X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

Análise de Algoritmos Genéticos Aplicados a Robôs em Ambientes Dinâmicos via Modelo Exato

Renato Tinós
Departamento de Computação e Matemática, FFCLRP, Universidade de São Paulo (USP)
rtinos@ffclrp.usp.br

Resumo – O comportamento do Algoritmo Genético (AG) quando aplicado no controle de um robô móvel em ambientes dinâmicos é investigado aqui de um ponto de vista teórico. Para tal, a abordagem por sistemas dinâmicos, conhecida como modelo exato e na qual o AG é visto com um sistema dinâmico discreto, é utilizada. Aqui, um robô simples sujeito a falhas é estudado. As mudanças provocadas pelas falhas no espaço de fitness do problema são analisadas, assim como o comportamento dinâmico do sistema em simulações. Observa-se que as mudanças provocadas pelas falhas geram problemas de otimização evolutiva dinâmica lineares e com permutação. Ainda, nota-se que o estudo dos estados metaestáveis do sistema dinâmico neste caso ajuda a explicar o desempenho do algoritmo.

Palavras-chave – Algoritmos Genéticos, Sistemas Dinâmicos, Teoria de Algoritmos Evolutivos, Robótica.

Abstract – In this work, the behavior of a Genetic Algorithm (GA) used to control a mobile robot in a dynamic environment is theoretically investigated. For this purpose, the dynamical system approach, known as exact model and where the GA is viewed as a discrete dynamical system, is employed. Here, a simple robot with faults is studied. The changes caused by the faults in the fitness landscape of the problem are analysed, as well the dynamical behavior of the system in simulations. The faults generate dynamic evolutionary optimization problems with linear transformations and permutation. One can still observe that the study of the metastable states of the dynamical systems helps, in this case, the undestanding of the algorithm's performance.

Keywords – Genetic Algorithms, Dynamical Systems, Theory of Evolutionary Algorithms, Robotics.

1. INTRODUÇÃO

Recentemente, diversas aplicações envolvendo robôs controlados ou projetados através de técnicas de Computação Evolutiva têm sido relatadas [1]. A principal motivação do uso de Algoritmos Evolutivos (AEs) em Robótica advém do fato de serem um meio interessante para prover adaptabilidade aos robôs em ambientes não-estruturados, flexíveis, e/ou parcialmente desconhecidos [2]. Interessantemente, neste caso, a conexão entre Robótica e Biologia não têm sentido único pois robôs podem ser uma importante ferramenta para a investigação e teste de modelos cognitivos, evolutivos, e comportamentais de sistemas biológicos.

As pesquisas em robôs controlados ou projetados através de AEs, chamados de Robôs Evolutivos, têm se concentrado quase que exclusivamente em robôs aplicados em ambientes estacionários. No entanto, diversos problemas exigem que o robô se adapte às mudanças no ambiente. Em geral, tais mudanças ocasionam alterações nas restrições das soluções ou na função de aptidão ($fitness$) associada ao problema durante o processo de otimização. Diversos motivos ocasionam as mudanças na prática, tais como a ocorrência de falhas, mudanças nas características do ambiente, problemas decorrentes da cooperação e competição entre robôs, mudança de tipo de robô, transferência de soluções simuladas para robôs reais e mudanças de estratégias. Entre as soluções envolvendo AEs aplicadas em Robôs Evolutivos em ambientes dinâmicos, podem ser citadas como exemplo: a adição de ruído [3], a avaliação da função de $fitness$ para diferentes condições do ambiente [4], a combinação com outros métodos de aprendizado [2], a manutenção do nível de diversidade por meio de Imigrantes Aleatórios [5], e a adaptação das distribuições de mutação [6]. Nestas aplicações, assim como quando AEs são aplicados em diversos outros problemas de otimização dinâmica, o desempenho do algoritmo é testado experimentalmente, ou seja, executando o algoritmo e analisando o desempenho obtido.

De acordo com o conhecimento do autor, não existem na literatura trabalhos teóricos envolvendo AEs aplicados em robôs em ambientes dinâmicos, ou seja, trabalhos que analisem o funcionamento do AE quando aplicado em robôs usados em ambientes dinâmicos de um ponto de vista teórico. Aliás a falta de investigações teóricas acerca do funcionamento do AEs não é exclusividade da área de robótica, mas ocorre também em uma infinidade de outras áreas. De forma geral, o número de investigações envolvendo teoria em AEs não acompanha o crescimento que se verifica no número de trabalhos envolvendo a aplicação ou o desenvolvimento de novos AEs ou operadores. Entretanto, poderia se argumentar, a análise teórica deveria ser geral, independente da área de aplicação, já que algumas das ferramentas de análise teórica não levam em conta as particularidades do espaço de busca associado ao problema. No entanto, tais ferramentas têm tido sucesso limitado em explicar o funcionamento dos AEs [7]. Por outro lado, as ferramentas em que as particularidades do espaço de busca devem ser conhecidas, como a análise por Cadeias de Markov ou através da Teoria de Sistemas Dinâmicos, têm tido mais exito, apesar de diversas restrições, sendo a principal delas a necessidade de tratar problemas simples devido ao tamanho dos modelos utilizados.

Particularmente no caso do uso da modelagem de Algoritmos Genéticos (AGs) como sistemas dinâmicos, desenvolvida principalmente por Vose [8] e cujo modelo é conhecido como Modelo Exato, apesar de demandar uma grande quantidade de equações para acompanhar todas as possíveis soluções representadas pelos indivíduos do AG, o seu uso é bastante interessante por permitir uma completa descrição da dinâmica populacional do AG [7]. Em [9], os autores propuseram o uso do Modelo Exato para estudar o AG em ambientes dinâmicos criados pelo gerador de problemas XOR [10]. Posteriormente, esta análise foi estendida para problemas dinâmicos criados pelo método descrito em [11] e para o problema da mochila 0-1 dinâmico [12].

