CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
MESTRADO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

COMPUTAÇÃO EVOLUCIONÁRIA

PROF. DR. ROGÉRIO MARTINS GOMES

3ª LISTA DE EXERCÍCIOS

ALGORITMOS GENÉTICOS

Charles Wellington de Oliveira Fortes

Março de 2016

Lista de Figuras

Sumário

1. Consideração sobre o desenvolvimento do trabalho

Para o desenvolvimento do trabalho foi criado um programa usando a linguagem de programação C# na plataforma .NET da Microsoft.

O trabalho foi desenvolvido usando como base a tarefa 2 entregue anteriormente para esta disciplina, onde os indivíduos foram alterados para possuírem representação real ao invés de representação binária, além dos ajustes das funções para atender tanto a nova representação quanto às solicitações feitas no enunciado do trabalho.

Apesar das N execuções dos experimentos, devido a semelhança entre os resultados, serão listados apenas 1 resultado de cada tipo de experimento.

Os itens comparativos entre a implementação Real e Binária foram executados utilizando os mesmo valores de parâmetro de configuração e as mesmas funções de crossover, avaliação, mutação e seleção, sendo a única diferença a representação dos cromossomos.

Todos os experimentos representados neste relatório foram concluídos com todos os elementos da população atingindo o valor ótimo da função.

2. Desenvolvimento: Representação Real vs Binária

O desenvolvimento do código do experimento usando representação por números reais é muito mais simples do que o desenvolvimento utilizando representação binária para os indivíduos.

É facilmente percebida a facilidade em trabalhar com as propriedades que representam os genes, uma vez que não são necessárias conversões entre binário e real, e as iterações dentro dos genes fica muito mais simples.

As funções de avaliação, crossover e mutação também apresentam melhor legibilidade quando usando valores reais para representação, simplificando o entendimento.

Durante a execução de múltiplos experimentos, ou elevando o número de gerações para 10 ou 100 vezes o valor proposto no trabalho, é perceptível o ganho do tempo de execução do experimento.

2.1 Diferença no resultado das mutações e do crossover na representação Binária e Real

Ao observar o cruzamento e a mutação nos indivíduos com representação binária, vemos um impacto maior nos descendentes, pois um bit trocado de 0 para 1 (e vice-versa) pode representar um salto grande de valor, enquanto nas técnicas usadas no trabalho com representação real apresenta uma variação mais suave dos genes, representando uma variação mais suave da população que pode ser vista ao compararmos as curvas dos gráficos que representam as médias de fitness das gerações dos experimentos da tarefa 2 (Figura 1) e da tarefa 3 (Figura 2).

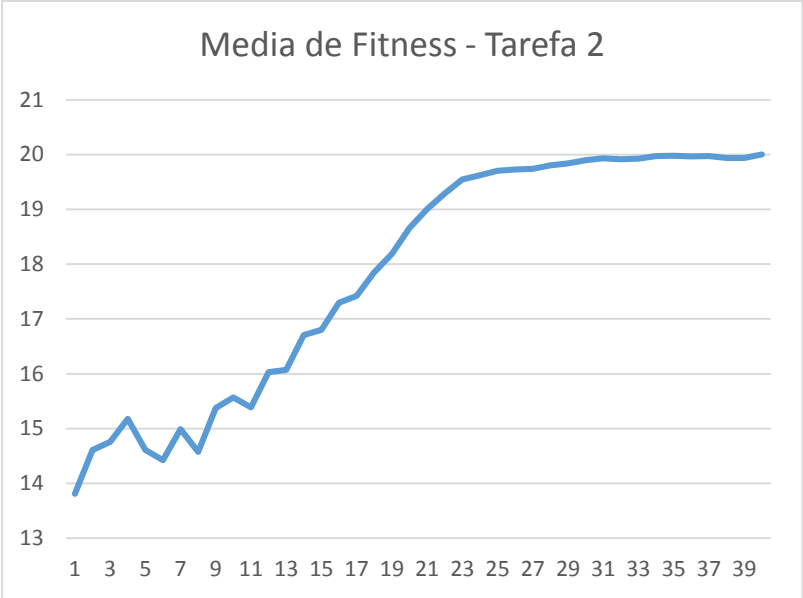


Figura 1 - Gráfico de média dos fiteness das populações da tarefa 2. Todos os indivíduos atingiram o valor ótimo na 39ª geração.

