Adriano Batista Prieto

Autovalores com Algoritmos Genéticos

Limeira

2015

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

Adriano Batista Prieto

Autovalores com Algoritmos Genéticos

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Tecnologia, na área de Tecnologia e Inovação.

Orientador: Prof. Dr. Vitor Rafael Coluci

Este exemplar corresponde à versão final da tese defendida pelo aluno Adriano Batista Prieto, e orientada pelo Prof. Dr. Vitor Rafael Coluci

_____

Limeira

2015

Inclua aqui o pdf com a ficha catalográfica fornecida.

Inclua aqui a Folha de aprovação.

Dedico esta tese à todo mundo.

Agradecimentos

Agradecimentos aqui.

"Escreva aqui a sua epígrafe"
(Citação)

Resumo

Colocar o resumos aqui.

Palavras-chaves: palavra-chave 1; palavra-chave 2; palavra-chave 3.

Abstract

Put abstract here.

Keywords: keyword 1; keyword 2; keyword 3.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Autovalores e o Quociente de Rayleigh

1.2 Algoritmos Genéticos

1.2.1 ONEMAX

Problema 1: ONEMAX, considerado o "*Hello World*" dos algoritmos genéticos.

Definir objetivo do problema.

Representação cromossomial: zeros e uns.

Fitness: soma dos genes.

Seleção: por roleta

Crossover: ponto único

Mutação: simples

Gráfico do comportamento do fitness.

Listar última geração, mostrando que o algoritmo chegou na solução.

1.2.2 Robocode com GA

Resumir o relatório final da disciplina de Inteligência Artificial.

1.3 NVidia CUDA

1.4 Objetivos

2 Autovalores e o Quociente de Rayleigh

Inseir apenas a álgebra linear necessária para o entendimento do método e da discussão.

3 Algoritmos Genéticos

4 Materiais e Métodos

Antes de atacar o método em si foi necessário estudo de Algoritmos Genéticos e Linguagem C, escolhida pensando em CUDA.

4.1 CUDA

Metodologia: revisão bibliográfica e aplicação no ONEMAX.

Material do ERAD.

4.2 Método

Artigo de 2004. Aproveitar conteúdo do segundo relatório de estudos dirigidos.

Apresentar o fitness *fitness* com $\rho - \rho_0$ utilizado no Artigo de 2006.

Apresentar e definir a matriz de Coope-Sabo (citar artigo original de 77), utilizada no pelos indianos no artigo de 2006.

4.3 Framework/Programa Serial

Optou-se por desenvolver do zero todo o programa. Motivo: controle sobre todas as características do método.

Linguagem C: linguagem nativa para CUDA.

Totalmente parametrizado.

Com foi será possível:

1. Estudo do método
2. Estudo de algoritmos genéticos
3. Novas aplicações como máximo de função
   a) Mudança no fitness
   b) Mudança nos critérios de parada

Descrição dos parâmetros.

Descrição de cada função, incluindo *print screen* das partes de código mais fundamentais:

1. Estruturas de dados
2. Geração de números pseudoaleatórios
3. Geração das Matrizes de Coope
4. Geração da População Inicial
5. Fitness: as várias equações
6. Fitness: cálculo de ρ_i
7. Fitness: cálculo de $\nabla\rho_i$
8. Seleção
9. Crossover
10. Mutação
11. Álgebra Linear: multiplicação de matrizes
12. Álgebra Linear: multiplicação de matriz por escalar
13. Álgebra Linear: subtração de matrizes

Qualidade dos números pseudo-aleatórios (base do GA)

1. Mostrar que a distribuição dos números segue o esperado para números aleatórios
2. Quanto maior a quantidade de pontos, melhor a distribuição
3. Gráfico: histograma de frequência.

Exemplo de execução no Windows.

Reprodutibilidade.

Exemplo de reprodutibilidade.

Código disponível em

`https://github.com/prietoab/msc_code`

5 Resultados e discussão

1. Parágrafo de introdução do capítulo. Citar que, basicmente, o leitor encontrará no capítulo:

   a) Resultados do ONEMAX, legitimando o uso do código para o programa mais complexo que foi utilizado no método dos indianos.

   b) o estudo dos tipos de *fitness*, operador responsável pelo elo entre o algoritmo e o problema (LINDEN, 2008), que, para o nosso caso, é encontrar autovalores. Ponte para o próximo: para cada tipo de *fitness*, um resultado diferente.

2. Os dois tipos de fitness dos indianos. Ideia central: dois tipos, resultados diferentes. Com $\nabla\rho$ chegamos a um autovalor qualquer, com $(\rho - \rho_0)^2$ podemos chegar ao mínimo, mas dá mais trabalho. Ponte para o próximo: proposta de dois novos fitness.

3. Combinação de $\nabla\rho$ com $(\rho - \rho_0)^2$. Se cada forma leva a comportamentos diferentes, tentamos combinar os dois termos em um único fitness. Uma hipótese seria a melhoria da qualidade dos resultados. A hipótese não foi confirmada. Ponte para o próximo: a busca pela qualidade levou à verificação da importância do parâmetro λ.

4. Além do que os indianos disseram, que λ é escolhido para não estourar a função exponencial, ele tem influência na convergência do algoritmo e na precisão (ou resolução) do fitness. Se na primeira população, geração inicial, o fitness médio é alto, isso provoca convergência precoce, fazendo com que o resultado final seja ruim. Por outro lado, se no início o fitness médio é muito baixo, não há muita discriminação entre os indivíduos, o fitness não cresce e não chegamos a uma solução. A medida que o fitness se aproxima de 1, a discriminação entre os indivíduos fica difícil, levando ao problema da resolução. Ponte para o próximo: vários testes levaram ao desenvolvimento de uma equação empírica para λ, restrita às matrizes de Coope−Sabo (COOPE; SABO, 1977).

5. Fórmula empírica. Por já conhecermos de antemão os autovalores das matrizes de Coope−Sabo, foi possível criar uma fórmula empírica para λ. Ela garante que na primeira população o *fitness* médio é baixo, previnindo o *underflow* do *fitness* e a convergência prematura.

5.1 Problemas com o mínimo global

Na seção 4.2 vimos que o *fitness* utilizado no artigo (NANDY *et al.*, 2004) foi

$$f_i = e^{-\lambda \|\nabla \rho_i\|^2}, \tag{5.1}$$

onde f_i é o *fitness* do i-ésimo indivíduo da população, λ é um parâmetro para evitar o estouro do *fitness* e $\|\nabla \rho_i\|^2$ é o módulo ao quadrado do vetor gradiente de ρ, dado por

$$\nabla \rho_i = \frac{2[H - \rho_i]C_i}{C_i^t C_i}, \tag{5.2}$$

em que C_i é um vetor candidato à solução do problema do autovalor

$$HC = EC. \tag{5.3}$$

Além disso, se C_i é de fato um dos autovetores, ρ é o autovalor associado E_i:

$$\rho_i = \frac{C_i^t H C_i}{C_i^t C_i} = E_i. \tag{5.4}$$

A fim de reproduzir os resultados, testamos o método com matrizes de Coope-Sabo de ordem 10, 20, 30 e 40, utilizando os mesmos parâmetros encontrados em (NANDY *et al.*, 2004): probabilidade de *crossover* $p_c = 75\%$, probabilidade de mutação $p_m = 50\%$ e intensidade de mutação $\Delta = 0,01$. Com um bom ajuste de λ, que será discutido em detalhes posteriormente, o *fitness* comportou-se conforme o esperado em todos os casos. Um exemplo está na figura 1, que apresenta o melhor fitness de cada geração para uma matriz de ordem N = 10. Na primeira geração o melhor *fitness* é pequeno, aproximadamente 0,1, cresce rapidamente e a partir da décima geração está próximo de 1.

O próximo passo foi verificar o comportamento de ρ, o Quociente de Rayleigh, e, especificamente, sua convergência para o menor autovalor E_0. Ainda conforme (NANDY *et al.*, 2004), obteríamos uma curva semelhante à da figura 1, mas invertida, ou seja, os primeiros valores de ρ seriam grandes e, rapidamente, diminuiriam até haver convergência para o autovalor mínimo. Na figura 2 há um exemplo. Os gráficos exibem os valores de ρ para a mesma execução apresentada na figura 1. Note no primeiro gráfico que até a geração 20 o quociente ρ teve caráter oscilatório e, então, aparentemente estabilizou-se entre 6 e 8, valores muito superiores ao autovalor mínimo para essa matriz, $E_0 = 0,38675$. Entretanto, ainda no primeiro gráfico, observa-se que há uma tendência de queda do ρ entre as gerações 40 e 50 e, portanto, existiria a possibilidade do algoritmo convergir para E_0. Porém, para esse exemplo especificamente, isso não aconteceu, como pode ser visto no segundo gráfico da figura 2. Para garantir a estabilidade, o programa foi executado até a

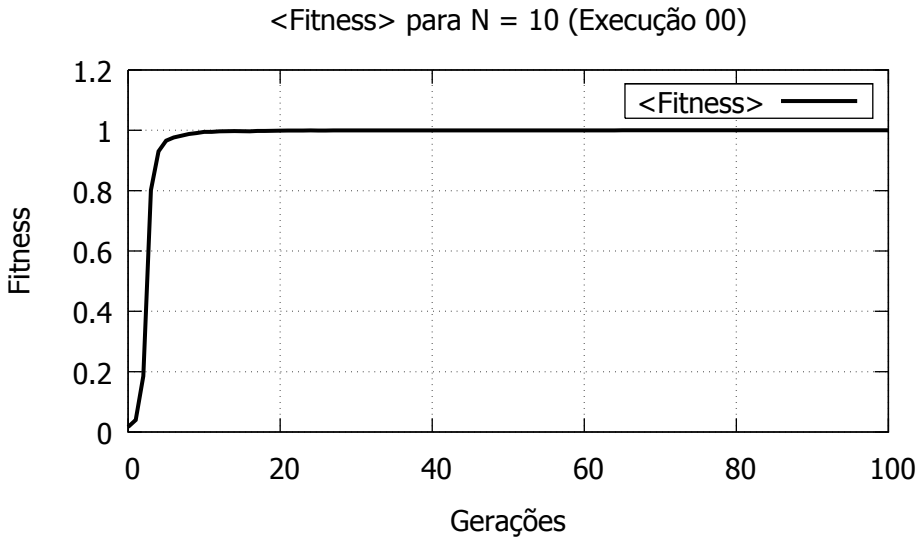

Figura 1 – Comportamento do *fitness* $f_i = e^{-\lambda\|\nabla\rho_i\|^2}$ para N = 10. Na primeira geração o melhor *fitness* é pequeno, aproximadamente 0,1, cresce rapidamente e a partir da décima geração está próximo de 1.

geração 400.000, e o valor médio obtido foi $< \rho >= 6,572898$. Para nossa surpresa, além do valor obtido de $< \rho >$ não ser o mínimo, ele não é um valor qualquer, mas corresponde, com erro menor que $0,00002\%$, ao quarto autovalor da matriz, $E_3 = 6,572897$. Um gráfico expandido dessa execução está na figura 3 da página 20. Pensamos, então, que poderia haver algo de errado com o nosso programa.

