**Trabalho Prático 2**

**Problema da Diversidade Máxima de Grupos**

Licenciatura em Engenharia Informática

Ano letivo 2020/2021

Unidade Curricular de Introdução à Inteligência Artificial

2ºano - 1ºsemetre

André Lima Domingues 2019127839

Filipe Daniel Ramos Morgado 2019137625

# Índice

# Introdução

A proposta de realização deste trabalho prático no âmbito da unidade curricular do 2ºano, 1ºsemetre de Introdução a Inteligência Artificial tem como objetivo a implementação de métodos de otimização que visem dividir um conjunto de elementos em subconjuntos menores, denominados grupos, de modo a que a diversidade entre os elementos em cada grupo seja a maior possível.

Dessa forma, os problemas de otimização são caracterizados por um grande universo de soluções o que dificulta a obtenção de uma solução de custo ótimo através de um algoritmo exato em tempo computacional admissível.

Neste trabalho são solucionadas algumas metodologias para gerar soluções para o problema mencionado:

* **Algoritmo de pesquisa local**
  + Trepa Colinas (1 vizinhança);
  + Trepa Colinas (1 vizinhança e aceitando soluções de custo igual);
  + Trepa Colinas (2 vizinhanças);
  + Trepa Colinas (2 vizinhanças e aceitando soluções de custo igual);
  + Trepa Colinas (probabilístico);
  + Recristalização Simulada
  + Pesquisa Tabu
* **Algoritmo evolutivo**
  + Estratégia de Penalização;
  + Estratégia de Reparação;
* **Método Híbrido**
  + Dois algoritmos anteriores;

# Algoritmos de Pesquisa Local

O algoritmo de pesquisa local, tem por objetivo procurar as melhores soluções explorando a vizinhança da solução inicial de modo a obter a melhor solução.

Este algoritmo parte de um estado inicial dado ou gerado aleatoriamente, para avaliar os seus estados sucessores o algoritmo consta com a estratégia da **melhor vizinhança,** que a partir de uma solução corrente, avalia cada estado gerado e escolhe o de valor maior. Em alguns algoritmos específicos, tal com o Trepa Colinas probabilístico, é possível aceitar soluções piores como forma de escapar a máximos locais. É possível também em alguns algoritmos ser aceite soluções iguais, caso o algoritmo se encontre numa planície, ou seja, num máximo local na maior parte das vezes, podendo assim escapar desse mesmo máximo local. O processo é repetido até que para quando atingir o número de reinícios máximo.

Assim foram implementados os seguintes algoritmos:

* **Trepa Colinas (1 vizinhança) –** neste algoritmo pretende-se avaliar o custo de uma solução atual (inicialmente solução inicial). Para isso, gera-se um vizinho, avalia-se o seu custo, e compara-se esse custo com o custo da solução atual. De seguida, como estamos perante um **problema de maximização** aceita-se o custo da vizinhança, caso este seja superior ao custo da solução atual.

É repetido este processo um certo número de iterações predefinido.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUM ELEMENTOS E SUBCONJUNTOS |  | 100  iterações | 1000  iterações | 5000  iterações | 10000  iterações |
| 10,2: n010.txt | **Melhor** | **1228** | **1228** | **1228** | **1228** |
| **MBF** | **1228** | **1228** | **1228** | **1228** |
| 12,4: n012.txt | **Melhor** | **1000** | **1000** | **1000** | **1000** |
| **MBF** | **966,7** | **980,24** | **984,92** | **987,5** |
| 30,5: n030.txt | **Melhor** | **4858** | **5179** | **5194** | **5194** |
| **MBF** | **4547,28** | **5018,76** | **5037,56** | **5064,74** |
| 60,6: n060.txt | **Melhor** | **16489** | **18012** | **18386** | **18520** |
| **MBF** | **15795,48** | **17560** | **18069,92** | **18189,54** |
| 120,10: n120.txt | **Melhor** | **37518** | **42427** | **44461** | **44756** |
| **MBF** | **36572,519** | **41483,8** | **43667,89** | **44118,55** |
| 240,12: n240.txt | **Melhor** | **124347** | **136519** | **143384** | **146191** |
| **MBF** | **120895,28** | **134411,78** | **142229,23** | **144448,18** |

## Algoritmo Trepa Colinas com 1 Vizinhança

## Variáveis da experiência em análise:

* Número de iterações entre 100 e 10000;
* Realização de 50 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

* Através da tabela de resultados, verifica-se que um maior número de iterações promove uma melhor solução em todos os ficheiros testados, uma vez que existe uma maior probabilidade de encontrar a diversidade máxima do problema.
* Como o número de elementos e subconjuntos vai aumentando progressivamente, cada vez mais o número de iterações influencia a qualidade da melhor solução. De maneira que, existindo vários elementos e subconjuntos ,100 iterações podem não ser suficiente para determinar a solução ótima.
* Contudo, este algoritmo é bastante simples uma vez que só tem em conta um vizinho e apenas substitui a melhor solução se o vizinho for maior que a solução atual tendo uma maior probabilidade de ficar preso num máximo de local.
* **Trepa Colinas (1 vizinhança e aceitando de soluções de custo igual) –** neste algoritmo pretende-se avaliar o custo de uma solução atual (inicialmente solução inicial). Para isso, gera-se um vizinho, avalia-se o seu custo, e compara-se esse custo com o custo da solução atual. De seguida, como estamos perante um **problema de maximização** aceita-se o custo da vizinhança, caso este seja superior ou igual ao custo da solução atual.

É repetido este processo um certo número de iterações predefinido.

