Mais um Algoritmo Genético para Resolver o Cubo Mágico

Daniel K. S. Vieira

12 de setembro de 2016

Resumo

Neste trabalho é proposto um Algoritmo Genético (GA) baseado no trabalho de Herdy e Patone (1994) desenvolvido para resolver o famoso jogo de quebra-cabeça chamado de cubo mágico. Este GA foi projetado para tentar resolver cubos mágicos de qualquer tamanho e testes mostram que ele consegue resolver cubos de tamanho 3x3. Para cubos maiores é necessário um estudo mais aprofundado das peculiaridades desses problemas que possuem um espaço de busca muito maior.

1 Objetivo

Este trabalho, como já foi dito anteriormente, visa projetar e implementar um sistema de otimização baseado em Algoritmo Genético que resolva o problema de quebra-cabeça cubo mágico. Dado um cubo de tamanho 3x3 existem 18 movimentos que podem ser aplicados, já que um cubo tendo 6 faces, cada face pode ser rotacionada em sentido horário, anti-horário, ou 180º.

A contribuição deste trabalho está em propor um modelo de Algoritmo Genético que consiga evoluir soluções para cubos mágicos de qualquer tamanho

2 Modelagem do Algoritmo Genético

No trabalho de Herdy e Patone (1994) foi descrito um algoritmo baseado em Estratégia Evolucionária ((1,50)-ES) que utiliza um conjunto de sequências movimentos que são concatenados para se aplicar a mutação nos indivíduos de forma que não se altere muito seu *fitness*. Isso ocorre devido ao fato do cubo resultante dos movimentos após a mutação não ser muito alterado se comparado ao seu estado antes da modificação do indivíduo.

A função de *fitness* que pondera a penalização aplicada a cada quadrado errado de cada face de acordo com sua distância em relação ao centro, ou seja, quadrados que estão nas quinas, caso estejam de uma cor diferente da do centro, são penalizados mais gravemente se comparados com a penalização de um quadrado de uma borda.

El-Sourani e Borschbach (2010) mostram uma expansão do número de sequências de movimentos adicionando mais 2 tipos aos 11 previamente existentes, isso levando em consideração a face da frente e o cubo não rotacionado.

O GA proposto utiliza estes conceitos mas os extrapola na tentativa de resolver cubos de qualquer tamanho.

2.1 Representação do Indivíduo

O cromossomo do indivíduo foi codificado de forma numérica, em que cada valor representa um movimento. Para um cubo de tamanho 3x3, existem 18 possíveis movimentos para serem feitos, mas para um cubo de tamanho maior, 4x4 por exemplo, essa quantidade aumenta já que neste caso não existe centro estático, as duas colunas e linhas centrais do cubo também se movem.

Portanto os valores que os genes comportam não são fixos em sua variação. Para tal foi feito um cálculo de indexação convertendo todas as características de um movimento em um único número.

Sendo as faces $F = 0, B = 1, U = 2, D = 3, L = 4, R = 5$, os tipo de movimentos em sentido horário, anti-horário e o de 180º, iguais a 0, 1, 2, respectivamente. Com apensas essas informações já é possível gerar os valores para todos os movimentos do cubo 3x3, mas não para o de 4x4. Por isso foi definido um conceito chamado de nível.

O nível específica em qual camada do cubo o movimento irá acontecer. O nível 0 é a superfície, todos os movimentos anteriormente mencionados ocorriam neste nível, onde também se rotaciona a face em si. Mas no caso de um movimento que ocorra em um nível l mais interno ($l > 0$), a face do movimento não é rotacionada, apenas um segmento das faces adjacentes.

Por exemplo, um movimento na face frontal em sentido horário mas com nível igual a 1 corresponde para baixo da segunda coluna da esquerda para a direita da face da direita. O número de níveis de um cubo é calculado por:

$$l = \left\lfloor \frac{s}{2} \right\rfloor, \tag{1}$$

sendo s o tamanho do cubo. Portanto 3x3 só possui 1 nível (indexado por 0).

O cromossomo do indivíduo foi definido como tendo tamanho fixo 100. Mas é importante notar que isso não significa que as soluções devem ter 100 movimentos especificamente. Existe um valor específico (-1) que um gene pode ter que define o não movimento. Também é possível que o valor daquele gene represente não apenas um mas uma sequência de movimentos, mas isso será descrito com mais detalhes na subseção de Mutação (2.5).

Portanto, um movimento v na face f, no sentido c, no nível l pode ser codificado no valor:

$$v = l \times 6 \times 3 + c \times 6 + f \tag{2}$$

em que 6 e 3 representam o número de faces e número de sentidos de movimento, respectivamente.

