Segundo Latombe (1991), as técnicas de construção de mapas de rotas podem ser divididas de três formas: o grafo de visibilidade, o diagrama de Voronoi e o probabilístico.

O grafo de visibilidade utiliza como vértices do grafo todos os cantos e quinas dos obstáculos, além das posições de origem e de destino (NISSOUX; SIMEON; LAUMOND, 1999). Os caminhos obtidos com esta técnica são os de menor comprimento possível, uma vez que são compostos por segmentos de retas que tangenciam os cantos e quinas dos obstáculos (FRACASSO, 2008). Fracasso (2008) afirma que “devido a este tipo de abordagem, a navegação do robô móvel pode ser uma tarefa perigosa, uma vez que o sistema de localização pode fornecer dados imprecisos ao sistema de navegação, fazendo com que haja uma real possibilidade de colisão entre o robô móvel e obstáculos”.

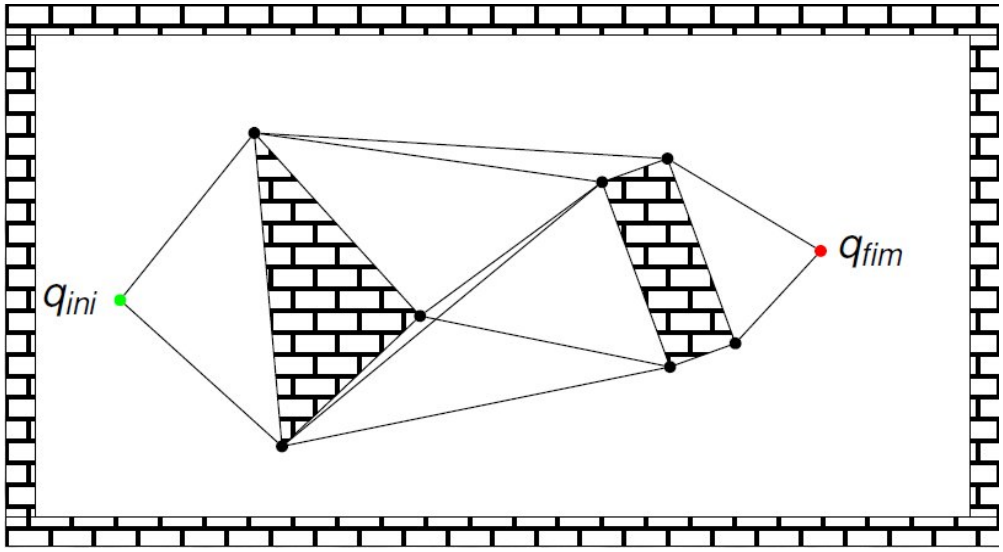

Figura 2 –Grafo de visibilidade aplicado em ambiente com obstáculos. Fonte: DIAS (2010).

O diagrama de Voronoi cria os vértices do grafo, baseando-se nas posições livres e equidistantes entre os elementos pertencentes ao ambiente, como obstáculos, limites, posições proibidas etc (CHOSET; BURDICK, 2000). Estas posições são denominadas “pontos geradores”. Entre estes pontos, são traçadas retas equidistantes entre si que irão formar as bordas dos polígonos. Cada polígono será fechado e adjacente a outro, contendo apenas um dos pontos geradores (REZENDE; ALMEIDA; NOBRE, 2000). Diferentemente do que ocorre no grafo de visibilidade, a resposta desta técnica está o mais distante possível do obstáculo, porém, este tipo de estratégia penaliza o comprimento dos caminhos resultantes, uma vez que são muito conservadoras (FRACASSO, 2008).

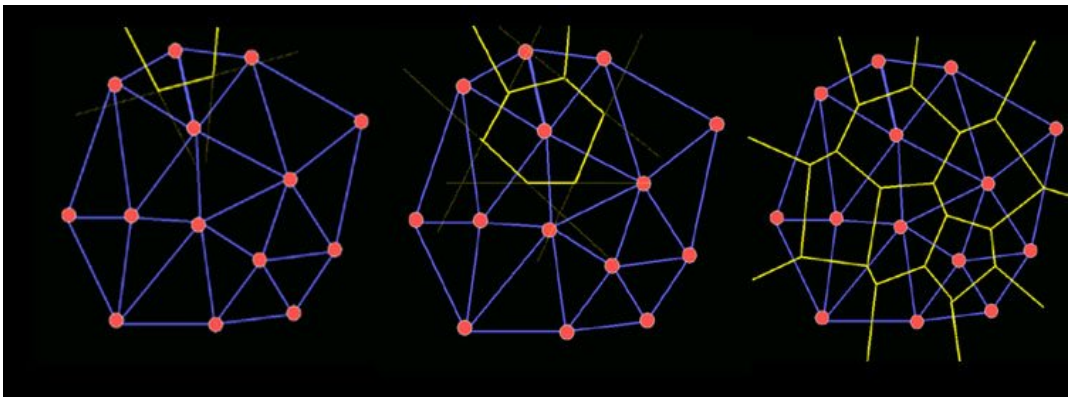

Figura 3 – Polígonos gerados pelo Diagrama de Voronoi. Fonte: WANGENHEIM (2012).

O planejamento probabilístico seleciona posições nos espaços livres e tenta conectá-las para formar um grafo, onde posteriormente são adicionadas as posições de origem e de destino em busca de um caminho factível entre elas. Segundo Fracasso (2008), “A adição da noção de não determinismo faz com que o algoritmo reduza a sua complexidade computacional e consequentemente é mais rápido, quando comparado com algoritmos deterministas”. Outra característica do algoritmo de planejamento probabilístico é que ele não é exato e sim aproximado, uma vez que trata somente amostras do espaço de configurações livres, isto diminui consideravelmente o tamanho da árvore de busca (FRACASSO, 2008).

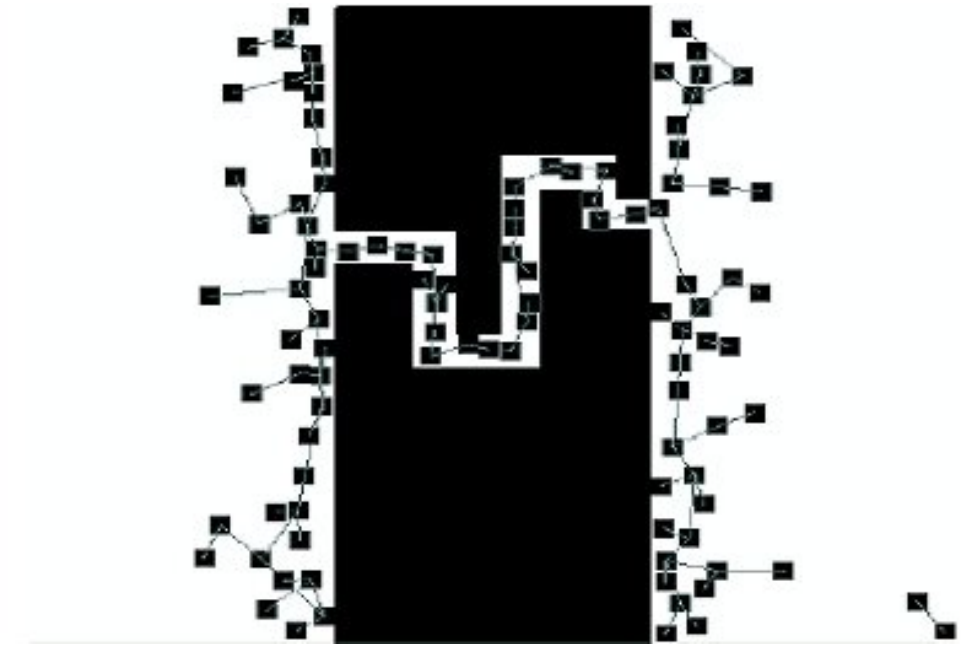

Figura 4 – Planejamento probabilístico realizado em um espaço de configurações livres. Fonte: BORGES (2012).

2.2 Algoritmos Genéticos (AG)

Vários algoritmos têm sido concebidos utilizando eventos biológicos como inspiração, o Algoritmo Genético é um destes. Algoritmos Genéticos (AG) são algoritmos inspirados na teoria da evolução de Darwin onde a solução do problema é obtida através da simulação de uma competição entre indivíduos em uma mesma população, na qual sobreviverá apenas o indivíduo mais apto. A população é iniciada com um determinado número de possíveis soluções (indivíduos) e, com as interações entre esses indivíduos, o indivíduo mais apto irá prevalecer e se perpetuar durante as gerações, até

que os genes desse indivíduo domine a população por inteiro ou parcialmente (HERRERA, 2010). A função de aptidão é que avalia cada um dos indivíduos dessa população e indica a aptidão deste indivíduo. O valor da aptidão define se um indivíduo irá “morrer” ou “viver” na próxima geração. Operadores genéticos irão simular o cruzamento, a mutação e a seleção dos indivíduos dentro do ambiente evolutivo. As técnicas de recombinação e mutação são consideradas como agentes introdutórios de diversidade genética na população, pois a partir destes operadores, novas possíveis soluções serão geradas para competir com as soluções já existentes na população (MITCHELL, 1996).

De uma forma geral, o AG se comporta conforme o Algoritmo 1(LUGER, 2004):

Início
ajuste o tempo (gerações) t = 0;
produza a população inicial P(t);
enquanto a condição de parada não for satisfeita faça
início
avalie a aptidão de cada indivíduo da população P(t);
selecione membros de P(t) com base na aptidão;
produza os descendentes destes escolhidos utilizando os operadores de recombinação e mutação;
substitua, com base na aptidão, indivíduos de P(t) pelos descendentes gerados;
fim da primeira geração t = t + 1;
fim
fim

Algoritmo 1 – Pseudocódigo do comportamento geral de um AG

A seguir serão apresentados os parâmetros iniciais do Algoritmo Genético além de outros aspectos importantes que precisam ser definidos, como a representação genética para as soluções do problema (cromossomo), a função de aptidão, a população inicial, os métodos de seleção e o critério de parada. A maioria destes parâmetros necessita ser definida de forma empírica, com exceção da forma de representar o cromossomo e a definição da função de aptidão. Por analogia aos eventos biológicos, muitos dos termos utilizados em torno dos algoritmos genéticos são referentes à biologia.

2.2.1 Cromossomos

O cromossomo, para os problemas abordados neste trabalho, consiste em uma estrutura de dados que armazena a ordem em que os vértices foram visitados no grafo, onde o valor da ligação entre esses vértices representa o peso das arestas que ligam cada par de vértices que compõem o cromossomo. A Figura 2 representa um grafo onde as setas não pontilhadas indicam o caminho percorrido, enquanto a Figura 3 representa este caminho na forma de um cromossomo.