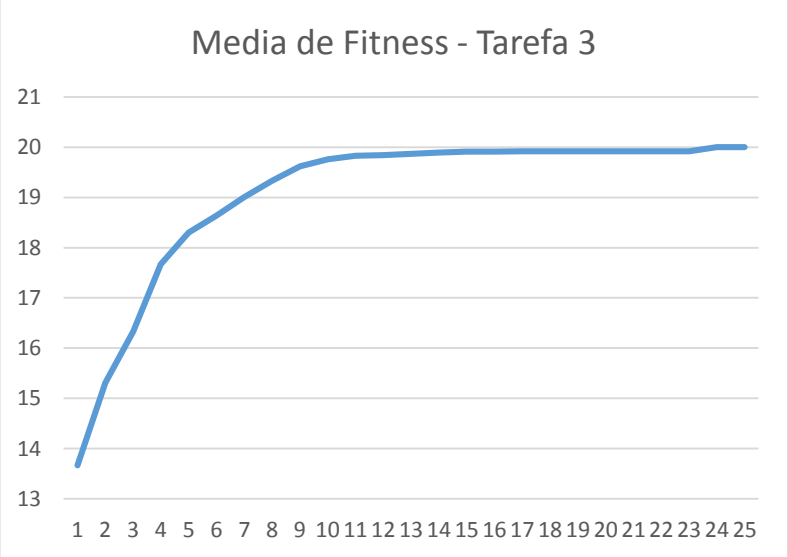


Figura 2 - Gráfico de média dos fitness das populações da tarefa 2. Todos os indivíduos atingiram o valor ótimo na 25ª geração.

2.2 Diferença de Desempenho computacional entre Binário e Real

Para medição do desempenho computacional foi utilizado o recurso de analise nativo da IDE Microsoft Visual Studio 2013 – “Performance and Diagnostics”, obtendo os resultados que apontam o consumo de 400ms (Figura 4) de CPU para o processamento da representação em real contra 492ms (Figura 3) para representação em binário.

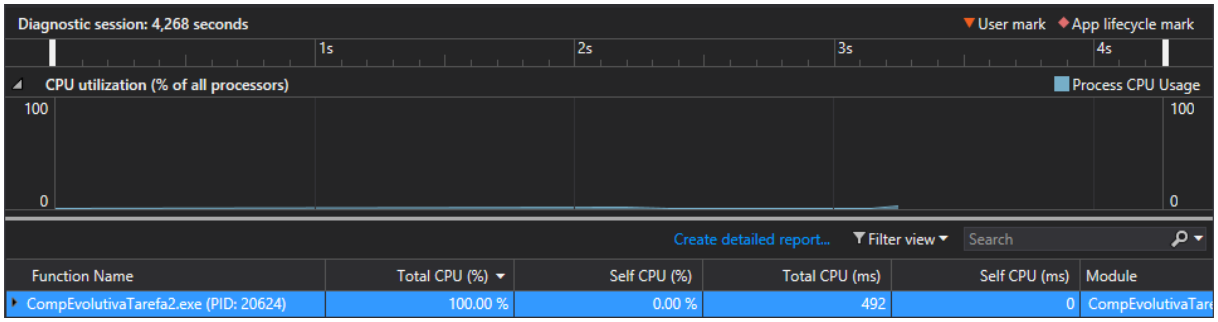


Figura 3 - Medição de tempo de CPU do experimento da tarefa 2

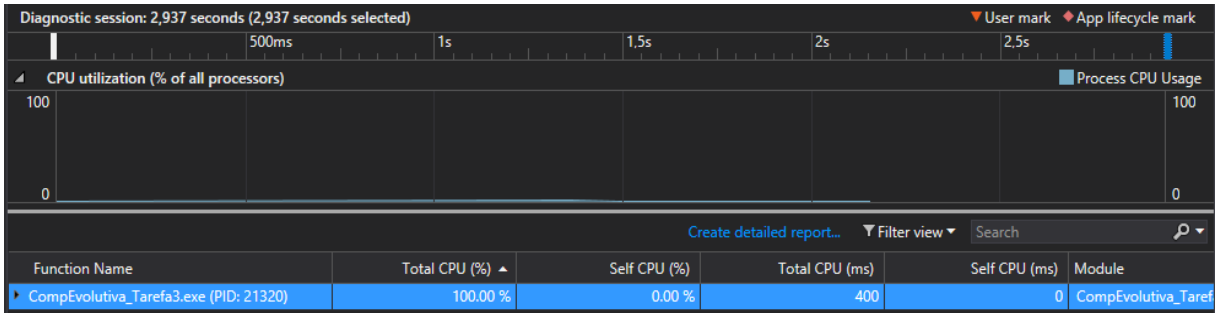


Figura 4 - Tempo de execução do experimento da tarefa 3

3. Avaliação dos Experimentos

3.1 Mutação Determinística

O uso de mutação determinística durante o desenvolvimento da tarefa, demonstra que a redução progressiva da taxa de mutação durante a evolução da população, faz com que a dispersão dos indivíduos em gerações mais avançadas (geralmente causadas pela mutação) seja menor.