2.2 Cálculo da *Fitness*

Inspirado no trabalho de Herdy e Patone (1994), o cálculo de *fitness* desenvolvida penaliza mais quadrados das quinas que estão diferentes do centro se comparados com quadrados das bordas.

Mas expandido essa ideia para cubos de qualquer tamanho um *kernel* foi gerado a partir de uma função Gaussiana modificada para ser utilizado como conjunto de pesos nas penalizações e a *fitness* era calculada aplicando uma convolução desse *kernel* em cada face.

Quando um quadrado estava diferente do centro a solução era penalizada com o peso do *kernel* na mesma posição do quadrado. Em cubos que não possuem centro, tenta-se encontrar uma cor de maioria nos quatro quadrados centrais, caso exista, esta é definida como a cor que a face deveria ter, caso contrário, toda a face é penalizada.

A função aplicada para se gerar o *kernel* é inspirada na função Gaussiana Guo (2011). Ela possui um aspecto inverso ao da função Gaussiana original, ou seja, o ponto com menor valor é o central (0) e os pontos com maior valor são os extremos do espaço.

Para se gerar um *kernel* a partir de uma função Gaussiana basta computar o valor da função em cada ponto. A função para se gerar o *kernel* de pesos pe definido como:

$$
\begin{aligned}
f(x,y) &= \left(\left\| 1 - e^{-\left(\frac{(x-\mu)^2}{2\sigma^2} + \frac{(y-\mu)^2}{2\sigma^2} \right)} \right\| (\mu + 1) \right)^2, \\
\sigma &= \frac{\mu}{2}, \\
\mu &= \left\lfloor \frac{s}{2} \right\rfloor,
\end{aligned}
\tag{3}
$$

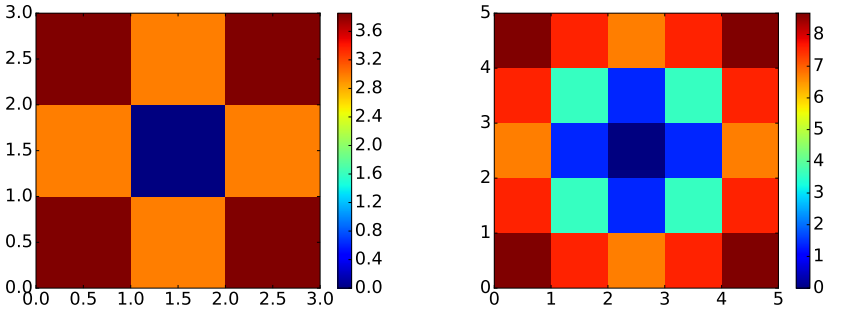
com s sendo o tamanho do cubo.

Sendo assim, a função de penalização apresentada na equação 3 resulta nas penalizações que podem ser vistas visualmente na figura 1.

Quando um cubo $k \times k$ possui tamanho par, são removidas a linha e coluna centrais do *kernel* gerado para um cubo de tamanho $k + 1 \times k + 1$. ou seja, no caso do cubo de tamanho 4×4, seria utilizado o *kernel* da figura 1b mas com a linha e coluna centrais removidas. Portanto, os quatro quadrados centrais possuem a mesma importância e não são penalizados com peso 0 (representando a não penalização), como no caso de um cubo com tamanho ímpar.

2.3 População Inicial

A população inicial é gerada com os valores dos genes definidos de forma aleatória, mas com uma probabilidade especifica dos genes serem definidos como sem movimento, ou seja, o valor de não movimento pode ter uma probabilidade muito



(a) Em um cubo de tamanho 3x3.

(b) Em um cubo de tamanho 5x5.

Figura 1: Pesos gerados pela equação 3.

maior de ser atribuído aos genes do que os outros movimentos, ou nenhuma chance. Esta probabilidade é definida como parâmetro de configuração do GA.

Esta probabilidade separada para o não movimento é definida para se ter a chance de fazer como q ele apareça muito mais vezes que os outros movimentos, e assim permitir que no processo evolutivo seja adicionado novos movimentos. Caso todos os genes fossem definidos como tendo algum movimento, operadores de mutação só poderiam substituir movimentos por outros, en ão adicionar, impactando fortemente o cubo resultante dos movimentos da solução e, consequentemente, sua *fitness*.

2.4 Cruzamento

O operador de cruzamento utilizado foi o *N-point*. Este operador vai definir N pontos no cromossomo, chamados de ponto de corte, e dados dois pais, são gerados dois filhos que possuem cromossomos determinados a partir da concatenação de segmentos de forma alternada definidos por estes pontos de corte.