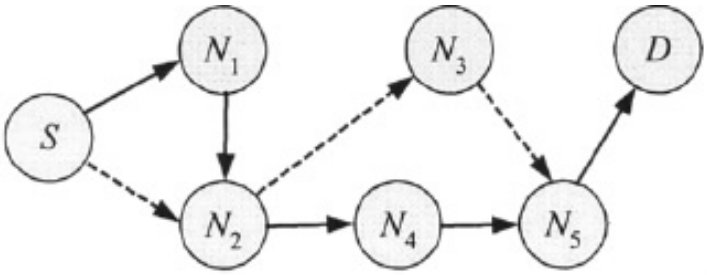

Figura 5 - Representação dos vértices de um grafo.Fonte: AHN e RAMAKRISHNA (2002).

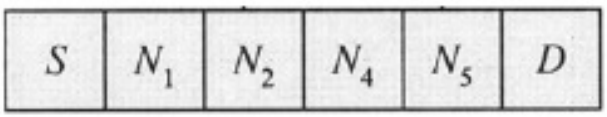

Figura 6 - Representação de um possível caminho como um cromossomo.Fonte: AHN e RAMAKRISHNA (2002).

Segundo Lacerda e Carvalho (1999), alguns cuidados devem ser tomados para que cromossomos idênticos não sejam avaliados mais de uma vez, sendo assim, deve-se: evitar gerar cromossomos idênticos na população inicial, verificar se o cromossomo gerado (denominado como filho) pela recombinação ou mutação não é igual ao cromossomo selecionado pelo operador genético e, antes de avaliar um filho, verificar se um cromossomo idêntico já existe na população.

2.2.2 População Inicial

Para desenvolver umAlgoritmo Genético é necessário definir, de forma empírica, o tamanho da população inicial, que nada mais é do que um grupo de cromossomos que serão gerados aleatoriamente no início da execução do algoritmo. Geralmente, utiliza-se um tamanho de população proporcional ao tamanho do cromossomo, isto é, quanto maior for o cromossomo maior deverá ser o tamanho da população a fim de manter uma diversidade razoável. Quanto maior o número de cromossomos na população inicial, maior será o espaço de busca do Algoritmo Genético. O Espaço de busca pode ser definido como o conjunto de possíveis soluções que estão sendo processados pelo AG.

2.2.3 Função de aptidão

A função de aptidão (*fitness*) é a função utilizada para indicar quão apto um indivíduo está para fazer parte de uma população. Determinar de maneira eficaz o valor de aptidão de cada solução é parte fundamental do problema e está diretamente ligada ao desempenho e qualidade da solução final do Algoritmo Genético. Para solucionar o problema do menor caminho em um grafo e o problema do caixeiro viajante, a abordagem mais simples para determinar a aptidão dos indivíduos é somar todos os pesos entre os vértices de um possível caminho. Para esta abordagem, quanto menor for este valor, mais apto estará o indivíduo. O caminho será descrito na forma de um cromossomo, como visto na Figura 6.

2.2.4 Métodos de Seleção

Dada uma população em que a cada indivíduo foi atribuído um valor de aptidão, existem vários métodos para selecionar os indivíduos sobre os quais serão aplicados os operadores genéticos de cruzamento e mutação. Estes indivíduos selecionados formarão uma população, que é conhecida como população intermediária. A maioria dos métodos de seleção é projetada para escolher preferencialmente indivíduos com maiores ou menores valores de aptidão, embora não exclusivamente, a fim de manter a diversidade da população. A seguir serão descritos dois métodos de seleção:

- **Seleção pela roleta**

O método de seleção da Roleta é um método de seleção muito utilizado, onde os indivíduos de uma geração são escolhidos para fazer parte de uma população intermediária através de um sorteio de roleta. Neste método cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão, como é mostrado na Figura 4. Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor da roleta. Finalmente, a roleta é girada um determinado número de vezes, dependendo do tamanho da população. Os indivíduos sorteados pela roleta serão os escolhidos para participar da população intermediária e poderão ser aprimorados na próxima geração;

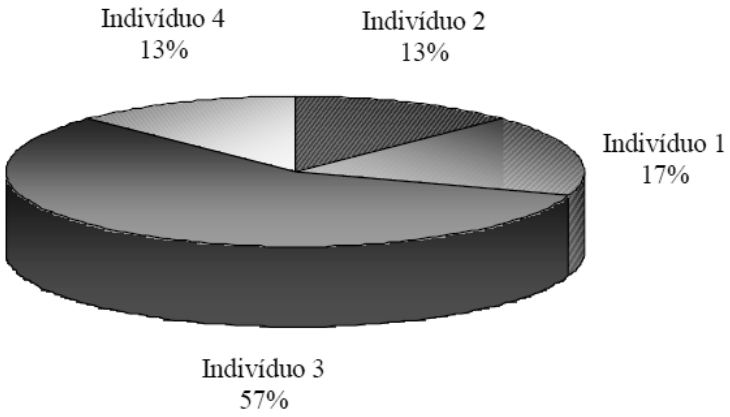

Figura 7 - Seleção por roleta. Fonte: BERTONI (2006).

- **Seleção por torneio**

No método de seleção por Torneio são escolhidos aleatoriamente (com probabilidades iguais) n indivíduos da população e o que possuir maior aptidão é selecionado para a população intermediária. O processo repete-se até completar a população intermediária;

- **Seleção Elitista**

Introduzido por Kenneth De Jong (1975), este método de seleção garante que uma quantidade n dos melhores indivíduos serão preservados e serão passados para a próxima geração. Esta abordagem garante que os melhores indivíduos não sejam destruídos pelos operadores de mutação e

cruzamento, auxiliando na convergência do algoritmo. Devido ao seu comportamento, este método costuma ser utilizado em conjunto com outro método de seleção (BERTONI, 2006).

2.2.5 Critério de Parada

Os critérios de parada de um Algoritmo Genético variam de acordo com a aplicação em questão. A finalização da execução pode ocorrer após um número de gerações pré-fixado, pode ser baseada na repetição sucessiva do melhor indivíduo durante as gerações, pode ser efetuada através da observância de que a média ou o desvio padrão do valor de aptidão não sofre alteração no decorrer das gerações, ou simplesmente parar ao encontrar a solução ótima, caso ela seja conhecida.

2.2.6 Operadores de Cruzamento (*crossover)*

Também conhecido pelo termo recombinação, os operadores de cruzamento são os principais responsáveis em introduzir novos indivíduos (possíveis soluções) em uma população. A partir de dois cromossomos já existentes, denominados pais, ocorre uma combinação que determina como serão formados os dois novos indivíduos após o cruzamento. O operador de cruzamento define como a combinação entre os pais irá ocorrer. Os operadores de cruzamento são aplicados em uma determinada população obedecendo a uma taxa de recombinação, que define quantos elementos da população irão passar por este processo. É importante definir esta taxa com precisão para evitar cruzamentos desnecessários, ou evitar que os cruzamentos sejam insuficientes para alcançar o resultado desejado. A maior parte da literatura recomenda uma taxa de cruzamento entre 75% e 95% (MITCHELL, 1996). A seguir serão listados alguns dos métodos mais comuns de cruzamento, utilizando exemplos de cromossomos utilizando codificação binária (representado por 0’s e 1’s) (BERTONI. 2006):

a) Cruzamento de Um-Ponto

Define um ponto entre os pais denominado de ponto de cruzamento. O cromossomo será dividido em dois neste ponto de cruzamento e os filhos serão

gerados a partir da combinação das metades dos seus pais, conforme ilustrado na Figura 8;

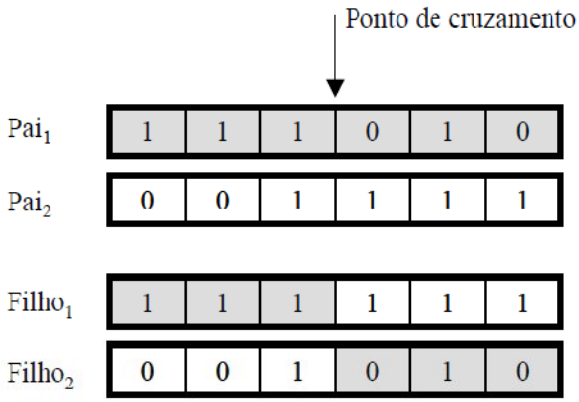

Figura 8 - Cruzamento de Um-Ponto. Fonte: BERTONI (2006).

b) Cruzamento uniforme

Neste operador, o filho é gerado através de sucessivas cópias dos genes dos pais.Estas cópias são feitas de forma alternada e são controladas pela máscara de cruzamento, que deve ser gerada aleatoriamente. Na Figura 6 a máscara determina que onde houver 1 o gene a ser copiado é do pai 1, e onde houver zero, o gene que deve ser copiado é o do pai 2.

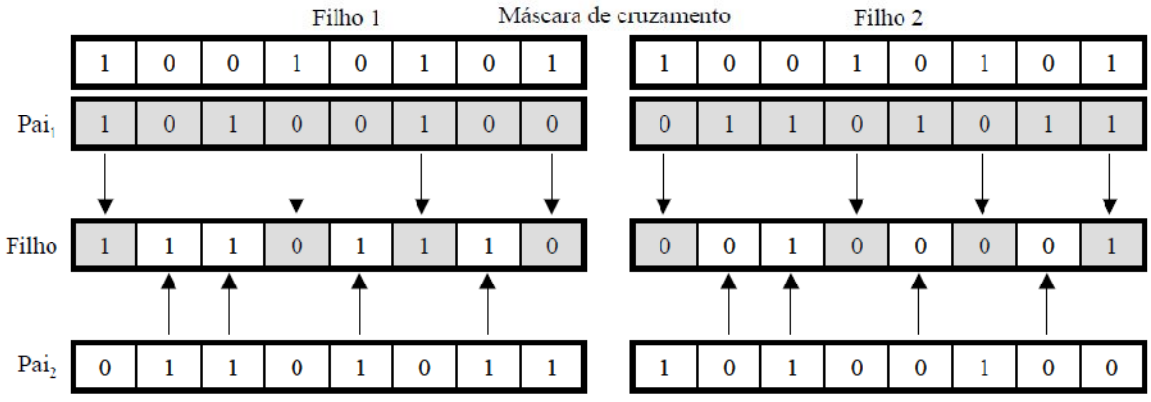

Figura 9- Cruzamento uniforme. Fonte: BERTONI (2006).

2.2.7 Operadores de Mutação

São os operadores que modificam com maior intensidade os indivíduos de uma população e por isso devem ser utilizados com moderação. A literatura recomenda uma taxa de mutação entre 0,5% e 1% (MITCHELL, 1996). Quando aplicado corretamente, o operador de mutação garante a diversidade de soluções a cada geração do Algoritmo Genético, porém se for aplicado exageradamente, pode levar a perda de toda informação adquirida nas gerações passadas. A seguir é apresentado um dos métodos de mutação mais comum.

- **Mutação Uniforme**

  O operador de mutação uniforme seleciona aleatoriamente um gene do cromossomo e substitui seu valor por um número aleatório gerado dentro do domínio definido para a variável em questão. O comportamento deste operador é ilustrado na Figura7.

