Otimização da Gestão de Projetos

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Inteligência Artificial

Grupo B5.3:
André Costa Moreira Maia - 201200674 - up201200674@fe.up.pt
Guilherme Vieira Pinto - 201305803 - up201305803@fe.up.pt
João Manuel Estrada Pereira Gouveia - 201303988 - up201303988@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

29 de Maio de 2016

Objetivo

Este trabalho é desenvolvido no âmbito da unidade curricular de Inteligência Artificial (IART), e tem como principal objectivo o estudo e investigação de algoritmos de otimização de soluções.
No projeto em questão, será abordada a otimização da gestão de projetos, isto é, a distribuição de trabalhadores pelas tarefas de um determinado projeto. Sendo que esta atribuição ocupa um vasto campo de soluções para projetos complexos com um considerável número de trabalhadores, recorremos a metodologias de otimização de soluções como algoritmos genéticos e arrefecimento simulado para alcançar o objectivo pretendido.

Tendo em conta a complexidade da organização de recursos orientados à gestão de projetos coordenados, temos como objetivo, o output de um valor ótimo para o problema.

Definiremos então um Projeto como sendo constituído por uym conjunto de Tarefas a serem concluídas por um ou mais Trabalhadores. As Tarefas podem ter precedências entre si e apresentam uma duração Trabalhador/mês. Cada tarefa não pode ser iniciada sem a totalidade das suas precedências estar completa.

Conteúdo

1 Especificação **4**
1.1 Algoritmos Genéticos 5
1.2 Arrefecimento Simulado 8

2 Desenvolvimento **9**

3 Experiências **12**
3.1 Algoritmos Genéticos 12
3.1.1 Redução do numero de Cromossomas 12
3.1.2 Redução do numero de Gerações 13
3.1.3 Aumento da probabilidade de Mutação 13
3.1.4 Aumento dos cortes por Crossover 14
3.1.5 Diminuição das seleções por geração 14
3.2 Arrefecimento Simulado 15
3.2.1 Aumento da Temperatura 15
3.2.2 Diminuição do Rácio de arrefecimento 15

4 Conclusões **16**

5 Melhoramentos **16**

1 Especificação

O problema apresentado enquadra-se no paradigma da otimização de projetos generalizados.

Tendo em conta as variáveis a ser trabalhadas, devemos ter em conta os seguintes pontos:

- Um *Projeto* é constituido por uma ou mais *Tarefas*;
- Uma *Tarefa* tem associadas as seguintes carateristicas:
  - Duração (Pessoas/Mês);
  - *Tarefas* precedentes;
  - *Scope* (Âmbito);
- Um *Trabalhador* é caracterizado pelas suas *Skills* (Competências);
- Uma *Skill* é constituída pelo tuplo *Scope-Performance*.

Relativamente a algumas definições dos termos acima mencionados, deve-se clarificar os seguintes:

- **Duração do Projeto** - O término do projeto deverá corresponder ao final da execução da última tarefa concluida;
- **Execução de uma Tarefa** - Uma tarefa só deve ser iniciada quando todas as suas precedências foram concluidas;
- **Duração em Pessoa/Mês** - Esta unidade de tempo foi interpretada de forma a que, dada uma tarefa com *Scope* **S**, com duração de **t** Pessoas/Mês e tendo a ela associados **n** trabalhadores com um somatório total de **P** *Performance* (sendo que apenas as performances consideradas são relativas a *Skills* equivalentes a **S**), temos: $\boldsymbol{RealDuration = \frac{t}{P}}$, sendo *RealDuration*, a duração definitiva da Tarefa após atribuição dos Trabalhadores;
- **Atribuição de Trabalhadores** - Na nossa abordagem, é possivel a atribuição de Trabalhadores a Tarefas cujas *Skills* não se enquadrem na *Scope* pretendida; no entanto, estes Trabalhadores em nada influenciarão o progresso da Tarefa;

Os resultados obtidos na execução do programa fornecerão uma solução otimizada para uma alocação eficaz de elementos de uma equipa de Trabalhadores, avaliando as diversas fases de desenvolvimento de um Projeto e, tendo em conta a possivel escalabilidade do caso para situações de maior porte, é possivel considerar a sua aplicação num enquadramento empresarial.

Para este problema, serão aplicados algoritmos orientados para a resolução de problemas de otimização, sendo eles Algoritmos Genéticos e Arrefecimento Simulado tendo em conta que a sua natureza probabilística permite o alcance e seleção de soluções aceitáveis sem ter que considerar todo o campo de hipóteses.