Caso $N = 1$, o ponto de corte pode ser definido em qualquer parte do cromossomo. Entretanto se $N = 2$, o primeiro ponto de corte só pode ser definido entre $[0, {}^{c}/_{2})$, e o segundo ponto de corte entre $[{}^{c}/_{2}, c)$, sendo c o tamanho do cromossomo. isto é feito para distribuir mais adequadamente os pontos de corte em todo o cromossomo. Quanto maior o número de pontos de corte, menor o segmento do cromossomo do indivíduo que cada um pode ocupar. O número de pontos de corte é um parâmetro de configuração do GA.

O cruzamento é aplicado de forma que todos os indivíduos tenham uma chance e gerar filhos. Após o cruzamento ocorre a etapa de mutação. Os filhos não são incluídos nesta etapa, mas poderão ser na próxima geração caso eles passem pela etapa de seleção.

2.5 Mutação

Dois operadores de mutação foram implementados. O primeiro é um simples operador de mutação aleatório em que cada gene possui uma probabilidade de ser alterado, modificando o movimento que está inserido naquele gene para qualquer outro possível no cubo que está sendo resolvido (lembrando que quanto maior o tamanho do cubo, maior o número de movimentos).

Já o segundo operador concatena (ou substitui, caso o cromossomo esteja todo ocupado por movimentos) um conjunto de movimentos que quando aplicados em sequência, alteram consideravelmente pouco o cubo resultante, possibilitando uma alteração muito mais suave do fenótipo do indivíduo e uma busca mais controlada sobre o espaço de busca. Enquanto o primeiro operador criar diversidade sobre a população, este proporciona uma estratégia de busca local.

Este conjunto de sequências de movimentos foi originalmente utilizado por Herdy e Patone (1994) e posteriormente expandido no trabalho de El-Sourani (2009) adicionando mais duas sequências totalizando 12 sequências.

Estes movimentos também citados por El-Sourani e Borschbach (2010) são referentes apenas a face frontal e sem o cubo ser rotacionado inteiramente para nenhum lado. Ou seja, de fato existem $12 \times 6 \times 2 \times 4 = 576$ sequências de movimentos, sendo 12 o número de sequências para a face frontal sem o cubo rotacionado (note que cubo rotacionado é diferente de face rotacionada), 6 o número de faces, 2 representando se o movimento é espelhado ou não (sequência inversa) e 4 o número de rotações que o cubo pode ter (não rotacionado, 90º, -90º, 180º).

2.5.1 Novos movimentos

Para um cubo de tamanho maior que 3×3 Apenas estas sequências não são capazes de movimentar níveis mais internos do cubo, apenas as extremidades do mesmo. Portanto este trabalho amplia ainda mais este conjunto de sequências de movimentos para o âmbito de cubos de tamanho maiores que 3×3. Para tal, além de todas as possibilidades de movimentos já existentes, para cada nível do cubo, todas as sequências de movimentos possuem uma versão com todos os movimentos daquela sequência referentes aos níveis mais internos existentes no cubo.

Por exemplo, a sequência *two edge flip cw* para um cubo de tamanho 6×6 possui as seguintes versões somente para a face frontal sem rotação do cubo nem espelhamento:

Ou seja, o número de de sequências de movimentos cresce automaticamente conforma o tamanho do cubo aumenta, sendo definido como $m = l_n \times 576$, em que l_n é o número de níveis.

Existem quatro casos especias que não alteram o cubo quando aplicados em níveis mais internos, não alteram em nada o cubo. São estes o *two corner flip cw*, *two corner flip cw*, *three corner swap cw* e *three corner swap cw*. Portanto, quando um movimento é mapeado para estas sequências mas com um nível maior, estes são automaticamente alterados para o mesmo tipo de sequência

Tabela 1: Exemplo de movimentos novos levanto em conta os níveis de um cubo 6×6.

Nome	Sequência
two edge flip cw l1	F R B L U Li U Bi Ri Fi Li Ui L Ui
two edge flip cw l2	2F 2R 2B 2L 2U 2Li 2U 2Bi 2Ri 2Fi 2Li 2Ui 2L 2Ui
two edge flip cw l3	3F 3R 3B 3L 3U 3Li 3U 3Bi 3Ri 3Fi 3Li 3Ui 3L 3Ui

mas para o nível 0.

2.5.2 Mapeamento

Todas as sequências são mapeadas para um único valor para que elas não ocupem muitos genes do cromossomo. para os índices não entrarem em conflito com os valores dos movimentos já existentes, depois do cálculo do valor que representa a sequência, é adicionado um *offset* somando o número de movimentos padrões já existentes.

