Introdução à Inteligência Artificial

TP2 - Problema de Otimização

2020 - 2021

TheForgotten
JOSEALM3IDA

Licenciatura de Engenharia Informática

17 de janeiro de 2021

Índice

1 Introdução

Este trabalho consiste em conceber, implementar e testar métodos de otimização que encontrem soluções de boa qualidade para diferentes instâncias do problema a seguir descrito.

Neste problema pretendemos obter soluções que maximizem a diversidade. Estas soluções consistem em maneiras diferentes de dividir um número M de elementos uniformemente por G subconjuntos de modo a que cada um tenha, respetivamente N elementos, sendo N = M / G.

A qualidade duma solução no problema apresentado é igual à soma das diversidades de todos os seus subconjuntos que, por consequência, é igual à soma das distâncias entre todos os pontos desse mesmo subconjunto.

Tanto o número de elementos como o número de subconjuntos a formar e as distâncias entre os elementos são dados de entrada.

2 Algoritmo de Pesquisa Local (Recristalização Simulada)

O método de pesquisa local implementado foi a recristalização simulada (também conhecido como simulated annealing). A estratégia usada por este para ultrapassar máximos locais consiste em prosseguir "durante algum tempo" a pesquisa no sentido descendente de modo a encontrar possivelmente outra solução ótima.

Este escolhe sempre um valor aleatório ao invés de escolher o estado seguinte de maior valor. Se esta for superior à que estava antes guardada, substitui-a, se não for tem chance de ser guardada na mesma de modo a evitar ficar preso numa solução que pareça a ótima porque toda a vizinhança é pior.

Para além de considerarmos o método de pesquisa melhor, foi também escolhida a recristalização simulada ao invés do trepa-colinas (por exemplo) porque pareceu ser mais otimizado, ou seja, o programa executava mais depressa e obtínhamos soluções melhores.

Foram testadas duas funções de arrefecimento para este algoritmo: versão linear e versão geométrica. A versão linear reduz a temperatura subtraíndo um valor constante, a versão geométrica vai reduzindo percentualmente.

2.1 Vizinhança 1

A primeira vizinhança testada no nosso algoritmo foi uma vizinhança bastante simples, que se baseia em aleatoriedade apenas.

Esta funciona da seguinte forma: escolhe, aleatoriamente, 2 subconjuntos diferentes A e B, e troca, de A para B e de B para A, um elemento, também aleatório.

Nos testes efetuados ao algoritmo fizemos variar o número de vizinhos, assim como a temperatura máxima e mínima e o fator de atualização. Estes últimos três parâmetros e a escolha da função de arrefecimento fazem variar o número de iterações do algoritmo.

Em todos os testes realizados neste trabalho foram usadas 50 repetições de modo a obter equilíbrio estatístico.

Os resultados foram avaliados em comparação com valores base obtidos com os seguintes parâmetros:

- Nvizinhos = 10;
- Tmax = 50;
- Tmin = 20 (arrefecimento linear), 5 (arrefecimento geométrica);
- F_atualiz = 0.25 (arrefecimento linear), 0.5 (arrefecimento geométrica).

2.1.1 Função de arrefecimento linear

Com os parâmetros mencionados anteriormente obtivémos um total de 120 iterações.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1144	942.28
n012.txt	861	591.64
n030.txt	4154	3542.56
n060.txt	14395	13407.72
n120.txt	33998	32533.60
n240.txt	117047	113895.30

Ao aumentar a diferença entre a temperatura máxima e a mínima foi registado um aumento do número de iterações associadas ao algoritmo, o que provocou, de modo geral, melhores resultados em todos os ficheiros. Foi também verificada uma ligeira melhoria nos resultados ao aumentar o fator de atualização, o que é contrário ao esperado pois foi diminuído o número de iterações. Atribuímos isto ao fator sorte associado à aleatoriadade do método usado para obter vizinhos.

Foi também variado o número de vizinhos gerado a cada iteração, o que melhorou de modo geral as soluções finais obtidas e nos forneceu a melhor solução para o ficheiro **n010.txt**.

