

Relatório ALGAV

**Grupo 03**

José Mota (1161263)

Pedro Real (1170689)

João Flores (1171409)

Patrick Timas (1171352)

1. Introdução incluindo identificação dos objetivos atingidos no sprint2.

Para esta iteração, foi-nos lançada a proposta de obter uma order de modo a realizar varia/s tarefas de modo a produzir lotes de produtos, tendo como critérios a realização do lote o mais rapido. Assim sendo, para este trabalho tivemos que criar uma heurística e modificar outra. Foi criado um para avaliar e obter a melhor solução. Por fim, foi pedido para criarmos o predicado Algoritmo genetico, a avaliação de quando deve terminar o predicado era opcional escolhemos por tempo e iterações.

2.

Uma imagem contendo captura de tela

Descrição gerada automaticamenteUma imagem contendo captura de tela

Descrição gerada automaticamenteUma imagem contendo interior, pessoa

Descrição gerada automaticamente

A usar no Algoritmo Genético (AG) a partir dos dados do sistema produtivo para a resolução deste sprint, era fundamental a criação das tarefas associadas a cada Produto. Ou seja cada tarefa corresponde a um lote de um produto de uma encomenda. Como sabemos, uma encomenda pode gerar várias tarefas. Cada tarefa é constituída por um idTarefa, o MakeSpan, o tempo de conclusão e a prioridade do cliente. Para a prioridade, o nosso grupo considerou que 1 correspondia a maior importância e o 5 a menor importância. Sabemos que o makespan corresponde ao tempo que um lote demora a ser produzido. Assim sendo, começamos por obter todos os produtos, através da obtenção de todas as encomendas conseguimos obter os produtos associados às mesmas. Assim tendo os produtos todos conseguimos criar as tarefas. como dito anteriormente, a tarefa tem um id, logo criamos um contador que começa e será incrementado até atingir o número de produtos. Ao criarmos as tarefas necessitamos das operações associadas ao fabrico do produto. Deste modo, obtemos uma lista de pares Operacao-tempo execução ordenada pela operação que tem o maior tempo de execução. a seguir calculamos o makespan onde Makespan corresponde ao TempoLote + SomaTempoExecaoRestante. porem o MakespanComTempoPreparacao corresponde ao Makespan - o tempo de preparação.

3.Algoritmo Genetico

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Neste algoritmo começamos por permutar a lista inicial para de modo a reordenar a lista, de seguida são realizados os cruzamentos que ira originar um lista nova, junta-se as duas gerações a antiga e a que foi gerada a partir dos cruzamentos, de seguida extrai os dois melhores resultados de guarda numa lista que posteriormente irá ser instroduzida na lista de melhores casos para a proxima iteração, neste momento não existem repetidos porque à nova lista foram removidos os repetidos e a lista da geração anterior não existiam repetidos.

Conforme os dados que contem, vai realizar os dados que tem na lista de geração sem os dois maiores.

Caso a lista contenha 6 elementos não é necessario realizar torneios, são todos escolhidos automaticamente, caso tenha menos que 12 faz torneio 2 a 2, os que perdem vão para uma lista auxiliar e são posteriormente verifica novamente, repete o processo até ter 6 elementos, caso tenha 12 faz o torneio a pares, 2 a 2 cada um que ganhe é adicionado a lista que posteriormente é adicionada aos 2 melhores.

A avaliação dos pares no torneio é conforme o seu valor é gerado um valor random de 0 até à sua avaliação e o que tiver menor valor é o retornado.

3) Criação da população inicial do Algoritmo Genético (AG) ACRESCENTAR Ambas as heurísticas ordenam num mapa de pairs, as tarefas, tarefas estas obtidas através de um findAll. E os respetivos valores da soma pesada dos tempos com a prioridade. Foi utilizado um EDD (menorAtrasoEncomendaComPrioridades\_SemShortestpath) e a condicionante foi o tempo de execução máximo dado pelo cliente aquando da encomenda e a sua prioridade de 1.0 a 1.4(simplificação dos valores de 1 a 5). Foi utilizado um SPT menorAtrasoEncomendaComPrioridades\_Shortestpath) e a condicionante foi o makespan das tarefas e a sua prioridade de 1.0 a 1.4(simplificação dos valores de 1 a 5). Para a prioridade, como referimos anteriormente,1 corresponde a prioridade máxima e 5 a mínima. Em ambos os casos, ordenando este mapa de pares<Tarefa,Soma Pesada> de forma crescente, conseguimos implementar as suas heurísticas correspondentes e o calculo do tempo dessas heurísticas com base no tempo de atraso das encomendas.

4.Aleatoriedade no cruzamento entre indivíduos da população

Usamos o random\_permutation(A,B) onde A é uma lista inicial e B corresponde a uma lista permutade de modo random, garantindo assim resultados mais aleatorios e promovendo melhores resultados.

Quando é necessário realizar os torneios também é realizada a permutação da lista .

5.Seleção da nova geração da população

A seleção é realizada atraves das heuristicas implementadas.

6. Analise de eficácia

Não foi possivel realizar os gráficos e tabelas necessarios, por isso decidimos realizar uma analise mais teórica.

A eficácia do AG depende do tipo de parametro e da dimensão, existem caso onde o tempo será o ideal para ter a melhor análise noutros o número de iterações basta para obter um resultado aceitavel, o ideal será uma parameterização onde procuramos o resultado ideal e só termina quando o resultado for alcançado ou ocurra algo que termine a sua execução.

É necessário realçar que a utlização do find all não tera muitos problemas quando a amostra é pequena, mas quando escalamos para algo mais o AG tira uma eficácia muito melhor.

7.Parametrização da condição de terminio do AG

O ultimo parametro no AG apresentado a cima é por iterações corre “n” vezes e retorna o valor obtido, na parametrização uma implementação possível era terminal apôs um intervalo de tempo desde o inicio do AG até ao momento atual.Uma imagem contendo captura de tela

Descrição gerada automaticamente

Neste parte do codigo valida se o tempo que passou é igual ou superior ao tempo que é aceite, o “tempoCriterio” é recebido por parametro do metido que chame do gera\_geracao/5

8. Conclusões relativas ao Sprint 2

Nesta iteração, concluímos que as heuristicas continua a apresentar os melhores resultados tendo um consideração os tempos e recursos gastos. Se pretendermos obter a solução através do cálculo de todas as soluções possíveis, então é mais adequado será o uso do predicado sem findall, dado ser mais eficiente em termos de tempo que com findall e que suporte um maior número de operações.

Em suma, o trabalho foi realizado todo com sucesso, apesar de varias dificuldades na implementação dos algoritmos, devido a problemas externos que nos foram aparecendo, mas com algum esforço e múltiplas tentativas, conseguimos executar as tarefas propostas.