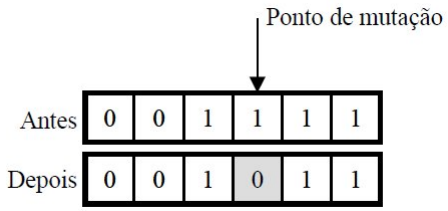

Figura 10 - Mutação uniforme. Fonte: BERTONI(2006).

2.3 O Problemado Caminho Mínimo (SP)

O problema do caminho mínimo (*SP – Shortest Path*) é considerado um problema de otimização combinatória clássico, que vem sendo estudado durante muitos anos. Este problema consiste em encontrar um caminho mínimo entre um par de vértices em um grafo qualquer. Muitos métodos já foram desenvolvidos para resolver este tipo de problema, como o conhecido algoritmo de Dijkstra, o algoritmo de Floyd, algoritmos heurísticos pertencentes à inteligência artificial, redes neurais, entre outros (WU; RUAN, 2004).

Os algoritmos tradicionais citados anteriormente conseguem resolver versões do problema SP de forma eficiente, porém, eles nem sempre são adequados para outras versões mais complexas que necessitam atender a certas condições ou restrições, como no problema do caixeiro viajante. Considerando tais versões do problema SP de maior complexidade, a utilização de Algoritmos Genéticos é encorajada devido a sua capacidade de lidar com uma variedade de problemas de otimização (WU; RUAN, 2004).

2.4 O Problema do Caixeiro Viajante (TSP)

O problema do caixeiro viajante (TSP – *traveling salesman problem*)é inspirado em uma situação onde um caixeiro viajante deve visitar N cidades. Para cada cidade que ele visitar haverá um custo do caminho da cidade anterior até a cidade atual. O problema visa encontrar um caminho onde todas as cidades são visitadas e cada uma dessas cidades só será visitada uma única vez, sendo que o ponto final deste caminho deve coincidir com o ponto inicial. O caminho que obtiver o menor custo de forma a atender a todos estes requisitos é a solução para o TSP.

O TSP tem várias aplicações, tais como: perfuração de placas de circuito impresso, a cristalografia de raios X e o roteamento no processo de fabricação de circuitos VLSI (*Very-large-scale integration*). Estes são problemas que requerem a visita de milhares de pontos com um caminho de custo mínimo. O TSP é um problema interessante e difícil, com muitas ramificações de estratégias de busca(LUGER, 2004). Neste trabalho, este problema será abordado utilizando umAlgoritmo Genéticoque possui operadores genéticos desenvolvidos especificamente para este problema.

Para resolver o problema do TSP, algumas modificações nos operadores do algoritmogenético são necessárias,para garantir as limitações do problema do caixeiro viajante.Operadores como cruzamento de um-ponto ou o cruzamento uniforme geram novos cromossomos, sem garantir que este seja um caminho válido para o problema do TSP. Vários pesquisadores criaram operadores que superam esses problemas e que permitem trabalhar com uma lista ordenada de cidades visitadas. Uma forma de solucionar a busca pelo menor caminho seria gerar e avaliar todas as permutações possíveis dos N elementos da lista de cidades. É necessário que os operadores de cruzamento e mutação sejam capazes de produzir todas estas permutações (LUGER, 2004).A seguir serão apresentados os operadores de recombinação e mutação que são capazes de produzir tais permutações.

2.4.1 Recombinação Ordenada

Considerando que a ordem dos números inteiros no vetor da Figura 8corresponde à ordem das cidades visitadas, a recombinação ordenada utiliza-se deste vetor para construir seus descendentes através da escolha de uma subsequência de cidades dentro do caminho de um dos pais. Ela preserva também a ordenação relativa de cidades do outro pai.

Primeiro, são selecionados dois pontos de corte, indicados pelo "|", que são inseridos no mesmo local em cada pai. A localização dos pontos de corte é aleatória, mas, uma vez selecionados, os mesmos locais são usados em ambos os pais (LUGER, 2004). Por exemplo, nas Figuras 8 e 9, para dois pais **p1** e **p2**, os pontos de corte estão após a terceira e a sétima cidade:

Figura 11 - Pai p1

Figura 12 - Pai p2

Nas figuras 10 e 11 estão ilustradas as criações de dois filhos **c1** e **c2** a partir dos segmentos gerados pelos pontos de corte existentes nos pais **p1** e **p2**.A ordem das cidades no segmento é mantida, garantindo que a ordem das cidades visitadas não será modificada no filho gerado.

Figura 13 - Filho c1

Figura 14 - Filho c2

Em seguida, partindo do segundo ponto de corte do pai**p2**, as cidades serão copiadas na mesma ordem, gerando uma lista de cidades que será encurtada ao remover as cidades que já estão presentes no filho **c1**. Assim, para a criação do filho **c1**, a sequência de cidades obtida do pai **p2** seria: 2, 3, 4, 5, 9, 1, 8, 7 e 6.Como 4, 6, 5 e 7 já estão presentes no filho **c1**, removemos estes pontos da sequência de cidades de **p2** e obtemos a lista encurtada 2, 3, 9, 1 e 8, que constitui as cidades remanescentes a serem visitadas pelo filho **c1.** Preservando assim, a ordenação encontrada em **p2** e garantindo que nenhuma cidade será visitada mais de uma vez**.** Preenchendo os pontos que faltam em **c1** com a lista encurtada obtemos o primeiro filho. De forma análoga, obtemos o filho **c2** ilustrados nas Figuras 12 e 13.

Figura 15 - Filho c1 completo

Figura 16 - Filho c2 completo

2.4.2 Cruzamento *Order-Based* (OBX)

O Cruzamento OBX seleciona um grupo de posições aleatoriamente, considerando que as posições escolhidas no pai1 devem ser as mesmas do pai2 (Figura 17). Todas as posições não selecionadas serão passadas do pai para o seu respectivo filho na mesma ordem, enquanto uma nova ordem será atribuída aleatoriamente para as posições escolhidas (Figura 18).

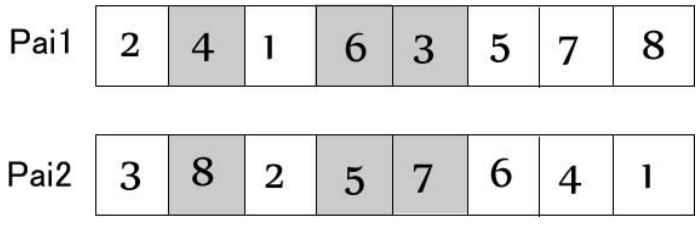

Figura 17 - Posições escolhidas aleatoriamente no cromossomo

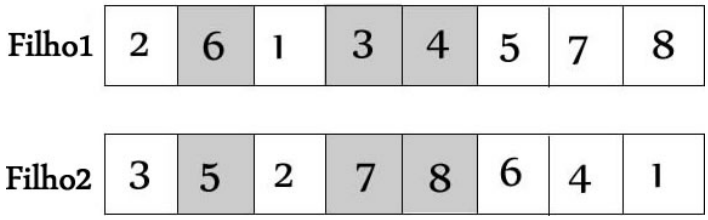

Figura 18 - Nova ordem das posições escolhidas

2.4.3 Mutação Ordenada

A Mutação Ordenada acontece de forma parecida com a do operador de recombinação ordenada. São definidosdois pontos de corte no cromossomo que irá sofrer a mutação. Por exemplo, para o filho c1 na Figura 13, o primeiro ponto de corte está situado após a terceira cidade e o segundo ponto de corte está situado após a sétima cidade. Seleciona-se, então, o segmento criado entre os pontos de corte e o inverte, gerando assim um novo caminho como ilustrado na Figura 19.

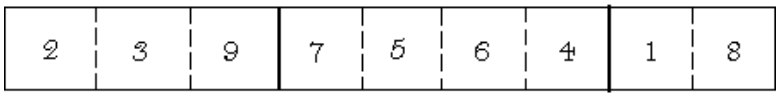

Figura 19 - Filho c1 após sofrer mutação

2.4.4 Mutação *Order-Based*

A Mutação Order-Based é um operador que apenas seleciona duas posições aleatoriamente e inverte os valores das posições escolhidas. A Figura 20demonstra o processo.

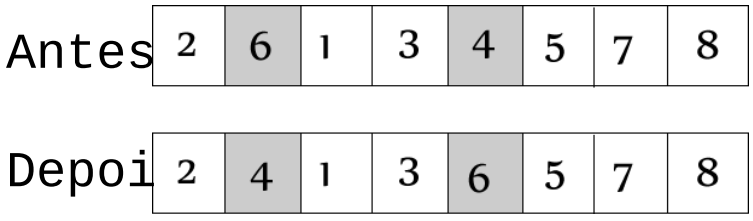

Figura 20 - Mutação Order-Based

3. MATERIAIS E METODOS

Antes de solucionar o problema do caixeiro viajante decidiu-se que seria necessário desenvolver um Algoritmo Genético capaz de encontrar o menor caminho em um grafo sem qualquer restrição. Partindo de um problema mais abrangente para depois solucionar um problema mais específico.

Na abordagem do problema do caminho mínimo deste trabalho, o AG utilizou o operador de cruzamento de um-ponto e o operador de mutação *order-based.*

Em seguida foi desenvolvido o Algoritmo Genético para solucionar o problema do caixeiro viajante, utilizando, inicialmente, os operadores de recombinação e mutação ordenada. Nestes problemas foram realizados vários testes, variando muitos dos parâmetros do AG em busca da melhor configuração. Após selecionar os grafos de caminhos possíveis, na sua maioria provenientes da TSPLib (REINELT, 2008)e com variadas quantidades de vértices, foi executado o Algoritmo Genético na procura pelo menor ciclo hamiltoniano.

Sendo assim, foram desenvolvidos dois programas na linguagem Java, utilizando a abordagem do Algoritmo Genético, onde um seria utilizado para solucionar o problema do menor caminho em um grafo e outro para solucionar o problema do caixeiro viajante. Ambos os programas utilizam o mesmo AG, alterando-se apenas os operadores genéticos para adequação das diferentes abordagens. Como as fontes dos grafos para cada problema, o cálculo da distância entre os vértices e outros métodos auxiliares são diferentes, optou-se por separar o programa em dois, um para cada problema.