2.1.2 Função de arrefecimento geométrica

Com os parâmetros mencionados anteriormente obtivémos um total de 4 iterações.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1210	941.38
n012.txt	847	599.6
n030.txt	4055	3527.4
n060.txt	14502	13252.94
n120.txt	34405	32649.96
n240.txt	117920	114019.18

A análise dos resultado é semelhante aos da função de arrefecimento linear, mas mais extremos devido ao aumento brusco do número de iterações e vizinhos testado: o aumento dos mesmos resultou numa melhoria geral nos resultados.

Foi também obtida a melhor solução na maior parte dos testes para o ficheiro **n010.txt**.

2.2 Vizinhança 2

A segunda vizinhança testada no nosso algoritmo é mais complexa que a anterior pois já não se baseia totalmente em aleatoriedade.

Esta funciona da seguinte forma: escolhe, aleatoriamente, 2 subconjuntos diferentes A e B, e troca, de A para B e de B para A, o pior elemento de cada subconjunto (ou seja, o elemento cuja soma das distâncias com os elementos do mesmo subconjunto é a menor).

Com esta diminuição de aleatoriedade, o fator sorte tem um peso muito menor nos resultados obtidos o que nos fornece resultados mais concisos.

Nos testes efetuados ao algoritmo fizemos variar o número de vizinhos, assim como a temperatura máxima e mínima e o fator de atualização. Estes últimos três parâmetros e a escolha da função de arrefecimento fazem variar o número de iterações do algoritmo.

Os resultados foram avaliados em comparação com valores base obtidos com os mesmos parâmtros usados para os valores base da vizinhança 1.

2.2.1 Função de arrefecimento linear

Com os parâmetros mencionados anteriormente obtivémos um total de 120 iterações.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1117.92
n012.txt	961	753.3
n030.txt	5045	4655.62
n060.txt	17931	17132.4
n120.txt	43839	42665.2
n240.txt	145829	143364

Comparando os valores base destes testes com os melhores valores da vizinhança anterior, podemos notar um aumento substancial na qualidade das soluções obtidas. Para além disso obtivemos também, constantemente, a solução máxima do ficheiro **n010.txt**.

Neste caso o aumento do número de iterações não provocou diferença considerável nos resultados (perante a base), mas a sua redução teve efeitos negativos nos resultados obtidos.

2.2.2 Função de arrefecimento geométrica

Com os parâmetros mencionados anteriormente obtivémos um total de 4 iterações.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1074.72
n012.txt	984	744.88
n030.txt	4960	4575.84
n060.txt	17021	16393.78
n120.txt	39285	38031.82
n240.txt	127128	124821.72

Comparando os valores base destes testes com os melhores valores da vizinhança anterior, podemos notar um aumento substancial na qualidade das soluções obtidas, mesmo usando apenas quatro iterações. Para além disso obtivemos também, constantemente, a solução máxima do ficheiro **n010.txt**.

O impacto do aumento do número de iterações, no entanto, tem rendimentos decrescentes claros, como se nota pela pequena variação entre resultados de testes com 180 e 459 iterações

Aqui foram obtidos os melhores valores da rescristalização simulada, que foram os seguintes:

- **n010.txt**: 1228;
- **n012.txt**: 984;
- **n030.txt**: 5045;
- **n060.txt**: 17931;
- **n120.txt**: 44265;
- **n240.txt**: 146135.

Estes valores foram obtidos com Nvizinhos = 10, Tmax = 50, Tmin = 5 e F_atualiz = 0.99 dando um total de 459 iterações.

3 Algoritmo Genético

O algoritmo genético é uma técnica para resolução de problemas que necessitem de otimização. É baseado na Teoria de Evolução de Darwin e tem um forte vínculo com conceitos da biologia.

Neste algoritmo ao invés de melhorarmos uma única solução, manipulamos uma população. Uma população é um conjunto de indivíduos (soluções) que por sua vez são compostos por cromossomas (no caso do nosso problema, pontos).

