Escola de Engenharia

Departamento de Produção e Sistemas

Mestrado em Engenharia de Sistemas

Universidade do Minho

Relatório
Escalonamento da Produção
Gestão da Produção

Discentes:
Ana Antunes PG35007
Cátia Pinto PG35004
Ivo Ferreira PG35375
Joana Dias PG35383

Docente:
Maria Leonilde Rocha Varela

Braga, janeiro de 2018

Resumo

O presente relatório tem como objetivo resolver um problema de escalonamento da produção usando três métodos ou abordagens heurísticas, considerando-o numa primeira fase em ambiente de linha flexível e noutra fase em ambiente de oficina flexível, devendo ser apresentada para cada fase os resultados através de gráficos para várias medidas de desempenho. Além disso, é objetivo do relatório fazer um resumo acerca de métodos/abordagens de escalonamento da produção que se conheça e que se possa sugerir como alternativos à resolução do problema.

Para isso, foi usado o sistema *Lekin*, onde se selecionou a heurística *General SB Routine* para várias funções objetivo e as regras SPT e CR, em ambiente de linha flexível e as regras LPT e SPT, em ambiente de oficina flexível.

Dos resultados obtidos, entre outras conclusões concluiu-se que o melhor método para resolver um problema de escalonamento da produção irá depender do objetivo que se tem, ou seja, da medida de desempenho que se pretende minimizar.

No entanto, neste problema obteve-se que em todos os escalonamentos os 9 lotes apresentam atraso. Além disso, concluiu-se que o ambiente em oficina flexível permitiu obter melhores escalonamentos.

Também se concluiu que existe uma grande variedade de métodos de escalonamento da produção, dependendo o método escolhido para a resolução de um problema de escalonamento da produção das características do problema, da capacidade da máquina que o processa e do tempo disponível de espera para encontrar uma solução. Como alternativa aos métodos usados para este problema podiam ter sido usados o algoritmo de *Hodgson e o* algoritmo de *Jonhson* aplicável a 3 máquinas para o problema em ambiente de linha flexível, técnicas de *Branch and Bound*, desde que o tempo de execução o permita, técnicas de procura local e outras regras de prioridade como a MWKR.

Índice

1. Introdução

O presente relatório insere-se no âmbito da unidade curricular Gestão da Produção e tem como objetivo resolver um problema de escalonamento da produção usando três métodos ou abordagens heuríticas, considerando-o numa primeira fase (parte A) em ambiente de linha flexível (*Flexible Flow Shop*, FFS) e numa segunda fase (parte B) em ambiente de oficina flexível (*Flexible Job Shop*, FJS). É também pretendido que se apresentem para cada uma das partes os resultados graficamente para as medidas de desempenho tempo total de produção ou *makespan*, tempo de percurso médio ou *mean flow time*, atraso máximo, atraso médio e número de lotes atrasados. Por fim (parte C), pretende-se que se faça um resumo acerca de métodos/abordagens de escalonamento da produção que se conheça e que se possa sugerir como alternativos à resolução do problema.

Numa linha de produção flexível, todos os trabalhos seguem a mesma sequência operatória em relação às máquinas existentes no sistema de produção. Assim, caso as máquinas sejam dispostas no espaço segundo essa sequência não há fluxos inversos ou regressivos dos trabalhos. Além disso, existem duas ou mais máquinas semelhantes para o processamento de cada operação de um trabalho. Este tipo de configuração de sistema de produção está representado na Figura 1 [1].

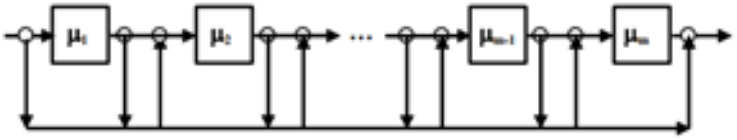

Figura 1 – Linha de Produção Geral – Flexível – (*General flexible flow-shop*).

Legenda:
μ: Conjunto de máquinas /agregado de meios que executam uma dada função
m: Máquinas
➡ Sentidos do fluxo de materiais autorizado no sistema

Já numa oficina de produção flexível, os trabalhos também têm duas ou mais operações com uma sequência de execução expressamente definida, que pode ser diferente de trabalho para trabalho em relação às máquinas existentes no sistema. Consequentemente, quando as máquinas estão dispostas no espaço numa dada sequência é de prever a existência de fluxos inversos ou regressivos dos trabalhos no sistema. Além disto, existem também, duas ou mais máquinas semelhantes para o processamento de cada operação de um trabalho. Este tipo de configuração de sistema de produção encontra-se representado na Figura 2 [1].