1.1 Algoritmos Genéticos

Os Algoritmos Genéticos pertencem à classe de algoritmos evolutivos, recorrendo a métodos de simulação comportamental de uma população de forma a que a seleção de soluções boas geram soluções progressivamente melhores. Para tal, são adotadas técnicas inspiradas na biologia evolutiva como mutação, seleção e reprodução (*crossover*).

Na nossa abordagem ao algoritmo, começamos pela definição da representação dos cromossomas a serem processados.

Figura 1: Representação de um cromossoma esquematizando um projeto constituído por três tarefas e seis trabalhadores associados.

Cromossomas Os cromossomas planeados serão estruturados como arrays binários com dimensão $T \times W$ (sendo que T simboliza o número de tarefas do projeto e W o número de trabalhadores a ele associados). O cromossoma representa, portanto, o projeto na íntegra, na medida em que inclui cada tarefa que o constitui. Esta representação permite identificar que trabalhadores estão associados para cada tarefa, sendo que o valor 1 num bit indica que o respetivo trabalhador lhe está destacado. Esta análise é explicada mais detalhadamente no cálculo de *fitness* de cada cromossoma.

Variáveis Ao iniciar o algoritmo, uma população com um determinado número de cromossomas é gerado. É definido o número de gerações a serem executadas, o número de cortes por cromossoma durante o *crossover* e a respetiva probabilidade de mutação na criação de um novo cromossoma dele derivado. É também possível definir o número de seleções por geração, caso se pretenda que o algoritmo opte por uma metodologia de seleção elitista ou, caso contrário, este adotará o método de seleção probabilistico, tendo em conta os *fitness's* da população.

População No decorrer do algoritmo, à medida que novas gerações são geradas, determinados indivíduos são excluidos para a próxima iteração, quer por motivos de penalização por *fitness* insatisfatório ou por seleção. De modo a preencher a população da geração seguinte, por cada elementos excluído, são gerados novos cromossomas por *crossover*, mantendo a integridade do domínio de soluções.

Seleção Na seleção de individuos de uma população, devemos ter em conta o tipo de análise a ser feita. Para tal, o array de cromossomas deve ser constantemente ordenado de geração para geração, por ordem crescente de *fitness*, e com os individuos de *fitness* negativo no final do array. Caso o número de seleções seja especificado, o método de escolha será elitista: sendo *N* o número de seleções pretendidas, o array ordenado de cromossomas é percorrido apartir do primeiro índice, até à posição *N-1*, eliminando todos os cromossomas com *fitness* negativo e aqueles que se encontrem apartir da posição de índice *N*, pelo que o array terminará sempre com um máximo de N elementos. Caso contrário, se o número elementos a selecionar não seja especificado, apenas os cromossomas com *fitness* negativo serão excluídos.

Crossover Relativamente ao *crossover*, este será executado tantas vezes quantas forem necessárias para completar a população, de modo a preencher as vagas geradas durante a seleção. Para tal, é preciso considerar se o algoritmo genétio está ser executado segundo uma prespetiva elitista ou não. Avaliando se o número de selecoes foi especificado: se foi, então são selecionados os 2 melhores cromossomas para *crossover*, caso contrário, são avaliados todos os cromossomas ainda presentes na população tendo em conta os seus *fitnesses* e, quanto melhor o seu valor, mais probabilidade terão de ser selecionados para *crossover*. Para além da determinação de cromossomas, é preciso tb o processamento de cortes: tendo em conta o número de cortes especificado, serão gerados índices de corte distintos de modo a determinar o ponto de *crossover* entre os cromossomas elegidos.

Figura 2: Representação do processo de crossover para uma situação com um corte (esquerda) ou dois cortes (direita)

Mutação Cada cromossoma resultante de um *crossover* terá os seus genes percorridos de modo a executar um sorteio de mutação para cada bit, tendo em conta a probabilidade de mutação definida. Caso seja definida mutação num determinado índice, o respetivo bit é modificado para o valor oposto.

Fitness A definição da função de *fitness* é o ponto crucial deste algoritmo, pelo que quantificará cada cromossoma e influenciará diretamente o decorrer da evolução da população a cada geração processada.