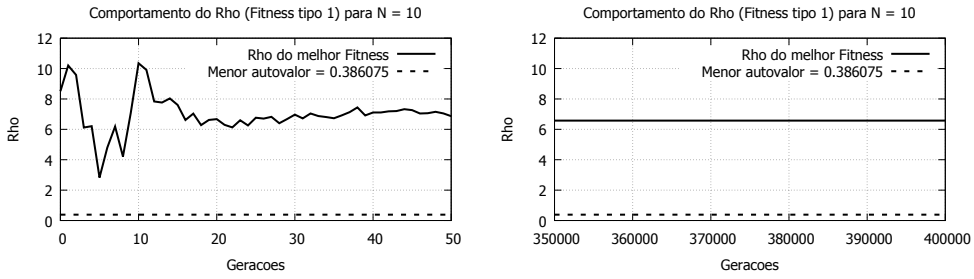

Figura 2 – Comportamento de ρ (Quociente de Rayleigh) para uma matriz de Coope−Sabo de ordem 10.

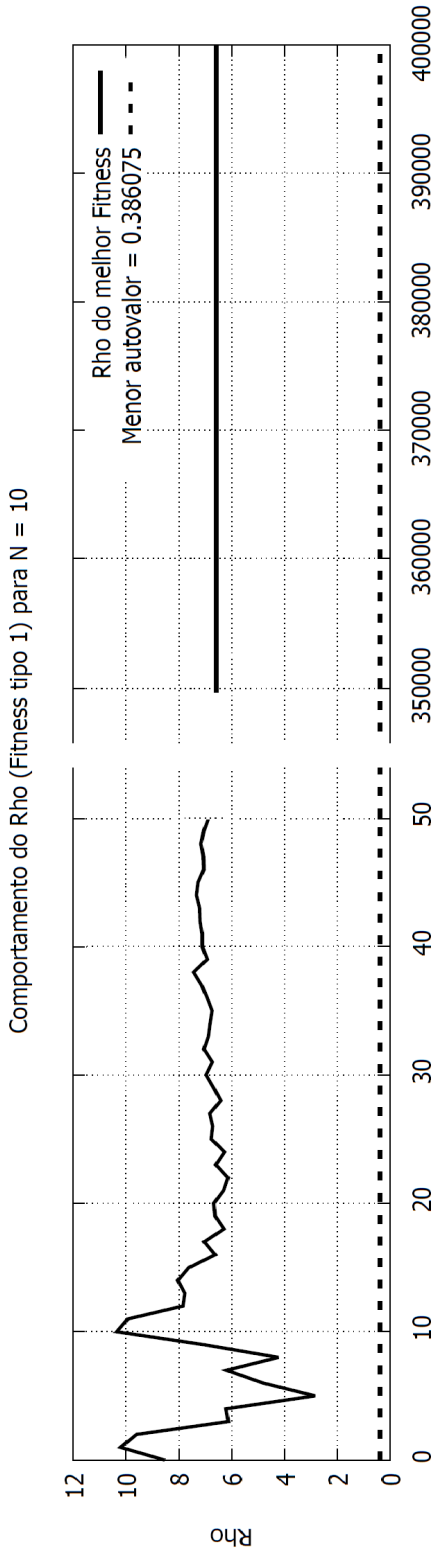

Figura 3 – Comportamento de ρ (Quociente de Rayleigh) para uma matriz de Coope−Sabo de ordem 10.

Após esses resultados preliminares executamos uma validação cuidadosa do programa, testando cada uma de suas quase 2500 linhas e comparando os resultados das operações e cálculos com os softwares Excel (Microsoft Corporation, 2007) e SciLab (Scilab Enterprises, 2012). A hipótese era a de que erros numéricos, principalmente nas funções de álgebra linear e nos operadores genéticos, pudessem ter levado ao comportamento incorreto da não convergência para o menor autovalor. De fato alguns erros foram encontrados.

Discutiremos a seguir os testes com a versão corrigida do programa e exibidos nas figuras 14, 15, 16 e 17. Visando brevidade, apresentaremos dados para matrizes de CoopeSabo de ordem 10, 20, 30 e 40 apenas, sem perda de generalidade. Foram cinco execuções para cada matriz, até a geração 400.000, gerando sempre dois gráficos, um do *fitness* médio (<*fitness*>) e outro do Quociente de Rayleigh médio (<ρ>), ambos em função do número de gerações, e dando ênfase às primeiras 100 gerações. Essas escolhas, número máximo da geração e uso de médias sobre cada população, visaram garantir, respectivamente, a convergência genética e boa precisão. A exibição de apenas as primeiras 100 gerações tem como objetivo olhar em detalhe (com *zoom*) o período em que o *crossover* tem mais peso, ou seja, onde há geralmente os saltos no espaço de soluções de um Algoritmo Genético. Em todos os gráficos de $< \rho >$ há indicado nas legendas o autovalor mínimo E_0 e o autovalor obtido após as 400.000 gerações (E_{obtido}). Na tabela 3 há a lista de todos os autovalores. Por exemplo, para uma matriz de ordem $N = 10$, o menor autovalor é E_0 =0,386075, e o quinto autovalor para $N = 30$ é $E_4 = 8{,}450274$.

Comecemos a discussão com o que foi encontrado em todas as execuções. Em qualquer gráfico do *fitness* observa-se estabilidade do comportamento conforme esperado pelo método: no início seu valor é baixo, próximo de zero, cresce rapidamente nas primeiras gerações e fica estável próximo de $< fitness >= 1$. Com relação ao ρ, há sempre oscilações, sejam pequenas variações em torno de uma clara linha de tendência, como na execução 02 para N = 10, ou grandes saltos, como nas execuções 05 de N = 20 e 05 de N = 30. Novamente, o menor autovalor não foi obtido em nenhuma execução, contradizendo os resultados de (NANDY *et al.*, 2004), mas, por outro lado, o algoritmo sempre encontrou algum autovalor.

De fato, verificando os dados da tabela 1, concluímos que tais valores não devem ser coincidência. Para todas as execuções o *fitness* médio chegou ao valor máximo ($< f >= 1,000000$). As médias de ρ sobre todos os indivíduos da última população possuem baixo desvio padrão ($\sigma <$ 0,0001), indicando que eles são muito parecidos entre si e que o algoritmo atingiu a convergência genética. Ou seja, não há variabilidade genética suficiente na população para alterar o rumo da busca de modo a atingir o menor autovalor, ou o mínimo global. Portanto, o algoritmo chegou em um mínimo local, corroborado pelos baixos erros relativos de $< \rho >$ quando comparado com o autovalor mais próximo. Por

exemplo, para N = 30, execução 4, $< \rho > = 40{,}772447$, correspondendo, com erro relativo absoluno menor que $0,001\%$, ao vigésimo primeiro autovalor, $E_{20} = 40,772850$. Apesar das evidências descritas acima, até esse ponto ainda há dúvidas sobre a validade do nosso programa e, obviamente, dos resultados produzidos. Então, buscamos embasamento mais rigoroso.

De acordo com (NANDY *et al.*, 2004), se algum C_i, em algum momento, é o autovetor fundamental (associado ao menor autovalor), o $\nabla\rho$ é nulo. Com o *fitness* da equação (5.1) os autores afirmam que "*Claramente,* $f_i \to 1$ *quando* $\nabla\rho_i \to 0$*, sinalizando que a evolução atingiu o verdadeiro autovetor fundamental de* H *em* C_i"[1]. Há duas relações distintas de causalidade nessa frase, e acreditamos que nelas residam a explicação dos resultados obtidos por nós até agora.

A primeira relação de causalidade refere-se à afirmação "$f_i \to 1$ *quando* $\nabla\rho_i \to 0$", que está absolutamente correta. Retomando a seção 4, o *fitness* definido pela equação 5.1 é limitado no intervalo (0,1] e, como $\lambda > 0$, só chega ao seu valor máximo quando $\nabla\rho_i = 0$. Em outras palavras, $\nabla\rho_i \to 0$ implica $f_i \to 1$.

Na afirmação "(...) *sinalizando que a evolução atingiu o verdadeiro autovetor fundamental de* H *em* C_i" reside a segunda relação de causalidade que, apesar de sutil, é muito poderosa:

$$\text{Se } f_i \to 1,\ C_i = C_0. \tag{5.5}$$

Ou seja, sempre que algum indivíduo C_i, de qualquer população, possuir *fitness* muito próximo de 1, isso implica que, além de ter uma excelente "nota", ele, ainda por cima, é um vetor especial, o autovetor fundamental C_0. Portanto, possui autovalor associado E_0, o autovalor mínimo (conforme equação 5.4). Grosso modo, $f_i(C_i) = 1$ implica que $C_i = C_0$ e que podemos obter $E_0(C_0)$:

$$f_i(C_i) = 1 \to C_i = C_0 \to E_0(C_0). \tag{5.6}$$

As relações de causa e efeito da equação acima estão erradas. Em sua obra clássica sobre o problema de autovalores em matrizes simétricas, (PARLETT, 1998) abre o capítulo introdutório frisando que "*em muitos lugares no livro, é feita referência a fatos mais ou menos bem conhecidos sobre a teoria de matrizes*"[2]. Conforme já dito no capítulo 2, um desses fatos diz que $\rho(u)$ é estacionário, ou seja, $\nabla\rho(u) = 0$, apenas se o vetor u é

[1] Tradução livre de "*Clearly,* $f_i \to 1$*, as* $\nabla\rho_i \to 0$*, signalling that the evolution has hit the true ground state eigenvector of* H *in the vector* C_i".

[2] Tradução livre de "*At many places in the book, reference is made to more or less well known facts from matrix theory*".

um autovetor w de $HC = EC$. Consequentemente, o encadeamento correto se apresenta como:

$$C_i \text{ é um autovetor} \rightarrow \nabla\rho(C_i) = 0 \rightarrow f_i = 1. \tag{5.7}$$

Então, se $f_i = 1$, o máximo que podemos concluir é que C_i é *algum* autovetor, e não necessariamente *o* autovetor fundamental. Ao fim de todos os nossos testes o *fitness* médio foi $<f> = 1$, a população final era composta por autovetores e foi possível, com boa precisão, obter os autovalores relacionados (não necessariamente o autovalor mínimo). Nossos dados confirmam a matemática e, assim, acreditamos que nosso programa não contém erros.

Apesar de não chegar ao mínimo, o método pode ser utilizado de maneira exploratória com relativa facilidade, bastando extrair ρ sempre que $f_i \rightarrow 1$ e $\nabla\rho \rightarrow 0$.

Resta a dúvida: afinal, como o autovalor mínimo foi obtido com o *fitness* definido pela equação 5.1? Não sabemos. Esse *fitness* foi utilizado não só em (NANDY *et al.*, 2004), mas também em (SHARMA *et al.*, 2006), (SHARMA *et al.*, 2008) e (NANDY *et al.*, 2009), seguindo exatamente o argumento resumido pela equação 5.6. Não identificamos nada nesses quatro artigos que pudesse levar à resposta. Seguimos o estudo com uma nova definição do *fitness* encontrada em (NANDY *et al.*, 2011).

Tabela 1 – Execuções para matrizes de Coope−Sabo.