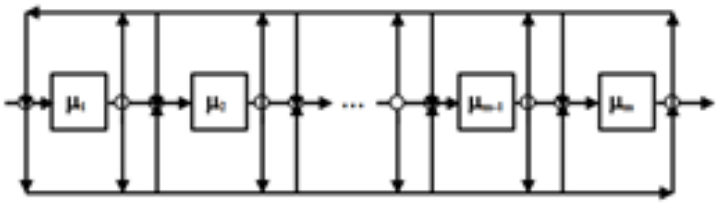

Figura 2 – Oficina de Produção – Flexível (*Flexible job-shop*).

Legenda:
μ: Conjunto de máquinas/ agregado de meios que executam uma dada função
m: Máquinas
➡ Sentidos do fluxo de materiais autorizado no sistema

O escalonamento da produção, também conhecido como programação da produção, tem como objetivo o bom funcionamento de um sistema produtivo, em que se obtenha a quantidade e qualidade expectável de bens a partir de dados recursos [1]. Envolve afetação de trabalhos a recursos de produção, sequenciamento de trabalhos objetos aos vários recursos de produção e, ainda calendarização ou programação detalhada das operações de cada trabalho em cada recurso [9].

Para isso, é necessário conhecer os trabalhos ou tarefas, os processadores, os meios auxiliares, os ambientes de escalonamento e de produção e os critérios de otimização, bem como os elementos que se relacionam e contribuem para uma boa execução de um dado trabalho [1].

Neste caso, o problema consiste em processar 9 lotes de trabalho em duas operações, operação 1 e operação 2, dispondo de 3 máquinas para a realização de cada uma das operações dos lotes: um grupo de máquinas A (A_1, A_2 e A_3) e um grupo de máquinas B (B_1, B_2 e B_3). No ambiente de produção na configuração de oficina flexível a partir do lote 5, inclusive, os lotes realizam as operações por ordem inversa do fluxo normal de produção verificado na configuração de linha flexível, ou seja, realizam primeiro a sua primeira operação nas máquinas do grupo B e a sua segunda operação no grupo de máquinas A, respetivamente.

Os tempos de processamento em cada uma das operações, p_{j1} e p_{j2}, respetivamente, assim como os prazos de conclusão (d_j) para cada lote (j) encontram-se na Tabela 1.

Tabela 1 – Tempos de processamento em cada uma das operações, p_{j1} e p_{j2}, respetivamente, e prazos de conclusão (d_j) para cada um dos 9 lotes (j)

Lotes j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
p_{j1}	25	32	19	18	26	37	22	22	25
p_{j2}	20	31	28	22	21	19	28	18	27
d_j	20	15	30	35	25	18	40	37	45

Note-se que, os valores de p_{j1} e p_{j2} da Tabela 1 foram obtidos para d_1 igual a 3 e d_2 igual a 8, correspondendo estes, respetivamente, ao último algarismo do número mecanográfico maior do grupo e ao penúltimo algarismo do mesmo, que corresponde a 35383.

Para resolver um problema de escalonamento da produção, vários métodos ou regras podem ser aplicados consoante o objetivo. Sendo que, estes podem ser aplicados manualmente ou através de softwares de escalonamento da produção, como o *Lisa* e o sistema *Lekin* [9].

O sistema *Lekin* foi desenvolvido pela universidade *Stern School of Business*, tendo sido as principais partes criadas e desenvolvidas por estudantes da universidade de *Columbia*. Os diretores do projeto foram os professores Michael L. Pinedo, Xiuli Chao e o professor Joseph Leung. Este programa foi criado como uma ferramenta educacional com o objetivo principal de apresentar aos alunos a teoria de escalonamento e as suas aplicações práticas [4].

No sentido de fazer o escalonamento, o *Lekin* possui um conjunto de oito regras de escalonamento de produção e de heurísticas, oferecendo ainda a possibilidade de fazer uma entrada manual.