Neste trabalho, o enfoque por sistema dinâmicos é utilizado para investigar o comportamento populacional de um AG aplicado para o controle de um robô móvel simples em um ambiente dinâmico. A escolha de uma aplicação simples envolvendo robôs móveis é justificada pela necessidade de uma descrição completa da dinâmica populacional do AG. Salienta-se que este é um trabalho teórico, sendo que, ao invés de executarmos o AG, este é simulado através de seu modelo exato. Isso permite estudarmos o comportamento dinâmico da população do AG, ajudando na análise do desempenho obtido na prática.

Este artigo está estruturado da seguinte forma: na Seção 2, o modelo exato do AG considerando-se ambientes estacionários é brevemente introduzido; na Seção 3, tal modelo é estendido para ambientes dinâmicos; o problema envolvendo um modelo de robô simples é apresentado na Seção 4; na Seção 5, o problema envolvendo o robô é descrito e analisado usando as definições descritas na Seção 3; simulações envolvendo o robô descrito na Seção 4 são apresentadas e analisadas na Seção 6; finalmente, as conclusões e perspectivas futuras do trabalho são apresentadas na Seção 7.

2 MODELO EXATO DO AG

Em [8], Vose propõe o modelo exato do AG simples no qual este algoritmo é descrito e analisado como um sistema dinâmico discreto. No AG simples, a codificação binária é utilizada, sendo que cada indivíduo da população representa uma solução $\mathbf{x} \in \{0,1\}^l$ do problema. No modelo exato, todas as possíveis soluções candidatas do problema de otimização são representadas em um espaço discreto com $n = 2^l$ dimensões. Assim, a população atual do AG pode ser descrita como um vetor n-dimensional no qual cada elemento define a proporção de cada possível solução na população, ou seja, $\mathbf{p} = \mathbf{v}/N$, sendo que o k-ésimo elemento de $\mathbf{v}$ indica o número de cópias da k-ésima possível solução candidata na população com tamanho N. Como a soma dos elementos de $\mathbf{p}$ é igual a 1, o vetor de população pode ser descrito como membro de um simplex Λ, i.e.,

$$\Lambda = \left\{ \mathbf{p} \in \mathbb{R}^n : p_k \geq 0, \text{ para } k = 0, 1, \ldots, n-1 \text{ e } \sum_{k=0}^{n-1} p_k = 1 \right\}, \tag{1}$$

sendo p_k o k-ésimo elemento do vetor de população $\mathbf{p}$. No simplex Λ, os vértices representam populações com cópias de uma única solução. No modelo exato, a evolução da população de soluções ao longo da execução do AG é descrita como uma trajetória no simplex Λ, sendo que os vetores de população são usados para descrever a distribuição de probabilidades dos indivíduos no espaço de busca. Assim, o operador geracional $\mathcal{G} : \Lambda \to \Lambda$ é definido por $\mathcal{G}(\mathbf{p}) = \mathbf{p}'$, sendo $\mathbf{p}'$ a distribuição de probabilidades amostrada para gerar a população posterior a $\mathbf{p}$, i.e., $\mathcal{G}(\mathbf{p})$ é a próxima geração esperada [8]. O vetor $\mathcal{G}(\mathbf{p})$ descreve a média sobre todas as possíveis populações da próxima geração com variância inversamente proporcional ao tamanho da população N, sendo que no limite $N \to \infty$, conhecido como caso de população infinita (modelo exato), a variância se aproxima de zero e, como consequência, a trajetória da população no simplex Λ pode ser deterministicamente descrita. Assim, podemos definir o AG da sequinte forma:

Definição 1 *O **AG** é um sistema dinâmico discreto definido pela aplicação sucessiva da regra:*

$$\mathbf{p}(t) = \mathcal{G}\big(\mathbf{p}(t-1), t\big), \tag{2}$$

sendo $\mathcal{G}(., t) : \Lambda \to \Lambda$ o operador geracional (mapa) na geração $t \geq 1$, e $\mathbf{p}(t)$ a população esperada em t.

Para o caso estacionário, $\mathcal{G}(., t) = \mathcal{G}(.)$ para todo $t \geq 1$, fazendo com que a trajetória da população seja dada por $\mathbf{p}(0)$, $\mathcal{G}\big(\mathbf{p}(0)\big)$, $\mathcal{G}^2\big(\mathbf{p}(0)\big)$, ... Assim, para o caso de população infinita:

$$\mathbf{p}(t) = \mathcal{G}^t\big(\mathbf{p}(0)\big). \tag{3}$$

Para o AG simples com mutação e seleção proporcional, o operator geracional é computado por:

$$\mathcal{G} = \mathcal{U} \circ \mathcal{F}, \tag{4}$$

sendo o operador de seleção proporcional igual a:

$$\mathcal{F}(\mathbf{p}) = \frac{F\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}} \tag{5}$$

na qual $\mathbf{f}$ é o vetor com o fitness de cada solução $\mathbf{x}_i$ pertencente ao espaço χ, e $F = \mathrm{diag}(\mathbf{f})$ é uma matriz diagonal gerada a partir de $\mathbf{f}$. O vetor de fitness $\mathbf{f}$ fornece informação sobre a estrutura do espaço de busca. Já o operador de mutação é dado por:

$$\mathcal{U}(\mathbf{p}) = U\mathbf{p}, \tag{6}$$

sendo U a matriz de mutação. Nesta matriz, cada elemento U_{ij} indica a probabilidade de gerar o i-ésimo elemento de χ (i-ésima solução candidata possível) a partir do j-ésimo elemento deste mesmo espaço. Através das equações 4-6, é possível escrever o operador geracional como:

$$\mathcal{G}(\mathbf{p}) = \frac{UF\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}}. \tag{7}$$

A análise da Eq. 7 pode fornecer informações importantes para entendermos o comportamento do AG simples. Os pontos fixos de $\mathcal{G}$, i.e., pontos onde $\mathcal{G}(\mathbf{y}) = \mathbf{y}$, são dados pelos autovetores de UF. Para cada autovetor $\mathbf{y}$, um autovalor $\mathbf{f}^{\mathrm{T}}\mathbf{y}$, correspondente ao fitness médio de $\mathbf{y}$, pode ser computado. Como UF tem somente valores positivos, existe um único autovetor dentro de Λ, associado ao autovalor com o maior valor absoluto [7]. Como consequência, todas as trajetórias em Λ convergem para este ponto fixo, i.e., o sistema dinâmico é assintoticamente estável [8, 13]. Os autovetores restantes não são propriamente pontos fixos, já que, por exemplo, podem ser localizados fora do simplex. Contudo, eles representam um papel importante para o processo evolutivo já que podem mudar a trajetória da população no simplex ao criar estados metaestáveis, podendo até aprisionar populações finitas por várias gerações [14].