Os parâmetros do Algoritmo Genético foram definidos como segue:

- **Representação genética:** para cada vértice do grafo foi dado uma representação numérica. Por exemplo: 1, 2, 3,..., 9 e assim por diante. Dessa forma, cada vértice tem um número que o identifica. Cada vértice do grafo representa uma cidade e as arestas são os caminhos entre cada par de cidades. Desta forma, o cromossomo é representado por uma sequência de números inteiros que representam a ordem das cidades visitadas naquele caminho;

- **População inicial:** variou-se o tamanho da população entre 10, 100, 1000, 5000 e 7000 e 10 000 cromossomos, para fins de validação e otimização da solução;

- **Função de avaliação:** representada pela soma dos pesos dos subcaminhos que compõem o caminho. O indivíduo mais apto será aquele que possuir o menor valor de aptidão;

- **Método de seleção:** para selecionar a população intermediária foram utilizados dois métodos de seleção: o método elitista e método de seleção por Torneio, no qual *n* indivíduos da população são escolhidos aleatoriamente com a mesma probabilidade. O cromossomo com melhor aptidão dentre estes é selecionado para a população intermediária. O processo se repete até que a população intermediária seja preenchida;

- **Operadores genéticos:** são os algoritmos utilizados durante a recombinação e a mutação. As taxas de cruzamento e mutação definem com que frequência estes operadores serão aplicados nos indivíduos. Estes valores foram variados em cada experimento em busca de uma melhor configuração, os detalhes desses experimentos serão discutidos na seção de Resultados e Discussões;

- **Critério de parada:** a finalização da execução ocorre quando o desvio padrão do valor de aptidão não sofre alteração no decorrer das gerações, ou seja, o algoritmo irá convergir quando o valor de aptidão de todos os indivíduos da população forem iguais.

3.1 Algoritmo GenéticoProposto

Na figura 18 será apresentado um fluxograma do Algoritmo Genético proposto e o fluxograma do mesmo (Figura 21), que possui como característica a não eliminação dos cromossomos selecionados após a mutação e recombinação.

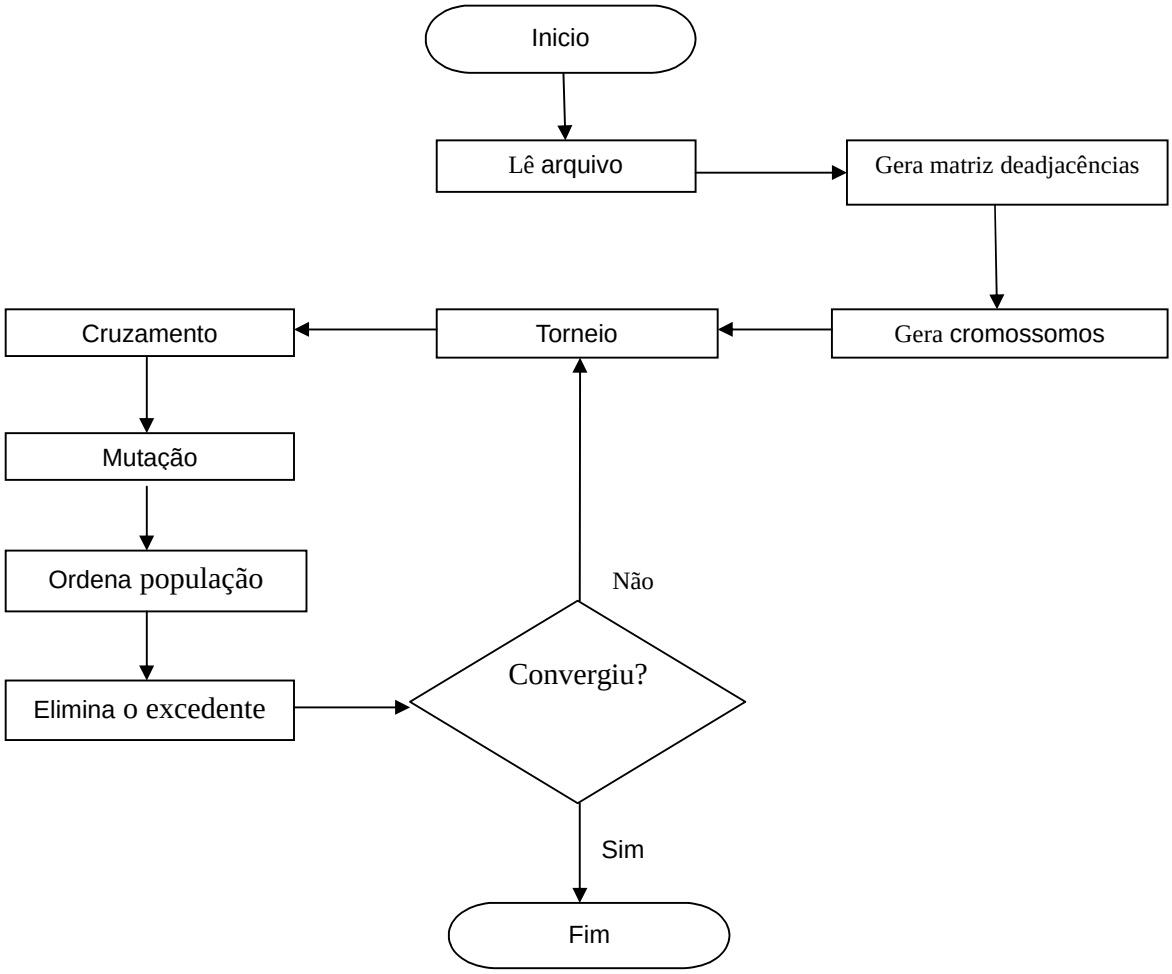

Figura 21 - Fluxograma do Algoritmo Genético proposto

Inicialmente é feita a leitura do arquivo que possui as coordenadas (x, y) de cada vértice do grafo (para o problema do caminho mínimo esse grafo é gerado aleatoriamente). A partir dessas coordenadas, utiliza-se o cálculo da distância euclidiana para gerar a matriz de adjacências que vai representar o grafo em questão. A distância euclidiana é utilizada para

definir a distância entre dois pares de vértices. O TSPLib sugere que o cálculo da distância euclidiana (dij) seja implementado da seguinte maneira para seus grafos (REINELT, 2008):

> Sejam x[i] e y[i] coordenadas de um vértice i.
> xd = x[i] - x[j];
> yd = y[i] - y[j];
> dij = nint (sqrt(xd*xd + yd*yd));
>
> Onde sqrt corresponde à raiz quadrada dos termos entre parênteses e nint a uma função de arredondamento, definida como:
>
> Seja x, o valor em ponto flutuante a ser arredondado.
> nint(x) = (int) (x+0.5), se x >= 0, e
> nint(x) = (int) (x-0.5), se x < 0.

Os cromossomos da população inicial serão gerados aleatoriamente e, terão seus valores de aptidão atribuídos através da soma dos subcaminhos que formam o caminho, representado pelo cromossomo. A distância de cada subcaminho existente no grafo estará presente na matriz de adjacências.

O operador de seleção escolhido para selecionar os indivíduos da população foi o torneio em conjunto com a seleção elitista. Depois de selecionados, cada cromossomo terá uma probabilidade de sofrer uma mutação ou recombinação. Tal probabilidade é definida pelas taxas de recombinação e mutação.

Após aplicar os operadores genéticos nos indivíduos escolhidos a população haverá aumentado, uma vez que o operador de recombinação gera dois novos indivíduos e o de mutação acrescenta mais um. Para evitar que a população cresça indiscriminadamente, antes de iniciar uma nova seleção a população é ordenada pelo valor de aptidão e o excedente é eliminado, mantendo a população sempre com o tamanho da população inicial.

O algoritmo irá convergir quando o primeiro indivíduo possuir o mesmo valor de aptidão do último colocado na população. Enquanto o critério de parada não for alcançado o processo retorna à seleção e o fluxo continua como definido na Figura 21.

4. RESULTADOS E DISCUSSÕES

Esta seção irá apresentar o comportamento desta implementação do AG em diversas situações. Partindo do problema do caminho mínimo e indo até a solução do TSP em grafos reais. Nas seções 4.1 e 4.2 os grafos foram gerados aleatoriamente no início de cada teste enquanto a seção 4.3 utiliza grafos que representam locais reais.

Utilizando um computador Intel® Pentium® D 3.0 GHz, com 4 GBs de memória RAM DDR2 e executando o sistema operacional Windows 7 Ultimate foram realizados 1029 testes (319 para o SP e 810 para o TSP) e devido a grande quantidade, apenas algum deles serão apresentados a seguir.

4.1 O ProblemadoCaminhoMínimo (SP)

Para solucionar este problema, o Algoritmo Genético (AG) foi testado utilizando três tamanhos de grafos diferentes, grafos com 25, 50, 100, 150 e 200 vértices que foram gerados aleatoriamente para cada sequência de testes. Em cada dimensão de grafo foi realizada uma sequência de testes, onde foram realizadas variações no tamanho das populações iniciais e nas taxas de cruzamento e mutação. Uma maior quantidade de testes foi feita utilizando o grafo de 25 vértices, pois quanto menor o grafo, mais rápido a resposta foi obtida, uma vez que o espaço de busca é menor.

Considerando que a cada nova sequência de testes um novo grafo aleatório é gerado, utilizou-se o algoritmo de Dijkstra para encontrar o menor caminho em cada grafo. O algoritmo de Dijkstra é um algoritmo determinístico e, portanto, garante 100% de precisão em todos os casos [LAFORE, 2003]. O valor obtido pelo algoritmo de Dijkstra será apresentado na seção 4.1 como o valor ótimo e será comparado ao valor obtido pelo Algoritmo Genético com o intuito de conhecer em quais configurações o AG retorna os melhores resultados.

Para solucionar o SP foram utilizados como operadores genéticos o cruzamento de um-ponto e a mutação uniforme.Alguns dos principais testes realizados para analisar o comportamento do Algoritmo Genético serão apresentados nas próximas seções.

4.1.1 Experimento 1 - Comportamento com a variação da taxa de recombinação

Para este experimento foram utilizados grafos com 25 vértices e uma população inicial de 500 cromossomos. O valor da taxa de recombinação foi variado visando observar como este operador genético influencia nos resultados gerados pelo algoritmo. Segundo Mitchell (1996), o valor sugerido para a taxa de recombinação deve ser entre 75% e 95%. Baseando-se nessa sugestão, os testes deste experimento se iniciaram com uma taxa de recombinação de 95%. Utilizando esta configuração, a frequência de acertos do valor ótimo foi de 100%, ou seja, em todos os testes realizados, o valor obtido pelo Algoritmo Genético foi igual ao obtido pelo algoritmo de Dijkstra com uma média de tempo de execução igual a 70,25 ms. Diminuindo a taxa de recombinação de 95% para 75% (Figura 22), a inserção de novos elementos nas populações intermediárias é reduzida, diminuindo assim a variedade de soluções no espaço de busca. Apesar da quantidade de elementos recombinados ter sido reduzida, não houve decréscimo na média do tempo de execução dos testes realizados. Para o grafo 25 de vértices, com o decréscimo da taxa de recombinação houve aumento no tempo de execução, pois o algoritmo necessitou de mais gerações para encontrar o resultado. Na Tabela 1 são apresentadas a média dos tempos de execuções em milissegundos e a quantidade de gerações que cada configuração necessitou para alcançar o seu resultado.