Os indivíduos da população vão-se alterando ao longo de várias gerações através de mecanismo genéticos como mutação e crossover. Os melhores indivíduos de cada geração, escolhidos através de torneios, são os que dão origem aos descendentes, ou seja, a próxima geração será composta por variações dos melhores indivíduos da anterior.

3.1 Mutações

O processo de mutação consiste em alterar a unidade genética mais básica (neste caso, o cromossoma) do indivíduo. Isto possibilita a mudança subtil dos indivíduos, permitindo desenvolver soluções diferentes (piores ou melhores) que possam ajudar a guiar as próximas gerações a obter a melhor solução possível.

Os resultados foram avaliados em comparação com valores base obtidos com penalização cega, recombinação 1 e os seguintes parâmetros:

- População = 100;
- Gerações = 1000;
- Prob. Reprodução = 0.2;
- Prob. Mutação = 0.1.

3.1.1 Mutação 1

Esta mutação consiste em escolher 2 subconjuntos aleatoriamente e trocar entre eles um elemento, o que pode gerar soluções inválidas. Sendo aleatória o fator sorte vai desempenhar um papel importante.

Os testes foram todos feitos sem reparação, recorrendo ao método de penalização cega (atribuir qualidade -1 a todas as soluções inválidas) para sinalizar soluções inválidas no problema.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	962.06
n030.txt	4434	4288.86
n060.txt	15131	15041.24
n120.txt	35805	35227.4
n240.txt	118529	118169.56

Com este método foi obtida quase sempre a melhor solução possível para os ficheiros **n010.txt** e **n012.txt**.

Ao alterar os parâmetros deste algoritmo chegámos às seguintes conclusões:

- Diminuir o número de indivíduos da população tem um efeito negativo nos resultados obtidos;
- Aumentar o número de gerações tem um efeito positivo nos resultados obtidos;
- Diminuir a probabilidade de reprodução e de mutação tem um efeito (ligeiramente) positivo nos resultados obtidos, sendo melhor o efeito da diminuição da probabilidade de reprodução;
- Aumentar a probabilidade de reprodução e de mutação tem um efeito negativo nos resultados obtidos.

3.1.2 Mutação 2

Esta mutação consiste em escolher 2 subconjuntos aleatoriamente e trocar entre eles o pior elemento, o que pode gerar soluções inválidas.

Os testes foram todos feitos recorrendo a diversas maneiras de lidar com soluções inválidas que serão abordadas mais tarde.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	994.02
n030.txt	4607	4572.78
n060.txt	15532	15369.88
n120.txt	35249	35159.08
n240.txt	118812	118538.24

Ao alterar os parâmetros deste algoritmo chegámos à conclusão de que a alteração da probabilidade de mutação pouco altera os resultados obtidos, melhorando ligeiramente uns e piorando ligeiramente outros.

3.2 Reparações

Os métodos de reparação consistem em corrigir um indivíduo (solução) não válido de modo a que este se torne válido novamente. Estes vêm substituir o método da penalização cega que apenas penalizava as soluções inválidas de modo a reduzir a chance para basicamente 0% de estas aparecerem na geração seguinte.

Com isto é esperado que as soluções melhorem porque ao corrigir soluções inválidas ao invés de descartá-las existe a chance de reaproveitar uma solução que estava próxima de ter bastante qualidade.

Os resultados foram avaliados em comparação com valores base obtidos com a mutação 1, recombinação 1 e os seguintes parâmetros:

- População = 100;
- Gerações = 1000;
- Prob. Reprodução = 0.2;
- Prob. Mutação = 0.1.

3.2.1 Reparação 1

Este reparação retira, aleatoriamente, elementos dos subconjuntos com excesso e coloca-os nos subconjuntos que precisam. Como funcionam aleatoriamente o fator sorte vai ser relevante nos resultados obtidos.