N	**Execução**	**Semente**	λ	**<*Fitness*>**	$<\rho>$	σ	**# autovalor**	**Autovalor**	**Erro relativo**
10	0	1445738835	0,128788	1,000000	2,461122	0,000023	1	2,461056	0,003%
10	1	1445780626	0,128788	1,000000	6,572898	0,000013	3	6,572897	0,00001%
10	2	1445780762	0,128788	1,000000	6,572883	0,000015	3	6,572897	-0,0002%
10	3	1445780907	0,128788	1,000000	6,572910	0,000016	3	6,572897	0,0002%
10	4	1445781049	0,128788	1,000000	12,765701	0,000016	6	12,765740	-0,0003%
10	5	1445781195	0,128788	1,000000	4,518952	0,000012	2	4,518931	0,0005%
20	1	1445795292	0,026665	1,000000	8,498192	0,000052	4	8,497626	0,007%
20	2	1445795501	0,026665	1,000000	12,551830	0,000018	6	12,551780	0,0004%
20	3	1445795718	0,026665	1,000000	12,551878	0,000020	6	12,551780	0,0008%
20	4	1445795953	0,026665	1,000000	14,578527	0,000035	7	14,578450	0,0005%
20	5	1445796166	0,026665	1,000000	18,634220	0,000062	9	18,633850	0,002%
30	1	1445796378	0,011171	1,000000	26,616790	0,000065	13	26,616670	0,0005%
30	2	1445796746	0,011171	1,000000	26,616595	0,000029	13	26,616670	-0,0003%
30	3	1445797109	0,011171	1,000000	22,580060	0,000051	11	22,580300	-0,001%
30	4	1445797473	0,011171	1,000000	40,772447	0,000071	20	40,772850	-0,001%
30	5	1445797882	0,011171	1,000000	30,655283	0,000022	15	30,655270	0,00004%
40	1	1445798248	0,006105	1,000000	26,554758	0,000040	13	26,554690	0,0003%
40	2	1445798838	0,006105	1,000000	54,773734	0,000078	27	54,773690	0,00008%
40	3	1445799429	0,006105	1,000000	58,819413	0,000087	29	58,819810	-0,0007%
40	4	1445800091	0,006105	1,000000	40,651473	0,000077	20	40,651140	0,0008%
40	5	1445800683	0,006105	1,000000	40,650764	0,000061	20	40,651140	-0,0009%

5.2 Outro *fitness* para encontrar o mínimo global

O novo *fitness*, apresentado em (NANDY *et al.*, 2011), é dado por

$$f_i = e^{-\lambda(\rho_i - E_L)^2}, \tag{5.8}$$

e apresenta semelhanças com o definido pela equação 5.1. Há uso de uma exponencial, o parâmetro λ foi mantido e possui exatamente o mesmo papel, f_i depende apenas de ρ e, como $(\rho_i - E_L)^2$ é claramente positivo, o *fitness* continua limitado ao conjunto (0,1]. As diferenças estão na ausência do $\nabla\rho$ e na inclusão do parâmetro E_L, que representa um limite inferior para o *menor* autovalor que estamos procurando[3]. Por exemplo, se soubermos de antemão que o autovalor *mínimo* é maior que zero, poderíamos definir $E_L = 0$.

A justificativa para o funcionamento do método em (NANDY *et al.*, 2011) segue a mesma estrutura de (NANDY *et al.*, 2004): "*Se* $\rho_i \to E_L$ *durante a busca,* $f_i \to 1$ *e* C_i *está próximo do autovetor fundamental de* H"[4]. Parece que, outra vez, não há garantia de que, se $f_i \to 1$, ρ tende, necessariamente, ao autovalor fundamental. E aqui há um agravante: nada na equação 5.8 está diretamente associado aos autovalores de H. Lembre-se que o *fitness* anterior (equação 5.1) contém $\nabla\rho$, que possui relação direta com os autovalores de H quando $\nabla\rho = 0$.

Repeti as execuções da tabela 1 alterando apenas o *fitness* e configurando o parâmetro E_L para $E_L = 0$, um pouco abaixo dos autovalores mínimos. Os resultados estão na página 26, e os gráficos da evolução do *fitness* e do quociente de Rayleigh estão nas páginas 47, 48, 49 e 50. Surpreendentemente, apesar do que foi dito no parágrafo anterior, o programa encontrou o menor autovalor em **todos** os casos. Assim como nas primeiras execuções, o desvio padrão (σ) da média de ρ na última geração (400.000) foi pequeno, indicando convergência genética. Entretanto, essa foi a única semelhança. Os próprios valores de σ são uma ordem de grandeza menores, sugerindo que os indivíduos são mais semelhantes entre si. O *fitness* médio só atingiu seu valor máximo para a matriz de ordem $N = 40$. Aliás, especificamente para E_L fixado em $E_L = 0$, o <*fitness*> final diminui com N, pois E_L está mais distante de E_0 na matriz de ordem 10 do que na de ordem 40. Os erros relativos não ultrapassaram 1%, mas foram substancialmente maiores comparados aos obtidos com o primeiro *fitness*. Enquanto nos testes anteriores seus valores permaneceram estáveis, agora os erros relativos apresentaram tendência de crescimento com N.

---

[3] L de *lower*.

[4] Tradução minha para "*If* $\rho_i \to E_L$ *during the search,* $f_i \to 1$ *and* C_i *approaches the ground eigenvector of* H".

Tabela 2 – Execuções novo *Fitness*.

N	**Execução**	**Semente**	λ	**<Fitness>**	$<\rho>$	σ	**# autovalor**	**Autovalor**	**Erro relativo (%)**
10	0	1445738835	0,128788	0,999044	0,386176	0,00005	0	0,3860745	0,03%
10	1	1445780626	0,128788	0,999044	0,386169	0,00003	0	0,3860745	0,02%
10	2	1445780762	0,128788	0,999045	0,386132	0,00002	0	0,3860745	0,01%
10	3	1445780907	0,128788	0,999044	0,386175	0,00005	0	0,3860745	0,03%
10	4	1445781049	0,128788	0,999043	0,386211	0,00003	0	0,3860745	0,04%
10	5	1445781195	0,128788	0,999044	0,386183	0,00005	0	0,3860745	0,03%
20	1	1445795292	0,026665	0,999954	0,341484	0,00005	0	0,3412367	0,07%
20	2	1445795501	0,026665	0,999954	0,341693	0,0001	0	0,3412367	0,1%
20	3	1445795718	0,026665	0,999954	0,34147	0,00006	0	0,3412367	0,07%
20	4	1445795953	0,026665	0,999954	0,341689	0,0001	0	0,3412367	0,1%
20	5	1445796166	0,026665	0,999954	0,34153	0,00007	0	0,3412367	0,09%
30	1	1445796378	0,011171	0,999995	0,320582	0,0001	0	0,319737	0,3%
30	2	1445796746	0,011171	0,999995	0,320772	0,0002	0	0,319737	0,3%
30	3	1445797109	0,011171	0,999995	0,320699	0,0001	0	0,319737	0,3%
30	4	1445797473	0,011171	0,999995	0,320755	0,0001	0	0,319737	0,3%
30	5	1445797882	0,011171	0,999995	0,320274	0,00007	0	0,319737	0,2%
40	1	1445798248	0,006105	1	0,306968	0,0001	0	0,306086	0,3%
40	2	1445798838	0,006105	1	0,307128	0,0001	0	0,306086	0,3%
40	3	1445799429	0,006105	1	0,307297	0,0002	0	0,306086	0,4%
40	4	1445800091	0,006105	1	0,307816	0,0002	0	0,306086	0,6%
40	5	1445800683	0,006105	1	0,30765	0,0002	0	0,306086	0,5%

As diferenças dos valores finais são indiscutíveis, sugerindo que o comportamento do *fitness* e do ρ ao longo da busca também deve ter sido alterado. Na figura 4 estão os gráficos referentes à execução zero para o Hamiltoniano de ordem 10, semente 1445738835. A primeira usa o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$, que chega ao autovalor mínimo, enquanto a segunda utiliza o $f_i = e^{-\lambda\|\nabla\rho_i\|^2}$. Ambos saem de valores muito baixos e convergem para 1, entretanto, o da esquerda é muito ruidoso e, aparentemente, essa é a causa da convergência mais lenta. Quando a curva da direita já está estável em $<f> \approx 1$ em torno da geração de número 15, a da esquerda ainda não ultrapassou o $<f> = 0,1$. A princípio, não podemos comparar os dois comportamentos diretamente, visto que cada um chegou em um autovalor diferente. A execução da direita, lembre-se, obteve apenas um mínimo local ($E_1 = 2,461056$, tabela 1).

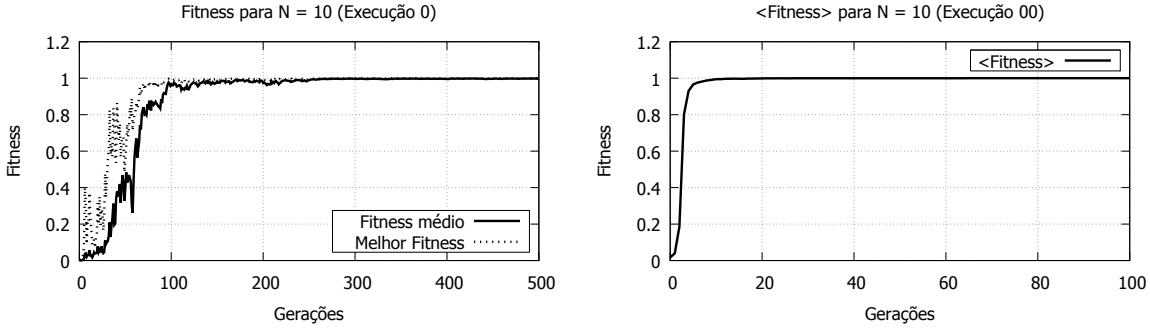

Figura 4 – Comportamento do *fitness* para as execuções zero do Hamiltoniano de ordem 10, semente 1445738835. A primeira usa o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$, que chega ao autovalor mínimo, enquanto a segunda utiliza o $f_i = e^{-\lambda\|\nabla\rho_i\|^2}$.

De todo modo, as duas execuções estão conectadas pois, como partiram da mesma semente de números pseudoaleatórios, a população inicial foi *exatamente* a mesma. Inclusive, na primeira geração, em ambas as execuções, os valores para $< \rho >$ e para o melhor ρ foram, respectivamente, $9,876075$ e $9,557892$, igualmente distantes do autovalor mínimo $E_0 = 0,386075$. Os gráficos da figura 5 permitem comparar a evolução do $< \rho >$ nos dois casos. Assim como na figura anterior, a imagem da esquerda refere-se ao uso do *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

É tentador afirmar que a causa de uma execução ter sido mais lenta do que a outra foi porque percorreu um caminho mais longo ao sair de $< \rho >= 9,876075$, passar por $E_1 = 2,461056$ e continuar até encontrar $E_0 = 0,386075$, enquanto a mais rápida saiu do mesmo $< \rho >$ e parou logo que encontrou E_1. Infelizmente essa conclusão estaria incorreta. A maneira como os Algoritmos Genéticos viajam no espaço de soluções tem forte base estocástica e, portanto, qualquer comparação linear é extremamente arriscada, quiçá impossível. Objetivamente, posso apenas concluir que os valores finais encontrados por cada *fitness* estão condizentes com a construção de cada função objetivo: $\nabla\rho_i$ leva a qualquer autovalor; $\rho_i - E_L$ encontra o autovalor mínimo.

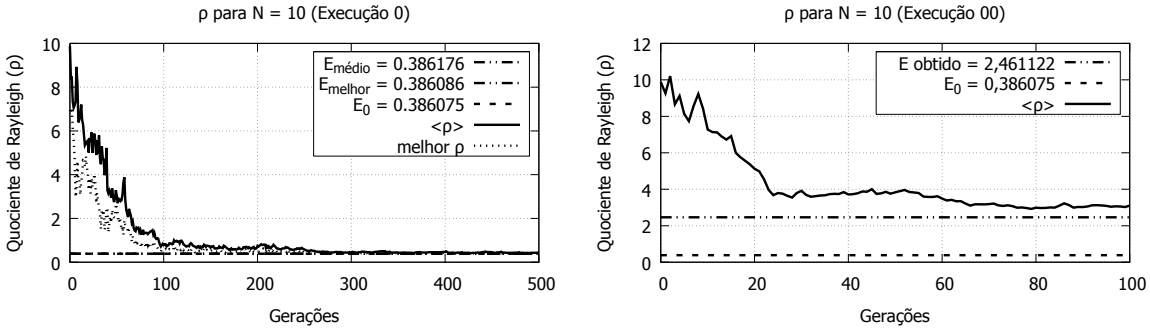

Figura 5 – Comportamento do ρ para as execuções zero do Hamiltoniano de ordem 10, semente 1445738835. A primeira usa o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$, que chega ao autovalor mínimo, enquanto a segunda utiliza o $f_i = e^{-\lambda\|\nabla\rho_i\|^2}$.