## Algoritmo Trepa Colinas com 1 Vizinhança e aceitando soluções de custo igual

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUM ELEMENTOS E SUBCONJUNTOS |  | 100  iterações | 1000  iterações | 5000  iterações | 10000  iterações |
| 10,2: n010.txt | **Melhor** | **1228** | **1228** | **1228** | **1228** |
| **MBF** | **1228** | **1228** | **1228** | **1228** |
| 12,4: n012.txt | **Melhor** | **1000** | **1000** | **1000** | **1000** |
| **MBF** | **968,18** | **982,53** | **987,7** | **988,89** |
| 30,5: n030.txt | **Melhor** | **4820** | **5169** | **5194** | **5194** |
| **MBF** | **4551,48** | **5028,82** | **5050,32** | **5066,82** |
| 60,6: n060.txt | **Melhor** | **16282** | **17952** | **18484** | **18461** |
| **MBF** | **15750,94** | **17486** | **18133,64** | **18192,84** |
| 120,10: n120.txt | **Melhor** | **37933** | **42603** | **44366** | **44730** |
| **MBF** | **36732,74** | **41460** | **43591,74** | **44200,58** |
| 240,12: n240.txt | **Melhor** | **122806** | **136738** | **143647** | **146712** |
| **MBF** | **120765,44** | **134607,3** | **142115,4** | **144590** |

## Variáveis da experiência em análise:

* Número de iterações entre 100 e 10000;
* Realização de 50 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

* Através da tabela de resultados, verifica-se que um maior número de iterações promove uma melhor solução em todos os ficheiros testados, uma vez que existe uma maior probabilidade de encontrar a diversidade máxima do problema.
* Como o número de elementos e subconjuntos vai aumentando progressivamente, cada vez mais o número de iterações influencia a qualidade da melhor solução. De maneira que, existindo vários elementos e subconjuntos ,100 iterações podem não ser suficiente para determinar a solução ótima.
* Este algoritmo é um pouco melhor que o anterior já que substitui a melhor solução se o vizinho for maior ou igual que a solução atual, aumentando assim a probabilidade de não ficar bloqueado num máximo local.
* **Trepa Colinas (2 vizinhanças) –** neste algoritmo pretende-se avaliar o custo de uma solução atual (inicialmente solução inicial). Comparativamente com os métodos anteriores, este método gera duas vizinhanças avaliando os seus custos e comparando esse custo com a solução atual. Seguidamente, como se trata de um **problema de maximização**, apenas se aceita novas soluções caso o custo da vizinhança 1 seja superior ou igual ao custo da solução atual e superior ou igual ao custo da vizinhança 2 ou caso o custo da vizinhança 2 seja superior ou igual ao custo da solução atual e superior ou igual ao custo da vizinhança 1.

É repetido este processo o número de interações predefinido, e como espectável este algoritmo exige um tempo computacional mais alargado devido ao elevado número de comparações entre soluções.

## Algoritmo Trepa Colinas com 2 Vizinhanças e aceitando soluções de custo igual

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUM ELEMENTOS E SUBCONJUNTOS |  | 100  iterações | 1000  iterações | 5000  iterações | 10000  iterações |
| 10,2: n010.txt | **Melhor** | **1228** | **1228** | **1228** | **1228** |
| **MBF** | **1228** | **1228** | **1228** | **1228** |
| 12,4: n012.txt | **Melhor** | **1000** | **1000** | **1000** | **1000** |
| **MBF** | **986,2** | **991,88** | **992,2** | **995,85** |
| 30,5: n030.txt | **Melhor** | **4912** | **5194** | **5194** | **5194** |
| **MBF** | **4710,98** | **5032,62** | **5062,32** | **5067,4** |
| 60,6: n060.txt | **Melhor** | **17187** | **18096** | **18524** | **18579** |
| **MBF** | **16344,62** | **17808,76** | **18175,51** | **18258,62** |
| 120,10: n120.txt | **Melhor** | **39027** | **43015** | **44817** | **45226** |
| **MBF** | **38038,12** | **42433,86** | **44201,32** | **44731,4** |
| 240,12: n240.txt | **Melhor** | **126073** | **139418** | **146003** | **148021** |
| **MBF** | **124054,9** | **138002** | **144306,42** | **146776** |

## Variáveis da experiência em análise:

* Número de iterações entre 100 e 10000;
* Realização de 50 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

* Através da tabela de resultados, verifica-se que um maior número de iterações promove uma melhor solução em todos os ficheiros testados, uma vez que existe uma maior probabilidade de encontrar a diversidade máxima do problema.
* Como o número de elementos e subconjuntos vai aumentando progressivamente, cada vez mais o número de iterações influencia a qualidade da melhor solução. De maneira que, existindo vários elementos e subconjuntos ,100 iterações podem não ser suficiente para determinar a solução ótima.
* Este algoritmo é melhor que o algoritmo em que esta incluído apena um vizinho pois tem em conta dois vizinhos, aumentando assim a probabilidade de encontrar o ótimo global.
* **Trepa Colinas (probabilístico) –** este algoritmo apresenta o mesmo método de avaliação de soluções do algoritmo “Trepa Colinas (1 vizinhança)”, mas aceita também soluções piores consoante uma certa probabilidade predefinida.

É repetido este processo um certo número de interações predefinido.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NUM ELEMENTOS E SUBCONJUNTOS |  | 100 iterações Prob = 0.01 | 5000 iterações Prob = 0.01 | 100 iterações Prob = 0.005 | 5000 iterações Prob = 0.005 |
| 10,2: n010.txt | **Melhor** | **1228** | **1228** | **1228** | **1228** |
| **MBF** | **1201,5** | **1207,9** | **1213,98** | **1218,22** |
| 12,4: n012.txt | **Melhor** | **924,7** | **930,4** | **955,74** | **969,27** |
| **MBF** | **1000** | **1000** | **1000** | **1000** |
| 30,5: n030.txt | **Melhor** | **4771** | **5006** | **4782** | **5153** |
| **MBF** | **4503,32** | **4802,62** | **4513,97** | **4900** |
| 60,6: n060.txt | **Melhor** | **16372** | **18102** | **16362** | **18018** |
| **MBF** | **15754,58** | **16951** | **15708,64** | **17428,86** |
| 120,10: n120.txt | **Melhor** | **37488** | **42126** | **38296** | **43085** |
| **MBF** | **36423,88** | **41050,94** | **36563** | **41889,98** |
| 240,12: n240.txt | **Melhor** | **123647** | **139013** | **123473** | **142178** |
| **MBF** | **120900,96** | **137182,703** | **120773,48** | **139298,16** |

## Variáveis da experiência em análise:

* Número de iterações entre 100 e 5000;
* Variação da probabilidade de 0.001 e de 0,005.
* Realização de 50 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