O valor de *fitness* calculado corresponderá ao valor do tempo de execução do projeto, caso a representação do cromossoma constitua uma alocação de trabalhadores válida para a a sua execução, caso contrário será quantificado como -1. Mais especificamente, entende-se como "válida", a alocação de trabalhadores tal que, para uma dada tarefa com *Scope* **S**, exista pelo menos um trabalhador a ela destacado que possua uma *Skill* compatível a **S**.

A determinação do tempo de execução de um projeto será equivalente ao momento de conclusão da última tarefa realizada, pelo que será esse mesmo valor que deveremos avaliar. Para tal, após verificar se a alocação de trabalhadores no cromossoma é válida, procede-se ao calculo da duração de cada tarefa, tendo em conta a execução das suas precedências e a disponibilidade de cada trabalhador a ela destacado.

O processamento deste cálculo efetua-se segundo as seguintes fases:

- Iniciar loop com **T** iterações, sendo **T**, o número de tarefas que constituem o projeto;
- Para cada iteração, percorrer **W** indices do cromossoma, sendo **W** o número de trabalhadores criados para o projeto;
- Para cada indice percorrido, caso o bit esteja assinalado com o valor "1", consultar os dados do trabalhador que lhe é respetivo, determinando se este possui alguma competência para influenciar o desenvolvimento da tarefa, acumulando a respetiva performance à performance total dedicada por todos os trabalhadores destacados à tarefa;
- Após reunir a informação relativa à performance total, determinar o tempo de inicio de execução da tarefa. Sendo que o tempo de início de execução de uma tarefa corresponde ao momento de finalização de todas as suas precedências, e tendo em conta que as tarefas são analisadas por ordem cronológica (não existe nenhuma situação em que uma dada tarefa seja processada antes das suas precedências), conseguimos obter facilmente este valor;
- Calcular o tempo de execução, sendo este determinado pela fórmula enunciada na página 4, ao clarificar a definição de duração de uma tarefa;
- Obtendo os resultados dos últimos dois pontos, é-nos possivel determinar o momento de finalização da tarefa a ser analizada na iteração atual;
- Finalmente, deve-se atualizar os dados dos trabalhadores relativos à disponibilidade de cada um e armazenando o tempo de finalização desta tarefa.

Após execução das instruções acima enunciadas, obtém-se uma lista de resultados relativa ao momento de finalização de cada tarefa. O valor de *fitness* que pretendemos obter consiste no máximo presente nesta lista.

Sendo que o objetivo do problema é determinar a melhor forma para diminuir o tempo de execução de um projeto, o cromossoma mais Apto será aquele que apresentar menor *fitness*.

1.2 Arrefecimento Simulado

O Arrefecimento Simulado (*Simulated Annealing*) consiste numa meta-heurística para otimização de problemas onde recorre a técnicas de busca local probabilística//aleatória, fundamentando-se numa analogia relacionada com termodinâmica. O processo denominado de *annealing* é utilizado na metalurgia e consiste no aumento de um metal, seguido de um arrefecimento lento e controlado, de modo a atingir-se um estado energético baixo o suficiente para garantir uma menor energia interna, reduzindo possiveis defeitos do resultado.

Na nossa abordagem ao problema, decidimos optar por uma adaptação dos calculos executados nos algoritmos genéticos, modificando-os de forma a que se enquadrassem no processamento deste algoritmo em específico. Mais concretamente, o codigo reutilizado consiste no cálculo do valor do estado atual: recorrendo ao mesmo processo de análise do tempo de execução de um projeto no Algoritmo Genético, executamos as mesmas operações. Contudo, existem pequenas alterações relativamente à penalização. Uma vez que pretendemos sempre avançar num sentido que favoreça, de modo individual, um determinado estado, convém nao generalizar os casos em que um projeto não pode ser concluido. Sendo assim, para cada tarefa que não possa ser concluida, por não ter trabalhadores aptos para a executar, acrescenta-se a penalização de dez meses acrescidos à duraçao original do projeto. Esta análise de cada estado torna-se simples na medida em que a representação do conhecimento é igual à representação dos cromossomas, nos Algoritmos Genéticos.

O algoritmo depende de duas variavéis: a temperatura e o rácio de diminuição da temperatura. Ambos os valores poderão ser especificados pelo utilizador, de modo a ser possivel retirar conclusões mais claras relativamente à eficácia do algoritmo. De acordo com a teoria, quanto maior a temperatura e/ou menor o rácio de diminuição, melhores serão os resultados obtidos.