Ok. Os dados mostraram que chega no menor autovalor. Mas, se não há relação direta no fitness, como isso acontece? Outra sutileza: E_L é um limite inferior para o *menor* autovalor. Roubada? Não parece muito útil, pois o fitness só foi próximo de 1 porque escolhi um E_L bem próximo de E_0. Mas, o que aconteceria se eu não soubesse por onde anda ou autovalor mínimo? Quatro cenários para E_L. Cenário 1: com sorte, o E_L escolhido está um pouco abaixo do E_0. Encontra o valor mínimo, conforme exemplos. Cenário 2: um pouco acima de E_0. Cenário 3: muito abaixo de E_0; Cenário 4: muito acima de E_0.

coloar as figuras dos cenários aqui.

Para as execuções da semente 1445738835.

Semente 1445738835. Tipo 1. EL um pouco acima.

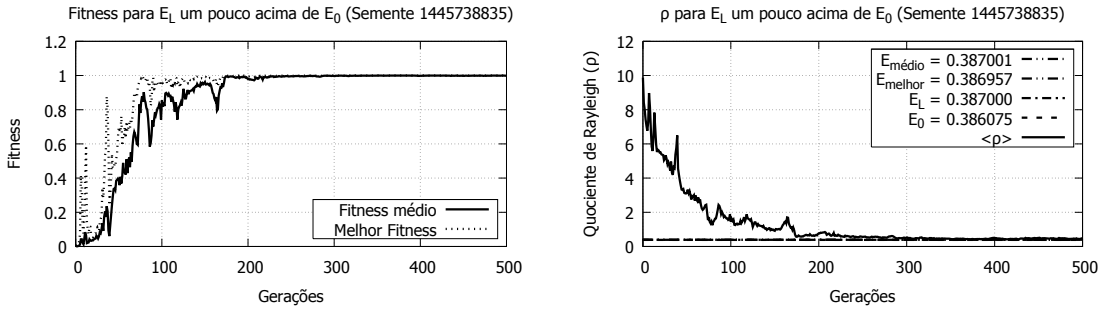

Figura 6 – Execução para a semente 1445738835. E_L um pouco acima de E_0 no *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

Semente 1445738835. Tipo 3.

Semente 1445738835. Tipo 4. Até 500 gerações.

Semente 1445738835. Tipo 4. Entre 30.000 e 40.000 gerações.

1. **O que acontece quando ρ_0 está um pouco acima de E_0?**

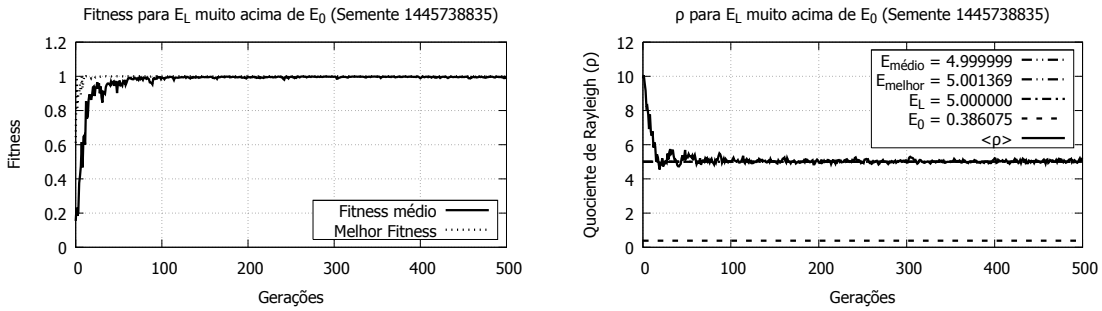

Figura 7 – Execução para a semente 1445738835. E_L muito acima de E_0 no *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

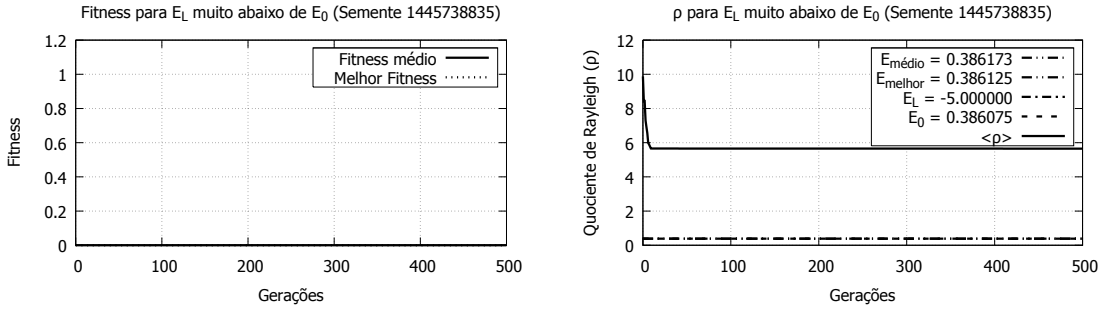

Figura 8 – Execução para a semente 1445738835. E_L muito abaixo de E_0 no *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$. Até geração 500.

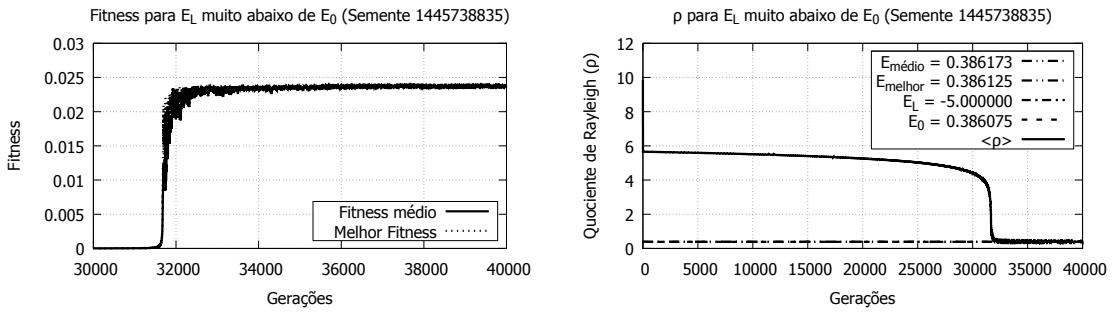

Figura 9 – Execução para a semente 1445738835. E_L muito abaixo de E_0 no *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$. Geração entre 30.000 e 40.000.

Para com $< \rho > \approx \rho_0$

Não encontra o menor autovalor (E_0).

Porém, verifica-se que o $< \nabla\rho > \approx 0$.

Então, estamos próximos ao menor autovalor.

2. **O que acontece quando ρ_0 está um pouco abaixo de λ_0**

   Encontra uma aproximação para o autovalor mínimo (λ_0).

   $< fitness >$ é próximo a 1 pois $< \rho >$ é próximo de ρ_0.

   $< \nabla\rho > \approx 0$.

3. **O que acontece quando ρ_0 está muito acima de E_0?**

   Acredito que não encontrará uma aproximação para E_0, pois a região de busca está muito distante. (verificar)

   Para com $< \rho > \approx \rho_0$

   Verifica-se que o $\nabla\rho >> 0$. Portanto, estamos distantes do menor autovalor.

4. **O que acontece quando ρ_0 está muito abaixo de E_0?**

   Acredito que não encontrará uma aproximação para E_0, pois a região de busca está muito distante.

   Para com $< \rho > \approx \rho_0$

   Verifica-se que o $\nabla\rho >> 0$. Portanto, estamos distantes do menor autovalor.

Citar brevemente a relação entre a dificuldade (parece uma inércia) de melhorar a precisão dos resultados no final, dizer que isso está relacionado com o formato da função *fitness*.

5.3 $f_i = e^{-\lambda(\rho_i - \rho_0)^2}$ leva ao autovalor mínimo (E_0), mas devemos saber aproximadamente onde ele está

Ponte para a discussão do erro *fitness*

5.4 $f_i = e^{[-\lambda\nabla\rho]}$ é mais rápido do que $f_i = e^{[-\lambda(\nabla\rho)^2]}$

Como um dos critérios de parada utiliza $\nabla\rho$ (sem quadrado), testamos essa forma no fitness.

Várias execuções.

Gráfico comparando o comportamento (um termina mais rápido)

Tabela com os detalhes explícitos do do ganho.

Ponte pra falar sobre o outro fitness que encontra o mínimo.

5.5 $f_i = e^{-\lambda(\rho_i - \rho_0)^2}$ próximo de ρ é intrinsicamente impreciso

Como λ e ρ_0 são constantes por definição, f_i é uma função apenas de ρ.

Função simétrica em torno de ρ_0.

Gráfico com diferentes ρ_0's para o domínio $\rho = [-200, 200]$.

Para uma dada função $fitness$, em torno de zero, por exemplo, dar um *zoom* no pico.

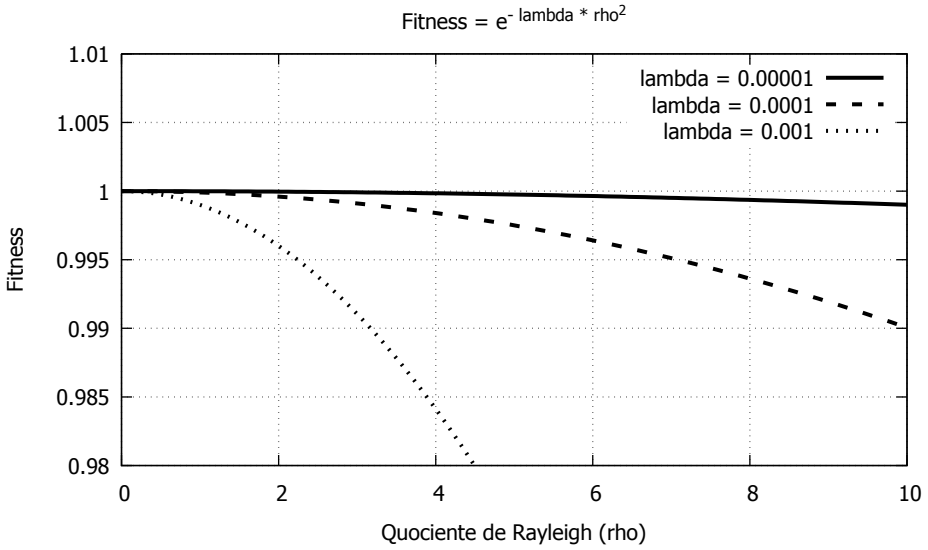

Figura 10 – Zoom próximo do pico

Gráficos com $\rho = [-5, 5]$ e $\rho = [-0.1, +0.1]$.

Verificar que no último gráfico o SciLab já apresenta dificuldades para diferenciar os pontos de f.

Gráfico da variação do fitness como função da variação do ρ (a derivada).

A derivada é

$$\frac{df}{d\rho} = -2\lambda(\rho - \rho 0)e^{-\lambda(\rho-\rho 0)^2}$$

Com $\rho = [-200, 200]$, gráfico da derivada.

Gráfico comparando f com $df/d\rho$. Verificar visualmente que $df/d\rho$ é muito pequeno, próximo de zero.