* Através da tabela de resultados, verifica-se que para o mesmo número de iterações quanto menor for a probabilidade melhor a solução em todos os ficheiros testados.
* Como o número de elementos e subconjuntos vai aumentando progressivamente, cada vez mais o número de iterações influencia a qualidade da melhor solução. De maneira que, existindo vários elementos e subconjuntos ,100 iterações podem não ser suficiente para determinar a solução ótima.
* Este algoritmo é melhor que o algoritmo “Trepa Colinas com 1 vizinhança” pois tem a probabilidade de em algumas das vezes que analisa uma vizinhança escolher uma vizinhança pior, em que aparentemente é uma pior decisão. No entanto, como a probabilidade de isto acontecer é bastante baixa, não corrompe o objetivo do algoritmo e tem uma pequena probabilidade de ao selecionar o vizinho pior, esse mesmo vizinho ter um outro vizinho que tenha a melhor solução encontrada até ao momento.

**Recristalização simulada –** O algoritmo local Recristalização Simulada é bastante semelhante ao algoritmo local “Trepa Colinas”, contudo pretende ultrapassar máximos locais tendo em conta o processo de arrefecimento. A temperatura contribui para a melhor solução possível, visto que a probabilidade de aceitar soluções piores é influenciada diretamente pela temperatura, e a probabilidade de aceitar soluções piores pode se revelar essencial para o algoritmo não ficar preso num ótimo local. No entanto como a temperatura baixa ao longo da sua execução, o algoritmo cada vez mais se torna semelhante ao “Trepa Colinas”. A variação lenta da temperatura é determinada com o aumento do número de iterações do algoritmo.

* Neste algoritmo, a probabilidade de aceitar vizinhos piores deteriora-se seguindo a seguinte fórmula:
  + - **eprob = exp((custo - custo\_viz) / temperatura).**
* Seguidamente aceita-se o custo da vizinhança, caso este seja superior ou igual ao custo da solução atual, uma vez que estamos perante um problema de maximização.

Contudo, caso seja inferior aceitasse o custo, mas apenas com uma certa probabilidade ***(random < eprob)***, que vai ter em conta o “*eprob*” entrando aqui o fator do arrefecimento.

* Este algoritmo baixa constantemente a temperatura com o número de iterações, proporcionado obter uma temperatura muito baixa e que a escolha de estados de pior avaliação quase nunca aconteça, tornando o algoritmo “quase” como o Trepa-Colinas.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NUM ELEMENTOS E SUBCONJUNTOS |  | Iterações = 1000 Tmax = 100 Tmin = 5 | Iterações = 1000 Tmax = 10 Tmin = 0.5 | Iterações = 1000 Tmax = 1 Tmin = 0.05 | Iterações = 10000 Tmax = 10 Tmin = 0.5 | Iterações = 10000 Tmax = 100 Tmin = 0.01 | Iterações = 10000 Tmax = 0.99 Tmin = 0.0001 |
| 10,2: n010.txt | **Melhor** | **1228** | **1228** | **1228** | **1228** | **1228** | **1228** |
| **MBF** | **1228** | **1228** | **1228** | **1228** | **1228** | **1228** |
| 12,4: n012.txt | **Melhor** | **1000** | **1000** | **1000** | **1000** | **1000** | **1000** |
| **MBF** | **935,15** | **931.32007** | **935,67993** | **988.96** | **1000** | **1000** |
| 30,5: n030.txt | **Melhor** | **4502** | **4402** | **4414** | **4487** | **4623** | **4617** |
| **MBF** | **41178,24** | **4179,18** | **4186.839844** | **4335.899** | **4490.97** | **4462,1401** |
| 60,6: n060.txt | **Melhor** | **15182** | **14917** | **15383** | **15281** | **15522** | **15418** |
| **MBF** | **14640,29981** | **14574,36** | **14636,09961** | **14984.259766** | **15193.139648** | **15177,62017** |
| 120,10: n120.txt | **Melhor** | **35710** | **35456** | **35313** | **35643** | **35972** | **366177** |
| **MBF** | **34489,199** | **345553,3** | **34456,73** | **35019,07813** | **35476,39844** | **35514.73821** |
| 240,12: n240.txt | **Melhor** | **118270** | **118959** | **119136** | **119983** | **120260** | **120704** |
| **MBF** | **117158,6996** | **117146,3438** | **117242,9609** | **118416,2185** | **119214,5** | **119201.882813** |

## Variáveis da experiência em análise:

* Número de iterações entre 1000 e 10000;
* Variação da temperatura máxima entre 0.99 e 100;
* Variação da temperatura mínima entre 0.0001e 5;
* Realização de 50 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

Devido ao aumento da temperatura máxima ser proporcional à diminuição da temperatura mínima, podemos verificar que os resultados acabam por ser bastantes semelhantes uns dos outros, no entanto, é visível que uma temperatura mais baixa e o aumento de iterações contribui para uma maior probabilidade de chegar ao ótimo global.

Relativamente ao algoritmo em si, comparativamente com os outros, é bastante visível que o algoritmo tem resultados piores, devido ao facto de quando encontra um máximo (podendo ser máximo local), o algoritmo prossegue “durante algum tempo”, aceitando soluções piores, com perspetiva de ultrapassar um ótimo local, no entanto, no final de contas, acaba por se tornar pouco eficaz na sua procura pelo ótimo global.

**Pesquisa Tabu –** o algoritmo local Tabu é um algoritmo que visa a exploração de novas zonas do espaço de procurar evitando dessa forma a entrada em ciclos. Dessa forma, verifica-se que a obtenção de uma solução ótima é calculada de forma rápida, utilizando um tempo computacional satisfatório.

O seu funcionamento consiste na procura de possíveis soluções, partindo de uma solução inicial. A cada iteração tenta avançar para outra solução na sua vizinhança que satisfaça um determinado parâmetro de paragem.

Assim, caso seja uma solução válida marca essa solução numa lista de soluções visitadas denominada lista tabu, evitando assim que seja visitada mais tarde. Desse modo, apresenta um comportamento determinista escolhendo sempre o melhor vizinho.