Uma análise semelhante pode ser feita quando o crossover padrão é adicionado ao operador geracional. Entretanto, neste caso, os estados metaestáveis devem ser obtidos através de um processo mais complicado que envolve a linearização das equações [8]. Por simplicidade, iremos nos ater aqui ao caso do AG simples com mutação e seleção proporcional, i.e., sem crossover.

3 OTIMIZAÇÃO EVOLUTIVA DINÂMICA

A seguir, mudanças e problemas de otimização dinâmica são definidos [12] no contexto do modelo exato do AG.

Definição 2 *Considere um AG (Definição 1) no qual o operador geracional $\mathcal{G}(.,t)$ na geração t é hiperbólico para todo $t \geq 1$ e possui n_f pontos fixos (estados metaestáveis), i.e., $\mathbf{y}_i(t) = \mathcal{G}\big(\mathbf{y}_i(t), t\big)$ para $i = 1, \ldots, n_f$, sendo $\mathbf{y}_i(t)$ o i-ésimo ponto fixo (estado metaestável) de $\mathcal{G}(.,t)$. Uma* ***mudança*** *em um AG ocorre na geração t quando $\mathbf{y}_i(t) \neq \mathcal{G}\big(\mathbf{y}_i(t-1), t\big)$ para pelo menos um $\mathbf{y}_i(t)$ para $i = 1, \ldots, n_f$, i.e., pelo menos um ponto fixo de $\mathcal{G}(.,t)$ não é preservado.*

Pode-se observar que nem todas as modificações no operador geracional $\mathcal{G}$ (Eq. 7 para o AG simples sem crossover) pode ser definida como mudança de acordo com a Definição 2. Outra importante observação é que nem toda mudança causa necessariamente uma alteração na trajetória das populações. Observa-se que aqui, consideramos que mudanças somente ocorrem entre duas aplicações consecutivas do operador geracional. Desta forma, Problema de Otimização Dinâmica (DOP - *Dynamic Optimization Problems*) no contexto de AGs pode ser definido de acordo com o enfoque por sistemas dinâmicos.

Definição 3 *Um* ***Problema de Otimização Dinâmica DOP*** *no contexto de AGs (Definição 1) é um problema de otimização onde pelo menos uma mudança (Definição 2) ocorre durante o processo evolucionário.*

Já que o operador geracional é modificado após uma mudança, a Eq. 3 não é mais válida para toda geração t em um DOP. Antes de apresentarmos a nova equação para a dinâmica do AG em um DOP, vamos definir ciclo de mudança.

Definição 4 ***Ciclo de mudança*** *é uma série de aplicações de um mesmo operador geracional entre duas mudanças consecutivas.*

A duração d_e de um ciclo de mudanças e é o número de gerações consecutivas deste ciclo. Se o ciclo de mudanças e começa na geração t, então

$$\mathcal{G}(.,t) = \mathcal{G}(,t+1) = \mathcal{G}(.,t+2) = \ldots = \mathcal{G}\big(.,t+d_e-1\big), \tag{8}$$

sendo $d_e > 0$. Em abuso de notação, definimos agora $\mathcal{G}(.,e)$ como o operador geracional no ciclo de mudanças e. Desta forma, para o caso de população infinita, a população na geração t é agora dada por:

$$\mathbf{p}(t) = \mathcal{G}^{(t-\sum_{i=1}^{e-1} d_i)}(.,e) \circ \mathcal{G}^{d_{e-1}}(.,e-1) \ldots \mathcal{G}^{d_2}(.,2) \circ \mathcal{G}^{d_1}\big(\mathbf{p}(0),1\big), \tag{9}$$

sendo $e > 0$. Nota-se que o DOP pode ser visto como uma sequência de processos estacionários definidos pelos ciclos de mudança, nos quais a população inicial no i-ésimo ciclo de mudança é a população final gerada no ciclo de mudança $i-1$. O valor mínimo de d_i é uma geração, que é o caso em que as mudanças são contínuas, ao passo que o máximo valor de d_i é igual ao índice da geração atual, que é o caso em que o problema é estacionário (até a geração atual) e, como consequência, a Eq. 9 reproduz a Eq. 3. No AG simples aqui tratado, uma mudança altera pelo menos um dos termos do operador geracional definido pela Eq. 4. Geralmente, mudanças na matriz de mutação U são relacionadas a mudanças no algoritmo, e.g., quando a taxa de mutação é aumentada durante o processo evolucionário. Entretanto, algumas mudanças podem também modificar os operadores de reprodução, e.g., quando existem mudanças nas restrições do problema e algumas soluções não são mais permitidas. Contudo, a comunidade que investiga DOPs no contexto de AEs em geral não está interessada em problemas deste tipo, sendo os DOPs que interessam aqueles em que há mudança na superfície de fitness e que são a seguir definidos.

Definição 5 *Um* ***DOP com mudanças na superfície de fitness*** *é um DOP (Defição 3) no qual a superfície de fitness do problema é alterada por uma mudança (Definição 2) pelo menos uma vez durante o processo evolucionário, i.e., $\mathbf{f}(e) \neq \mathbf{f}(e-1)$, em pelo menos um ciclo de mudança $e > 1$, sendo $\mathbf{f}(e)$ o vetor de fitness no ciclo de mudança e.*

A Definição 5 é bastante geral, sendo que nem todos os DOPs com mudanças na superfície de fitness têm atraído a atenção da comunidade de AEs. Se a nova superfície de fitness não mantém pelo menos um mínimo de similaridade em relação a superfícies antigas, o melhor procedimento é reiniciar o processo de busca. Entretanto, se a nova superfície de fitness após a mudança pode ser relacionada com superfícies antigas, então o conhecimento adquirido durante o processo de busca das soluções antigas pode de alguma forma ser utilizado na busca pelas novas soluções [15]. Um caso especial de DOPs com mudanças na superfície de fitness que tem atraído a atenção de pesquisadores é o de DOPs com dependência do último ambiente.