Ao analisarmos os gráficos com os valores de fitness, verificamos que com este modelo, o nível de evolução é bem alto nas gerações iniciais e elas vão estabilizando conforme o experimento avança (Figura 7). Além disto é possível observar que os valores de mutação baixo fazem com que os piores indivíduos da população avancem de forma mais uniforme aos valores ótimos quando comparado aos outros experimentos (Figura 8) testados neste trabalho.

Como efeito colateral, em comparação aos outros experimentos, a população (Figura 5) levou um maior número de gerações (47 gerações) até que todos os indivíduos atingisse o valor ótimo da função (Figura 6).

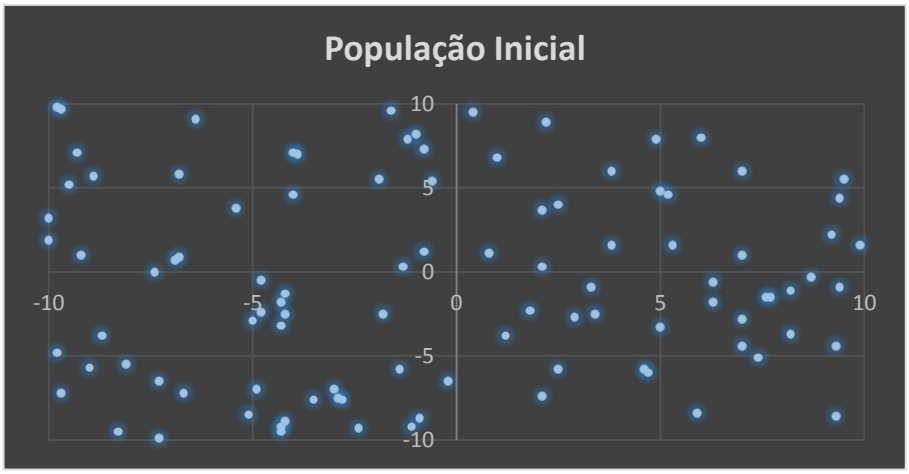


Figura 5 - População Inicial do Experimento com Mutação Determinística

Figura 6 - População Final do Experimento usando Mutação Determinística

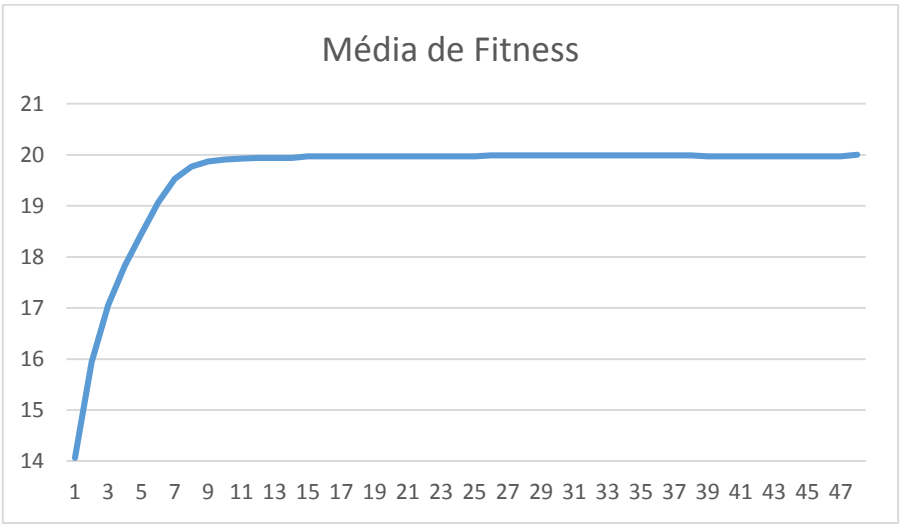


Figura 7 - Média de Fitness Usando Mutação Determinística

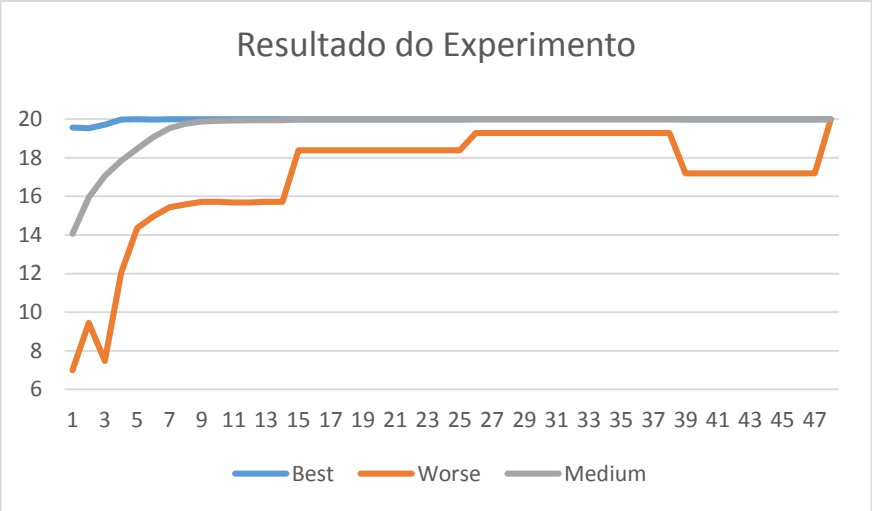


Figura 8 - Curvas de Resultado com o Fitness dos Melhores e Piores Indivíduos de cada geração em comparação à média de fitness da geração

```
225         public static void DeterministicMutation(List<double[]> offsprings)
226         {
227             var rdn = new Random((int)DateTime.Now.Ticks);
228             _deterministicMutationFactor = 0.01 - 0.009 * _currentGeneration / Parameters.MaxGenerations;
229
230             foreach (var cromossome in offsprings)
231             {
232                 for (var gene = 0; gene < cromossome.Length; gene++)
233                 {
234                     var testNumber = rdn.NextDouble();
235                     var apply = testNumber < _deterministicMutationFactor;
236                     if (apply)
237                         cromossome[gene] = rdn.Next(Parameters.MinReal, Parameters.MaxReal);
238                 }
239             }
240         }
```

Figura 9 - Código C# da Mutação Determinística Usada na Tarega

3.2 Mutação por Feedback

Com a mutação por feedback, é possível verificar que o algoritmo no decorrer da evolução, vai progressivamente reduzindo a taxa de mutação da população.

No início do experimento é possível ver o impacto das diferentes taxas de mutação ao observarmos os fitness dos piores indivíduos (Figura 13). Com o passar das gerações a influência de taxas de mutações não ideais vão sendo reduzidos, fazendo com que nas gerações finais as variações sejam mais controladas.

Neste modelo todos os indivíduos da população atingiram o ponto ótimo da função em menos gerações do que na versão determinística, porém a curva de evolução observada pela média de fitness com mutação por feedback (Figura 12) é maior do que a observada na determinística (Figura 7).