Portanto, o valor i que representa uma sequência é calculado da seguinte forma:

$$i = m \times r \times t \times f \times l + v_n, \tag{4}$$

sendo que $m \in \{0, 1\}$ especifica se o movimento é do tipo espelhado ou não, $r \in \{0, 1, 2, 3\}$ determina a rotação do cubo (sem rotação, 90°m -90°, 180°, respectivamente), $t \in \{0, 1, 2, \cdots, 11\}$ define o tipo de sequência de movimentos utilizada (a sequência numérica respeita a ordem apresentada na Tabela 1 do trabalho de El-Sourani e Borschbach (2010)), $f \in \{0, 1, 2, \cdots, 5\}$ especifica a face, $l \in \mathbb{Z}$ diz respeito ao nível do movimento e v_n é o número de movimentos padrões possíveis no cubo (lembrando que em um cubo de tamanho 3×3 existem 18 movimentos).

2.5.3 Atribuição

Estas sequências de movimentos (agora mapeadas em um único valor) só cumprem seu papel de modificar minimamente o fenótipo da solução se estes forem concatenados ao final do cromossomo. Para tal, é feito uma troca de informação dos genes de forma que todos os valores que especifiquem não movimento (-1) estejam ao final do cromossomo, isto obviamente é feito sem alterar a ordem dos movimentos. Após isso o valor que representa a sequência de movimentos a atribuída ao último gene do cromossomo.

Por fim, os valores de não movimento são aleatoriamente distribuídos no cromossomo para que a operação de cruzamento não seja impactada pelo fato do segmento final do cromossomo dos indivíduos provavelmente conter apenas valores de não movimento.

2.6 Seleção

Ocorrendo após a etapa de mutação, a seleção é aplicada sobre a população gerada pelos indivíduos existentes desde o início da geração, os filhos e os indivíduos gerados por mutação. Possuindo elitismo, este GA separa os n melhores indivíduos para serem unidos aos selecionados para continuarem para a próxima geração.

A seleção é realizada utilizando o método do torneio. Sendo k o tamanho do torneio de n o número de indivíduos que deve ser retornados depois da execução da etapa de seleção, são selecionados n vezes k indivíduos aleatoriamente e o melhor entre eles é escolhido para continuar para a próxima geração, sendo que n é o tamanho da população inicial.

3 Experimentos

Foram definidas 11 configurações distintas alterando boa parte dos parâmetros que este GA possui. Estes parâmetros são:

- n_i: número de indivíduos;
- n_g: número de gerações;
- p_c: probabilidade de cruzamento
- p_m: probabilidade de mutação
- t: tamanho do torneio
- e: especifica se será utilizado elitismo ($e \in \{0, 1\}$)
- n_e: número de elites
- n_c: número de pontos de corte
- p_f: probabilidade de alterar um dado gene (mutação aleatória)
- n_f: número de gerações que o operador de mutação aleatório será utilizado. Após n_f gerações o operador de sequência de movimentos passa a ser utilizado.
- p_{-1}: probabilidade de ser atribuído o valor de não movimento a um dado gene.

Foram definidos 9 configurações (chamados de c_n, em que n é o número da configuração) alterando estes parâmetros de forma encontrar indícios de como os parâmetro devem ser configurados, e 2 (chamadas de h_n) configurações definidas de formar a tentar extrair o melhor do GA. A tabela 2 apresenta os parâmetros utilizados em cada configuração.

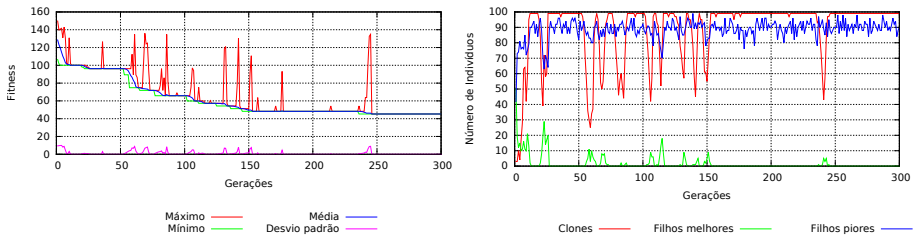
Todos os testes foram condizidos utilizando semente estática para o gerador de números aleatórios, possibilitando assim, uma comparação mais justa entre

Tabela 2: Configurações de parâmetros utilizadas nos experimentos.