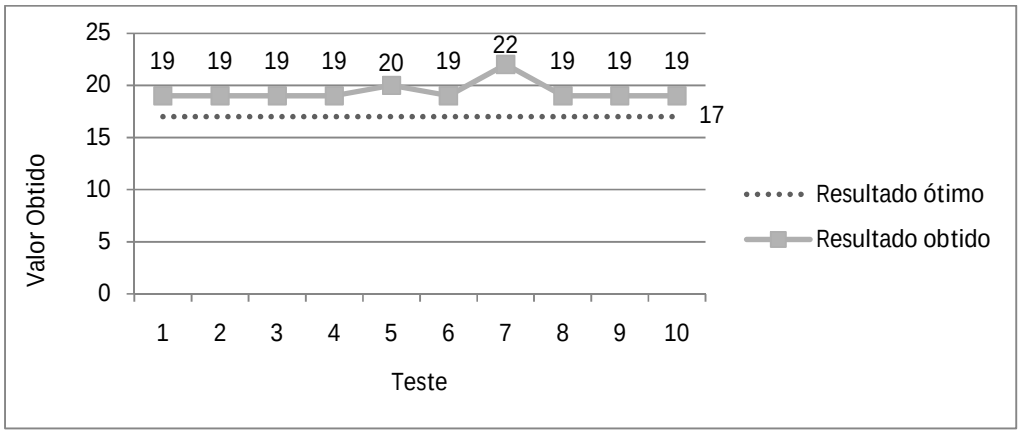

Figura 22 - Resultado obtido com25 vértices (1% de mutação e 75% de recombinação)

Tabela 1 – Tempo de execução obtido com a variação da taxa de recombinação

Porcentagens (Mutação/Recombinação)	Média de gerações necessárias	Média do tempo de execução (ms)
1/95	3	70,25
1/75	5	126,375
1/50	4	121,25

Ainda investigando o comportamento do algoritmo com o decréscimo da taxa de recombinação, foi realizada mais uma sequência de testes utilizando a taxa de recombinação em 50%. Os resultados obtidos com esta configuração são descritos na Figura 23, onde podemos observar que a eficácia do algoritmo continuou a piorar. Assim, conclui-se que diminuir ainda mais a taxa de recombinação não seria algo vantajoso devido a perca de eficácia e do custo computacional.

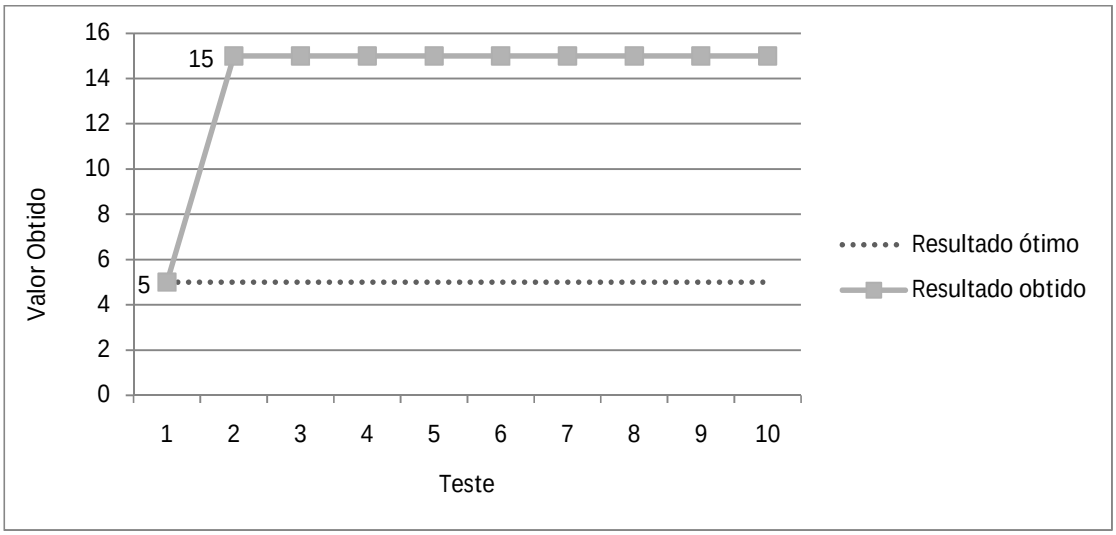

Figura 23 - Resultado obtido com25 vértices (1% de mutação e 50% de recombinação)

4.1.2 Experimento 2 - Comportamento com a variação da taxa de mutação

Neste experimento também foram utilizados grafos com 25 vértices e uma população inicial de 500 cromossomos. Considerando os resultados da seção 4.1.1, a taxa de recombinação foi definida como 95%. A literatura correlata sugere valores entre 1% e 5% para a taxa de mutação (MITCHELL, 1996), sendo assim, os testes deste experimento foram inicialmente realizados com uma taxa de mutação a 1% e gradativamente este valor foi

aumentado para as sequências de testes posteriores. Como já foi relatado, utilizando 1% de mutação e 95% de recombinação foi obtida uma frequência de acerto de 100%. A mesma frequência foi mantida utilizando a taxa de mutação em 5% e 10%. Com o aumento em 5%,o tempo de execução sofreu um pequeno decréscimo enquanto que o aumento da taxa para 10% resultou em um aumento considerável no tempo de execução como descrito na Tabela 2.

Tabela 2 - Tempo de execução obtido com a variação da taxa de mutação

Porcentagens (Mutação/Recombinação)	**Frequência de acerto do valor ótimo**	**Média de gerações necessárias**	**Média do tempo de execução (ms)**
1/95	100%	3	70,25
5/95	100%	3	63
10/95	100%	5	125,16

No Algoritmo Genético proposto, o operador de mutação não elimina o pai (cromossomo que sofreu a mutação). A implementação tradicional do operador de mutação sugere eliminar o pai e utilizar baixos valores na taxa de mutação, valores estes que podem ser verificados na literatura correlata (MITCHELL, 1996). Utilizar altas taxas de mutação e aplicar o operador de mutação, sem preservar o conteúdo do indivíduo que está sendo mutado, pode levar a uma busca aleatória pela melhor solução (CARVALHO, 2012), o que resulta em um maior custo computacional para alcançar a resposta e, uma possível perda na eficácia da resposta encontrada. Segundo Mitchell (1996), altos valores de mutação são aplicáveis em alguns tipos de problemas e que, em alguns casos, uma estratégia *hill-climbing* pode ser mais eficiente do que um AG com alta taxa de recombinação. Considerando que o AG proposto neste trabalho apresenta esta diferença da implementação tradicional, resolveu-se variar a taxa de mutação em valores não recomendados pela literatura correlata para observar o seu comportamento. Taxas de mutação em 20% e 40% foram utilizadas, e com isso,foi possível observar o prejuízo na eficácia causado pela alta taxa de mutação. Uma alta taxa de mutação pode levar a área de busca para locais indesejados, atrasando a convergência ou diminuindo a precisão. Na Figura 24 podemos observar que a eficácia do resultado diminuiu de 100% para 70% e o tempo de execução apresentou uma média de 98,1 ms.

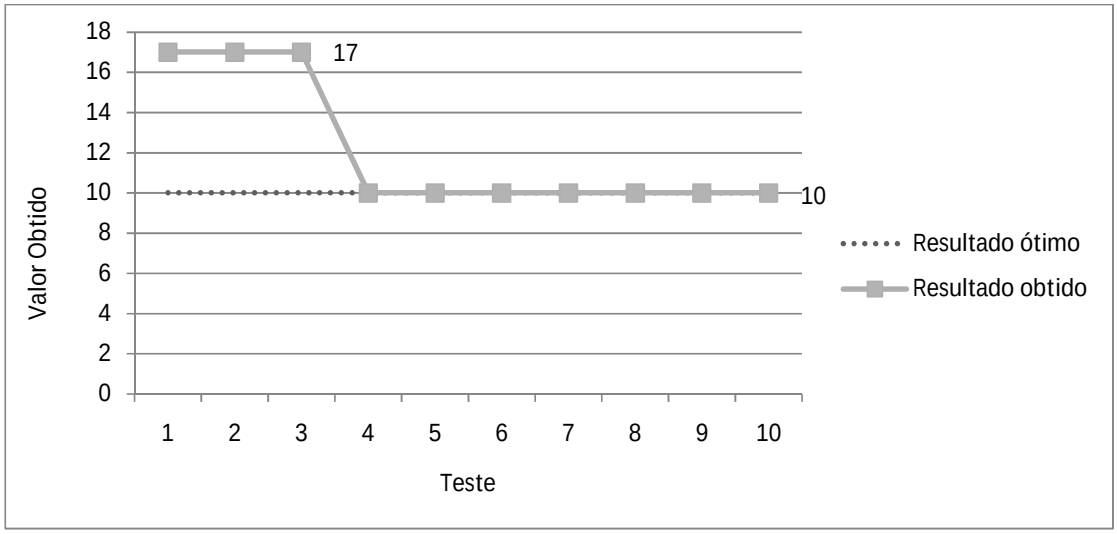

Figura 24 - Resultado obtido com25 vértices (20% de mutação e 95% de recombinação)

Para verificar se a eficácia continuaria piorando foi realizada a sequência de testes descrita na Figura 25. A frequência de acerto se manteve em 70% e houve pouca divergência entre o pior valor encontrado e o valor ótimo. O tempo de execução também manteve uma média de 103,5 ms. Assim, chega-se a conclusão de que não haverá melhoras nos resultados utilizando altas taxas de mutação, sendo assim, a taxa de mutação mais adequada seria a de 5%, como visto na Tabela 2.

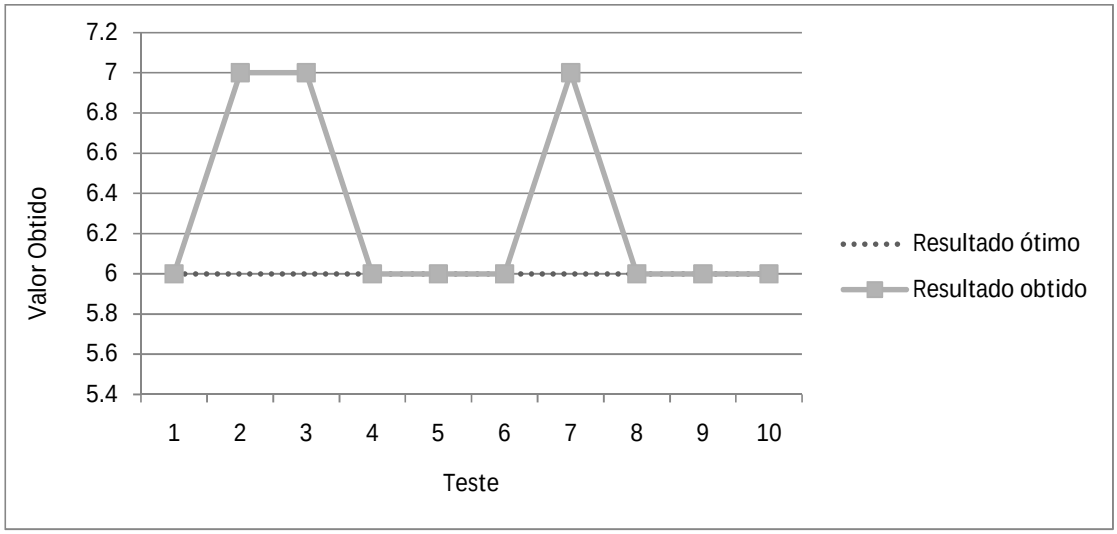

Figura 25- Resultado obtido com25 vértices (40% de mutação e 95% de recombinação)

4.1.3 Experimento 3 - Comportamento com a variação dos vértices do grafo

Para estudar o comportamento do Algoritmo Genético utilizando uma maior quantidade de vértices foram realizados testes utilizando grafos com 25, 50, 100, 150 e 200 vértices. A população inicial foi composta por 500 cromossomos e foram utilizadas 5% como taxa de mutação e 95% como taxa de recombinação. Para grafos com até 100 vértices a frequência de acerto do valor ótimo foi de 100% variando-se apenas a média do tempo de execução dos testes realizados, como pode ser visto na Tabela 3.