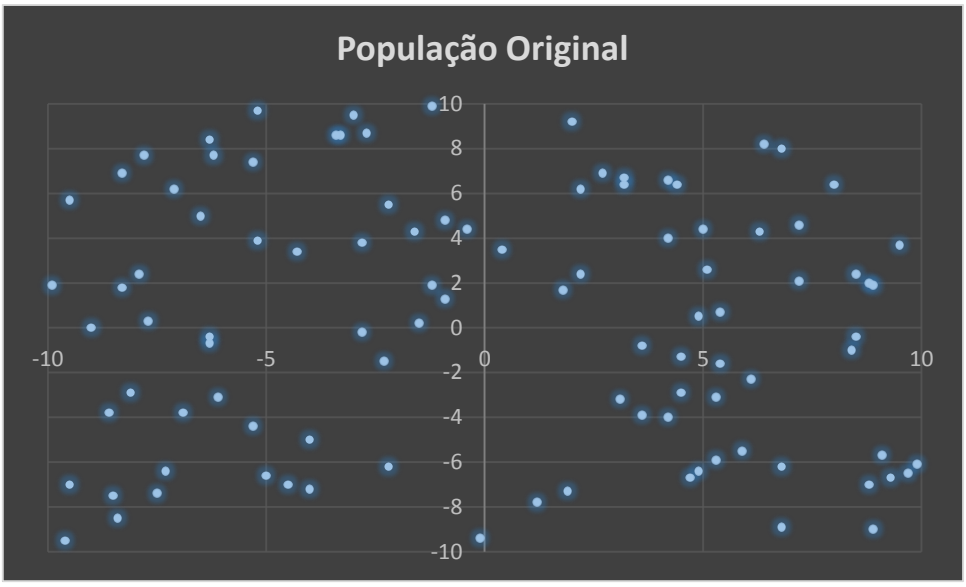


Figura 10 - População Inicial do Experimento com Mutação por Feedback

Figura 11 - População Final do Experimento com Mutação por Feedback

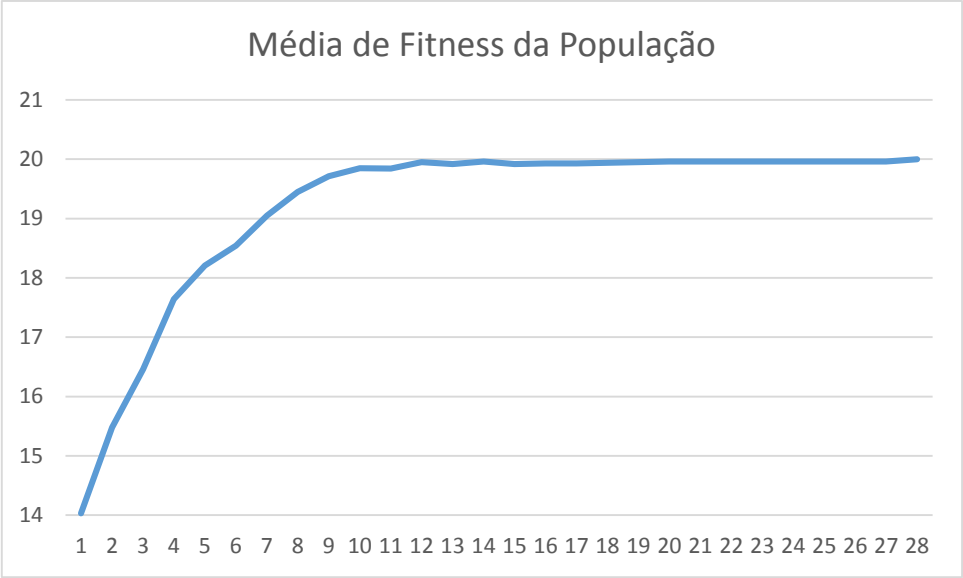


Figura 12 - Gráfico de Média de Fitness do Experimento com Mutação por Feedback.

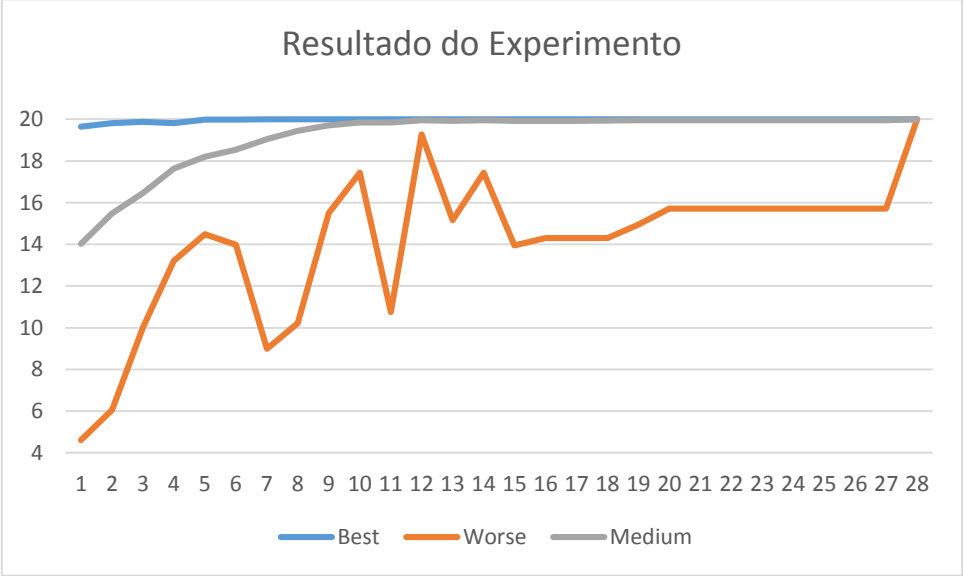


Figura 13 - Curvas de Resultado com o Fitness dos Melhores e Piores Indivíduos de cada geração em comparação à média de fitness da geração

```
242         public static void FeedbackMutation(List<double[]> offsprings)
243         {
244             var rdn = new Random((int)DateTime.Now.Ticks);
245             var notMutatedPopulation = new List<double[]>(offsprings);
246             var mutatedPopulation = new Dictionary<int, double[]>();
247 
248             for (var i = 0; i < offsprings.Count; i++)
249             {
250                 for (var gene = 0; gene < offsprings[1].Length; gene++)
251                 {
252                     var testNumber = rdn.NextDouble();
253                     var apply = testNumber < _deterministicMutationFactor;
254 
255                     if (apply)
256                     {
257                         offsprings[1][gene] = rdn.Next(Parameters.MinReal, Parameters.MaxReal);
258                         if (!mutatedPopulation.ContainsKey(i))
259                             mutatedPopulation.Add(i, offsprings[1]);
260                     }
261                 }
262             }
263 
264             var successFactor = getMutationSuccessFactor(notMutatedPopulation, mutatedPopulation);
265 
266             if (successFactor > 0.2)
267                 _deterministicMutationFactor /= (rdn.Next(817, 1000) / 1000.0); //Numero aleatorio entre 0.817 e 1
268             else if (successFactor < 0.2)
269                 _deterministicMutationFactor *= (rdn.Next(817, 1000) / 1000.0); //Numero aleatorio entre 0.817 e 1
270 
271         }
```