Os testes foram realizados usando os tipos de mutação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5194
n060.txt	18476	18422.26
n120.txt	44406	44316.56
n240.txt	143678	143662.4

Com apenas a mudança no tratamento de soluções inválidas passamos a obter quase sempre a melhor solução para os ficheiros **n010.txt**, **n012.txt** e **n030.txt**.

Podemos também concluir que o método de mutação utilizado desempenha um grande papel na qualidade das soluções obtidas e que a alteração dos parâmetros referidos anteriormente não altera de forma relevante os resultados obtidos.

3.2.2 Reparação 2

Este reparação retira os piores elementos dos subconjuntos com excesso e coloca-os nos subconjuntos que precisam.

Os testes foram realizados usando os tipos de mutação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5190.7
n060.txt	18546	18432.54
n120.txt	45299	45233.04
n240.txt	147466	147385.16

Comparando com o método de reparação anterior, foram obtidos resultados ligeiramente melhores.

Podemos novamente concluir que o método de mutação utilizado desempenha um grande papel na qualidade das soluções obtidas e que a alteração dos parâmetros referidos anteriormente não altera de forma relevante os resultados obtidos.

3.3 Recombinações

Os métodos de recombinação consistem em recombinar o material genético dos indivíduos duma população para gerar filhos.

Desta maneira obtemos dois indivíduos (filhos) que são uma "mistura" genética de dois indivíduos duma população anterior (pais) o que em teoria irá ajudar a obter melhores soluções.

Os resultados foram avaliados em comparação com valores base obtidos com a mutação 1, reparação 2 e os seguintes parâmetros:

- População = 100;
- Gerações = 1000;
- Prob. Reprodução = 0.2;
- Prob. Mutação = 0.1.

3.3.1 Recombinação 1

Esta recombinação consiste em encontrar um ponto de corte aleatório e gerar dois filhos a partir de rearranjos das duas partes obtidas de cada pai, ficando um filho com primeira parte da mãe, segunda parte do pai e o outro com primeira parte do pai e segunda parte da mãe. Isto pode gerar soluções inválidas.

Os testes foram realizados usando os métodos de mutação e reparação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5190.7
n060.txt	18546	18432.54
n120.txt	45299	45233.04
n240.txt	147466	147385.16

Sendo esta a recombinação "base" os resultados obtidos têm o único objetivo de servir de comparação para os próximos métodos de recombinação.

Podemos concluir que alterar parâmetro tem efeito quase nulo nos resultados obtidos, sendo estes apenas afetados substancialmente com a alteração do método de reparação e/ou mutação.

3.3.2 Recombinação 2

Esta recombinação consiste em encontrar três pontos de corte aleatórios e gerar dois filhos a partir de rearranjos das 4 partes obtidas de cada pai, ficando um filho com primeira parte da mãe, segunda parte do pai, terceira parte da mãe, quarta parte do pai e o outro com primeira parte pai, segunda parte mãe, terceira parte pai, quarta parte mãe. Isto pode gerar soluções inválidas.

Os testes foram realizados usando os métodos de mutação e reparação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5187.84
n060.txt	18430	18398.12
n120.txt	45348	45212.98
n240.txt	147675	147605.6

Esta recombinação é, de um modo geral, pior que a anterior, obtendo apenas uma solução melhor para o ficheiro **n240.txt**.

Tal como na anterior, podemos concluir que alterar parâmetro tem efeito quase nulo nos resultados obtidos, sendo estes apenas afetados substancialmente com a alteração do método de reparação e/ou mutação.

3.3.3 Recombinação 3

Esta recombinação, conhecida como recombinação uniforme, consiste em distribuir aleatoriamente os cromossomas da mãe e do pai por dois filhos. Isto pode gerar soluções inválidas.

Os testes foram realizados usando os métodos de mutação e reparação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5166.92
n060.txt	18452	18336.48
n120.txt	45366	45302.84
n240.txt	148230	148167.88

Esta recombinação é, de um modo geral, melhor que as anteriores.

Ao contrário das reparações anteriormente sugeridas, esta parece obter melhores resultados quanto maior for a probabilidade de reprodução.