Definição 6 *Considere um DOP com mudanças na superfície de fitness (Definição 5) no qual a superfcíe de fitness no ciclo de mudança e seja dependente somente da superfície de fitness no ciclo $e-1$, i.e., $\mathbf{f}(e) = h\big(\mathbf{f}(e-1), e\big)$, sendo $e > 1$ e $h(., e)$ uma função real. Tal DOP é chamado* ***DOP com dependência do último ambiente****.*

Vários tipos de DOPs com dependência do último ambiente podem ser definidos, entre eles, os DOPs lineares.

Definição 7 *Um* ***DOP linear*** *é um DOP com dependência do último ambiente (Definição 6) no qual a superfície de fitness no ciclo de mudança $e-1$ é modificada de acordo com uma transformação linear, i.e., $\mathbf{f}(e) = \mathbf{A}(e)\mathbf{f}(e-1)$, sendo $\mathbf{A}(e) \in \mathbb{R}^{n\times n}$.*

Um caso especial de DOPs linear é definido a seguir.

Definição 8 *Um* ***DOP com permutação*** *é um DOP linear (Definição 7) no qual a superfície de fitness no ciclo de mudança $e-1$ é modificada de acordo com uma matriz de permutação, i.e., $\mathbf{f}(e) = \sigma(e)\mathbf{f}(e-1)$, sendo $\sigma(e)$ uma matriz de permutação no ciclo e, a qual mapeia um elemento em uma posição do vetor $\mathbf{f}(e-1)$ para um elemento em uma posição do vetor $\mathbf{f}(e)$.*

Em um DOP com permutação (Definição 8), os valores de fitness são preservados no espaço de busca, i.e., o vetor de fitness é apenas reordenado. Em [9], os autores apresentam um estudo sobre DOPs criados através do gerador de problemas binários dinâmicos XOR [10], que permite tornar dinâmico qualquer problema estacionário de otimização binária. Este gerador tem sido bastante utilizado para comparar o desempenho de diferentes AEs em DOPs, de modo que a descrição das propriedades dos ambientes produzidos pelo gerador XOR é de fundamental importância para entendermos os resultados obtidos pelos algoritmos. Os autores em [9] mostraram que os DOPs produzidos pelo gerador XOR são do tipo DOPs com permutação (Definição 8), com um tipo de matriz de permutação $\sigma(e) = \sigma_{\mathbf{k}(e)}$ que mapeia o elemento na posição $\mathbf{i}$ do vetor $\mathbf{f}(e-1)$ em um elemento na posição $\mathbf{i} \oplus \mathbf{k}(e)$ do vetor $\mathbf{f}(e)$, sendo $\oplus$ o operador XOR, ou adição módulo 2. O vetor $\mathbf{i} \in \{0,1\}^l$ indica a posição do elemento no vetor de fitness. O vetor $\mathbf{k}(e) \in \{0,1\}^l$ controla a permutação dos elementos do vetor de fitness.

Um possível questionamento acerca do gerador XOR é sobre a relação dos ambientes gerados com DOPs reais. Para isso, deve-se questionar se DOPs reais podem ser do tipo DOP com permutação, como definido anteriormente.

4 ROBÔ MÓVEL SIMULADO

Neste artigo, um robô móvel com um único sensor frontal é simulado em um ambiente com 4 paredes. Considera-se que o robô possa ocupar 9 posições (quadrados) deste ambiente. O sensor frontal produz um sinal igual a $I = 1$ caso o robô esteja de frente para uma parede, e $I = 0$ caso contrário. O objetivo é encontrar através do AG uma lei de controle que permita o robô navegar pelo ambiente sem que se choque com as paredes durante o período de 10 iterações. O robô é controlado por uma máquina de estado finito (automato) com l bits que determinam o que o robô deve fazer para cada possível condição de estado interno e entrada proveniente do sensor. Dois modelos são aqui estudados, um com $l = 4$ bits e outro com $l = 8$ bits.

No primeiro modelo, o robô pode executar apenas duas ações: seguir em frente (0), ou seja mover-se para próxima posição (quadrado) localizada em sua frente; ou girar no sentido horário (1), ou seja, mudar a direção sem que a posição seja alterada. Neste caso, além do sinal do sensor (I), o robô possui um bit de estado (S), ou memória interna, que indica se o último movimento foi uma rotação ($S = 0$) ou não ($S = 1$). Desta forma, o automato tem 4 bits que definem qual a ação para cada possível combinação de estado/posição, ou seja, (I, S) = { 0,0 ; 0,1 ; 1,0 ; 11 }. Por exemplo, se o automato é dado por $\mathbf{x} = [0, 0, 0, 1]^{\mathrm{T}}$, o robô irá girar apenas quando ele seguiu em frente na última iteração e encontrou uma parede em frente da posição atual. Assim, o espaço de busca é composto por apenas $n = 16$ possíveis soluções. Já no segundo modelo, 4 ações são permitidas: ficar parado (00), girar no sentido horário (01), girar no sentido anti-horário (10) e seguir em frente (11). Desta forma, o automato tem agora 8 bits, sendo dois para cada um das 4 combinações possíveis de posição/estado. Portanto, o espaço de busca é composto por $n = 256$ soluções possíveis.

Em ambos os modelos, o fitness é dado pelo número de posições ocupadas pelo robô (ou seja, o número de vezes que ele segue em frente mais 1) até que este se choque com uma parede ou, em caso contrário, até um limite de 10 iterações. Como o robô sempre começa em uma mesma posição e orientação (na primeira posição e voltado para a direita), o máximo valor de fitness que pode ser alcançado é 8, pois ele deve girar (sem sair da posição) pelo menos 3 vezes. O objetivo deste estudo é investigar o sistema dinâmico do AG simples (sem crossover) aplicado para buscar os automatos que geram o maior valor de fitness. Salienta-se que um forma simples do problema de navegação do robô móvel foi escolhida de maneira a permitir o acompanhamento de todas as equações resultantes do sistema dinâmico do AG. Como estamos interessados em DOPs, mudanças no problema foram introduzidas através da simulação de 3 tipos de falhas nas leituras do sensor.