Tabela com os valores mostrando que, próximo à ρ_0, uma mudança de $\Delta\rho = 0.1$ leva a uma variação de $\Delta f <= 0.000001$.

Para o algoritmo genético isso é ruim. Cada rho está associado a um indivíduo e o fitness deve diferenciar cada um, de modo que a maior nota é dada ao indivíduo mais próximo da solução.

Veja que para $\rho = [-0.3, +0.3]$ o fitness é igual a 1. Nessa região a o fitness falha na diferenciação dos indivíduos.

Para fitness muito pequeno há o mesmo problema. Com isso, pode-se definir uma região do fitness onde a função é apropriada para o uso dos algoritmos genéticos. Figura 11.

Para Futuro, continuação do trabalho: há alguma maneira de lidar com os parâmetros de f de modo a ficarmos sempre na região boa?

Região boa: entre as duas linhas, quase linear.

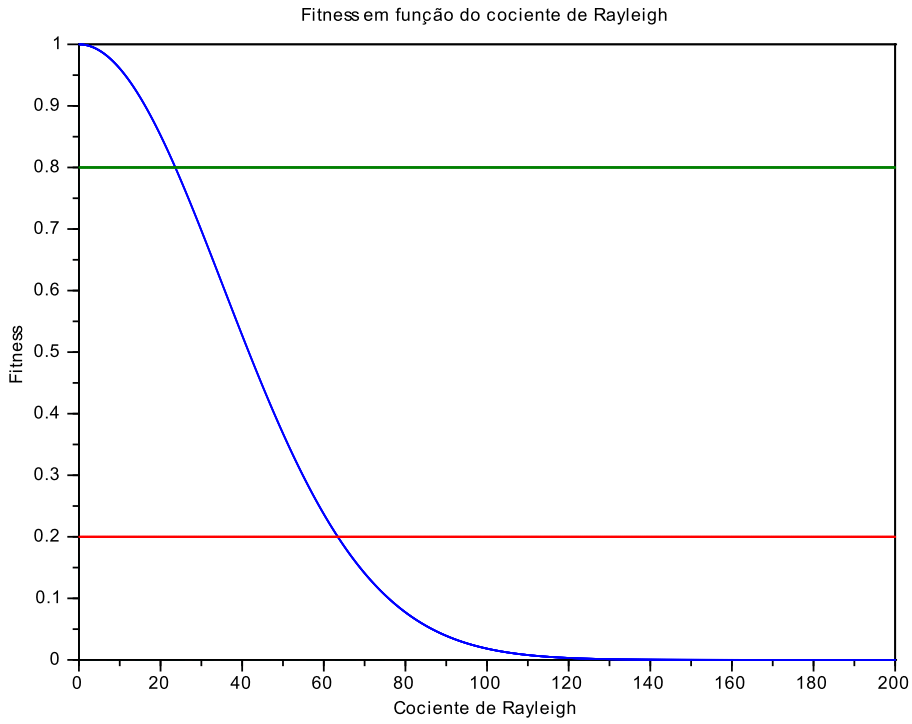

Figura 11 – Boa região para o fitness (entre as duas linhas)

Há um limite de precisão intrínseco com o uso desse fitness.

Por isso a dificuldade em melhorar a precisão.

rho	*f*
-0,1	0,999996
-0,09	0,999997
-0,08	0,999997
-0,07	0,999998
-0,06	0,999999
-0,05	0,999999
-0,04	0,999999
-0,03	1
-0,02	1
-0,01	1
0	1
0,01	1
0,02	1
0,03	1
0,04	0,999999
0,05	0,999999
0,06	0,999999
0,07	0,999998
0,08	0,999997
0,09	0,999997
0,1	0,999996

O parâmetro λ também influencia fortemente o fitness e o algoritmo genético.

Ponte para a discussão do λ.

5.6 Por que o λ deve ser escolhido cuidadosamente?

Execuções para N=10 com diferentes λ's. Com os gráficos, explicar o que o artigo de 2004 quis dizer com *fitness overflow/underflow*.

Gráficos com rho entre 0 e 250 (exemplo pra N=10), mas com cortes em diferentes rhos.

Explicar que uma boa escolha do λ deve cobrir todos os autovalores. Citar as execuções anteriores (boas e ruins em função de cada λ).

Gráfico com λ fazendo o fitness cortar em um ρ muito baixo. Discutir puxando as execuções anteriores.

Outro gráfico, mas com λ fazendo o fitness cortar em um ρ muito alto. Discutir puxando as execuções anteriores.

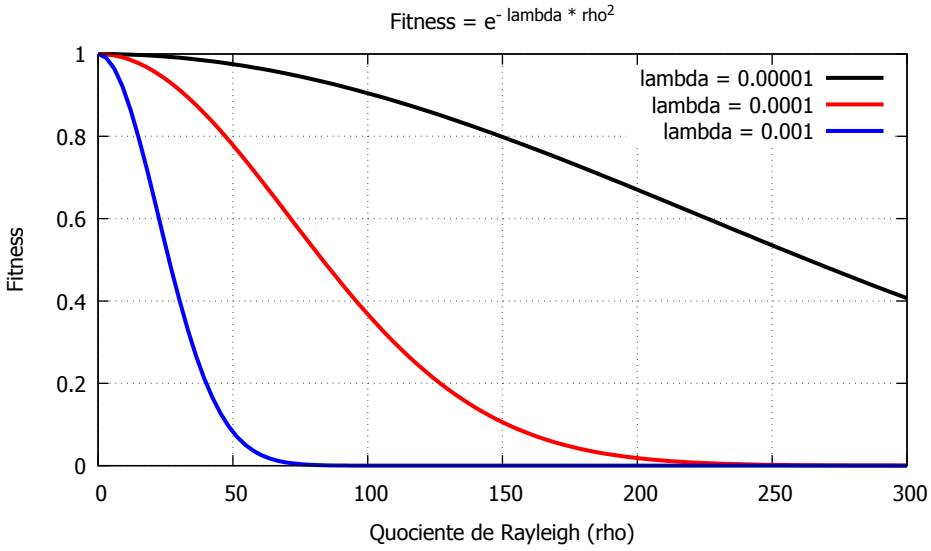

Figura 12 – Fitness em função do lambda – colorido

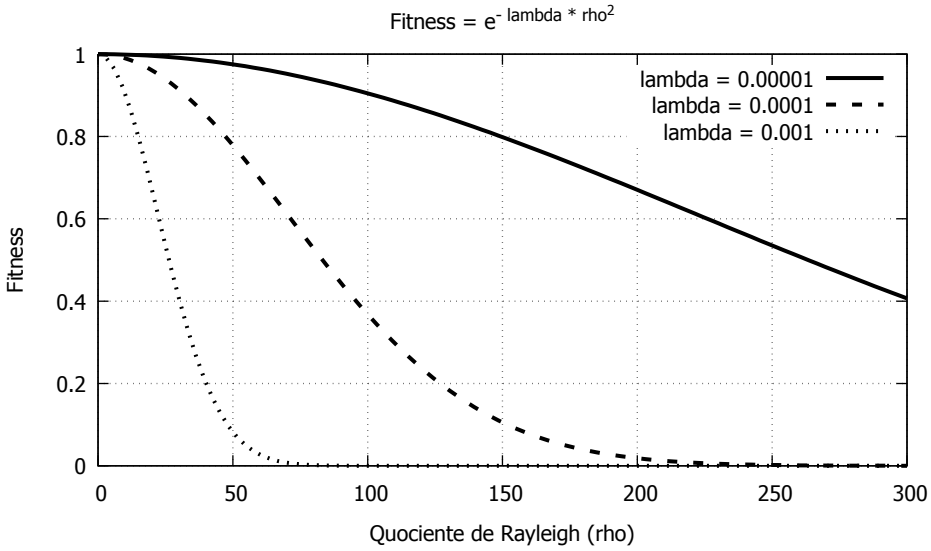

Figura 13 – Fitness em função do lambda

Gráfico com uma boa escolha de λ. Discutir puxando as execuções anteriores.

Após estimativa, refinar a obtenção do λ. Alterar o *lambda* (valores em torno da estimativa), executar o programa para verificar se o fitness médio da primeira população é baixo. (se a população inicial tem fitness muito grande, há convergência prematura).

Tabela com alguns *lambdas* encontrados dessa maneira (estimativa e refinamento).

Infelizmente, para cada matriz, um λ diferente.

Ponte pra equação empírica do λ.

5.7 Equação empírica para o λ

Delineamento da equação como feito na reunião de 29/09.

Isolar λ a partir da $f = e^{-\lambda*(\rho-\rho_0)^2}$

Fazer $f = 0.00001 \approx 0$.

Substituir $(\rho - \rho_0)^2$ por $E_{central} - E_{mínimo}$. Justificar.

Regressão linear para $E_{central} - E_{mínimo}$ com função apenas da ordem da matriz (N).

Inserir a Equação obtida na regressão na equação de λ.

Fator 0.65: obtido empiricamente de modo que o λ seja semelhante aos encontrados pelo processo de estimativa e refinamento.

Exemplo de execução com λ automático.

Explicitar que essa equação é válida apenas para matrizes de Coope−Sabo. Apesar disso, foi importante para o estudo pois permitiu automação completa.

5.8 A mistura de $(\rho - \rho_0)^2$ com $\nabla\rho$ não leva a melhores resultados

Como em seção anterior verificamos que $f_i = e^{[-\lambda\nabla\rho]}$ é mais rápido do que $f_i = e^{[-\lambda(\nabla\rho)]}$, e que o $\nabla\rho$ está diretamente associado aos autovalores, pensei na seguinte hipótese: inserir $\nabla\rho$ ao fitness com $(\rho - \rho_0)^2$ traria resultados mais rápidos.

Justificativas para a hipótese:

1. Inserir $\nabla\rho$ no fitness puniria os ρ's que, apesar de próximos de ρ_0, não fossem autovalor. Em outras palavras, o termo $\rho - \rho_0 \approx 0$, mas $\nabla\rho >> 0$ e, portanto, o fitness ficaria pequeno.

2. Como o fitness, a princípio, estaria diferenciamento melhor os bons indivíduos, o algoritmo teria uma taxa de convergência maior.

Executar 10 para o primeiro fitness, e, utilizando as mesmas dez sementes, executar outros 10 testes com ou outro fitness.

Comparação dos resultados: gráficos do comportamento do fitness e tabela comparando a velocidade de convergência (em que geração o critério de parada foi atingido), tempo de execução e erro relativo ao menor autovalor "exato" (obtido no SciLab).

5.9 Resultados preliminares na GPU

5.9.1 ONEMAX na GPU

ERAD: artigo + poster

5.9.2 Método paralelizado (versão atual, com ganho de $1,4$

Falar um pouco.

Referências

COOPE, J. A. R.; SABO, D. W. A new approach to the determination of several eigenvectors of a large hermitian matrix. *Journal of Computational Physics*, v. 23, p. 404–424, 1977.

LINDEN, R. *Algoritmos Genéticos. Uma importante ferramenta da Inteligência Computacional*. [S.l.]: BRASPORT, 2008.

Microsoft Corporation. *Microsoft Excel 2007*. Redmond, Washington, 2007. Disponível em: <https://products.office.com/pt-br/excel>.

NANDY, S.; CHAUDHRY, P.; BHATTACHARYYA, S. P. Diagonalization of a real-symmetric hamiltonian by genetic algorithm: A recipe based on minimization of rayleigh quotient. *J. Chem. Sci*, Indian Academy of Sciences, v. 116, p. 285–291, September 2004.