	n_i	n_g	p_c	p_m	t	e	n_e	n_c	p_f	n_f	p_{-1}
c_1	100	300	0.9	0.1	2	1	4	1	0.09	50	0.5
c_2	1000	300	0.9	0.1	2	1	4	1	0.09	50	0.5
c_3	10000	300	0.9	0.1	2	1	4	1	0.09	50	0.5
c_4	10000	1000	0.9	0.1	2	1	4	1	0.09	50	0.5
c_5	10000	300	0.9	0.5	2	1	4	1	0.09	50	0.5
c_6	10000	300	0.9	0.9	2	1	4	1	0.09	50	0.5
c_7	10000	300	0.9	0.9	1	1	4	1	0.09	50	0.5
c_8	10000	300	0.9	0.9	5	1	4	1	0.09	50	0.5
c_9	10000	300	0.9	0.9	5	0	N/A	1	0.09	50	0.5
h_1	10000	300	0.19	0.62	3	1	6	3	0.09	55	0.78
h_2	10000	500	0.32	0.87	2	1	10	5	0.09	65	0.74

configurações. As instâncias utilizadas consistem em três cubos de 3×3, um cubo de 4×4 e outro 5×5, todos em estados randômicos. Todas as configurações foram testadas 5 vezes em todas as instâncias e em cada teste foram coletados os dados de *fitness* do pior indivíduo (maior), melhor indivíduo (menor), média e desvio padrão dos *fitness* da população, juntamente com o número de clones (*fitness* iguais), número de filhos com *fitness* menor (melhor) e maior (pior) que a média dos pais em cada geração.

Observando os gráficos da Figura 2 pode-se notar que a configuração c_1 não possui uma diversidade considerável. muitos indivíduos são iguais na maior parte do tempo e somente ocorre uma alteração quando se encontra uma solução melhor. Neste caso alguns indivíduos pioram muito enquanto outros melhoram (possível de notar até mesmo pelos filhos melhores gerados pelo cruzamento). Outro fato interessante de perceber é a queda da *fitness* após a geração 50, exatamente no momento que ocorre a troca de operador de mutação.



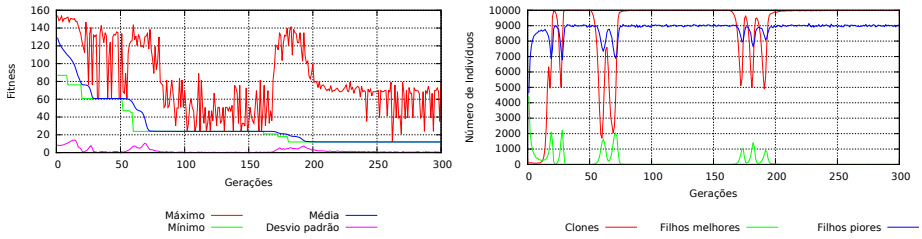
(a) Informação de *fitness*.

(b) Estatística da população.

Figura 2: Desempenho em uma execução da configuração c_1 na instância 3_3_1 (instância 1 do cubo de tamanho 3×3).

Na tentativa de aumentar a diversidade da população, o número de in-

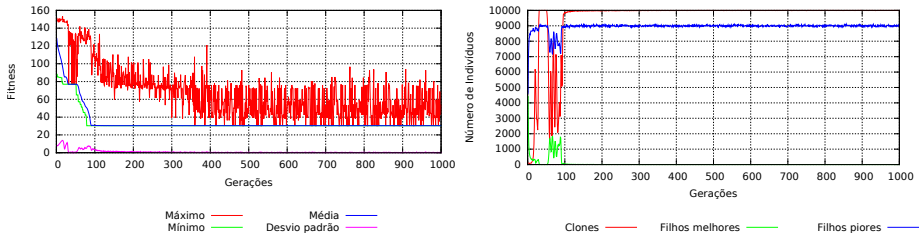
divíduos foi acrescido (configurações c_2 e c_3). No caso da configuração c_2 não houve uma diferença significativa, mas no caso da configuração c_3, com 10000 indivíduos, apesar do número de clones ainda ser quase igual ao tamanho da população, o pior indivíduo não segue a média e o melhor. É possível perceber na Figura 3 também a melhora na *fitness* do melhor indivíduo.



(a) Informação de *fitness*. (b) Estatística da população.

Figura 3: Desempenho em uma execução da configuração c_3 na instância 3_3_1.

A configuração c_4 aumenta o número de gerações para prover ao GA mais tempo para que ele saia do mínimo local. Mas como é possível observar na Figura 4 isso não acontece.



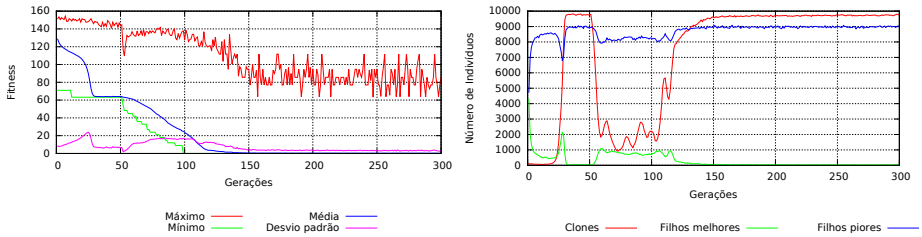
(a) Informação de *fitness*. (b) Estatística da população.