Na primeira falha (falha 1), o sensor do robô tem os sinais zerados, ou seja, independente de haver um obstáculo a sua frente ou não, o sensor fornecerá sinal igual a 0. Este tipo de falha pode ocorrer, por entre outros motivos, pelo mal-funcionamento

do sensor ou por um rompimento dos cabos que ligam o sensor ao micro-controlador responsável pelo controle do robô. No segundo tipo de falha (falha 2), o inverso ocorre, ou seja, o sensor sempre gerá sinal igual a 1, o que pode ocorrer em caso de um curto-circuito. Já no terceiro tipo de falha (falha 3), o sinal proveniente do sensor é invertido, ou seja, quando existe um obstáculo em sua frente, o sinal é igual a $I = 0$, sendo enviado um sinal $I = 1$ caso contrário. Este tipo de falha pode ocorrer devido a mal-funcionamento do sensor ou das portas de entrada do micro-controlador.

5 ANÁLISE DO SISTEMA DINÂMICO DO ALGORITMO GENÉTICO

Para entendermos como as mudanças provocadas pelas falhas afetam o comportamento dinâmico do AG, devemos investigar como a superfície de fitness é alterada na transição dos ciclos de mudança. No caso da falha 1, o sinal de entrada será sempre $I = 0$, o que resulta no fato de as combinações de estado/posição serem reduzidas a (I, S) = { 0,0 ; 0,1 }. Como consequência, as ações dadas pelo terceiro e quarto elementos do vetor $\mathbf{x}$ serão respectivamente iguais às ações dadas pelo primeiro e segundo elementos deste mesmo vetor. Efeito similar ocorre no caso da falha 2, na qual o sinal do sensor é sempre $I = 1$. Neste caso, (I, S) = { 1,0 ; 1,1 } e, como consequência, as ações dadas pelo primeiro e segundo elementos do vetor $\mathbf{x}$ é que serão respectivamente iguais às ações dadas pelo terceiro e quarto elementos deste vetor. Já no caso da falha 3, na qual o sinal de entrada é invertido, existirá uma permutação entre os elementos do vetor $\mathbf{x}$, sendo que as trocas serão entre o primeiro e terceiro elementos e entre o segundo e quarto elementos. Assim, podemos escrever que, quando ocorre a falha no ciclo de mudança e, supondo que o robô não apresenta falhas no ciclo $e - 1$, o vetor $\mathbf{x}(e)$ fica:

$$\mathbf{x}(e) = \mathbf{B}(e)\mathbf{x}(e-1) \tag{10}$$

na qual, para o modelo 1 (com $l = 4$), $\mathbf{B}(e)$ é definido respectivamente para as falhas 1, 2 e 3 como:

$$\mathbf{B}_{f1}(e) = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}, \mathbf{B}_{f2}(e) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix}, \mathbf{B}_{f3}(e) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}.$$

sendo $\mathbf{0}_2$ uma matriz 2×2 composta apenas por zeros e $\mathbf{I}_2$ uma matriz identidade 2×2. Como consequência da Eq. 10, o vetor de fitness no ciclo de mudanças e (com falhas), poderá ser calculado através do vetor de fitness no ciclo de mudanças $e - 1$ (sem falhas). Para o caso da falha 1, os valores de fitness para as soluções do espaço de busca nas quais as ações (elementos do vetor $\mathbf{x}$) são iguais para $I = 0$ e $I = 1$ no ciclo $e - 1$, substituirão no ciclo e os valores de fitness para as soluções cujas respectivas ações para $I = 0$ são iguais (ou seja, que apresentam a mesma primeira metade do vetor $\mathbf{x}$). O mesmo ocorre para a falha 2, mas agora substituindo as soluções cujas respectivas ações para $I = 1$ são iguais (ou seja, que apresentam a mesma segunda metade do vetor $\mathbf{x}$). Finalmente, para a falha 3, haverá uma permutação entre os elementos do vetor de fitness correspondentes às soluções que apresentam ações (elementos do vetor $\mathbf{x}$) diferentes para $I = 0$ e $I = 1$. Para o restante dos elementos do vetor de fitness (ou seja, elementos que apresentam a primeira e segunda metades do vetor $\mathbf{x}$ iguais), os valores permaneceram inalterados. Assim:

$$\mathbf{f}(e) = \mathbf{A}(e)\mathbf{f}(e-1), \tag{11}$$

na qual, para o modelo 1 (com $l = 4$), $\mathbf{A}(e)$ é definido respectivamente para as falhas 1, 2 e 3 como:

$$\mathbf{A}_{f1}(e) = \begin{bmatrix} \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e \end{bmatrix}, \mathbf{A}_{f2}(e) = \begin{bmatrix} \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \end{bmatrix},$$

$$\mathbf{A}_{f3}(e) = \begin{bmatrix} \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 \\ \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} \\ \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \end{bmatrix}.$$

sendo: $\mathbf{M}_a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{M}_b = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{M}_c = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{M}_d = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \mathbf{M}_e = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$

Estas transformações são condizentes com o espaço de fitness do robô para cada uma das condições (ver Figura 1, na qual o vetor de $\mathbf{f}$ é apresentado para o modelo 1). Pode-se observar que as melhores soluções para o robô sem falhas são as soluções 1 ($\mathbf{x} = [0, 0, 0, 1]^{\mathrm{T}}$) e 3 ($\mathbf{x} = [0, 0, 1, 1]^{\mathrm{T}}$), ou seja, quando o robô irá girar apenas se ele seguiu em frente na última iteração e encontrou uma parede em frente da posição atual (solução 1), ou quando ele encontrou uma parede em frente da posição atual independente do seu estado (solução 3). Em ambos os casos, o valor de fitness é igual ao valor de fitness máximo (8), sendo que a estratégia adotada pelo robô é navegar em sentido horário nas posições ao lado das paredes. As soluções que apresentam melhor fitness na sequência (soluções 5 e 7), apresentam a estratégia de girar sempre após um movimento para frente. Desta forma, apresentam um fitness igual a 6, já que nestra estratégia o robô gira 5 vezes ao passo que nas estratégias que apresentam fitness máximo, o robô gira por 3 vezes. Nota-se, entretanto, que esta estratégia apresenta fitness máximo quando o robô está

com as falhas 1 e 2, pelo fato de não usar o sensor para navegar. Assim, pode-se observar que algumas soluções (as quais a falha no sensor não afeta a estatégia desenvolvida), apresentam o mesmo fitness após as falhas 1 ou 2 no robô, ao passo que outras soluções têm seu fitness alterado de acordo com o exposto anteriormente. Já para o caso da falha 3, observa-se que os valores de fitness são mantidos, sendo apenas reordenados de acordo com a inversão do sinal do sensor (agora, as melhores soluções são as de índice 4 e 12).