Enquanto a Temperatura **T** não for igual a 0, o algoritmo encontrar-se-à num loop constante, onde T variará no sentido decrescente, dependendo do rácio estabelecido. A cada iteração, um novo estado **N**, diferente do atual **A**, é gerado e avaliado. É, de seguida, calculada a diferença entre os dois estados, tendo que $\triangle \mathrm{E} = \mathrm{N} - \mathrm{A}$. Tendo este cálculo determinado, definimos que:

$$\begin{cases} \mathrm{A} = \mathrm{N} & \text{if } \triangle \mathrm{E} > 0 \\ \begin{cases} \mathrm{A} = \mathrm{A} & \text{if Math.random()} > \mathrm{e}^{\frac{\triangle \mathrm{E}}{T}} \\ \mathrm{A} = \mathrm{N} & \text{if Math.random()} < \mathrm{e}^{\frac{\triangle \mathrm{E}}{T}} \end{cases} \end{cases}$$

Tendo em conta o cálculo de $\triangle$E, não podemos quantificar cada estado como nos Algoritmos Genéticos, pelo que, para definir que um estado é melhor que o outro quando o seu valor é superior, decidimos pela atribuição do valor equivalente a $(\frac{\text{projectDuration}}{\text{fitness}})^2$, pelo que *fitnesses* mais elevados, gerarão valores mais baixos, e vice-versa

2 Desenvolvimento

Para a implementação deste projeto, recorreremos à linguagem Java que, para além de nos facultar a documentação e estruturas de dados mais importantes para a conceção do código, também suporta, de forma consistente e relativamente simples, a criação de uma interface gráfica importante para a comunicação com o utilizador, de modo a que este possa manipular as variáveis.

A nível de Ferramentas/APIs utilizadas neste projeto, são de referir o Eclipse (usado na elaboração do código), Github (usado um repositório para partilha de código) e TexStudio (usado na elaboração do relatório).

Relativamente à GUI, recorremos à biblioteca Swing de Java para estruturar uma interação muito simplificada para receber os inputs pretendidos pelo utilizador.

A seguir demonstra-se o diagrama de classes do referido projeto.