Figura 4: Desempenho em uma execução da configuração c_2 na instância 3_3_1.

Por conta disso o número de gerações foi reduzido ao seu valor original. Como a média ainda está extremamente próxima da melhor *fitness*, ou seja, não possui a diversidade desejada. A configuração c_5 tenta aumentar a diversidade com uma mutação de 50%. Com isso é obtido um aumento considerável na diversidade e, consequentemente, em alguns testes foi possível resolver o cubo, como mostrado na Figura 5.

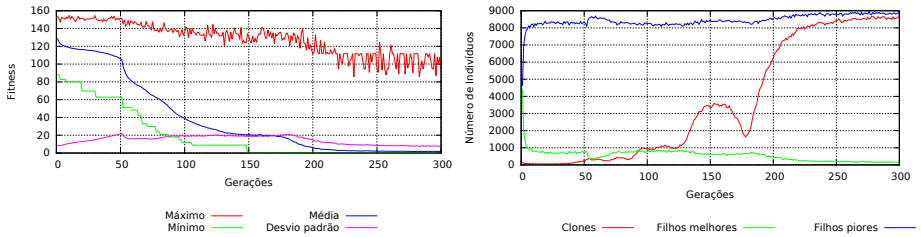
Com esta melhora expressiva, a configuração c_6 coloca a probabilidade de mutação em 90% e como mostra a Figura 6 a diversidade aumenta (já que a média fica menos perto da melhor solução). O núemro de clones também só chega próximo ao tamanho da população no final da execução do GA neste teste.

A configuração c_7 diminui o tamanho do torneio para 1, transformando a



(a) Informação de *fitness*. (b) Estatística da população.

Figura 5: Desempenho em uma execução da configuração c_5 na instância 3_3_1.



(a) Informação de *fitness*. (b) Estatística da população.

Figura 6: Desempenho em uma execução da configuração c_6 na instância 3_3_1.

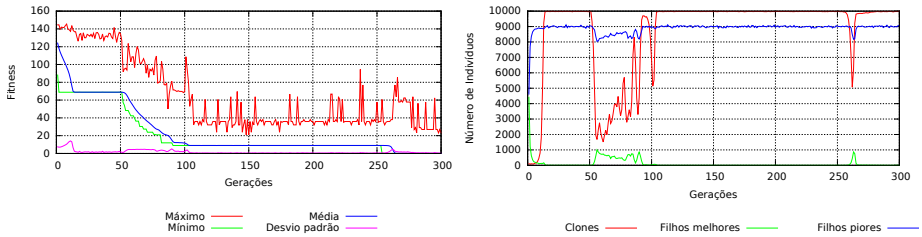
seleção em um processo de escolha aleatória de indivíduos. A diminuição na pressão seletiva de fato aumenta a diversidade, mas isso torna o Ga instável e em nenhuma das 5 execuções na instância 3_3_1 uma solução que resolvesse o cubo foi encontrada.

Já a configuração c_8 faz exatamente o contrário e aumenta a pressão seletiva aumentando o tamanho do torneio para 5. Se já estava difícil manter a diversidade das soluções com um torneio de tamanho 2, aumentar provavelmente não iria ajudar e isso é comprovado nos testes realizados. Nenhuma execução do GA utilizando a configuração c_8 consegui resolver a instância 3_3_1.

O elitismo foi retirado na configuração c_9 e curiosamente, como pode ser visto na Figura 7, em um caso foi até possível encontrar uma solução para a instância 3_3_1 mesmo com a pressão seletiva extremamente alta.

$h1$ é definido como uma tentativa de testar uma configuração que equilibra os parâmetros. p_{-1} foi aumentado para diminuir a probabilidade da adição de movimentos aleatórios no inicio que poderiam atrapalhar o processo de evolução (principalmente mutação) dos indivíduos.