## Variáveis da experiência em análise:

* Número de iterações entre 1000 e 5000;
* Variação do parâmetro tabu descidas entre 1 e 10;
* Realização de 30 testes para cada ficheiro.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NUM ELEMENTOS E SUBCONJUNTOS |  | 1000 iterações Tabu Descidas=5 | 5000 iterações Tabu Descidas = 5 | 1000 iterações Tabu Descidas=10 | 5000 iterações Tabu Descidas=10 | 1000 iterações Tabu Descidas=1 | 5000 iterações Tabu Descidas=1 |
| 10,2: n010.txt | **Melhor** | **1228** | **1228** | **1228** | **1228** | **1228** | **1228** |
|  | **MBF** | **1228** | **1228** | **1228** | **1228** | **1228** | **1228** |
| 12,4: n012.txt | **Melhor** | **1000** | **1000** | **1000** | **1000** | **1000** | **1000** |
|  | **MBF** | **993.23** | **985.2** | **986,63362** | **988.23** | **987.7** | **990.73337** |
| 30,5: n030.txt | **Melhor** | **5194** | **5194** | **5194** | **5194** | **5194** | **5194** |
|  | **MBF** | **5013.46** | **5080.63** | **4998.66** | **5133,86** | **4998.60** | **5077.66504** |
| 60,6: n060.txt | **Melhor** | **17938** | **18451** | **18185** | **18955** | **18029** | **18447** |
|  | **MBF** | **17587,22** | **18135,90039** | **17679.03** | **18528.166** | **17565.066406** | **18119,13281** |
| 120,10: n120.txt | **Melhor** | **42631** | **44303** | **42917** | **45703** | **42466** | **44186** |
|  | **MBF** | **41886.93359** | **43726,3** | **42173.73** | **45158.5** | **41612.800781** | **433680.80** |
| 240,12: n240.txt | **Melhor** | **138668** | **144503** | **139056** | **145676** | **136487** | **143349** |
|  | **MBF** | **1368833,703** | **142780,06** | **137986.40** | **144803.359375** | **135128.593750** | **142143.734375** |

## Análise do efeito da variação de parâmetros:

Através da tabela de resultados verifica-se que para um maior número de iterações e de descidas o algoritmo é mais preciso obtendo melhores soluções. É preciso ter em conta também que o número de descidas não pode ser elevado demais senão o algoritmo acaba por ficar sem soluções possíveis.

Estes mesmos resultados conseguem ser melhores que os anteriores verificados devido ao facto de o algoritmo guardar uma lista de soluções já antes visitadas, como já mencionado anteriormente, possibilitando desta forma, o algoritmo não ficar preso em ótimos locais, e, em simultâneo poder aceitar soluções piores da lista de vizinhos para fugir destes mesmo máximos locais, a junção destas qualidades num só algoritmo é essencial para a eficácia dele.

# Algoritmo Evolutivo

Os algoritmos evolutivos são algoritmos que apresentam mecanismos de evolução biológica, visto que permitem determinar a quantidade de recursos necessária para resolver “problemas” de sobrevivência. Assim têm como base a teoria da Evolução de Darwin, uma vez que contam com processos de **recombinação** e **mutação**.

A **recombinação** ou **crossover** são operadores genéticos usados para variar o processo de programação de cromossomas de uma geração para a próxima, ou seja, alteram a composição genética dos filhos durante a reprodução. Os operadores trabalham sobre a codificação de possíveis soluções(genótipo) e não sobre as soluções(fenótipos) propriamente ditas.

A **recombinação** de pais pode ser considerada uma operação de custo computacional relativamente alta para o problema de diversidade máxima de grupo uma vez que é possível calcular a diferença da qualidade entre duas soluções sem te que que necessariamente calcular a qualidade destas duas soluções, ou seja, a recombinação entre dois pais acontece com uma determinada probabilidade na descendência de um filho.

A **mutação** é um operador que após ser aplicado a um genótipo gera um filho. Similar a recombinação, a mutação é um operador sempre estocástico: o seu resultado filho depende dos resultados de uma série de escolhas aleatórias.

Assim, a variação de parâmetros influência o comportamento dos algoritmos genéticos, para que se possa conforme as necessidades do problema e recursos disponíveis estabelecê-los da melhor forma.

* **Tamanho da População**(**popsize)** – o tamanho da população afeta de forma significativa o desempenho global dos algoritmos genéticos.

Com uma população pequena o desempenho pode cair, pois deste modo a população fornece uma pequena cobertura do espaço de procura do problema. Por outro lado, uma grande população fornece uma cobertura já considerada representativa do domínio do problema. Contudo, grandes populações exigem maiores recursos computacionais.

* **Probabilidade de mutação**(**pm\_swap**) – uma alta probabilidade de mutação procura-se essencialmente uma maior aleatoriedade entre filhos.

Por outro lado, uma menor probabilidade previne que uma dada posição fique estagnada em determinado valor, para alem de que possibilita que se chegue em qualquer ponto do espaço de procura.

* **Probabilidade de Cruzamento**(**pr)**– quanto maior for a probabilidade de cruzamento, mais rapidamente novas estruturas serão introduzidas na população. Contudo, uma elevada probabilidade pode provocar a perda de estruturas de alta aptidão, ou seja, grande parte da população com boas aptidões é substituída.

Por outro lado, uma menor probabilidade de cruzamento, torna o algoritmo mais lento e com mais dificuldades de obter novas estruturas.

* **Número de gerações(numGenerations)** - O número de gerações contribui diretamente no tempo de computação necessário para o fim da pesquisa. Não influenciando diretamente a convergência dos algoritmos genéticos.
* **Tamanho do torneio(t\_size) –** um grupo de indivíduos são escolhidos aleatoriamente de uma população e o melhor deles é escolhido. Este processo é repetido até que se obtenha uma nova população.

**Algoritmo Evolutivo Base (Mutação Binária com 1 Ponto de Corte):** a implementação deste algoritmo tem como principal objetivo a mutação aleatória de genes e recombinação dos subconjuntos com 1 ponto de corte.  
Para alem disso, este algoritmo rejeita soluções inválidas, ou seja, soluções em que o número de elementos de subconjunto esperado é diferente do número de elementos obtido.   
  