De acordo com a Eq. 11, pode-se observar que as falhas geram DOPs lineares (Definição 7). Além disso, como a matriz $\mathbf{A}_{f3}(e)$ é uma matriz de permutação, a falha 3 gera um DOP com permutação (Definição 8), que apresenta características similares àquelas que ocorrem em DOPs produzidos pelo gerador de DOPs XOR, o que responde a questão sobre DOPs com permutação de fato ocorrerem em problemas do mundo real. No entanto, a matriz de permutação neste caso é gerada de forma diferente. Enquanto que, nos DOPs produzidos pelo gerador XOR, os elementos do vetor $\mathbf{f}$ são reordenados de acordo com a regra $\mathbf{i} \oplus \mathbf{k}(e)$ (Seção 3), o que produz uma permutação uniforme, a falha 3 ocasiona uma permutação não-uniforme de acordo com a matriz $\mathbf{A}_{f3}(e)$, sendo alguns dos elementos reordenados, enquanto outros permanecem fixos. De qualquer forma os valores de fitness são mantidos quando ocorre a falha 3, sendo apenas reordenados.

Como analisado em [9], em um DOP com permutação como o que ocorre no caso do robô com falha 3, o i-ésimo autovetor $\mathbf{y}_i(e)$ de $U(e)F(e)$ no ciclo de mudança e (Eq. 7), o qual define um estado metaestável, pode ser obtido pela permutação do respectivo autovetor no ciclo de mudança $e-1$. Assim, qualquer estado metaestável no ciclo de mudança e (robô com falha 3), pode ser obtido através da permutação de um estado metaestável no ciclo de mudança $e-1$ (robô sem falhas), e vice-versa. Além disso, os autovalores de $U(e)F(e)$ para os dois ambientes (robô sem falhas e com falha 3) são iguais, o que implica em valores iguais de fitness médio nos estados metaestáveis. Já nos casos das falhas 1 e 2, a transformação linear do vetor de fitness ocasiona o desaparecimento de alguns valores de fitness antes presentes. Desta forma, os estados metaestáveis são diferentes em cada ciclo de mudança, assim como os seus valores de fitness médio. Estes resultados podem ser observados nas simulações dos modelos do robô apresentadas na seção a seguir.

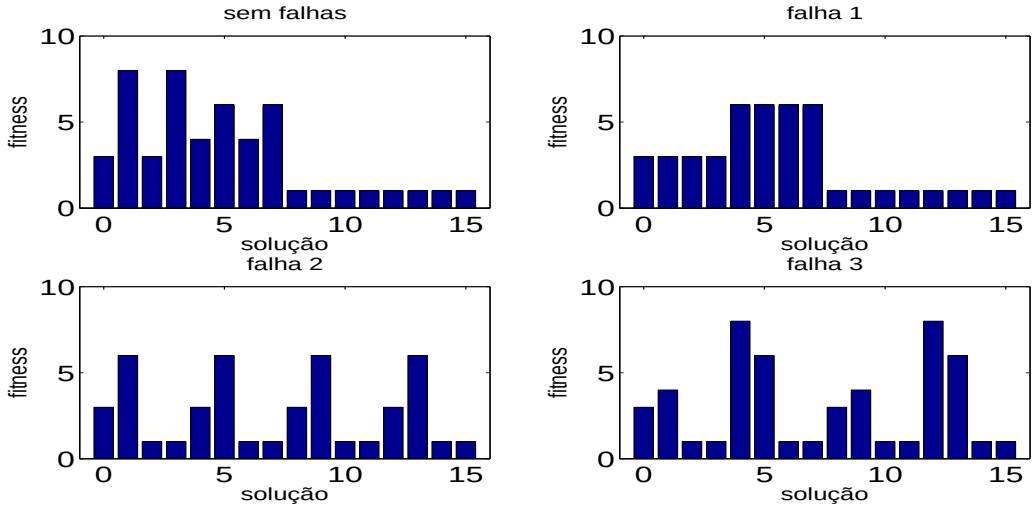

Figura 1: Espaço de fitness (vetor $\mathbf{f}$) para o modelo 1 ($l = 4$). As soluções são apresentadas de acordo com a conversão do vetor $\mathbf{x}$ de binário para inteiro.

6 SIMULAÇÕES

Nesta seção, as trajetórias das populações do AG com mutação e seleção proporcional quando aplicado no sistema descrito na Seção 4 são analisadas. Nas simulações aqui apresentadas, ao invés de executarmos o AG, o seu sistema dinâmico é simulado, ou seja, acompanha-se a evolução do vetor populacional dado pela Eq. 9. Considera-se que a população inicial ($\mathbf{p}_0$) é uniformemente distribuida e que a taxa de mutação é igual a $0,01$. Nestas simulações, o sistema dinâmico correspondente ao AG aplicado para controle do robô sem falhas é simulado por $\tau = 50$ (ou seja, a duração do ciclo de mudanças é 50 gerações), ocorrendo um dos três tipos de falhas na sequência. Estas mudanças são equivalentes a alterações no espaço (vetor) de fitness, como considerado na seção anterior. Desejamos observar como os principais estados metaestáveis da população se comportam quando ocorre a mudança no processo evolutivo.