Figura 14 - Código C# da Mutação por Feedback

```
273 private static double getMutationSuccessFactor(List<double[]> notMutatedPopulation, Dictionary<int, double[]> mutatedPopul
274 {
275     var totalMutated = mutatedPopulation.Count;
276     var sucess = 0;
277 
278     if (totalMutated == 0)
279         return 1;
280 
281     foreach (var mutant in mutatedPopulation)
282     {
283         //fitness do individuo após a mutação
284         var mutantEvaluation = EvaluatePopulation(new List<double[]> {mutant.Value}).First().Item2;
285 
286         //fitness do individuo antes da mutação
287         var originalEvaluation = EvaluatePopulation(new List<double[]> {notMutatedPopulation[mutant.Key]}).First().Item2;
288 
289         //se for melhor incrementa o sucesso
290         if (mutantEvaluation > originalEvaluation)
291             sucess++;
292     }
293     return sucess/totalMutated;
294 }
```

Figura 15 - Código C# pra Avaliação de Sucesso da Mutação

3.3 Mutação Auto-Adaptativa

De todas as implementações, esta foi a de maior complexidade. Durante a execução de experimentos com esta configuração foi percebido que os melhores indivíduos de cada geração tendem a assumir valores iguais ou muito aproximados de mutação.

Além disto foi observado que os melhores indivíduos de cada população não necessariamente eram os com a maior ou com a menor taxa de mutação. Porém ao observarmos as curvas dos piores indivíduos (Figura 19) e as comparar com os outros experimentos usando mutações determinísticas e por feedback, vemos que estes fatores mais contribuem para a geração dos piores indivíduos da população do que atrapalham os melhores indivíduos (não foi usado elitismo durante o trabalho).