Este algoritmo conta com o processo de **mutação binária** que consiste em percorrer todos os indivíduos da população e para cada um percorrer os seus elementos do conjunto. De seguida, é gerado um número aleatório e comparado com a probabilidade de mutação(**pm\_swap).** Caso o número gerado seja menor que a probabilidade de mutação é trocado um bit aleatoriamente com uma certa probabilidade.

Para além deste processo, conta também com a **recombinação com 1 ponto de corte**, em que é percorrida toda a população e seguidamente se um número aleatoriamente sorteado for menor que a probabilidade de cruzamento cria um ponto de corte e faz o devido corte nesse ponto. Este corte consiste em criar o cromossoma do filho tendo em conta metade dos genes dos seus pais, ou seja, metade dos genes do pai e metade dos genes da mãe. Os filhos gerados vão ter ainda bastantes semelhanças com os seus pais.  
  
O algoritmo evolutivo base implementado tem em conta a variação de alguns dos parâmetros acima referidos como é possível observar no quadro seguinte:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Algoritmo: Mutação binária + Recombinação de 1 ponto de corte** | | | | | | | | | | | |
|  |  | **n010.txt** | | **n012.txt** | | **n030.txt** | | **n060.txt** | | **n120.txt** | | **n240.txt** | |
| Parâmetros Fixos | Parâmetros a variar | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF |
| ger = 2500 | pr = 0.3 | 1228,0 | 1228,0 | 1000,0 | 890.7 | 4265 | 4202.1 | 15198 | 149918.9 | 35019 | 34473.5 | 118659 | 117360.7 |
| pop = 100 | pr = 0.5 | 1228,0 | 1228,0 | 945,0 | 872.7 | 4466 | 4221.8 | 15408 | 1459.5 | 34598 | 34251.5 | 117903 | 117254.2 |
| pm = 0.01 | pr = 0.7 | 1228,0 | 1228,0 | 961 | 886.7 | 4280 | 4161.7 | 15185 | 14792.5 | 34682 | 34302.3 | 118828 | 117291.6 |
| ger = 2500 | pm = 0.0 | 1228,0 | 1226.5 | 887 | 856.2 | 4183 | 4047.8 | 14575 | 14330.7 | 34989 | 34296.5 | 118145 | 117151.9 |
| pop = 100 | pm = 0.001 | 1228,0 | 1228.0 | 984 | 899.5 | 4275 | 4166.2 | 15051 | 14858.5 | 35139 | 34850.3 | 119334 | 1182928 |
| t\_size=5 | pm=0.01 | 1228 | 1228 | 1000 | 904.4 | 4342 | 4150.6 | 15569 | 14728.2 | 35428 | 34350.6 | 118137 | 117233.5 |
| pr = 0.7 | pm = 0.05 | 1228 | 1228 | 1000 | 903.9 | 4149 | 4076.1 | 14766 | 14491.5 | 344452 | 34235.3 | 119069 | 117286.7 |
| pr = 0.7 | pop = 10 (ger = 25K) | 1228 | 1228 | 945 | 807.5 | 4366 | 4105.6 | 14525 | 13980 | 34703 | 33896.0 | 116817 | 115903.6 |
| pm = melhor valor obtido=0.01 | pop = 50 (ger = 5K) | 1228 | 1228 | 967 | 866.9 | 4397 | 4155.5 | 15153 | 14884.5 | 34971 | 34299.1 | 117870 | 116978.2 |
| t\_size=5 | pop = 100 (ger = 2.5K) | 1228 | 1228 | 961 | 864.7 | 4324 | 4162.2 | 15147 | 14639.4 | 34761 | 34351.1 | 117442 | 117053.6 |

## Variáveis da experiência em análise:

* Número de iterações 1000;
* Tamanho do torneio 5;
* Número de gerações entre 2500 e 25000;
* Tamanho da população entre 10 e 100;
* Probabilidade de cruzamento entre 0.3 e 0.7;
* Probabilidade de mutação entre 0.0 e 0.05;
* Realização de 30 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

Relativamente a probabilidade de cruzamento, é possível verificar que uma probabilidade muito alta ou uma probabilidade muito baixa é o menos ideal, sendo a probabilidade mediana, de 0.5% a melhor, estas alterações divergem pouco entre si.

Comparativamente à probabilidade de mutação, podemos verificar que uma probabilidade muito baixa de mutação, de 0.001% é a melhor probabilidade a ser usada, pois aumentando a mutação, as soluções acabam por se estragar devido à elevada aleatoriedade.

As últimas 3 linhas da tabela têm como objetivo testar o impacto do número da população e do número de gerações existentes no algoritmo, podemos verificar assim que quanto maior o número de população, mais fácil será de chegar ao máximo ótimo do problema, no entanto as gerações têm o mesmo impacto, só que não é tão significativo, daí o teste que tem mais população tem uma melhor solução do que o teste que se foca mais no maior número de gerações.

**Algoritmo Evolutivo (Mutação binária com recombinação Uniforme):**  este algoritmo tem como principal foco a recombinação uniforme dos subconjuntos tal como a mutação aleatória de genes.

O funcionamento do **algoritmo de recombinação** **uniforme** passa pela criação da sua “Crossover Mask” e de seguida, a criação dos seus descendentes é decidida conforme cada bit da “Crossover Mask”, caso o *1º* bit da “Corssover Mask” seja **‘1’**, vai ao progenitor “pai” buscar o seu *1º* bit, caso o *2º* bit seja **‘0’** vai buscar o *2º* bit ao progenitor “mãe”, e vise versa.