Relativamente às regras, este sistema possui ATCS, EDD, MS, LPT, SPT, WSPT, FCFS e CR. Sendo que, as regras EDD e a MS dependem das datas de entrega e lançamento, as regras LPT, SPT e WSPT dependem dos tempos de processamento, enquanto a ATCS e a CR combinam regras [8].

Concretamente, a regra EDD (*Earlist Due Date*) pretende minimizar o atraso das entidades existentes em fila de espera dando prioridade ao trabalho com a menor data de entrega. Para isso, sempre que uma máquina fica desimpedida seleciona a entidade que tem a data de entrega mais cedo [6]. A regra MS (*Minimum slack*) é uma variação da regra EDD, em que quando uma máquina fica desimpedida, de todas as entidades em fila de espera é escolhida para processamento a entidade que tem menor folga ou maior urgência (dj – pj). Ou seja, o tempo de folga até à data de entrega (d_j) é encontrado, retirando à data de entrega o tempo necessário para processamento (p_j) [6] [8].

Já na regra LPT (*Longest Processing Time*), quando uma máquina fica livre é selecionada a entidade com maior tempo de processamento nessa máquina para entrar em produção. Contrariamente, na regra SPT (*Shortest Processing Time*), de todas as entidades existentes entra em processamento a entidade com menor tempo de processamento nessa máquina. Já na regra WSPT (*Weighted Shortest Processing Time*), após uma máquina estar livre é escolhida de todas as entidades para entrar em produção a tarefa que tem a menor razão entre o tempo de processamento e a prioridade existente nessa máquina e para essa tarefa [2][8].

A regra ATCS (*Apparent Tardiness Cost* (com *setups*)) combina as regras de prioridade WSPT e MS, tendo como objetivo reduzir o atraso total pesado do sistema. As tarefas escalonadas com esta regra são escolhidas de acordo com um índice, calculado sempre que a máquina está livre, em que a tarefa que apresentar o maior valor é a próxima a ser executada. Já a regra CR (*Critical Ratio Rule*) é uma combinação entre a regra EDD e LPT, que quando uma máquina fica livre, seleciona de todas as entidades aquela que tem menor razão crítica, ou seja, menor razão entre tempo disponível e tempo de processamento [2][8].

Por último, na regra FCFS (*First Come First Serve*) após uma máquina estar livre é dada prioridade à entidade que chega em primeiro ao sistema, garantindo que nenhuma entidade fica em fila de espera por tempo indeterminado [8].

Em relação às heurísticas, o *Lekin* possui a *General SB Routine*, o algoritmo *Shifting bottleneck sum(Wt) e Tmax, o* algoritmo *SB-LS* e a meta-heurística em pesquisa local *(Local Search).*

A *General SB Routine* é uma heurística que sequencia as máquinas uma a uma sucessivamente, re-otimizando localmente todas as sequências previamente estabelecidas sempre que uma nova máquina é sequenciada [12].

O algoritmo *Shifting bottleneck* é utilizado nas oficinas de produção para reduzir o "*makespan*" de um sistema. Sendo que, em cada iteração um problema de máquina única é resolvido e tem como objetivo tratar todas as máquinas que fazem parte do sistema [5] [12].

A meta-heurística em pesquisa local funciona apenas para um ambiente comum em que parte de uma solução inicial e pesquisa uma única solução através de um processo iterativo com o objetivo de melhorar o proposto para o plano de escalonamento. Neste caso, consegue-se escolher o período de tempo que o algoritmo pode executar ou até mesmo parar o algoritmo a qualquer momento [12].

Por último, o *algoritmo SB-LS* é uma combinação da heurística *Shifting bottleneck* com a heurística de pesquisa local, especificamente desenhado para as linhas de produção

flexíveis. Este consegue superar outros algoritmos quando o número de trabalhos é maior que o número de máquinas [12].

Em suma, cada método tem uma finalidade diferente, pelo que é necessário trabalhar com vários e analisar todos os resultados, de modo a optar pela solução mais adequada e ajustada a cada problema. Encontrando a solução mais ajustada gera-se mais valor para todos os *stakeholders* da organização.

2. Resolução

2.1. Parte A – Problema em ambiente de linha flexível

Nesta parte pretende-se resolver o problema de escalonamento da produção considerando-o em ambiente de linha flexível (*Flexible Flow Shop*, FFS), usando três métodos ou abordagens heurísticas. E também, apresentar os resultados graficamente para as medidas de desempenho tempo total de produção ou *makespan*, tempo de percurso médio ou *mean flow time*, atraso máximo, atraso médio e número de lotes atrasados.