3.4 Torneios

Os torneios servem como método de seleção: são usados para escolher os melhores indivíduos de uma determinada geração e determinar quais "sobrevivem" para a próxima geração.

Um torneio consiste em escolher k indivíduos. Esses k indivíduos "competem" entre eles, e o indivíduo com maior qualidade "ganha" o torneio e é adicionado ao vetor de indivíduos sobreviventes para a próxima geração.

Não há qualquer restrição na repetição de indivíduos: o mesmo indivíduo pode ser adicionado várias vezes ao vetor de sobreviventes, fazendo com que os indivíduos de maior qualidade tenham um peso maior na população da nova geração.

De modo a popular a nova geração com o mesmo número de indivíduos que a anterior, estes torneiros são realizados tantas vezes quanto o tamanho da população, visto que só é escolhido um indivíduo por torneio.

Um torneio onde k = 2 chama-se torneio binário. Este é o tipo de torneio que tem vindo a ser utilizado até agora, e que vai ser alterado e testado para vários valores de k (tsize).

Os resultados foram avaliados em comparação com valores base obtidos com a mutação 1, reparação 2, recombinação 3 e os seguintes parâmetros:

- População = 100;
- Gerações = 1000;
- Prob. Reprodução = 0.2;
- Prob. Mutação = 0.1.

3.4.1 Torneio 1

Também conhecido como torneio binário, neste torneio participam apenas 2 indivíduos.

Os testes foram realizados usando os métodos de mutação, reparação e recombinação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5166.92
n060.txt	18452	18336.48
n120.txt	45366	45302.84
n240.txt	148230	148167.88

Sendo este o torneio "base" os resultados obtidos têm o único objetivo de servir de comparação para os próximos torneios.

3.4.2 Torneio 2

Neste torneio participam 4 indivíduos.

Os testes foram realizados usando os métodos de mutação, reparação e recombinação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5189.2
n060.txt	18442	18412.6
n120.txt	45285	45222.16
n240.txt	149002	148699.52

De um modo geral a diferença dos resultados com a anterior é quase nula e pode ser atribuída ao caráter aleatório de algum dos métodos.

3.4.3 Torneio 3

Neste torneio participam 10 indivíduos.

Os testes foram realizados usando os métodos de mutação, reparação e recombinação mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5165.26
n060.txt	18452	18445.88
n120.txt	45346	45274.44
n240.txt	149105	148484.06

De um modo geral a diferença dos resultados com as anteriores é quase nula e pode ser atribuída ao caráter aleatório de algum dos métodos.

4 Algoritmo Híbrido (combinação das duas abordagens anteriores)

Este algoritmo é uma combinação dos algoritmos de pesquisa local (recristalização simulada) e do algoritmo genético discutidos anteriormente.

Os métodos escolhidos foram os que obtiveram melhores resultados e os valores escolhidos para os parâmetros foram os que pareciam ter melhor relação qualidade dos resultados / tempo de execução, ou seja:

- Algoritmo genético:
  - Reparação 2;
  - Recombinação Uniforme;
  - Mutação Aleatória;
  - População = 100;

  - Prob. Reprodução = 0.2;
  - Prob. Mutação = 0.1;
  - Tsize = 10.

- Algoritmo de pesquisa local (recristalização simulada):
  - Vizinhança 2;
  - Arrefecimento geométrico;
  - Nvizinhos = 10;
  - F_atualiza = 0.5;
  - Tmax = 50;
  - Tmin = 5.

Os resultados obtidos serão comparados com os resultados obtidos para os mesmos parâmetros e métodos no algoritmo genético.

A diferença entre este algoritmo híbrido e o algoritmo genético anteriormente falado é que este invoca, em diferentes etapas da execução, o algoritmo de pesquisa local, de modo a melhorar o indivíduo escolhido.

4.1 S.A. 1

Nesta versão do algoritmo híbrido o algoritmo de pesquisa local é invocado para melhorar a solução inicial do algoritmo genético.