Comparativamente aos algoritmos com recombinação com 1 ponto de corte e 2 pontos de cortes, é expectável uma ligeira melhoria a nível de soluções obtidas, devido a uma maior aleatoriedade na criação dos seus descentes relativamente aos seus progenitores, aumentando assim ligeiramente a chance de uma solução ótima.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Algoritmo mutacao binaria +recombinacao uniforme** | | | | | | | | | | | |
|  |  | **n010.txt** | | **n012.txt** | | **n030.txt** | | **n060.txt** | | **n120.txt** | | **n240.txt** | |
| Parâmetros Fixos | Parâmetros a variar | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF |
| ger = 2500 | pr = 0.3 | 1228,0 | 1228,0 | 1000,0 | 907,7 | 4378 | 4243,3 | 15634 | 15070,3 | 35305 | 34246 | 117911 | 117117,1 |
| pop = 100 | pr = 0.5 | 1228,0 | 1228,0 | 1000,0 | 897,0 | 4375 | 4199,2 | 15766 | 14966,1 | 35571 | 34282 | 118350 | 117528,5 |
| pm = 0.01 t\_size=5 | pr = 0.7 | 1228,0 | 1228,0 | 1000 | 901,5 | 4356 | 4218,9 | 15724 | 14871,8 | 35128 | 34366 | 117355 | 11700,4 |
| ger = 2500 | pm = 0.0 | 1228,0 | 1228,0 | 967 | 852,3 | 4111 | 4057,7 | 14725 | 14480,8 | 35028 | 34325 | 117924 | 117294,5 |
| pop = 100 | pm = 0.001 | 1228,0 | 1228,0 | 956 | 869,8 | 4286 | 4152 | 15248 | 14801,2 | 35197 | 34907 | 120772 | 118791 |
| t\_size=5 | pm=0.01 | 1228,0 | 1228,0 | 984 | 904,2 | 4453 | 4221,1 | 15386 | 14967,2 | 34888 | 34383 | 117792 | 117059 |
| pr = 0.7 | pm = 0.05 | 1228,0 | 1228,0 | 1000 | 915,2 | 4119 | 4040,2 | 14522,3 | 14522,3 | 35221 | 34370 | 118959 | 117399,8 |
| pr = 0.7 | pop = 10 (ger = 25K) | 1228,0 | 1228,0 | 918 | 776,4 | 4215 | 4069,9 | 14341 | 14019,8 | 34109 | 33558 | 116732 | 115935,1 |
| pm = melhor valor obtido = 0.01 | pop = 50 (ger = 5K) | 1228,0 | 1228,0 | 1000 | 870 | 4247 | 4093,5 | 15043 | 14811,7 | 34660 | 34130.2 | 117102 | 116874,2 |
| t\_size=5 | pop = 100 (ger = 2.5K) | 1228,0 | 1228,0 | 1000 | 891 | 4367 | 4235,1 | 15258 | 15019 | 35247 | 34358 | 117500 | 117050 |

## Variáveis da experiência em análise:

* Número de iterações 1000;
* Tamanho do torneio 5;
* Número de gerações entre 2500 e 25000;
* Tamanho da população entre 10 e 100;
* Probabilidade de cruzamento entre 0.3 e 0.7;
* Probabilidade de mutação entre 0.0 e 0.05;
* Realização de 30 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

Relativamente a probabilidade de cruzamento, é possível verificar que uma probabilidade muito alta ou uma probabilidade muito baixa é o menos ideal, sendo a probabilidade mediana, de 0.5% a melhor, estas alterações divergem pouco entre si.

Comparativamente à probabilidade de mutação, podemos verificar que uma probabilidade muito baixa de mutação, de 0.001% é a melhor probabilidade a ser usada, pois aumentando a mutação, as soluções acabam por se estragar devido à elevada aleatoriedade.

As últimas 3 linhas da tabela têm como objetivo testar o impacto do número da população e do número de gerações existentes no algoritmo, podemos verificar assim que quanto maior o número de população, mais fácil será de chegar ao máximo ótimo do problema, no entanto as gerações têm o mesmo impacto, só que não é tão significativo, daí o teste que tem mais população tem uma melhor solução do que o teste que se foca mais no maior número de gerações.

**Algoritmo Evolutivo (Mutação binária com 2 pontos de corte):** a implementação deste algoritmo tem como principal objetivo a mutação aleatória de genes a e recombinação dos subconjuntos com 2 pontos de corte.

Relativamente ao algoritmo com 1 ponto de corte, pressupõe-se uma melhoria uma vez que existe uma **recombinação** mais elaborada com **2 pontos de corte**, em que o cromossoma filho recebe genes dos seus pais alternadamente. Assim, invés de receber a mesma quantidade de genes de cada parente, recebe genes do pai, seguidamente genes da mãe e por fim novamente genes do pai. Este processo pode também ser realizado de forma inversa sendo os genes iniciais do cromossoma filho os genes do seu pai. Os filhos gerados com 2 pontos de corte em vez de apenas 1, vão garantir uma aleatoriedade um pouco maior consequentemente gerando melhores soluções.  
Este algoritmo evolutivo implementado tem em conta a variação de alguns dos parâmetros acima referidos como é possível observar no quadro seguinte:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Algoritmo: mutacao binaria + recombinacao com 2 pontos de corte** | | | | | | | | | | | |
|  |  | **n010.txt** | | **n012.txt** | | **n030.txt** | | **n060.txt** | | **n120.txt** | | **n240.txt** | |
| Parâmetros Fixos | Parâmetros a variar | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF |
| ger = 2500 | pr = 0.3 | 1228,0 | 1228,0 | 954,0 | 885.2 | 4402 | 4229.0 | 15295 | 14930.9 | 35010 | 34358.3 | 117986 | 117508.5 |
| pop = 100 | pr = 0.5 | 1228,0 | 1228,0 | 957,0 | 893.5 | 4470 | 4258,5 | 15514 | 14962,6 | 34922 | 34270 | 117270 | 118401 |
| pm = 0.01 t\_size=5 | pr = 0.7 | 1228,0 | 1228,0 | 1000 | 909 | 4417 | 4205 | 14998 | 14728,4 | 34889 | 34372 | 117774 | 117071,1 |
| ger = 2500 | pm = 0.0 | 1228,0 | 1228,0 | 984 | 874 | 4077 | 4034,8 | 14582 | 14356,4 | 34521 | 34238 | 118153 | 117340,5 |
| pop = 100 | pm = 0.001 | 1228,0 | 1228,0 | 1000 | 911 | 4313 | 4191.7 | 15094 | 14747.2 | 35058 | 34668 | 121712 | 119168.10 |
| t\_size=5 | pm=0.01 | 1228 | 1228 | 984 | 915.4 | 4317 | 4195 | 15102 | 14780.7 | 34513 | 34250.6 | 118082 | 117180.2 |
| pr = 0.7 | pm = 0.05 | 1228 | 1228 | 1000 | 910,9 | 4117 | 4035.6 | 14665 | 14409,8 | 34717 | 34345 | 118382 | 117428.2 |
| pr = 0.7 | pop = 10 (ger = 25K) | 1228 | 1228 | 1000 | 819.0 | 4280 | 4026 | 14584 | 14060.59 | 34003 | 33490.8 | 117469 | 116072.2 |
| pm = melhor valor obtido = 0.01 | pop = 50 (ger = 5K) | 1228 | 1228 | 967 | 882.2 | 4255 | 4113.7 | 14981 | 14508.5 | 34631 | 34085.5 | 118957 | 116846.7 |
| t\_size=5 | pop = 100 (ger = 2.5K) | 1228 | 1228 | 1000 | 892.0 | 4459 | 4262.20 | 15128 | 14956.5 | 34959 | 34473.8 | 118364 | 11703.3 |