Tabela 3–Resultados obtidos com a variação da quantidade de vértices

Tamanho do grafo	Frequência de acerto do valor ótimo	Desvio padrão dos valores obtidos	Média de gerações necessárias	Média do tempo de execução (ms)
25 vértices	100%	0	3	63
50 vértices	100%	0	4	103,9
100 vértices	100%	0	4	166
150 vértices	0%	3,741	5	118,4
200 vértices	0%	11,294	5	124,7

Foi possível observar que com o aumento da quantidade de vértices, o volume de dados processados pelo algoritmo também aumentou, ocasionando em um maior custo computacional e perca na eficácia. Utilizando os mesmos parâmetros iniciais (tamanho da população inicial, taxas de mutação e recombinação) dos grafos menores, os grafos com 150 e 200 vértices apresentaram 0% de frequência de acerto do valor ótimo, mas ainda assim, o melhor valor obtido é um valor próximo do ótimo, como pode ser observado nas Figuras 19 e 20.Considerando que é recomendado utilizaruma população inicial a qual tenha um tamanho proporcional ao tamanho do cromossomo (MITCHELL, 1996), era esperado que esse aumento na quantidade de vértices ocasionasse em uma redução na eficácia, uma vez que não foi aumentado o tamanho da população inicial. Nota-se pelas Figuras 19 e 20 que com o aumento da quantidade de vértices, os resultados também pioraram gradativamente, pois o melhor e o pior resultado obtido no grafo com 200 vértices (Figura 27) foram mais distantes do que no grafo com 150 (Figura 28). Tal variação encontrada entre o pior e o melhor valor também podem ser analisada através do aumento do desvio padrão de 3,741 para 11,294, indicando que houve uma maior variação nos resultados obtidos com o grafo de 200 vértices.

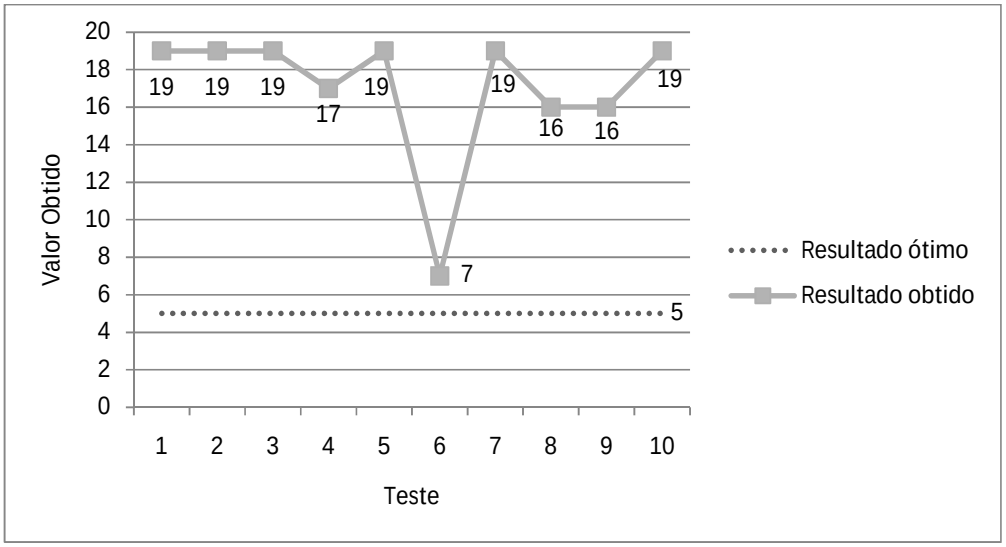

Figura 26 - Resultado obtido com 150 vértices (5% de mutação e 95% de recombinação)

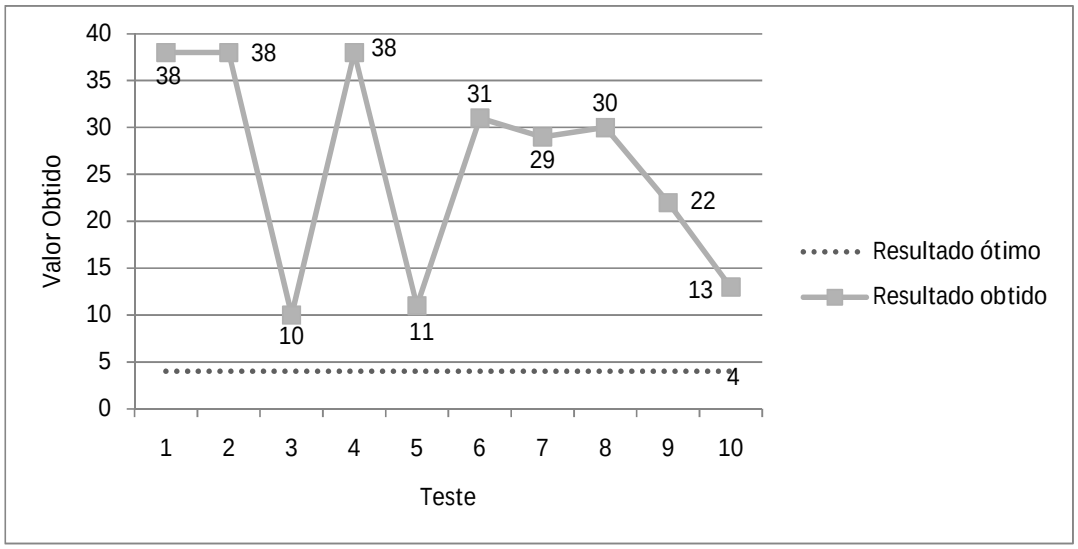

Figura 27 - Resultado obtido com 200 vértices (5% de mutação e 95% de recombinação)

4.1.4 Melhores Configurações Encontradas

Para obter os melhores resultados em um Algoritmo Genético é necessário encontrar o equilíbrio entre os valores das taxas de mutação e de recombinação (MITCHELL, 1996). Nesta subseção serão apresentados quais os valores foram utilizados nos parâmetros iniciais do AG para encontrar os melhores resultados.

Tabela 4 - Melhores resultados encontrados para o SP

Tamanho do grafo	Porcentagens (Mutação/Recombinação)	População Inicial	Frequência de acerto do valor ótimo	Média do tempo de execução (ms)
25 vértices	5/95	500	100%	63
50 vértices	5/95	500	100%	103,9
100 vértices	5/95	500	100%	166
150 vértices	5/95	5000	50%	7588
200 vértices	5/95	5000	40%	7243

Analisando os melhores resultados obtidos conclui-se que para o SP, os melhores resultados foram obtidos utilizando baixas taxas de mutação e altas taxas de mutação. Além disso, para grafos acima de 150 vértices a eficácia do algoritmo começou a diminuir consideravelmente, para otimizar a eficácia uma grande quantidade de cromossomos na populaçãoinicial foi necessária, levando a um grande aumento no custo computacional necessário para obter os resultados.

4.2 O Problemado Caixeiro Viajante (TSP)

O mesmo Algoritmo Genético foi utilizado para solucionar o problema do TSP, alterando-se apenas os operadores de recombinação e mutação. Nesta seção pretende-se discutir os primeiros experimentos na tentativa de solucionar o TSP. Baseando-se nos resultados já demonstrados na seção 4.1, optou-se por basear os testes em três experimentos: verificar a máxima eficácia com os operadores a 100%, verificar a eficácia com valores semelhantes ao da literatura correlata e uma discussão em voltada relação entre precisão/eficiência obtida. Em cada um destes testes o número de cromossomos na população inicial e as taxas de cruzamento e mutação foram alterados para avaliar qual combinação de parâmetros seria capaz de produzir o melhor resultado.

A seguir serão apresentados alguns dos testes realizados para cada experimento. Muitos dos testes serão apresentados em forma de tabela. Em cada tabela haverá duas células em destaque que indicam o melhor e o pior valor obtido para aquele experimento. Além disso, na última linha de cada tabela será apresentada a média aritmética do GAP e o desvio padrão dos valores obtidos. GAP é a diferença percentual entre o melhor valor conhecido (Opt) e o melhor valor obtido (Sol) pelo Algoritmo Genético (GOMES, 2008).Esta diferença pode ser definida como: 100 * (Sol – Opt)/Opt. Para realizar os testes que serão demonstrados a seguir

foi utilizado o grafo Oliver30 TSP que possui 30 vértices. As coordenadas deste grafo são provenientes do *"A study of permutation crossover operators on the travellingsalesman problem."*escrito por OLIVER, SMITH e HOLLAND (1987). Todos os testes foram executados com uma população inicial de 3000 cromossomos.

4.2.1 Experimento 1 – Utilizando valores semelhantes à literatura correlata

A literatura correlata sugere utilizar entre 0,5% e 1% como taxa de mutação e um intervalo de 75% até 95% para a taxa de recombinação (MITCHELL, 1996). Seguindo esse critério, foram gerados os seguintes testes.

Tabela 5 - Experimento 1 (a): 1% mutação e 75% recombinação

Valor esperado	Valor Obtido	GAP
420	**542**	29,04761905
420	**517**	23,0952381
420	**446**	6,19047619
420	**481**	14,52380952
420	**523**	24,52380952
420	**450**	7,142857143
420	**538**	28,0952381
420	**501**	19,28571429
420	**495**	17,85714286
420	**475**	13,0952381
	33,914	**18,28571429**

Tabela 6 - Experimento 1 (b): 1% mutação e 95% recombinação

Valor esperado	Valor Obtido	GAP
420	**438**	4,285714286
420	**510**	21,42857143
420	**507**	20,71428571
420	**521**	24,04761905
420	**450**	7,142857143
420	**488**	16,19047619
420	**468**	11,42857143
420	**537**	27,85714286
420	**483**	15
420	**482**	14,76190476
	31,159	**16,28571429**

Observando as Tabelas 2 e 3 pode-se concluir que o aumento na taxa de recombinação melhorou os resultados obtidos, fazendo a média da diferença percentual cair para 16,285 e o desvio padrão dos valores obtidos para 31,159.