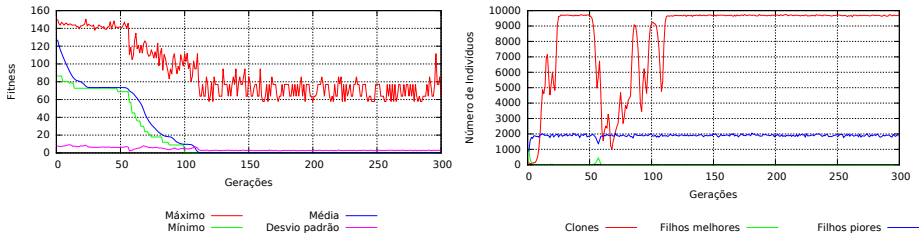
Apesar de ter conseguido resolver o cubo em duas das 5 execuções na instância 3_3_1, seu desempenho não foi melhor que a configuração c_6 (que também resolveu duas vezes a mesma instância). Acredita-se que a pressão seletiva alta causada pelo torneio de tamanho 3 e a probabilidade de mutação baixa, diminuindo a intensidade de evolução, tenham causado esse desempenho insatisfatório. A



(a) Informação de *fitness*. (b) Estatística da população.

Figura 7: Desempenho em uma execução da configuração c_9 na instância 3_3_1.

Figura 8 mostra uma das execuções feitas utilizando a configuração h_1.



(a) Informação de *fitness*. (b) Estatística da população.

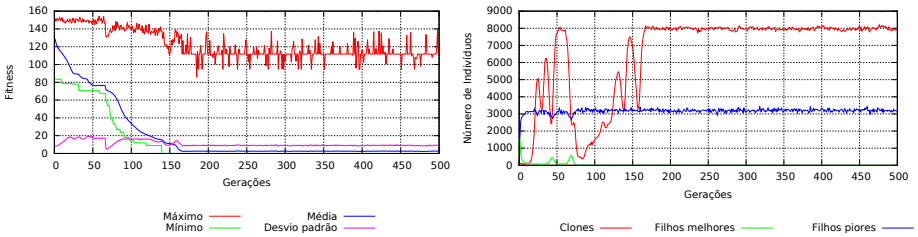
Figura 8: Desempenho em uma execução da configuração h_1 na instância 3_3_1.

A configuração h_2 é a junção de todas as propriedades que a configuração deve ter para que a população tenha uma evolução controlada. Principalmente alta mutação e baixa pressão seletiva. O número de gerações e de gerações com mutação aleatória foram aumentados com o intuito de aumentar as chances de se encontrar a solução e maximizar a diversidade na etapa inicial do algoritmo.

Como resultado o h_2 consegui resolver a instância 3_3_1 em todas as 5 execuções. A Figura 9 mostra uma das execuções feitas na instância 3_3_1.

As Tabelas 3, 4, 5, 6 e 7 mostram o resultados gerais de cada configuração. Interessante notar que apesar de h_2 ter sido o melhor na instância 3_3_1, nas outras instâncias de tamanho 3×3 ele obteve uma média pior que c_6, mostrado que com um cruzamento ainda mais alta possivelmente poderia apresentar resultados ainda melhores

Perceba também que nas instâncias 4_4_1 e 5_5_1 nenhuma conseguiu alcançar a solução, demonstrando a necessidade de mudança de estratégia para cubos de tamanho maiores que 3×3.



(a) Informação de *fitness*.

(b) Estatística da população.

Figura 9: Desempenho em uma execução da configuração h_2 na instância 3_3_1.

Tabela 3: Resultados para a instância 3_3_1

Configuração	Melhor	Pior	Média	Desvio padrão
c_1	41.07238412	47.917782	43.79684227	2.49067271061
c_2	17.94348114	54.7631767	41.231531688	14.0568363573
c_3	11.962321	30.3721662	23.008228264	6.88991182383
c_4	11.962321	42.8008588	26.69019926	12.6515382067
c_5	0.0	23.924641	10.76608882	7.70618587655
c_6	0.0	11.96232114	6.579276838	5.4818242585
c_7	8.97174117	36.35332916	17.438638914	10.4244927377
c_8	8.971741	27.381588	15.644290572	7.32596152652
c_9	0.0	11.9623211	8.373624932	4.39523736827
h_1	0.0	30.372166	12.746983	11.9257369618
h_2	0.0	0.0	0.0	0.0

Tabela 4: Resultados para a instância 3_3_2

Configuração	Melhor	Pior	Média	Desvio padrão
c_1	32.89638117	62.472813	46.455427018	12.6704360783
c_2	29.9058003	50.044125	36.60575026	7.5146706718
c_3	8.9717412	36.353325	22.82168186	9.88175253638
c_4	11.962321	29.11006216	22.064418432	5.64012781228
c_5	8.97174111	21.40042713	12.653710468	4.57328951109
c_6	0.0	11.96232112	4.784928608	5.86031644602
c_7	17.94348116	42.3344888	27.101767772	8.65725409911
c_8	0.0	17.943481	8.37362504	7.17739222
c_9	0.0	30.3721663	19.246685084	10.5699428564
h_1	8.971741	17.943481	11.962321024	3.27601625205
h_2	0.0	21.400427	9.0650141	8.16455902092