## Variáveis da experiência em análise:

* Número de iterações 1000;
* Tamanho do torneio 5;
* Número de gerações entre 2500 e 25000;
* Tamanho da população entre 10 e 100;
* Probabilidade de cruzamento entre 0.3 e 0.7;
* Probabilidade de mutação entre 0.0 e 0.05;
* Realização de 30 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

Relativamente a probabilidade de cruzamento, é possível verificar que uma probabilidade muito alta ou uma probabilidade muito baixa é o menos ideal, sendo a probabilidade mediana, de 0.5% a melhor, estas alterações divergem pouco entre si.

Comparativamente à probabilidade de mutação, podemos verificar que uma probabilidade muito baixa de mutação, de 0.001% é a melhor probabilidade a ser usada, pois aumentando a mutação, as soluções acabam por se estragar devido à elevada aleatoriedade.

As últimas 3 linhas da tabela têm como objetivo testar o impacto do número da população e do número de gerações existentes no algoritmo, podemos verificar assim que quanto maior o número de população, mais fácil será de chegar ao máximo ótimo do problema, no entanto as gerações têm o mesmo impacto, só que não é tão significativo, daí o teste que tem mais população tem uma melhor solução do que o teste que se foca mais no maior número de gerações.

**Algoritmo Evolutivo (Mutação por troca com 2 pontos de corte):** o foco principal deste algoritmo é a recombinação com dois pontos de troca e de seguida a implementação da mutação por troca.

As diferenças entre este algoritmo e o anterior é o tipo de mutação que é efetuada,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Algoritmo Base mutacao por troca + recombinacao com 2 pontos de corte** | | | | | | | | | | | |
|  |  | **n010.txt** | | **n012.txt** | | **n030.txt** | | **n060.txt** | | **n120.txt** | | **n240.txt** | |
| Parâmetros Fixos | Parâmetros a variar | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF |
| ger = 2500 | pr = 0.3 | 1228,0 | 1228,0 | 1000,0 | 992.7 | 5194 | 5061 | 18324 | 17872,7 | 43662 | 42825 | 139947 | 138487.2 |
| pop = 100 | pr = 0.5 | 1228,0 | 1228,0 | 1000,0 | 998,1 | 5194 | 5084 | 18335 | 17956.9 | 43756 | 42673.1 | 139967 | 138643,7 |
| pm = 0.01 t\_size=5 | pr = 0.7 | 1228,0 | 1228,0 | 1000 | 991.1 | 5194 | 5079.7 | 18295 | 17921.9 | 43005 | 42542.9 | 139223 | 138145.1 |
| ger = 2500 | pm = 0.0 | 1228,0 | 1228,0 | 954 | 869.5 | 4236 | 4079.9 | 15000 | 14435.9 | 35160 | 34381.1 | 118436 | 117335.7 |
| pop = 100 | pm = 0.001 | 1228,0 | 1228,0 | 1000 | 988.2 | 5049 | 4803.2 | 17100 | 16696.4 | 39799 | 38661 | 127913 | 126738.4 |
| t\_size=5 | pm=0.01 | 1228 | 1228 | 1000 | 997.8 | 5194 | 5043.4 | 18363 | 17921.3 | 43173 | 42626.1 | 139450 | 138189.9 |
| pr = 0.7 | pm = 0.05 | 1228 | 1228 | 1000 | 996.7 | 5194 | 5091.3 | 18460 | 18183.6 | 44807 | 44239.8 | 145665 | 144117.8 |
| pr = 0.7 | pop = 10 (ger = 25K) | 1228 | 1228 | 1000 | 993.7 | 5156 | 5029 | 18243 | 17904 | 43298 | 42779.3 | 140138 | 138866.1 |
| pm = melhor valor obtido = 0.01 | pop = 50 (ger = 5K) | 1228 | 1228 | 1000 | 998.93 | 5194 | 5052 | 18127 | 17895.7 | 43288 | 42715.4 | 140321 | 128644.4 |
| t\_size=5 | pop = 100 (ger = 2.5K) | 1228 | 1228 | 1000 | 995.2 | 5194 | 5060.16 | 18355 | 17849.5 | 43237 | 42580.8 | 140940 | 138344.6 |

## Variáveis da experiência em análise:

* Número de iterações 1000;
* Tamanho do torneio 5;
* Número de gerações entre 2500 e 25000;
* Tamanho da população entre 10 e 100;
* Probabilidade de cruzamento entre 0.3 e 0.7;
* Probabilidade de mutação entre 0.0 e 0.05;
* Realização de 30 testes para cada ficheiro.

## Análise do efeito da variação de parâmetros:

Relativamente a probabilidade de cruzamento, é possível verificar que uma probabilidade muito alta ou uma probabilidade muito baixa é o menos ideal, sendo a probabilidade mediana, de 0.5% a melhor, estas alterações divergem pouco entre si.

Comparativamente à probabilidade de mutação, podemos verificar que uma probabilidade muito baixa de mutação, de 0.001% é a melhor probabilidade a ser usada, pois aumentando a mutação, as soluções acabam por se estragar devido à elevada aleatoriedade.