Investigando as consequências de um aumento na frequência da mutação, foram realizados mais dois testes variando esta taxa.

Tabela 7 - Experimento 1 (c): 5% mutação e 95% recombinação

Valor esperado	Valor Obtido	GAP
420	476	**13,33333333**
420	486	**15,71428571**
420	497	**18,33333333**
420	497	**18,33333333**
420	495	**17,85714286**
420	486	**15,71428571**
420	471	**12,14285714**
420	520	**23,80952381**
420	513	**22,14285714**
420	486	**15,71428571**
	15,217	**17,30952381**

Tabela 8 - Experimento 1 (d): 10% mutação e 95% recombinação

Valor esperado	Valor Obtido	GAP
420	440	**4,761904762**
420	434	**3,333333333**
420	438	**4,285714286**
420	483	**15**
420	448	**6,666666667**
420	553	**31,66666667**
420	491	**16,9047619**
420	433	**3,095238095**
420	488	**16,19047619**
420	466	**10,95238095**
	37,777	**11,28571429**

Apesar dos resultados obtidos na Tabela 4 não se mostrarem satisfatórios, o Experimento 1 (d) inspirou uma investigação mais profunda devido a razoável melhora encontrada. Sendo assim, foram realizados mais testes aumentando o valor da mutação em intervalos maiores.

Tabela 9 - Experimento 1 (e): 25% mutação e 95% recombinação

Valor esperado	Valor Obtido	GAP
420	470	**11,9047619**
420	490	**16,66666667**
420	431	**2,619047619**
420	500	**19,04761905**
420	465	**10,71428571**
420	527	**25,47619048**
420	496	**18,0952381**
420	459	**9,285714286**
420	444	**5,714285714**
420	469	**11,66666667**
	28,497	**13,11904762**

Tabela 10 - Experimento 1 (f): 50% mutação e 95% recombinação

Valor esperado	Valor Obtido	GAP
420	429	**2,142857143**
420	450	**7,142857143**
420	449	**6,904761905**
420	484	**15,23809524**
420	490	**16,66666667**
420	459	**9,285714286**
420	442	**5,238095238**
420	511	**21,66666667**
420	454	**8,095238095**
420	508	**20,95238095**
	28,570	**11,33333333**

Fica claro pelas Tabelas 9 e 10 que o aumento ainda maior na taxa de mutação, não trouxe grandes melhoras nos resultados obtidos, pois a média dos GAPs obtidos permaneceram iguais ou piores ao do Experimento 1 (d). É interessante notar que no experimento com maior valor de mutação (Experimento 1(f)) foi encontrado o valor (célula em destaque) mais próximo do esperado entre os testes realizados até então. Além disso, o pior valor foi menor do que o pior obtido no experimento 1 (d), o que leva a concluir que apesar de não otimizar o conjunto de resultados, aumentar a taxa de mutação otimizou os extremos, reduzindo razoavelmente o desvio padrão dos valores obtidos, que foi reduzido de 37,777 para 28,570. Tal otimização pode claramente ser visualizada nas Figuras 28 e 29.

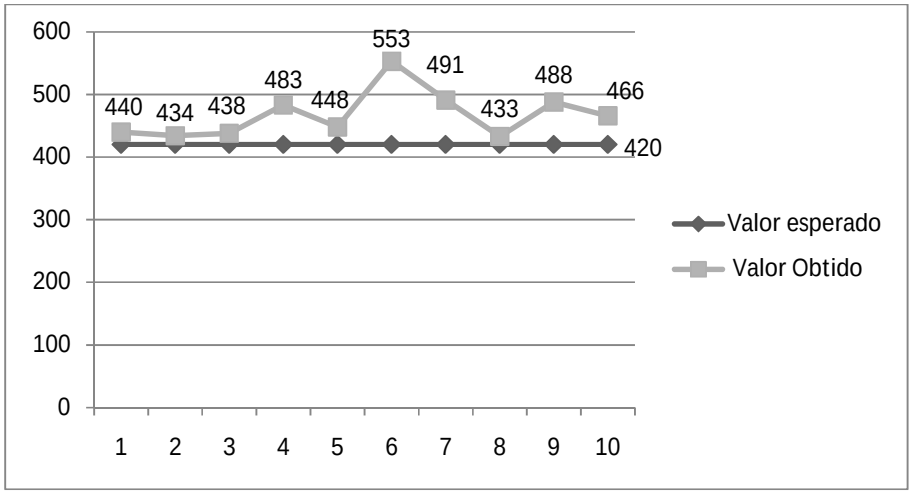

Figura 28- Experimento 1 (d): 10% mutação e 95% recombinação

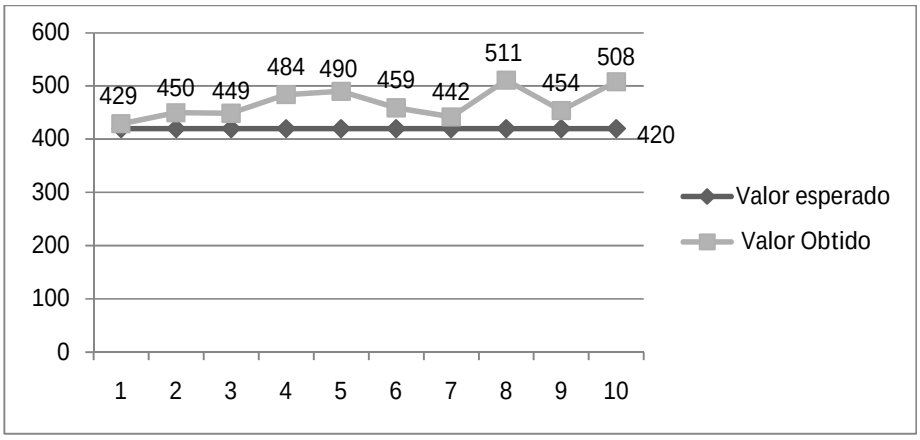

Figura 29 - Otimização alcançada com o aumento da mutação. Experimento 1 (f): 50% mutação e 95% recombinação

Aumentando ainda mais a taxa de mutação, foram obtidos os resultados do Experimento 1 (g).

Tabela 11 - Experimento 1 (g): mutação 75% e recombinação 95%

Valor esperado	Valor Obtido	GAP
420	438	**4,285714286**
420	468	**11,42857143**
420	453	**7,857142857**
420	454	**8,095238095**
420	449	**6,904761905**

420	455	**8,333333333**
420	435	**3,571428571**
420	464	**10,47619048**
420	438	**4,285714286**
420	423	**0,714285714**
	13,984	**6,595238095**

No Experimento 1 (g) a otimização foi considerável. Tanto o menor valor, quanto o GAP médio e o desvio padrão tiveram um decréscimo acentuado. Esta configuração se mostrou bastante promissora.

É possível concluir que, apesar da literatura correlata geralmente utilizar valores baixos de mutação, na abordagem deste problema o algoritmo proposto se comporta de maneira diferente. Segundo Mühlenbein (1992), "O poder da mutação é subestimado nos algoritmos genéticos tradicionais" e os testes realizados esta seção sugerem a utilização de valores maiores na taxa de mutação. Entretanto vale ressaltar que muito mais do que escolher entre um operador ou outro, deve se encontrar o equilíbrio entre mutação e recombinação em cada problema que se deseja trabalhar (MITCHELL, 1996).

4.2.2 Experimento 2 – Operadores a 100%

Como discutido na seção 4.2.1, houve uma melhora nos resultados obtidos com o aumento da ocorrência da mutação entre os indivíduos. Considerando esses resultados, foram realizados testes com ambos os operadores a 100%.

Tabela 12 - Experimento 2: ambos operadores a 100%

Valor esperado	Valor Obtido	GAP
420	441	**5**
420	458	**9,047619**
420	450	**7,142857**
420	455	**8,333333**
420	436	**3,809524**
420	439	**4,52381**
420	462	**10**
420	503	**19,7619**
420	424	**0,952381**
420	441	**5**
	21,573	**7,357143**

Com todos os indivíduos sofrendo mutação e recombinação durante cada geração obteve-se resultados melhores que os dos Experimentos 1 (e) e (f). O GAP médio reduziu consideravelmente e o melhor valor obtido aproximou-se bastante do valor esperado. Porém, o experimento 1 (g) apresenta resultados melhores em todos os quesitos: gerou o valor mais próximo do valor esperado, GAP médio menor, um menor tempo de execução (83,6 segundos, resultando em 28,4 segundos a menos do que o Experimento 2), e o menor desvio padrão dos valores obtidos, dessa forma, a configuração apresentada no Experimento 1 (g) continua sendo a mais promissora quando se deseja precisão no resultado final.

4.2.3 Experimento 3 – Comparativo entre custo computacional e eficácia

A partir da análise dos resultados das seções anteriores considerou-se a configuração do Experimento 1(g) (75% mutação e 95% recombinação) como a configuração que pode retornar os resultados mais eficazes. É importante ressaltar que os experimentos realizados anteriormente não garantem que o melhor valor sempre será gerado pelo algoritmo nesta configuração, porém, o esperado é que isso aconteça com esta configuração ou em alguma configuração similar.

Apesar do Experimento 1(g) apresentar a melhor taxa de eficácia ele não é a melhor configuração quando se trata de eficiência. Neste caso, entende-se eficiência como um equilíbrio entre bons resultados e tempo de execução. A seguir serão apresentados na Tabela 10, vários testes com as mais variadas configurações e as suas respectivas médias do tempo de execução, em destaque estão a melhor e a pior média de execução do algoritmo e o pior e melhor GAP.

Tabela 13 - Experimento 3: configuração de maior equilíbrio

Taxa de mutação (%)	**Taxa de recombinação (%)**	**GAP**	**Média do tempo de execução (s)**
1	40	21,2619	**59,589**
1	75	18,2857	**59,492**
1	95	16,2857	**64,981**
5	60	18,0476	**51,749**
5	75	12,5714	**58,88**
5	95	17,3095	**60,127**
10	60	14,5952	**50,795**
10	95	11,2857	**61,946**

25	95	13,119	**71,032**
50	95	11,3333	**70,062**
50	100	14,2142	**76,608**
75	95	6,5952	**83,6**
100	50	9,5952	**68,121**
100	100	7,3571	**112**
		13,7040	**65,515**

O valor médio do tempo de execução sugere algumas configurações como ponto de equilíbrio, tais configurações estão em destaque na Tabela 13. A configuração com a menor média de tempo de execução, 10% de mutação e 60% de recombinação, nos oferece um GAP razoável, estando apenas um pouco acima da média dos GAPs obtidos. Variando um pouco dessa configuração temos duas outras configurações que também podem ser muito promissoras, elas possuem 5% e 75%, e 10% e 95% de mutação e recombinação, respectivamente. Estas duas configurações também fornecem GAPs razoáveis a um acréscimo relativamente pequeno de tempo em relação à configuração que tem a menor média de tempode execução (10% e 60%).