NANDY, S.; CHAUDHURY, P.; BHATTACHARYYA, S. P. Workability of a genetic algorithm driven sequential search for eigenvalues and eigenvectors of a hamiltonian with or without basis optimization. In: YU, W.; SANCHEZ, E. (Ed.). *Advances in Computational Intelligence*. Berlin: Springer-Verlag, 2009. p. 259–268.

NANDY, S.; SHARMA, R.; BHATTACHARYYA, S. P. Solving symmetric eigenvalue problem via genetic algorithms: Serial versus parallel implementation. *Applied Soft Computing*, Elsevier, v. 11, p. 3946–3961, 2011.

PARLETT, D. N. *The Symmetric Eigenvalue Problem*. 2. ed. Philadelphia, USA: SIAM - Society for Industrial and Applied Mathematics, 1998. (Classics in Applied Mathematics).

Scilab Enterprises. *Scilab: Free and Open Source software for numerical computation*. Orsay, France, 2012. Disponível em: <http://www.scilab.org>.

SHARMA, R.; NANDY, S.; BHATTACHARYYA, S. P. On solving energy-dependent partitioned eigenvalue problem by genetic algorithm: The case of real symmetric hamiltonian matrices. *PRAMANA Journal of Physics*, Indian Academy of Sciences, v. 66, p. 1125–1130, June 2006.

SHARMA, R.; NANDY, S.; BHATTACHARYYA, S. P. On solving energy-dependent partitioned real symmetric matrix eigenvalue problem by a parallel genetic algorithm. *Journal of Theoretical and Computational Chemistry*, World Scientific Publishing Company, v. 7, n. 6, p. 1103–1120, 2008.

Apêndices

APÊNDICE A – Lista de autovalores

Tabela 3 – Lista de autovalores para matrizes de Coope−Sabo de ordem 10, 20, 30 e 40.

#	**10**	**20**	**30**	**40**
0	0,386075	0,341237	0,319737	0,306086
1	2,461056	2,397247	2,36844	2,350583
2	4,518931	4,436173	4,401134	4,379909
3	6,572897	6,468521	6,427419	6,4031
4	8,628524	8,497626	8,450274	8,42294
5	10,69057	10,52507	10,47105	10,44068
6	12,76574	12,55178	12,4905	12,457
7	14,86753	14,57845	14,50908	14,47232
8	17,03654	16,60562	16,52713	16,48692
9	22,07215	18,63385	18,54488	18,501
10		20,6637	20,56255	20,5147
11		22,69588	22,5803	22,52816
12		24,73127	24,59828	24,54146
13		26,77114	26,61667	26,55469
14		28,81733	28,6356	28,56792
15		30,87288	30,65527	30,58122
16		32,94325	32,67586	32,59466
17		35,04014	34,6976	34,60831
18		37,19805	36,72077	36,62223
19		45,2308	38,74571	38,63648
20			40,77285	40,65114
21			42,80277	42,6663
22			44,83625	44,68204
23			46,87444	46,69846
24			48,91902	48,71568
25			50,97274	50,73385
26			53,04052	52,75311
27			55,13271	54,77369
28			57,27946	56,79581
29			68,37101	58,81981
30				60,84608
31				62,87517
32				64,90781
33				66,94504
34				68,98845
35				71,04053
36				73,10578
37				75,19353
38				77,33102
39				91,50634

APÊNDICE B – Execuções para o *fitness* $f_i = e^{-\lambda||\nabla\rho||^2}$

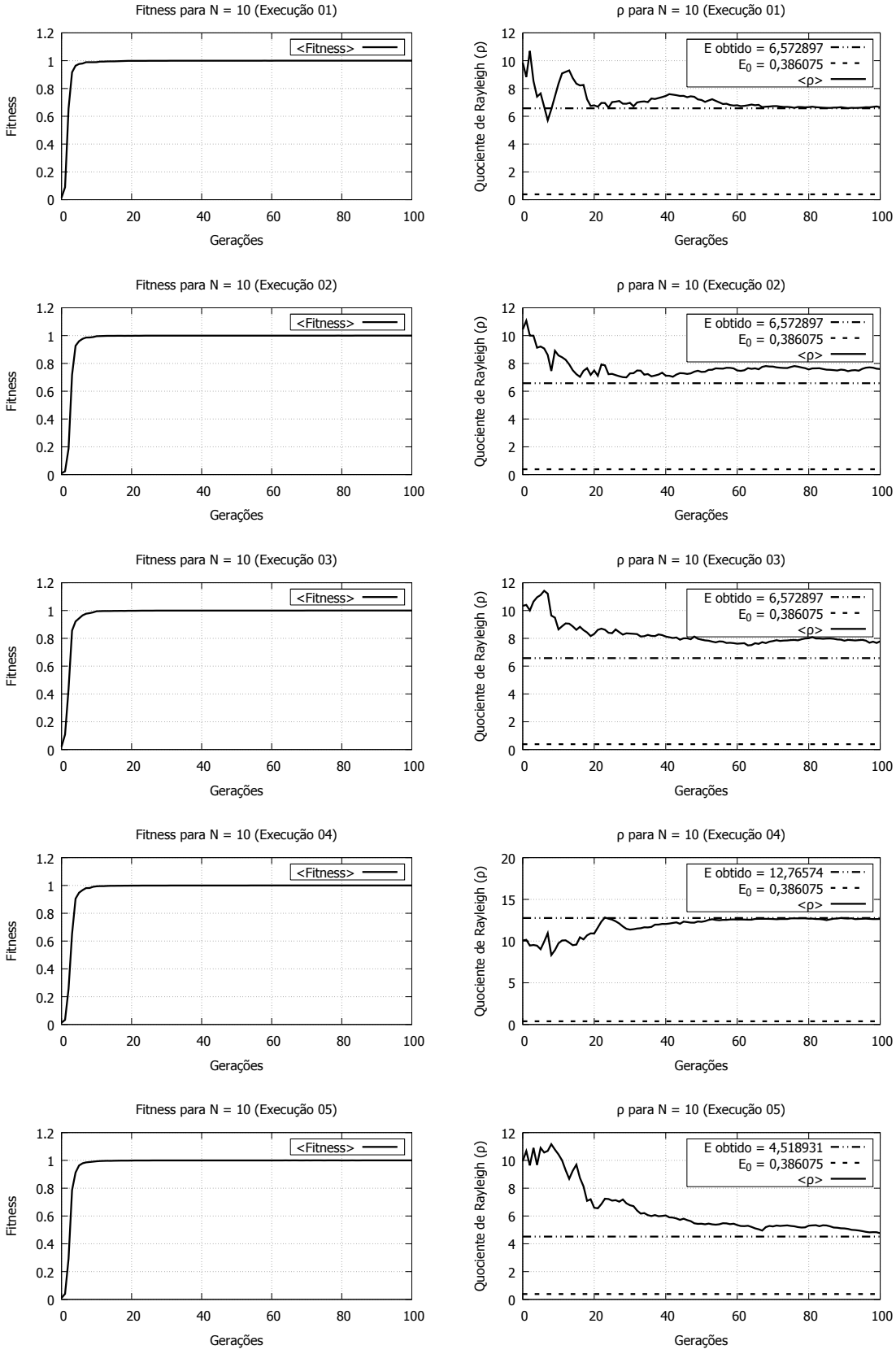

Figura 14 – Execuções N = 10.

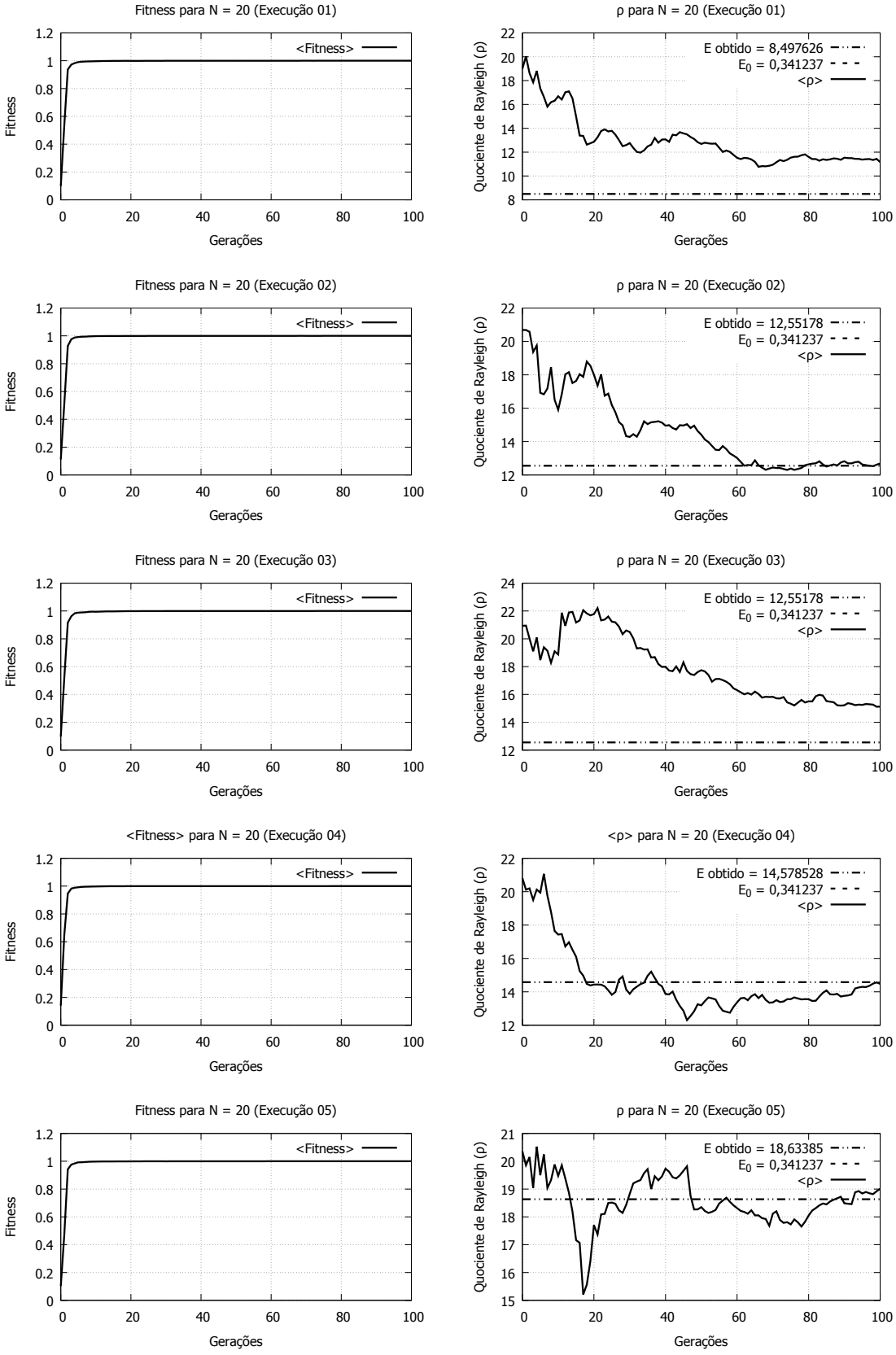

Figura 15 – Execuções N = 20.