Os testes foram realizados usando os métodos de mutação, reparação, recombinação, vizinhança e arrefecimento mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	**Melhor**	**MBF**
n010.txt	1228	1228.00
n012.txt	1000	1000
n030.txt	5194	5179.42
n060.txt	18513	18394.66
n120.txt	45516	45477.28
n240.txt	149252	149082.46

Em comparação com os valores base nota-se uma ligeira melhoria nos resultados obtidos.

4.2 S.A. 2

Nesta versão do algoritmo híbrido o algoritmo de pesquisa local é invocado para melhorar a melhor solução de cada execução.

Os testes foram realizados usando os métodos de mutação, reparação, recombinação, vizinhança e arrefecimento mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	1000.00
n030.txt	5194	5176.78
n060.txt	18505	18461.82
n120.txt	45480	45308.18
n240.txt	149669	148835.66

4.3 S.A. 3

Nesta versão do algoritmo híbrido o algoritmo de pesquisa local tem chance de ser invocado entre gerações para um ou mais indivíduos de modo a melhorar a qualidade total da solução.

Os testes foram realizados usando os métodos de mutação, reparação, recombinação, vizinhança e arrefecimento mencionados anteriormente.

Assim, com uma probabilidade de 10% de ser invocado, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	999.68
n030.txt	5194	5188.48
n060.txt	18615	18425.2
n120.txt	45352	45286.3
n240.txt	149993	149781.22

Em comparação com os valores anteriormente obtidos no algorítmo híbrido, podemos notar uma melhoria razoável. Para além da melhoria nos valores foi também verificado um enorme aumento no tempo de execução.

4.4 S.A. Misto

Esta versão do algoritmo híbrido é uma combinação de todas as versões mostradas anteriormente, ou seja, o algoritmo de pesquisa local é invocado para melhorar a solução inicial, melhorar a melhor solução de cada execução e tem chance de ser invocado entre gerações para um ou mais indivíduos de modo a melhorar a qualidade total da solução.

Os testes foram realizados usando os métodos de mutação, reparação, recombinação, vizinhança e arrefecimento mencionados anteriormente.

Assim, com uma probabilidade de 10% de ser invocado entre gerações, os valores base obtidos foram os seguintes:

Ficheiro	Melhor	MBF
n010.txt	1228	1228.00
n012.txt	1000	999.68
n030.txt	5194	5191.9
n060.txt	18612	18588.46
n120.txt	45619	45381.28
n240.txt	149744	149448.14

Este algoritmo foi o que apresentou os melhores resultados mas também demorou substancialmente mais que todos os outros a executar.

5 Conclusão

Assim, e concluindo o trabalho efetuado, depois de analisados os resultados, marcamos como principais os seguintes pontos gerais:

- Quando o algoritmo se baseia em lógica altamente aleatória, os resultados não são de todo estáveis e, de certa forma, prejudicam o estudo estatístico dos mesmos;
- Os resultados obtidos dependem largamente da criatividade e qualidade das funções de vizinhança, mutação, recombinação e reparação, sendo que há várias possibilidades para estas (com qualidade semelhante);
- Em geral, um número elevado de iterações (caso da pesquisa local) ou gerações / número de indivíduos da população aumenta a qualidade final dos resultados, mas requer mais tempo e memória para o algoritmo;
- No caso do algoritmo genético, o aumento da probabilidade de reprodução e mutação pode ter ou não efeitos benéficos ao problema, devendo ser estudada a sua variação e escolhido o melhor valor para cada função de mutação / reprodução;
- Um bom algoritmo requer, inevitavelmente, uma maior complexidade temporal para melhores resultados em problemas maiores, sendo que, para alcançar a solução ótima nestes problemas, é necessário uma grande capacidade de computação e tempo;
- Não é necessário um algoritmo com alta complexidade para resolver problemas mais simples, como os dos ficheiros **n010.txt** e **n012.txt**.

Tendo alcançado o objetivo deste trabalho, damos o mesmo por concluído.