4.3 Testesem GrafosdeVariadosTamanhose Inserçãode Novos Operadores

A partir dos experimentos realizados nas subseções anteriores foi possível compreender como o algoritmo proposto se comporta, entendendo com qual configuração de parâmetros é possível obter os resultados mais precisos, ou obter um melhor desempenho.

Nesta subseção o objetivo será apresentar testes do algoritmo proposto com variados operadores de recombinação/mutação em variados tamanhos de grafos. Os grafos utilizados neste experimento pertencem à TSPLib e ao Oliver30 TSP. Todos os grafos utilizados nos experimentos são simétricos, ou seja, a distância de um vértice A para um vértice B é igual a distância do vértice B para o vértice A. Por padrão, o final do nome de cada grafo vem acompanhando pela sua quantidade de vértices, por exemplo: Oliver30 possui 30 vértices.

As Tabelas 11, 12,13 e 14 apresentam os melhores resultados obtidos com determinada combinação de operadores.

Tabela 14- Recombinação Ordenada e Mutação Ordenada

Grafo	Valor esperado	Melhor Valor Obtido	GAP	Porcentagens (mutação/recombinação)	População Inicial
Ch130	6110	7217	18,1178396	75/95	3000
KroA100	21282	27629	29,8233248	75/95	3000
Eil76	538	572	6,31970260	75/95	3000
St70	675	737	9,185185185	75/95	3000
Berlin52	7542	8029	6,45717316	100/100	5000
Eil51	426	441	3,52112676	100/100	3000
Oliver	420	423	0,71428571	75/95	3000

Tabela 15 - Cruzamento OBX e Mutação Ordenada

Grafo	Valor esperado	Melhor Valor Obtido	GAP	Porcentagens (mutação/recombinação)	População Inicial
Ch130	6110	7707	26,1374795	100/100	3000
KroA100	21282	26657	25,2560849	75/95	3000
Eil76	538	617	14,6840148	75/95	3000
St70	675	771	14,22222222	75/95	3000
Berlin52	7542	8051	6,74887297	100/100	3000
Eil51	426	462	8,57142857	100/100	3000
Oliver	420	436	3,80952381	100/100	3000

Tabela 16 - Recombinação Ordenada e Mutação Order-Based

Grafo	Valor esperado	Melhor Valor Obtido	GAP	Porcentagens (mutação/recombinação)	População Inicial
Ch130	6110	10930	78,8870703	100/100	3000
KroA100	21282	45160	112,198101	75/95	3000
Eil76	538	720	33,8289962	75/95	3000
St70	675	880	30,37037037	75/95	3000
Berlin52	7542	8824	16,9981437	100/100	3000
Eil51	426	501	17,6056338	100/100	3000
Oliver	420	441	5	100/100	3000

Tabela 17 - Cruzamento OBX e Mutação Order-Based

Grafo	Valor esperado	Melhor Valor Obtido	GAP	Porcentagens (mutação/recombinação)	População Inicial
Ch130	6110	15308	150,5400982	100/100	3000
KroA100	21282	54574	156,4326661	100/100	3000
Eil76	538	838	55,76208178	100/100	3000

St70	675	898	33,03703704	75/95	3000
Berlin52	7542	9146	21,26756828	100/100	3000
Eil51	426	575	35,47619048	100/100	3000
Oliver	420	447	6,428571429	100/100	3000

Comparando os resultados obtidos nas Tabelas 11, 12, 13, e 14 pode-se concluir que, entre os operadores testados, os melhores são a recombinação ordenada e a mutação ordenada, pois foi esta configuração que apresentou os resultados mais próximos do esperado na maioria dos grafos. A única exceção foi o KroA100 utilizando cruzamento OBX e a mutação ordenada (Tabela 15).

5. CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um Algoritmo Genético (AG) capaz de planejar rotas para robôs autônomos. Este algoritmo foi testado utilizando os problemas do caminho mínimo (SP) e o do caixeiro viajante (TSP). Para cada problema foi realizada uma sequência de testes na intenção de investigar o comportamento do algoritmo proposto.

Os experimentos demonstraram que para grafos entre 25 e 100 vértices a precisão do algoritmo é satisfatória para o problema do SP, enquanto que para o TSP, os melhores resultados foram encontrados nos grafos entre 30 e 52 vértices. Tanto no SP, quanto no TSP, existiram configurações que trouxeram soluções próximas do esperado. Observou-se também que, quanto maior o grafo, maior deve ser o tamanho da população inicial visando aumentar a representação do espaço de busca, embora isso implique em um aumento considerável no custo computacional do algoritmo.

A vantagem do AG está em permitir o tratamento de problemas NP-difíceis, como o problema do caixeiro viajante e a flexibilidade de trabalhar com mapas dinâmicos, onde a modificação de um único obstáculo no mapa não implicaria em executar o algoritmo inteiro novamente. Bastaria modificar a matriz de adjacência, gerar um novo valor de aptidão para o antigo melhor indivíduo e gerar uma nova população inicial com este indivíduo presente nela.

Considerando-se os resultados obtidos conclui-se que ainda é necessário uma otimização no algoritmo proposto em relação à eficiência do algoritmo para problemas TSP. Exigir que um robô móvel espere mais de 1 minuto até receber a sua rota, não é viável em um sistema de tempo real. Uma vez realizada a otimização, pretende-se utilizar o algoritmo proposto em um robô real associado a uma técnica de construção de mapas.

REFERÊNCIAS

AHN, W. C.; RAMAKRISHNA, S. R. A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. In: IEEE Transactions on Evolutionary Computation, v. 6, 2002.

ARAÚJO, S. A.; LIBRANTZ, A. F. H. Navegação Autônoma de Robôs. Exacta, São Paulo, v. 4, n. especial, p. 81-83, 2006.

BERTONI, F. C.Computação Evolutiva. FAFEM: Faculdades da Fundação de Ensino de Mococa. Notas de aula. 2006.

BORGES, A. G. Planejamento Probabilístico de Rotas. Disponível em: <http://lara.unb.br/~gaborges/pesquisa/robotica/movel_prm/index.htm>. Acesso em: 13 ago. 2012.

CARVALHO, A. P.L. Algoritmos genéticos. Disponível em: <http://www.icmc.usp.br/~andre/research/genetic/index.htm#oper>. Acesso em: 18 jul. 2012.

DIAS, M. A. Aula 11 – Espaço de Configuração e Planejamento de Caminhos. UEFS: Universidade Estadual de Feira de Santana. Notas de Aula. 2010.

FRACASSO,P. T. *Análise de técnicas para amostragem e seleção de vértices no planejamento probabilístico de mapa de rotas*. 2008. 107f. Dissertação (Mestrado em Engenharia Elétrica)-Universidade de São Paulo, São Paulo, 2008.

GOMES, N. M.; SENNE, E. L. F. Um Algoritmode Busca Tabupara Soluçãode Problemasde Localizaçãode P-Medianas: A integração de cadeias produtivas com a abordagem da manufatura sustentável. In: Encontro Nacionalde Engenhariade Produção, v. 28, p. 8, 2008, Rio de Janeiro.

HERRERA, F. P. V. *Genetic algorithms for the traveling salesman problem*. Redmond: Digipen Institute Of Technology, 2010.

JONG, K. A. *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. 1975. 271f. Tese (Doutorado em Ciência da Computação e Comunicação)-University Of Michigan, Michigan, 1975.

KORTENKAMP, D., BONASSO, R. P. e MURPHY, R. Artificial Intelligence and Mobile Robots. Massachusetts: The MIT Press, 1998.

LACERDA, E. G. M.; CARVALHO, A. C. F. Introdução aos algoritmos genéticos. In: XIX Congresso Nacional da Sociedade Brasileira de Computação. Rio de Janeiro, 1999. Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação, p. 119-124, 1999.

LATOMBE, J.C.*Robot Motion Planning*. Norwell: Kluwer Academic Publishers, 1991.

LUGER, G. *Inteligência artificial: estruturas e estratégias para a resolução de problemas complexos*. 4. ed. Porto Alegre: Bookman, 2004.

MANDOW, A. et al. The autonomous mobile robot aurora for greenhouse operation. IEEE Robotics and Autonomous Magazine, v. 3, n. 4, p. 18-28, 1996.

MARTIN, L. Multisensor inspection&characterization robot for small pipes. Technology Development Data Sheet, Federal Energy Technology Center, p. 161-162, 1997.

MITCHELL, M. *An Introduction to Genetic Algorithms.* Massachusetts: The MIT Press, 1996.

Mühlenbein,H.How genetic algorithms really work: mutation and hill-climbing. In R. Männer and B. Manderick, Parallel problem solving from nature, 1992, Amsterdam.

MURPHY, R. R. Marsupial and shape-shifting robots for urban search and rescue.IEEE Intelligent Systems, 2000.

NETO, R. T.; COELHO, L. S. Planejamento de Rotas para Robôs de Inspeção Usando uma Nova Abordagem de Swarm Intelligence. In: Workshop em Nanotecnologia e Computação Inspirada na Biologia,2004, Rio de Janeiro. Proceedings do Nanobio, p. 1-2, 2004.

NETO, R. T.; COELHO, L. S. Planejamento de Rotas para Robôs de Inspeção Usando um Algoritmo Híbrido de Colônia de Formigas e Algoritmo Cultural. In: VII Simpósio Brasileiro de Automação Inteligente/II Latin-American Robotics Symposium, 2005, São Luís.

NISSOUX, C.; SIMEON, T.; LAUMOND, J.-P. Visibility-based probabilistic roadmaps. In: Proceedings of the 1999 IEEE International Conference on Intelligent Robots and Systems (IROS'93),1999, South Korea. v. 3,p. 1316–1321, 1999.

OLIVER, M.;SMITH, D. J.; HOLLAND, J. R. C. A Study of Permutation Crossover Operators on the Travelling Salesman Problem**.** In:Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, p. 224-230, 1987.

PIRES, M. G. *Abordagem Neuro-Genética para Mapeamento de Problemas de Conexão em Otimização Combinatória*. 2009. 103f. Tese (Doutorado em Engenharia Elétrica)-Escola de Engenharia de São Carlos, São Carlos, 2009.

REINELT, G. TSPLIB 95**.** Heidelberg: Universität Heidelberg.Institut für Angewandte Mathematik,p. 17, 2008.

REZENDE F. A. V. S., ALMEIDA R. M. V., NOBRE F. F.. Diagramas de Voronoi para a definição de áreas de abrangência de hospitais públicos no Município do Rio de Janeiro. Rio de Janeiro, Cad. Saúde Pública, 2000.

SALANT, M. A. Introdução à robótica. São Paulo: McGraw-Hill, 1990.

WANGENHEIM, V. A. Introdução a Reconhecimento de Padrões. Disponível em: <http://www.inf.ufsc.br/~patrec/intro.html>. Acesso em: 13 ago. 2012.

WU, W.; RUAN, Q. A gene-constrained genetic algorithm for solving shortest path problem. In: International Conference Signal Processing, 2004, Beijing. ICSP Proceedings, p. 1-2, 2004.