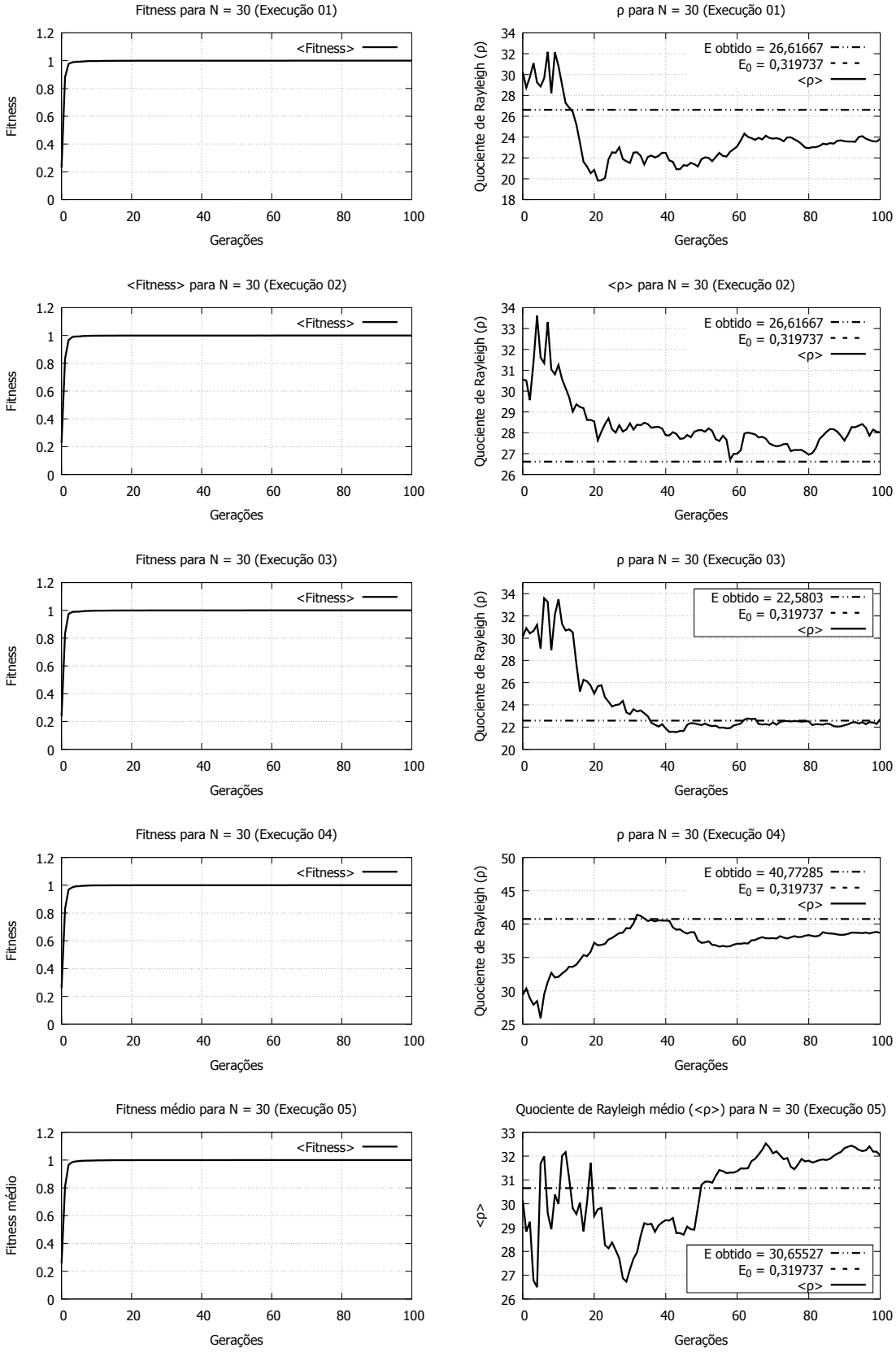

Figura 16 – Execuções N = 30.

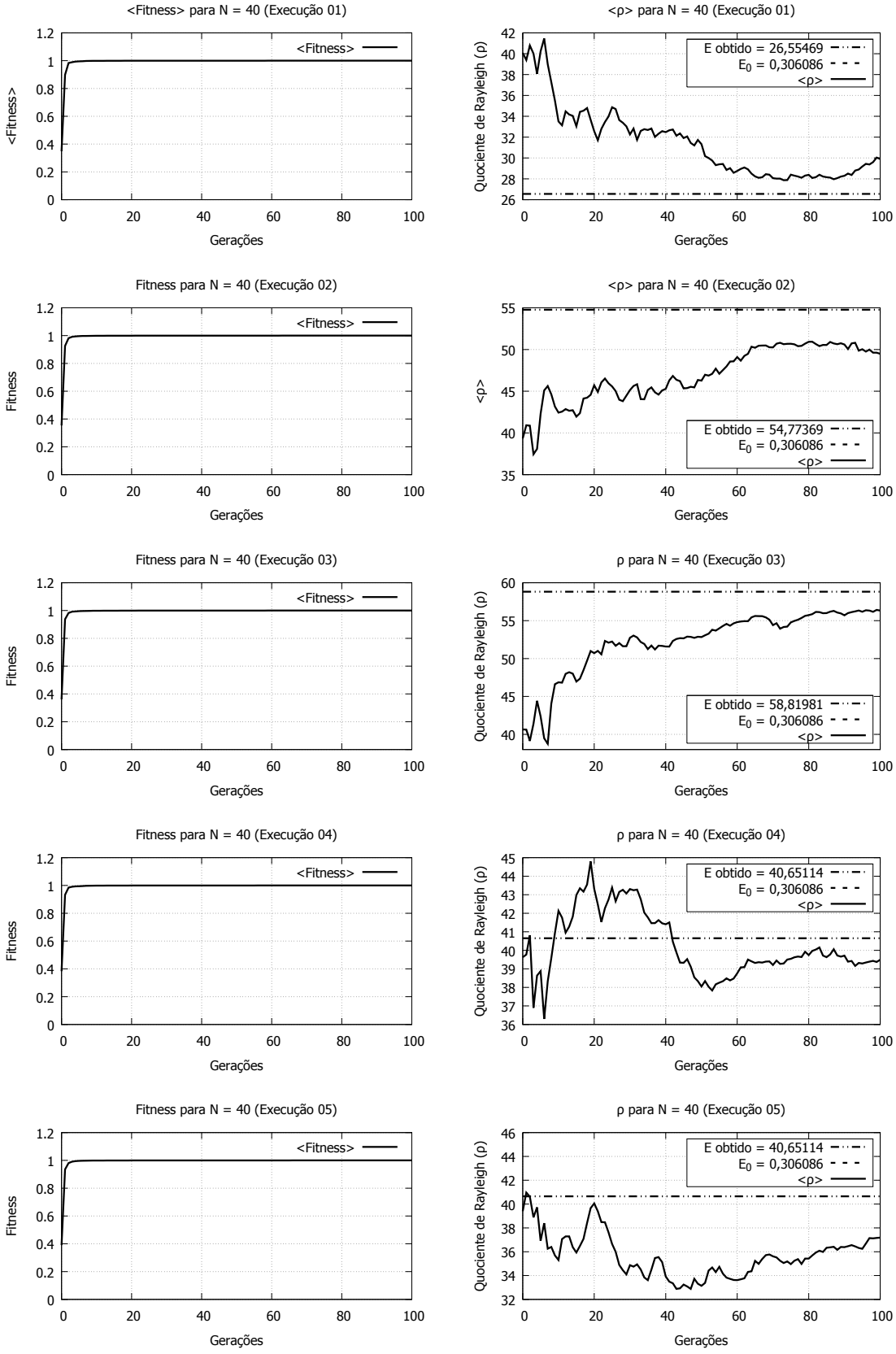

Figura 17 – Execuções N = 40.

APÊNDICE C – Execuções para o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$

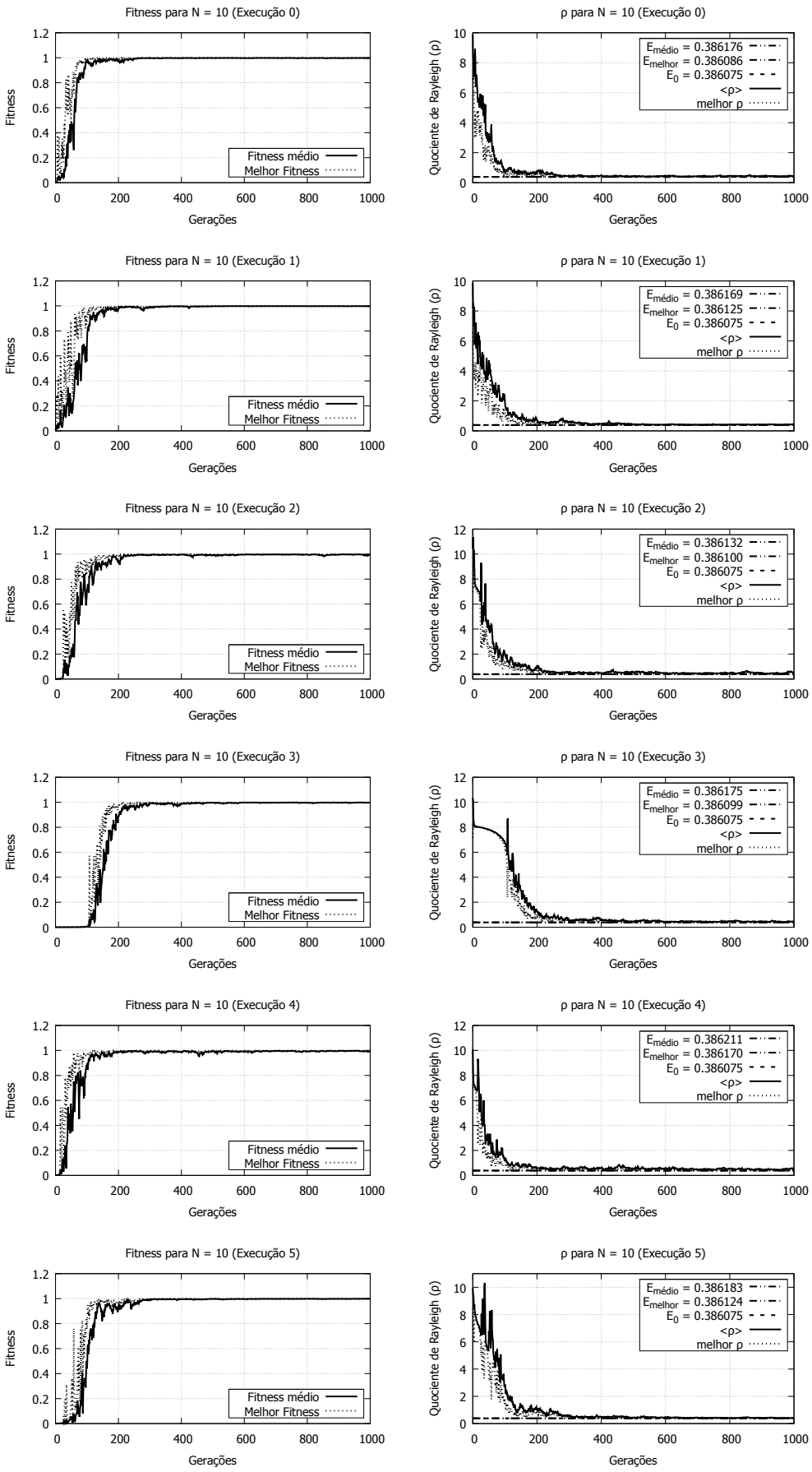

Figura 18 – Execuções para N = 10 com o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

