

ALGAV

2019-2020

RELATÓRIO

SPRINT 2

Carlos Moreira – 1161882

Marco Pinheiro – 1170483

Pedro Barbosa – 1150486

Pedro Mendes – 1161871

Índice

[Introdução 3](#_Toc26621507)

[Extração das tarefas a usar no Algoritmo Genético (AG) a partir dos dados do sistema produtivo 4](#_Toc26621508)

[Criação da população inicial do Algoritmo Genético (AG) 8](#_Toc26621509)

[Aleatoriedade no cruzamento entre indivíduos da população 11](#_Toc26621510)

[Seleção da nova geração da população 11](#_Toc26621511)

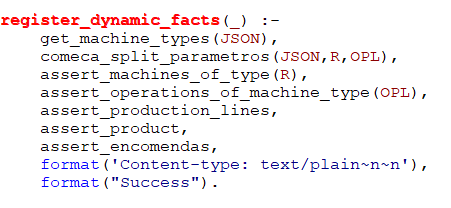
[Análise de eficácia 12](#_Toc26621512)

[Conclusão 13](#_Toc26621513)

# Introdução

Este relatório surge no âmbito da disciplina de Algoritmia Avançada (ALGAV), do terceiro ano da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia do Porto. Este sprint tem como objetivo a concretização de conhecimentos da disciplina, nomeadamente a aplicação de algoritmos genéticos de forma a gerar um sequenciamento de tarefas otimizado para o somatório dos tempos de atraso. Para este efeito, criamos ainda um cliente http de forma a obter os dados (GET) da factoryApi, productionApi e ordersApi.

# Extração das tarefas a usar no Algoritmo Genético (AG) a partir dos dados do sistema produtivo



O predicado **regista\_dynamic\_facts** tem como função invocar todos os predicados que criam fatos a partir de dados extraídos das Apis.

Uma imagem com captura de ecrã

Descrição gerada automaticamenteUma imagem com captura de ecrã

Descrição gerada automaticamente

Os predicados apresentados acima, realizam chamadas http ás respetivas Apis, de modo a extrair os dados necessários.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Os predicados apresentados acima retiram das respostas Json a informação necessária à criação dos factos. Através destes predicados são criados os seguintes factos:

type\_of\_machine/2, que faz a ligação entre uma maquina e o seu tipo.

operationtype/3, que define o tempo de setup e o tempo de execução de um tipo de operação.

operation/2, que faz a ligação entre uma ferramenta e um tipo de operação.

machine\_type\_operation/3, que faz a ligação entre um tipo de maquina e uma operação.

machines\_of\_line/2, que faz a ligação entre uma linha de produção e uma lista de maquinas.

product\_plan/2, que define uma lista de operações como plano de produção de um produto.

order/4, que representa a informação da encomenda, nomeadamente o cliente, lista de produtos a ser encomendados, data de entrega e a prioridade do cliente.

Uma imagem com captura de ecrã

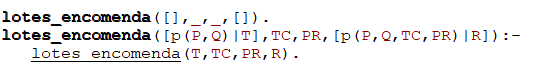
Descrição gerada automaticamente

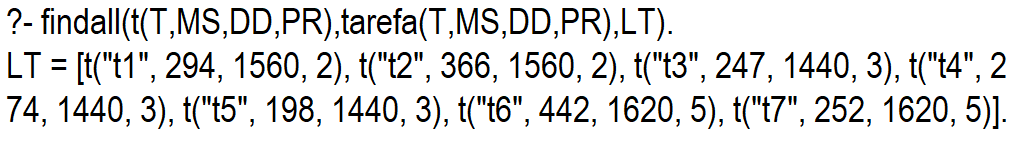
Foi também criado o predicado register\_dynamic\_facts\_offline que permita a criação de factos necessários ao algoritmo genérico sem recorrer a ligação com a base de dados.

# Criação da população inicial do Algoritmo Genético (AG)

Uma imagem com captura de ecrã

Descrição gerada automaticamente





Os predicados apresentados acima, criam factos do tipo tarefa/4, que incluem um id da tarefa, o makespan da tarefa, a data de entrega e a prioridade do cliente. É criada uma tarefa por cada lote de produtos de uma encomenda. O makespan é calculado através da formula Q\*M + Σm + Ts, onde Q representa a quantidade, M é o tempo de execução da operação mais longa, m é o tempo de execução das restantes operações e Ts é o tempo de setup que alcança maior amplitude no começo do lote.