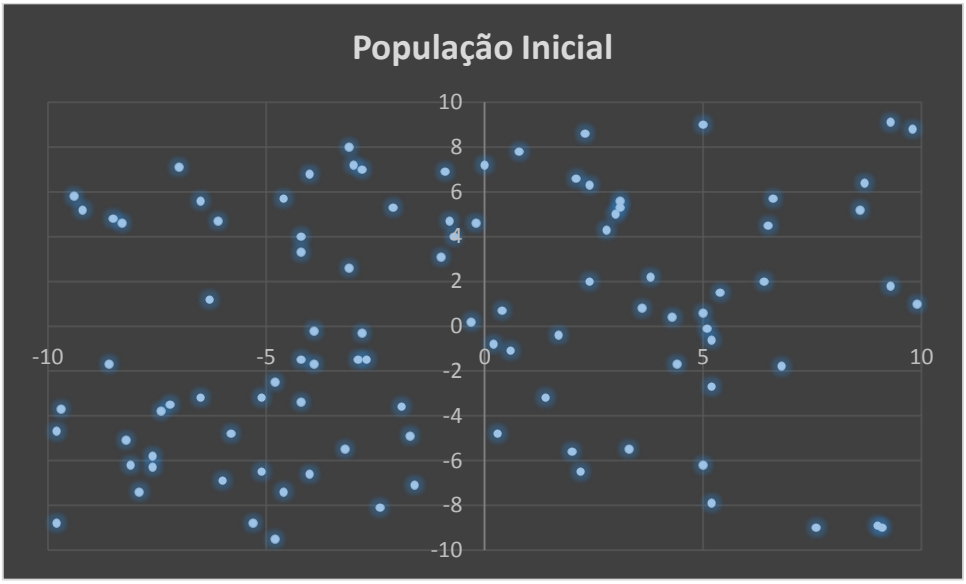


Figura 16 - População Inicial do Experimento com Mutação Auto Adaptativa

Figura 17 - População Final do Experimento com Mutação Auto Adaptativa

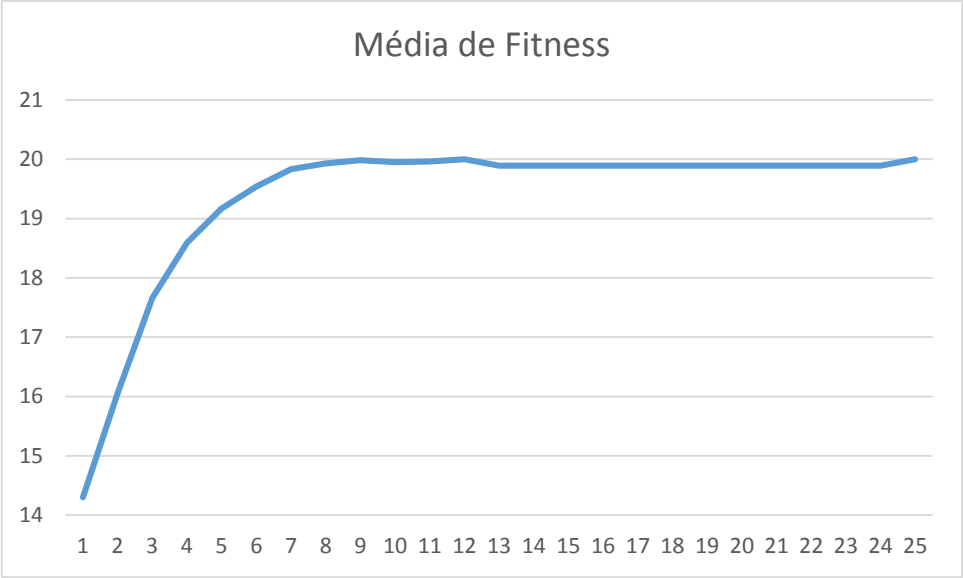
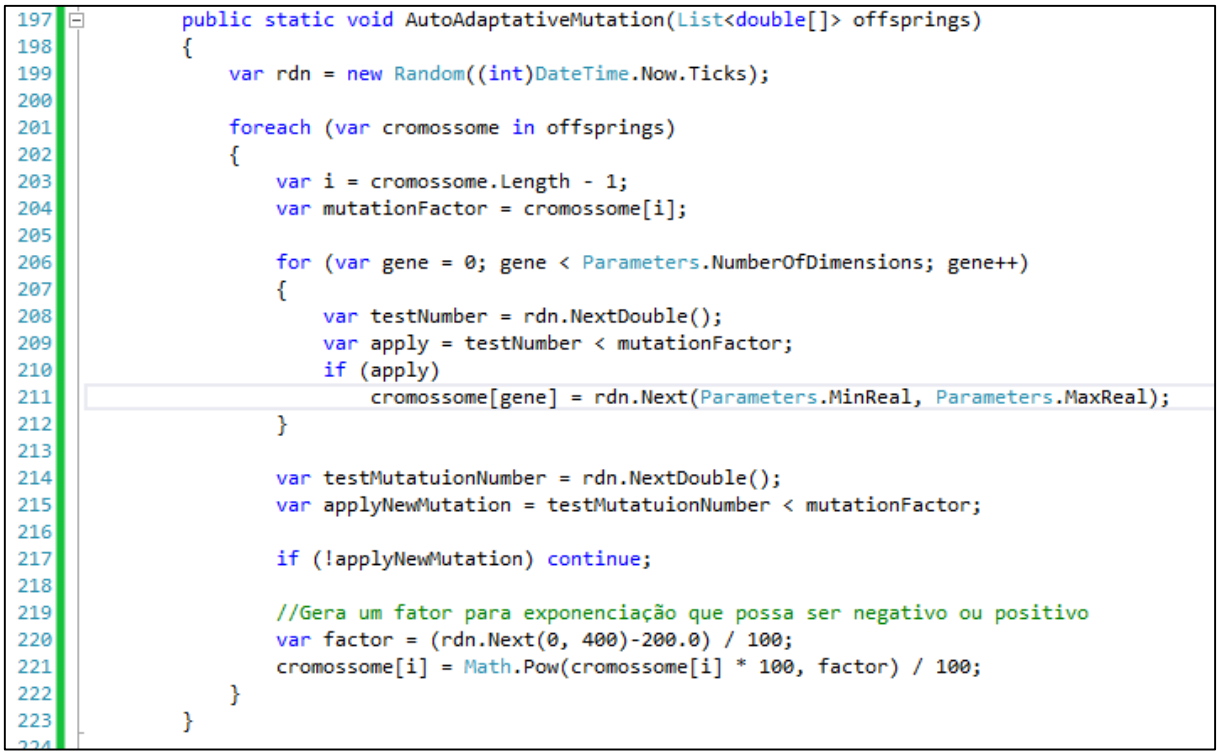