Para isso, foi usado o sistema *Lekin*.

Inicialmente, começou-se por introduzir os dados no sistema. Para isso, começou-se por abrir o *software* e selecionar no menu principal a opção *Flexible Flow Shop*. Depois, colocou-se no número de centros de trabalho e de tarefas (*jobs*) a serem executadas, respetivamente, 2 e 9, correspondendo estes ao número de operações e de lotes de trabalho do problema, respetivamente. Posteriormente, caraterizou-se cada centro de trabalho, colocando como nome operacao1, no primeiro centro de trabalho e, depois operacao2 no segundo centro de trabalho, o número de máquinas como 3, a data de disponibilidade (*Availability date*) como 0 e estado inicial (*starting status*) como A no primeiro centro de trabalho e B no segundo centro de trabalho. Por fim, inseriram-se os dados das tarefas (*jobs*) a serem executadas, ou seja, os dados de cada lote de trabalho. Sendo que, para cada lote definiu-se o nome como lote1, lote 2 e assim sucessivamente, a data de lançamento (*release date*), ou seja, a data a partir da qual começa a ser feito como 0, a data de vencimento (*due date*), isto é, a data em que se quer entregar com o valor do prazo de conclusão (d_j) respetivo ao lote e, ainda o peso como 1. Além disso, editou-se a rota, colocando para cada centro de trabalho (operação 1 e operação 2), o tempo de processamento (*Pr.Tm*) como o valor do tempo de processamento do lote (p_{j1} e p_{j2}) e o estado (*status*) como A, na operação1 e B na operação 2. Note-se que, o parâmetro tempo de processamento (*processing time*) é preenchido automaticamente, correspondendo à soma de p_{j1} e p_{j2}.

Após a introdução dos dados, procedeu-se ao escalonamento. Para isso, na barra de ferramentas superior em *Schedule* selecionou-se a opção *Heuristic*. Sendo que, apesar de existirem as 5 heurísticas referidas na introdução, o sistema apenas permite selecionar duas, a *General SB Routine* e o algoritmo *SB-LS.*

Nesse sentido, começou-se por selecionar o algoritmo SB-LS. Sendo que, apesar de várias tentativas, onde se selecionaram-se vários segundos clicando de seguida em *Interrupt*, tal como mostra a Figura 3, não se obtiveram resultados desse algoritmo.

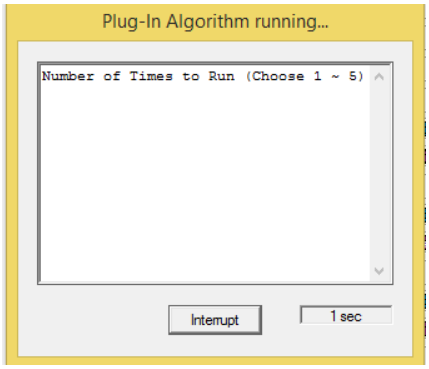

Figura 3 – Demonstração do algoritmo SB-LS.

Depois, optou-se pela heurística *General SB Routine*, tendo-se selecionado na caixa diálogo da função objetivo sucessivamente cada uma das funções objetivo, à exceção do tempo de percurso total pesado (*Total Weighted Flow Time*) e do atraso total pesado (*Total Weighted Tardiness*). Pois, como nesta instância todos os lotes têm o mesmo peso (1) obter-se-iam os mesmos resultados do escalonamento para as funções objetivo, tempo de percurso total (*Total Flow Time)* e atraso total (*Total Tardiness*), respetivamente.

Como resultados da aplicação da heurística com cada uma das funções objetivo, obtém-se um Diagrama de Gantt com a sequência de cada lote em cada uma das máquinas ao longo do tempo e, ainda duas janelas, a *Sequence-Seqs* e a *Job Pool* completa. Ou seja, o algoritmo faz a afetação dos lotes às máquinas e o respetivo sequenciamento.