As últimas 3 linhas da tabela têm como objetivo testar o impacto do número da população e do número de gerações existentes no algoritmo, podemos verificar assim que quanto maior o número de população, mais fácil será de chegar ao máximo ótimo do problema, no entanto as gerações têm o mesmo impacto, só que não é tão significativo, daí o teste que tem mais população tem uma melhor solução do que o teste que se foca mais no maior número de gerações.

# Algoritmo Híbrido

introdução

**Algoritmo híbrido (Algoritmo evolutivo + Trepa Colinas Simples):**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Algoritmo Híbrido + Trepa Colinas** | | | | | | | | | | | |
|  |  | **n010.txt** | | **n012.txt** | | **n030.txt** | | **n060.txt** | | **n120.txt** | | **n240.txt** | |
| Parâmetros Fixos | Parâmetros a variar | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF |
| ger = 2500 | pr = 0.3 | 1228,0 | 1228,0 | 1000,0 | 995,2 | 5087 | 4989,3 | 18341 | 18042,1 | 43663 | 43179,7 | 141166 | 140410,4 |
| pop = 100 | pr = 0.5 | 1228,0 | 1228,0 | 1000,0 | 998,3 | 5156 | 4984 | 18074 | 17554 | 41787 | 41357,3 | 141161 | 140389,8 |
| pm = 0.01 t\_size=5 | pr = 0.7 | 1228,0 | 1228,0 | 1000 | 992 | 5153 | 5055,8 | 18284 | 18064,6 | 43502 | 43036.9 | 140795 | 140127,1 |
| ger = 2500 | pm = 0.0 | 1228,0 | 1228,0 | 1000 | 989,2 | 5179 | 5050 | 17840 | 17467,5 | 42094 | 41520,5 | 136386 | 134698,5 |
| pop = 100 | pm = 0.001 | 1228,0 | 1228,0 | 1000 | 987 | 5108 | 4975,3 | 18026 | 17670,7 | 42110 | 41815,3 | 137805 | 135712,7 |
| t\_size=5 | pm = 0.01 | 1228,0 | 1228,0 | 1000 | 992 | 5153 | 5055,8 | 18284 | 18064,6 | 43502 | 43036.9 | 140795 | 140127,1 |
| pr = 0.7 | pm = 0.05 | 1228,0 | 1228,0 | 1000 | 998,4 | 5194 | 5093,6 | 18517 | 18296,2 | 44527 | 44258,6 | 144802 | 144333,7 |
| pr = 0.7 | pop = 10 (ger = 25K) | 1228,0 | 1228,0 | 1000 | 982,9 | 5140 | 5046 | 18368 | 18047,1 | 43788 | 43346,1 | 141994 | 140481 |
| pm = melhor valor obtido = 0.05 | pop = 50 (ger = 5K) | 1228,0 | 1228,0 | 1000 | 995 | 5156 | 5081,9 | 19168 | 18024,9 | 43581 | 43211,3 | 141152 | 140499 |
| t\_size=5 | pop = 100 (ger = 2.5K) | 1228,0 | 1228,0 | 1000 | 995 | 5194 | 5034,3 | 18219 | 18052,3 | 43320 | 43035 | 141175 | 140009,6 |

**Algoritmo híbrido (Algoritmo evolutivo + Trepa Colinas 2 vizinhos):**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Algoritmo Hibrido + Trepa Colinas 2 vizinhaças** | | | | | | | | | | | |
|  |  | **n010.txt** | | **n012.txt** | | **n030.txt** | | **n060.txt** | | **n120.txt** | | **n240.txt** | |
| Parâmetros Fixos | Parâmetros a variar | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF | Best | MBF |
| ger = 2500 | pr = 0.3 | 1228,0 | 1228,0 | 1000,0 | 996,0 | 5194 | 5064,1 | 18410 | 18059 | 43734 | 43343,7 | 143654 | 141544,2 |
| pop = 100 | pr = 0.5 | 1228,0 | 1228,0 | 1000,0 | 996,8 | 5173 | 5061,8 | 18527 | 18182,8 | 43643 | 43303,2 | 141847 | 141275,3 |
| pm = 0.01 t\_size=5 | pr = 0.7 | 1228,0 | 1228,0 | 1000 | 993,6 | 5194 | 5107 | 18128 | 17961,4 | 43745 | 43296 | 141807 | 141185,7 |
| ger = 2500 | pm = 0.0 | 1228,0 | 1228,0 | 1000 | 971 | 5194 | 5080,7 | 18063 | 17726 | 43373 | 42579,8 | 138972 | 138058,2 |
| pop = 100 | pm = 0.001 | 1228,0 | 1228,0 | 1000 | 1000 | 5194 | 5048 | 18288 | 17897 | 43238 | 42570,2 | 140019 | 138774,9 |
| t\_size=5 | pm = 0.01 | 1228,0 | 1228,0 | 1000 | 993,6 | 5194 | 5107 | 18128 | 17961,4 | 43745 | 43296 | 141807 | 141185,7 |
| pr = 0.7 | pm = 0.05 | 1228,0 | 1228,0 | 1000 | 998,4 | 5136 | 5025,6 | 18525 | 18235 | 44542 | 44257,9 | 145877 | 144868,3 |
| pr = 0.7 | pop = 10 (ger = 25K) | 1228,0 | 1228,0 | 1000 | 1000 | 5194 | 5055,7 | 18554 | 18344,4 | 44765 | 44477,5 | 143766 | 142847 |
| pm = melhor valor obtido = 0.05 | pop = 50 (ger = 5K) | 1228,0 | 1228,0 | 1000 | 1000 | 5153 | 5037,7 | 18547 | 18252,2 | 44457 | 44282,2 | 144238 | 143383,9 |
| t\_size=5 | pop = 100 (ger = 2.5K) | 1228,0 | 1228,0 | 1000 | 993,6 | 5194 | 5107 | 18128 | 17961,4 | 43745 | 43296 | 141807 | 141185,7 |

**Algoritmo híbrido (Algoritmo evolutivo + Tabu): TABELA DO EXCEL ESTA MAL FEITA.**