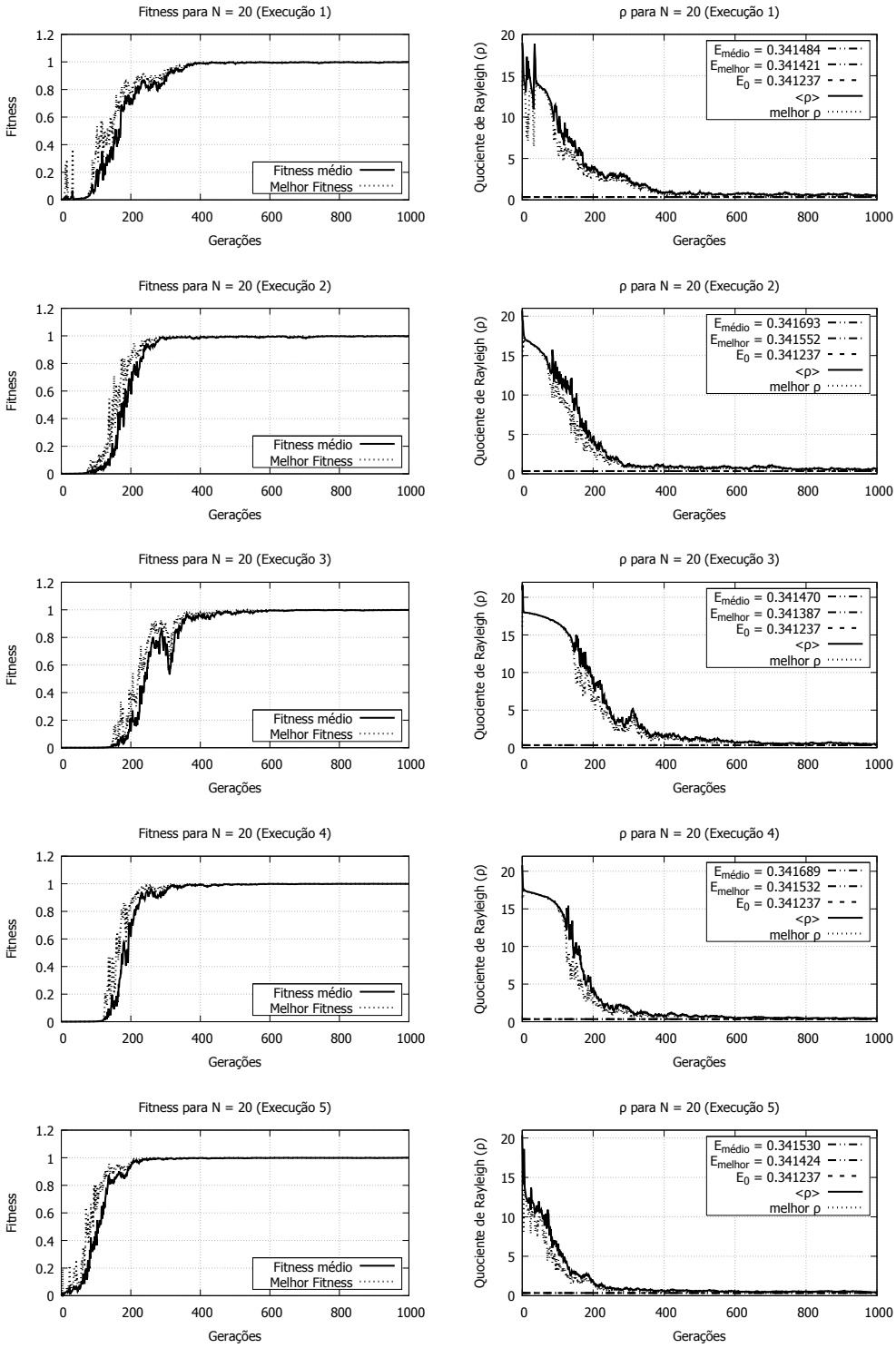

Figura 19 – Execuções para N = 20 com o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

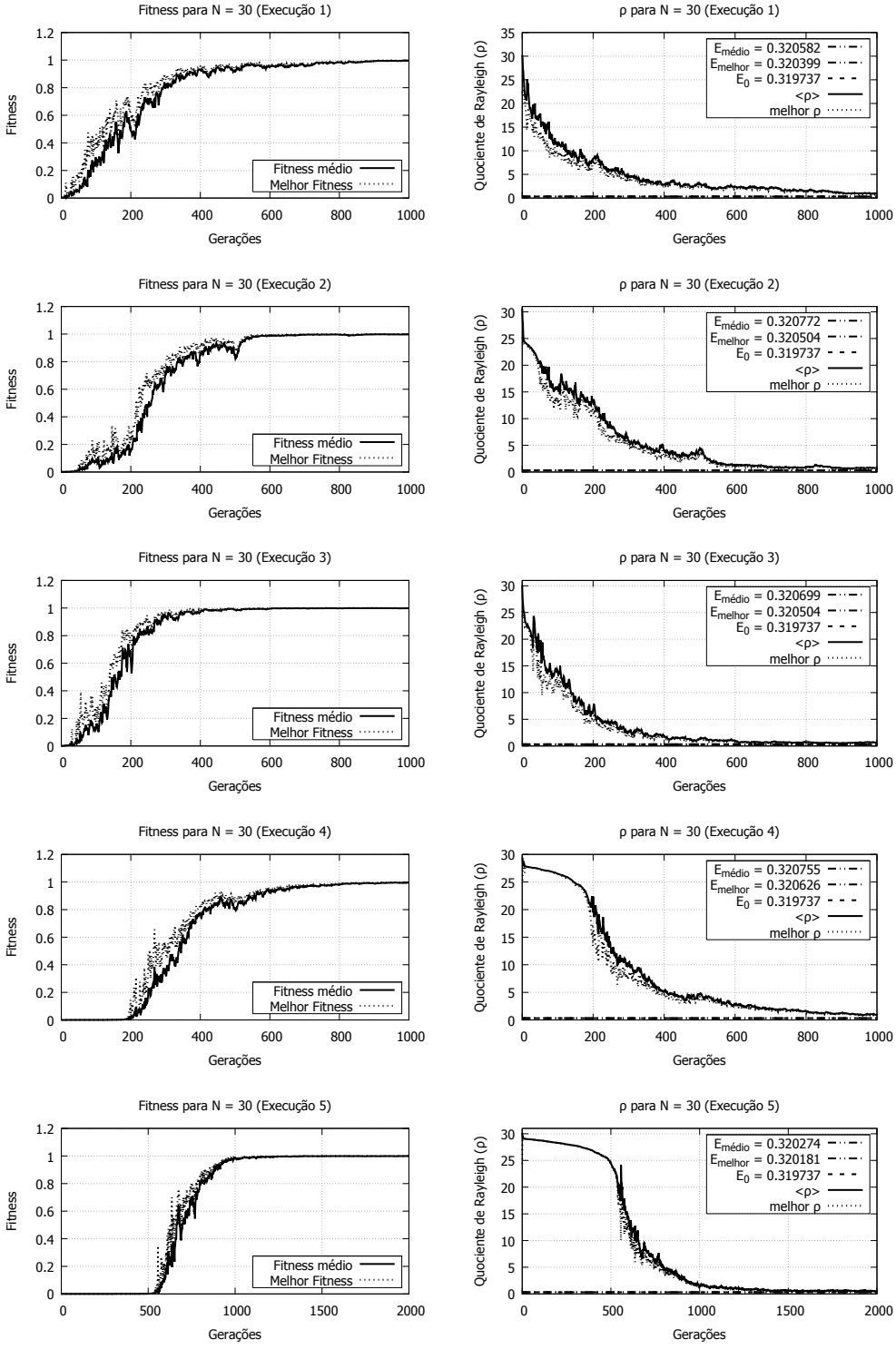

Figura 20 – Execuções para N = 30 com o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

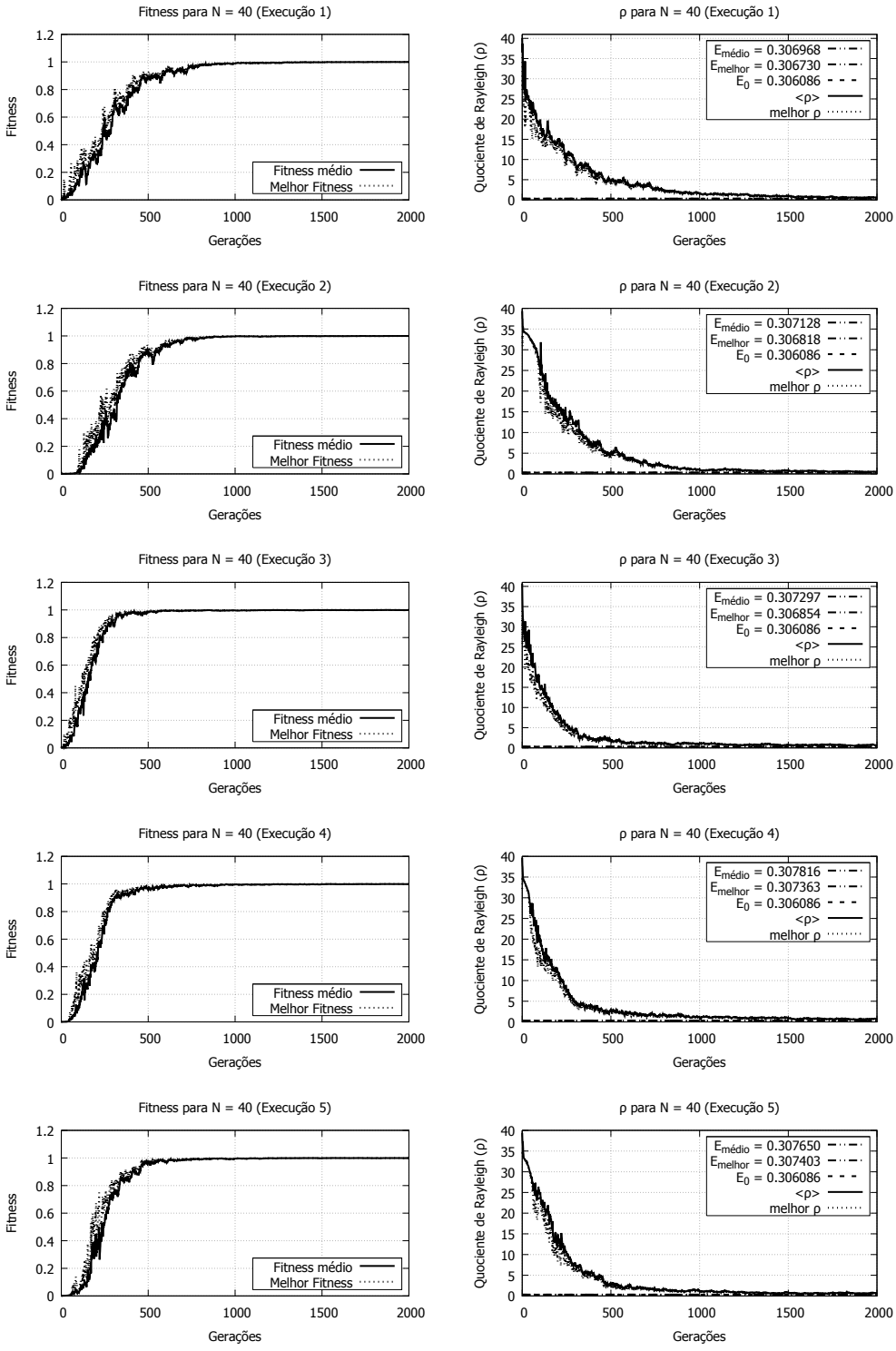

Figura 21 – Execuções para N = 40 com o *fitness* $f_i = e^{-\lambda(\rho_i - E_L)^2}$.

Anexos

ANEXO A – Título do Anexo X

Texto aqui.

ANEXO B – Título do Anexo Y

Texto aqui.