Figura 18 - Média de Fitness da População do Experimento com Mutação Auto Adaptativa

Figura 19Curvas de Resultado com o Fitness dos Melhores e Piores Indivíduos de cada geração em comparação à média de fitness da geração



```
197         public static void AutoAdaptativeMutation(List<double[]> offsprings)
198         {
199             var rdn = new Random((int)DateTime.Now.Ticks);
200
201             foreach (var cromossome in offsprings)
202             {
203                 var i = cromossome.Length - 1;
204                 var mutationFactor = cromossome[i];
205
206                 for (var gene = 0; gene < Parameters.NumberOfDimensions; gene++)
207                 {
208                     var testNumber = rdn.NextDouble();
209                     var apply = testNumber < mutationFactor;
210                     if (apply)
211                         cromossome[gene] = rdn.Next(Parameters.MinReal, Parameters.MaxReal);
212                 }
213
214                 var testMutatuionNumber = rdn.NextDouble();
215                 var applyNewMutation = testMutatuionNumber < mutationFactor;
216
217                 if (!applyNewMutation) continue;
218
219                 //Gera um fator para exponenciação que possa ser negativo ou positivo
220                 var factor = (rdn.Next(0, 400)-200.0) / 100;
221                 cromossome[i] = Math.Pow(cromossome[i] * 100, factor) / 100;
222             }
223         }
```

Figura 20 - Código C# para Mutação Auto-Adaptativa

4. Conclusão

Como conclusão, apesar das diferentes formas de implementação, pude verificar que as mais diversas formas de mutação nos permitem atingir os resultados esperados, porém com diferentes impactos sobre a população.

Durante as execuções dos experimentos para o problema apresentado, a ideia de reduzir a incidência de mutação na população nas gerações mais avançadas demonstrou assertiva. Dentre as opções verificadas, a determinística demonstra ser a que mais demora para fornecer os objetivos. Já o ajuste por feedback na mutação parece ser o mais flexível aos mais diversos cenários. Sendo que o melhor resultado parece ter sido atingido pela função auto-adaptativa, por ter tido o resultado ótimo de todos os indivíduos da população atingido com menos gerações e ter tido uma curva média mais homogênea (Figura 21).

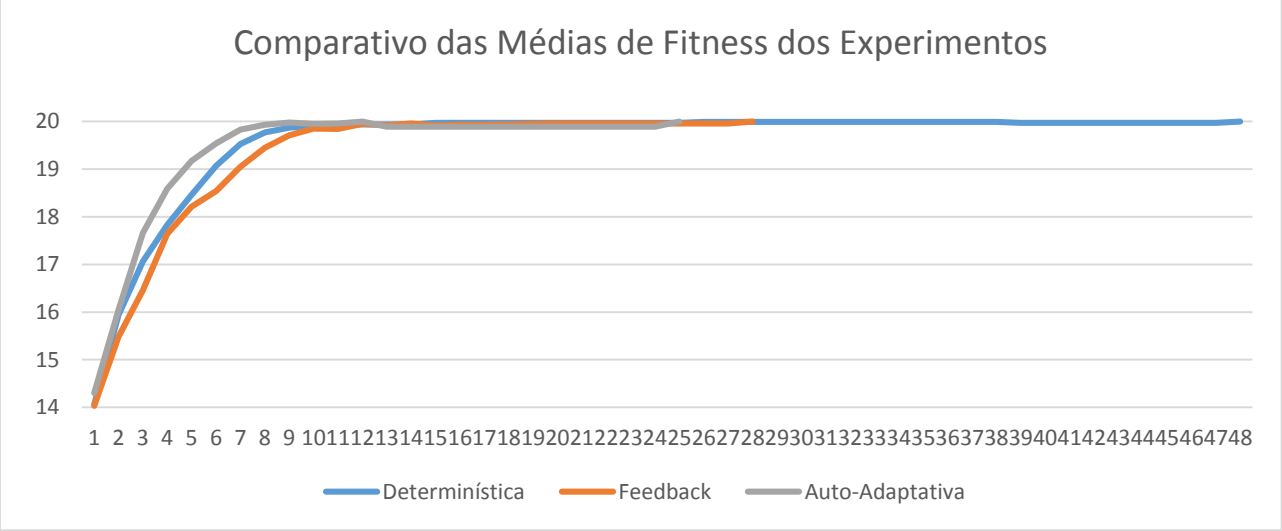


Figura 21 - Comparativo entre as Médias dos Experimentos por Tipo de Mutação