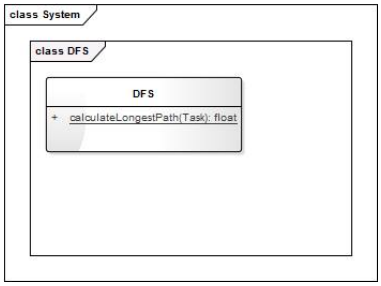

Figura 3: Pacote auxiliar para calculo da duraçao original do projeto

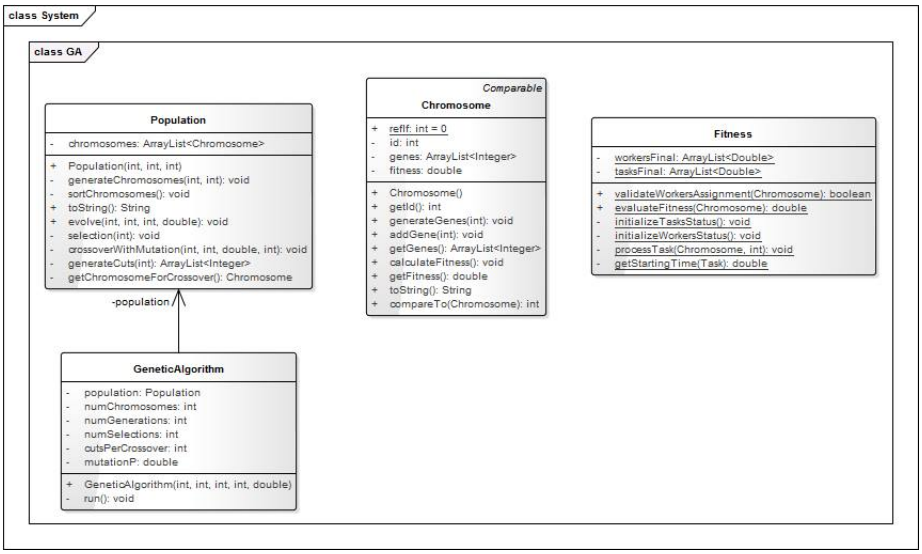

Figura 4: Pacote de Algoritmos Geneticos

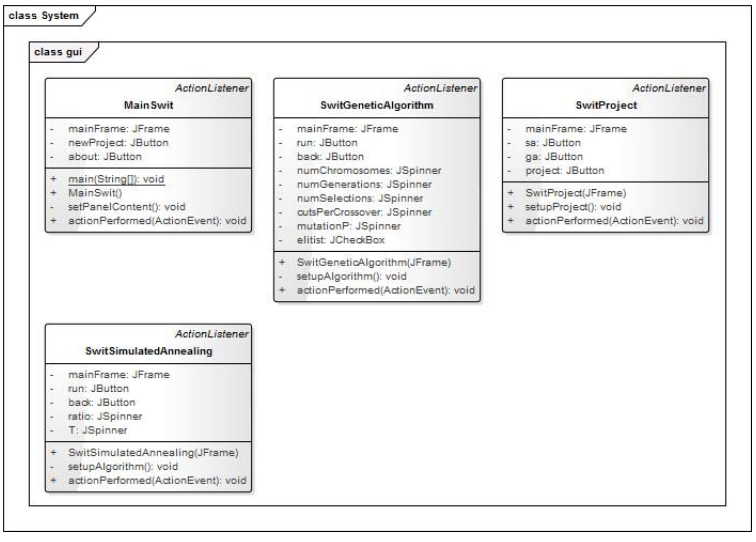

Figura 5: Pacote respetivo à interface Gráfica

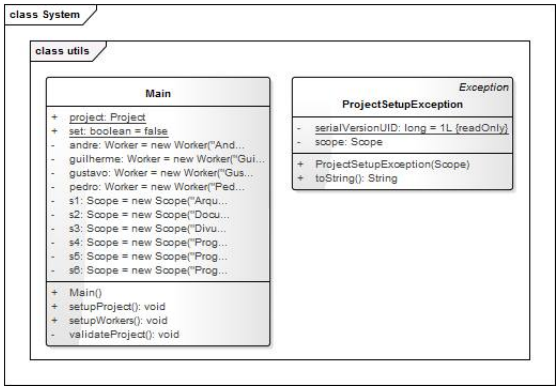

Figura 6: Pacote responsável pela criação dos projetos

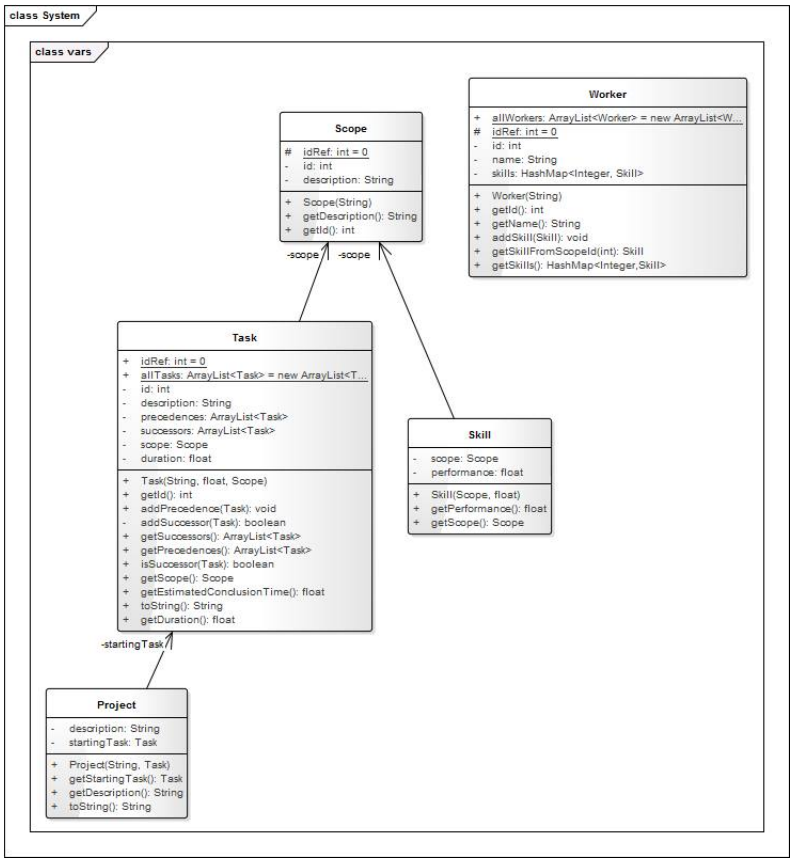

Figura 7: Pacote com as variaveis de estudo

3 Experiências

3.1 Algoritmos Genéticos

De modo a padronizar os resultados, avalariaremos o desempenho dos Algoritmos Genéticos com os seguintes parametros-base:

- **Numero de cromossomas:** 50
- **Numero de geraçoes:** 100
- **Probabilidade de Mutação:** 0.01
- **Cortes por Crossover:** 2
- **Numero de Seleções:** 25

Resultados	
1º Ensaio	6.889394
2º Ensaio	6.283333
3º Ensaio	6.392424
Média	6.521717

3.1.1 Redução do numero de Cromossomas

- **Numero de cromossomas: 25**
- **Numero de geraçoes:** 100
- **Probabilidade de Mutação:** 0.01
- **Cortes por Crossover:** 2
- **Numero de Seleções:** 12 (alteração proporcional)

Resultados	
1º Ensaio	6.586363
2º Ensaio	6.675252
3º Ensaio	7.192424
Média	6.818013

3.1.2 Redução do numero de Gerações

- **Numero de cromossomas:** 50
- **Numero de geraçoes: 50**
- **Probabilidade de Mutação:** 0.01
- **Cortes por Crossover:** 2
- **Numero de Seleções:** 25

Resultados	
1º Ensaio	7.192424
2º Ensaio	7.067965
3º Ensaio	7.192424
Média	7.150938

3.1.3 Aumento da probabilidade de Mutação

- **Numero de cromossomas:** 50
- **Numero de geraçoes:** 100
- **Probabilidade de Mutação: 0.1**
- **Cortes por Crossover:** 2
- **Numero de Seleções:** 25

Resultados	
1º Ensaio	6.889394
2º Ensaio	6.089394
3º Ensaio	6.889394
Média	6.622727

3.1.4 Aumento dos cortes por Crossover

- **Numero de cromossomas:** 50
- **Numero de geraçoes:** 100
- **Probabilidade de Mutação:** 0.01
- **Cortes por Crossover: 4**
- **Numero de Seleções:** 25

Resultados	
1º Ensaio	7.061616
2º Ensaio	6.372222
3º Ensaio	6.978283
Média	6.804040

3.1.5 Diminuição das seleções por geração

- **Numero de cromossomas:** 50
- **Numero de geraçoes:** 100
- **Probabilidade de Mutação:** 0.01
- **Cortes por Crossover:** 2
- **Numero de Seleções: 0**

Resultados	
1º Ensaio	9.676282
2º Ensaio	10.202076
3º Ensaio	10.288235
Média	10.062197

3.2 Arrefecimento Simulado

De modo a padronizar os resultados, avalariaremos o desempenho do Arrefecimento Simulado com os seguintes parametros-base:

- **Temperatura:** 5000
- **Rácio de arrefecimento:** 0.9

Resultados	
1º Ensaio	7.874739
2º Ensaio	8.292819
3º Ensaio	8.358585
Média	8.175381

3.2.1 Aumento da Temperatura

- **Temperatura: 10000**
- **Rácio de arrefecimento:** 0.9

Resultados	
1º Ensaio	8.058585
2º Ensaio	7.831871
3º Ensaio	8.281203
Média	8.057220

3.2.2 Diminuição do Rácio de arrefecimento

- **Temperatura:** 5000
- **Rácio de arrefecimento: 0.99**

Resultados	
1º Ensaio	7.412679
2º Ensaio	7.864551
3º Ensaio	7.443122
Média	7.573450

4 Conclusões

Relativamente ao trabalho desenvolvido, consideramos que a aplicação de algoritmos evolutivos ao planeamento de projetos pode-se tornar bastante eficiente para obtenção de uma solução eficaz.

Pelos testes realizados, podemos ver que é possível manipular o tipo de resultados que se pretendem, quer nos algoritmos genéticos (aumento do numero de cromossomas, de geraçoes ou seleções, por exemplo), quer no algoritmo de arrefecimento simulado (aumento da temperatura e diminuição do racio de arrefecimento), para obtenção de soluções otimizadas.

Realçamos ainda que, embora tenhamos testado a variação de todas variaveis possiveis, os resultados obtidos não são muito constrastantes, devido ao exemplo de projeto relativamente simples que estruturamos, para teste.

Consideramos este, um ótimo e interessante projeto, que permitiu a aplicação de conhecimentos de outras (biologia e metalurgia) à conceção de algoritmos consideravelmente eficazes para problemas como este.

5 Melhoramentos

Como melhorias ao projeto, focar-nos-iamos à criação de uma interface de utilizador mais completa, que permitisse a criação de um projeto e trabalhadores e capaz de fornecer estatísticas completas de forma clara e conclusiva.