Tabela 5: Resultados para a instância 3_3_3

Configuração	Melhor	Pior	Média	Desvio padrão
c_1	44.9272004	57.287392	50.894662046	4.30994190783
c_2	26.91522	50.044125	39.437182534	7.57926485836
c_3	8.97174115	44.0629623	25.65311451	12.0097300309
c_4	11.96232112	39.34391017	24.70930388	9.3512587066
c_5	8.971741	17.94348116	14.354785232	3.48758569076
c_6	0.0	11.9623214	2.392464334	4.784928533
c_7	14.95290214	45.32506917	31.381853822	12.9865371872
c_8	8.97174114	35.09122111	19.41953373	11.6647765572
c_9	0.0	17.9434816	7.17739262	8.79047459718
h_1	11.962321	21.40042715	17.53191143	4.55073891591
h_2	0.0	11.962321	4.784928468	5.86031646235

Tabela 6: Resultados para a instância 4_4_1

Configuração	Melhor	Pior	Média	Desvio padrão
c_1	279.678253	398.1764536	343.965344584	39.357507705
c_2	265.488525	328.44751	310.1408324	23.1062560164
c_3	240.6891941	316.81228629	288.611947924	28.1075956624
c_4	264.91949524	326.90133721	299.32283349	25.5269808257
c_5	193.64430218	290.8575132	238.325988776	33.6104225846
c_6	178.86882	290.15799023	229.398929146	43.2453183932
c_7	247.88159222	341.4332891	311.221374904	34.8624465015
c_8	178.478088	249.280548	224.035296818	27.1506090022
c_9	190.82959028	275.317261	223.0135013	34.5956724877
h_1	182.464996	272.420593	245.4339874	33.1758428379
h_2	204.432892	273.51083427	232.851880082	24.6296719686

Tabela 7: Resultados para a instância 5_5_1

Configuração	Melhor	Pior	Média	Desvio padrão
c_1	394.6114501	423.80233818	413.21580209	9.91071069797
c_2	328.55786125	385.001831	355.887060494	24.230085981
c_3	332.03643827	349.242981	335.77831437	6.7361318726
c_4	311.492523	379.78286722	355.767443932	28.4577157532
c_5	287.94522129	317.53970329	302.065277218	12.2501355881
c_6	252.38349921	311.65506	283.373019556	23.8778525189
c_7	343.16906735	408.469238	374.124096802	22.4704390737
c_8	306.783234	331.871277	321.69253558	9.51414085112
c_9	310.63562	360.31445322	328.762805328	17.1101699896
h_1	291.544067	360.4346628	317.658673194	25.0598505246
h_2	281.786133	311.43121324	298.484991526	10.3088598559

4 Conclusões

O cubo mágico faz valer seu slogan sendo realmente difícil de resolver. Sendo um problema combinatória com um espaço de busca exponencial, resolver o cubo mágico testando todos os estados que cada sequência de movimentos existente pode levar se torna inviável. Portanto o uso de algoritmos de otimização não ótimos como o GA se fazem necessários para se obter um resposta razoavelmente voa em um tempo praticável.

Este trabalho expande o operador de mutação proposto por Herdy e Patone (1994) com o intuito de criar um sistema de otimização capaz de aprimorar soluções para cubos de qualquer. Enquanto o cubo de tamanho 3×3 foi solucionado com um *tunning* apropriado, cubos de tamanho maior ainda se mostram desafiadores e apesar de indivíduos serem otimizados na decorrência do processo evolutivo, o GA fica preso em ótimos locais e a solução para o cubo não é encontrada.

A busca por novos operadores de mutação e cruzamento que consigam ter bons resultados em cubos maiores que 3×3 é uma proposta de trabalho futuro interessante, bem como a modificação do GA proposto para que seja minimizado o número de movimentos realizados pela solução.

Referências

N El-Sourani. *Design and Benchmark of different Evolutionary Approaches to Solve the Rubiks Cube as a Discrete Optimization Problem.* PhD thesis, Diploma Thesis, WWU Muenster, Germany, 2009.

Nail El-Sourani e Markus Borschbach. Design and comparison of two evolutionary approaches for solving the rubik's cube. Em *International Conference on Parallel Problem Solving from Nature*, pages 442–451. Springer, 2010.

Hongwei Guo. A simple algorithm for fitting a gaussian function. *IEEE Signal Process. Mag.*, 28(5):134–137, 2011.

Michael Herdy e Giannino Patone. Evolution strategy in action. 10 es-demonstrations. Em *International Conference On Evolutionary Computation*, 1994.