Na janela *Sequence-Seqs* é apresentada para cada máquina a sequência de lotes, com indicação do tempo de *setup*, instante de início (*Start*), instante de fim (*Stop*) e tempo de processamento (*Pr.tm*) para cada lote e ainda, o tempo de *setup* e de processamento total da máquina, ou seja, a soma destes tempos dos lotes alocados a essa máquina. Além disto, nesta janela também se obtém um sumário (*Summary*), no qual se apresentam os resultados para as diversas medidas de desempenho.

Na segunda janela é apresentada a informação da janela *Job Pool* inicial acrescida de informação do instante de início (*Bgn*), de fim (*End*) e o tempo de atraso (T) e de atraso pesado (wT), para cada lote com descriminação em cada operação.

Os diagramas de Gantt obtidos para as funções objetivo *makespan*, atraso máximo (*max. Tardiness*), tempo de percurso total e atraso total, encontram-se, respetivamente, nas Figuras 4, 5, 6 e 7.

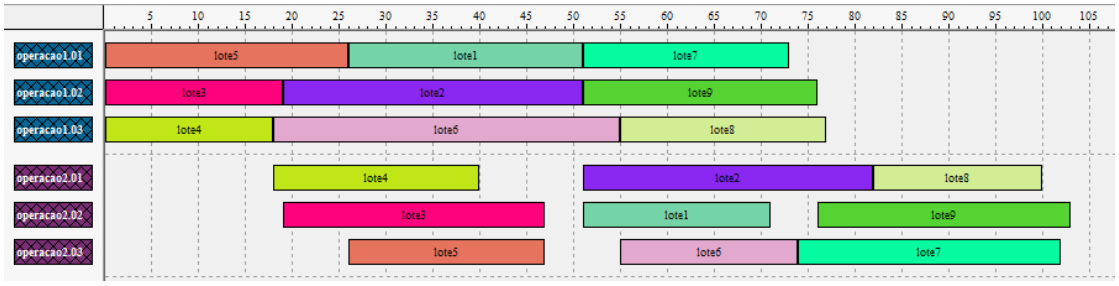

Figura 4 – Diagrama de Gantt obtido pela heurística *General SB Routine* com a função objetivo *makespan*.

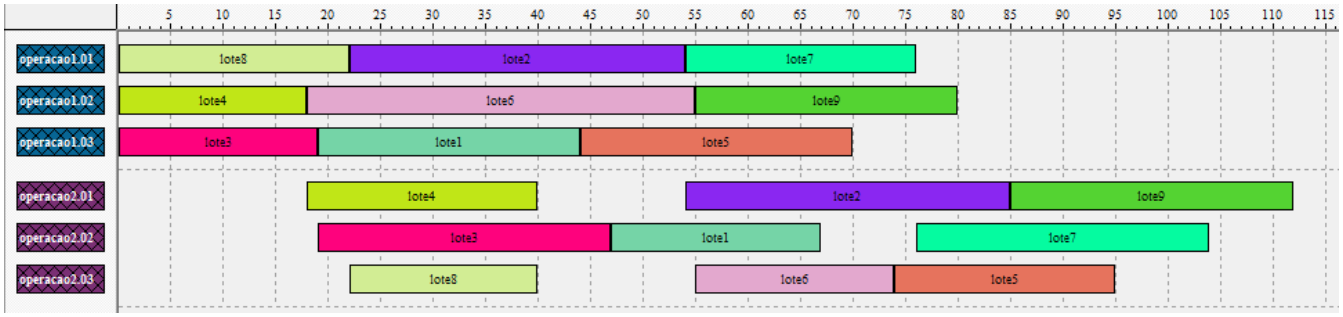

Figura 5 – Diagrama de Gantt obtido pela heurística *General SB Routine* com a função objetivo atraso máximo *(max.Tardiness)*.