Uma imagem com objeto

Descrição gerada automaticamenteUma imagem com captura de ecrã

Descrição gerada automaticamente

Para a criação da população inicial, usamos o predicado gera\_população para o qual utilizamos 3 heuristicas: SPT (ordenação por makespan), EDD (ordenação por data de entrega), e PCO (ordenação por prioridade do cliente). No caso de serem gerados dois indivíduos iguais usamos o predicado gera\_radom para gerar um individuo diferente e aleatório.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

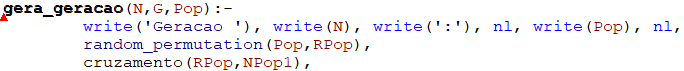
Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

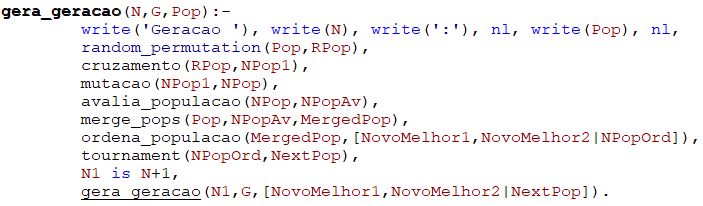
As heurísticas acima referidas utilizam um algoritmo “insert sort” para ordenar as listas conforme os parâmetros definidos.

# Aleatoriedade no cruzamento entre indivíduos da população



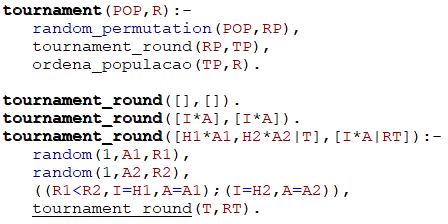
De modo a garantir aleatoriedade no cruzamento entre indivíduos da população, é utilizado o predicado nativo random\_permutation/2, que gera uma permutação aleatória da população.

# Seleção da nova geração da população





De forma a selecionar a população da próxima geração, é feita uma junção da geração corrente com os seus descendentes obtidos por cruzamento e mutação, sendo esta população conjunta posteriormente ordenada. Os dois melhores indivíduos são garantidos de passar à próxima geração e a restante população é submetida a um torneio de forma a serem escolhidos os restantes indivíduos da geração seguinte.



Um torneio consiste na geração de um valor aleatório entre 1 e a avaliação de dois indivíduos. Estes valores aleatórios são comparados e o indivíduo cuja avaliação tenha gerado um menor valor, passa para a população resultado.

# Análise de eficácia

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nº de Tarefas | Valor Real | Melhor valor do AG Base | Melhor valor do AG Final | Valor Médio do AG Base | Valor Médio do AG Final |
| 6 | 238.5 | 1017 | 238.5 | 1030 | 259.8 |
| 7 | 696 | 1464 | 696 | 2390.6 | 827.6 |
| 8 | 1224.0 | 2887 | 1232.1 | 3729 | 1502.8 |
| 9 | 3197 | 4269 | 3209.4 | 4746 | 3883.8 |
| 10 | 3836 | 5796.9 | 3836 | 6534.8 | 4944.8 |
| 11 | 5067.2 | 8240 | 5147.3 | 8497 | 6717.9 |
| 12 | 5780.1 | 7682 | 5956.5 | 8647.6 | 8375.5 |

# Conclusão

Este trabalho foi importante na consolidação de conhecimentos respeitantes a programação lógica, usando o Prolog como linguagem de programação.

Relativamente a utilização do algoritmo genético concluímos que os resultados obtidos são melhores que o método de pesquisa utilizado no sprint anterior (A\*).

Quanto à comparação com o algoritmo genético base verificamos que as alterações por nós efetuadas tornou-o mais eficaz , visto que quando aplicada a passagem de parte da população corrente para geração seguinte garantimos que as melhores soluções não são descartadas, aumentando assim a probabilidade de encontrar a melhor solução possível na geração final.