Devido a limitação de espaço, apresentamos aqui apenas os resultados para as simulações do modelo 2 ($l = 8$). Entretanto, observa-se comportamento similar nas simulações do modelo 1 ($l = 4$). A Figura 2 mostra simulações do sistema dinâmico do AG para o modelo 2 ($l = 8$). Nesta figura, o fitness médio da população e a distância euclidiana entre o vetor de população na geração atual e os autovetores que apresentam o maior autovalor em um dado momento da simulação são apresentadas. O primeiro autovetor (o que apresentada maior autovalor) corresponde sempre ao principal estado metaestável do sistema, ou seja, àquele em que o número de indivíduos da população nos ótimos globais é maior do que o número de indivíduos em qualquer

outro lugar do espaço. Os outros autovetores correspondem a outros estados metaestáveis que apresentam importância para o entendimento da dinâmica da população.

As simulações apresentadas ajudam a entender o funcionamento do AG. Note que no primeiro ciclo de mudança, quando o robô não apresenta falhas, o vetor da população rapidamente converge para o estado metaestável principal, sendo que, neste caso, grande parte da população encontra-se distribuída entre os dois ótimos globais da superfície de fitness e, como consequência, o fitness médio da população é próximo do fitness máximo permitido. Quando ocorre uma falha, a população encontra-se localizada no principal estado metaestável correspondente ao primeiro ciclo de mudanças. Esta posição é diferente daquela correspondente à do principal estado metaestável no segundo ciclo de mudanças (ou seja, quando o robô apresenta falhas), sendo que o vetor de população deve migrar para esta nova posição. Neste caso, dependendo do tipo de falha, duas situações ocorrem. Na primeira situação, mesmo com a mudança do principal estado metaestável, este ainda é o estado mais perto da posição atual da população (ou seja, da posição do antigo estado mestaestável principal). Desta forma, o vetor da população não tem dificuldades em alcançar o novo estado metaestável principal. Este é o caso da falha 2, na qual o estados metaestáveis principais antes e depois da falhas encontram-se próximos devido ao fato de uma das soluções que apresenta fitness máximo antes da falha também apresenta fitness máximo depois da falha. Note que, para a falha 2, o fitness médio rapidamente converge para o valor próximo ao fitness máximo (valor igual a 6).

Repare, entretanto, que isso não ocorre para o caso das falhas 1 e 3. Nestes casos, quando ocorre a falha, existem estados metaestáveis mais próximos da posição atual do vetor de população (correspondente à posição do estado metaestável para o ciclo de mudanças 1, ou seja, antes da falha) do que o estado metaestável principal após a falha. Note que, nestes casos, a população primeiro aproxima-se destes estados, ficando um tempo em sua vizinhança, antes de convergir para o estado metaestável principal. Como consequência, nota-se que a população demora mais para convergir para o novo ponto ótimo, fato que pode ser observado nos gráficos do fitness médio da população. Desta forma, as falhas 1 e 3 representam mudanças mais severas do que a falha 1 para este problema de otimização dinâmica. Outro fator importante, que pode ser observado em problemas de otimização dinâmica, é a duração dos ciclos de mudança (d_e). Repare que, no caso das falhas 1 e 3, ciclos de mudanças menores (ou seja, no processos nos quais as mudanças são mais frequentes, como no caso de falhas intermitentes que ocorrem com curta duração), o desempenho será mais afetado devido ao que foi explicitado no parágrafo anterior.

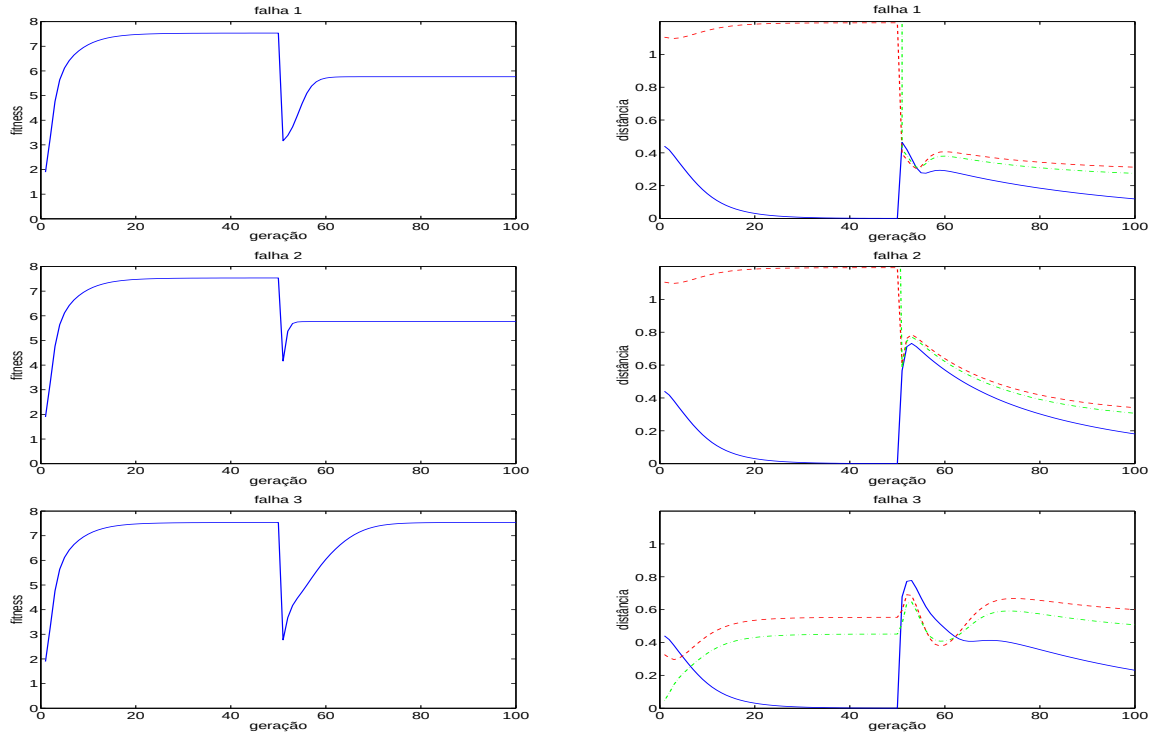

Figura 2: Média do fitness da população e distância para estados metaestáveis em simulações para os três tipos de falhas. A distância para o atual estado metaestável principal é apresentada pela linha sólida.

7 CONCLUSÕES

Neste trabalho, o comportamento do AG simples quando aplicado para o controle de um robô móvel em ambientes dinâmicos foi estudado de um ponto de vista teórico. Para isso, a abordagem por sistemas dinâmicos, conhecida como modelo exato, foi utilizada. Nesta abordagem, o AG é visto com um sistema dinâmico discreto, tornando possível investigar a trajetória da população durante o processo evolutivo. Aqui, um robô simples sujeito a falhas foi estudado. Na Seção 5, as mudanças provocadas pelas falhas no espaço de fitness do problema foram analisadas. Observou-se que tais mudanças geram DOPs lineares (Definição 7),

na qual o vetor de fitness é alterado de acordo com uma transformação linear. Ainda, notou-se que para o caso da falha 3, o problema é ainda do tipo DOP com permutação (Definição 8). DOPs com permutação são produzidos também pelo gerador de DOPs XOR. Desta forma, este trabalho responde um importante questionamento acerca da utilizada deste gerador em produzir DOPs similares aos que ocorrem no mundo real.

Além disso, na Seção 6, simulações do sistema dinâmico do AG para o problema estudado foram analisadas. Observou-se que os DOPs gerados pelas falhas 1 e 3 são mais difíceis do que aqueles gerados pela falha 2. Isso ocorre devido ao fato de que, para estas falhas, estados metaestáveis diferentes do principal (ou seja, no qual grande parte da população encontra-se em um ótimo global) após a falha estarem mais pertos do estado metaestável principal de antes da falha. Desta forma, a população fica por um tempo na vizinhança destes estados antes de convergir para o estado metaestável principal (de depois da falha). Como consequência, o AG leva mais tempo para chegar no ótimo global, o que ocasiona uma piora no desempenho (em relação à falha 2). O enfoque por sistemas dinâmicos aqui apresentado ajunda a explicar o comportamento do AG de um ponto de vista teórico, o que é extremamente útil em AEs nos quais o estudo do desempenho baseiam-se quase que exclusivamente em análises experimentais. A desvantagem principal deste método teórico é a aplicabilidade limitada a problemas pequenos, requisito para permitir o acompanhamento das equações que descrevem o sistema dinâmico do AG. Como trabalhos futuros, o enfoque por sistemas dinâmicos deverá ser aplicado em outros DOPs, inclusive em variantes do problema envolvendo robôs aqui estudado. Além disso, AGs especiamente desenvolvidos para DOPs deverão ser investigados usando a abordagem aqui descrita.

Agradecimento: O autor agradece à FAPESP (Proc. 2010/09273-1) pelo apoio financeiro a este trabalho.

REFERÊNCIAS

[1] Y. Jin and Y. Meng. "Special Issue on Evolutionary and Developmental Robotics [Guest Editorial]". *IEEE Computational Intelligence Magazine*, vol. 5, no. 3, pp. 9, 2010.

[2] S. Nolfi and D. Floreano. *Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines*. MIT Press/Bradford Books: Cambridge, USA, 2000.

[3] N. Jakobi. "Half-baked, ad-hoc and noisy: minimal simulations for evolutionary robotics". In *Proc. of the Fourth European Conference on Artificial Life*, edited by P. Husbands and I. Harvey, pp. 348–357. MIT Press, 1997.

[4] A. Thompson. "On the automatic design of robust electronics through artificial evolution". In *Proc. of the 2nd Int. Conf. on Evolbable Systems: From Biology to Hardware*, edited by M. Sipper and A. Prez-Uribe, pp. 13–24. Springer, 1998.

[5] D. Floreano, S. Nolfi and F. Mondada. "Co-evolution and ontogenetic change in competing robots". In *Advances in the Evolutionary Synthesis of Intelligent Agents*, edited by M. Patel, V. Honavar and K. Balackishnan. MIT Press, 2001.

[6] R. Tinós and A. C. P. L. F. Carvalho. "Use of gene dependent mutation probability in evolutionary neural networks for non-stationary problems". *Neurocomputing*, vol. 70, no. 1-3, pp. 44–54, 2006.

[7] C. R. Reeves and J. E. Rowe. *Genetic algorithms - principles and perspectives: a guide to GA theory*. Kluwer Academic Publishers, 2003.

[8] M. D. Vose. *The simple genetic algorithm: foundations and theory*. The MIT Press, 1999.

[9] R. Tinós and S. Yang. "An Analysis of the XOR Dynamic Problem Generator Based on the Dynamical System". In *Parallel Problem Solving from Nature - PPSN XI*, edited by R. Schaefer, C. Cotta, J. Kolodziej and G. Rudolph, volume 6238 of *Lecture Notes in Computer Science*, pp. 274–283. Springer Berlin / Heidelberg, 2011.

[10] S. Yang and X. Yao. "Experimental study on population-based incremental learning algorithms for dynamic optimization problems". *Soft Computing*, vol. 9, no. 11, pp. 815–834, 2005.

[11] S. Yang. "Constructing dynamic test environments for genetic algorithms based on problem difficulty". In *Proc. of the 2004 Congress on Evolutionary Computation*, volume 2, pp. 1262–1269, 2004.

[12] R. Tinós and S. Yang. "Analyzing Evolutionary Algorithms for Dynamic Optimization Problems Based on the Dynamical System". In *To appear in the book Evolutionary Computation for Dynamic Optimization Problems*, edited by S. Yang and X. Yao, pp. 1–25. Springer, 2011.

[13] C. Hayes and T. Gedeon. "Hyperbolicity of the fixed point set for the simple genetic algorithm". *Theoretical Computer Science*, vol. 411, no. 25, pp. 2368–2383, 2010.

[14] E. V. Nimwegen, J. P. Crutchfield and M. Mitchell. "Finite populations induce metastability in evolutionary search". *Physics Letters A*, vol. 229, no. 3, pp. 144–150, 1997.

[15] Y. Jin and J. Branke. "Evolutionary optimization in uncertain environments - a survey". *IEEE Trans. on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.