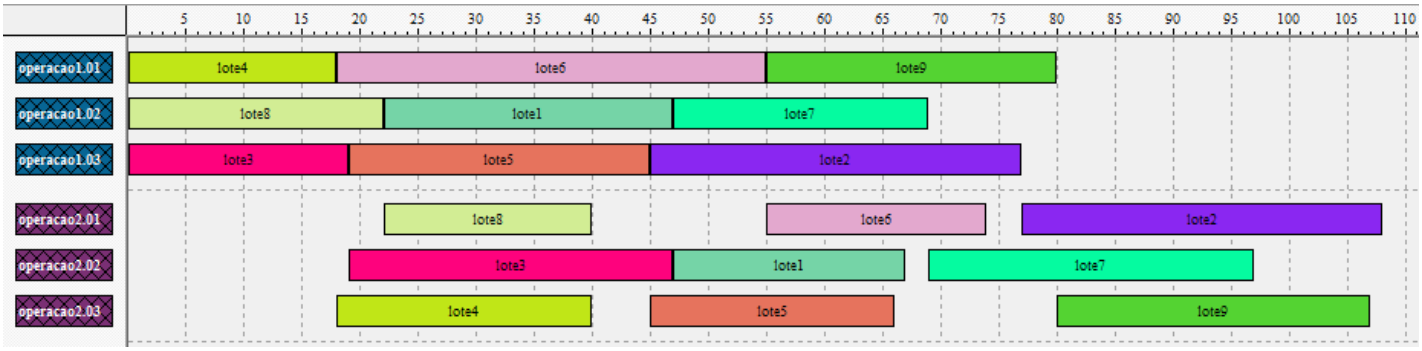

Figura 6 – Diagrama de Gantt obtido pela heurística *General SB Routine* com a função objetivo tempo de percurso total *(Total Flow Time)*.

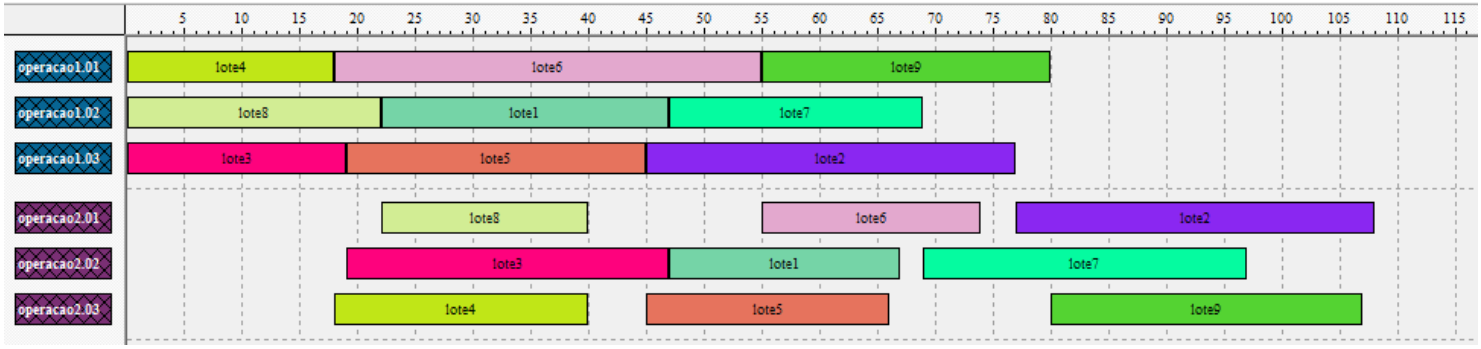

Figura 7 – Diagrama de Gantt obtido pela heurística *General SB Routine* com a função objetivo atraso total *(Total Tardiness)*.

Note-se que, de modo a facilitar a leitura dos digramas de Gantt, clicou-se na barra de ferramentas superior da janela do Gráfico de Gantt (*Gantt Chart - Seqs*) em *Grid lines* para obter as linhas que permitem ver melhor onde começa e acaba cada lote, em *Job names* para aparecer o nome de cada lote nas barras e, em *wider* e *shorter* para alargar e encurtar os lotes, respetivamente.

Como é pretendido nesta alínea resolver o problema de escalonamento da produção usando três métodos ou abordagens heurísticas, só se tendo obtido resultados a partir de um método, foram usadas também regras.

Tendo em conta a definição das regras, concluiu-se que as regras que melhor se adequariam ao problema seriam a EDD, a MS, a SPT, a LPT e a CR. Sendo que, optou-se por usar a regra SPT e a regra CR, pois como esta última é um compromisso entre a regra EDD e a LPT [12] de algum modo está-se a ter as duas em conta. A regra MS não foi usada, pois como é uma variação da regra EDD [12] de algum modo também é otimizada na regra CR.

Em relação às regras FCFS e WSPT, estas não foram consideradas, pois, a primeira faz o escalonamento das tarefas em ordem às datas de lançamento [12], que neste problema são nulas e, a regra WSPT é uma versão pesada da regra SPT [12], pelo que como neste problema os pesos são todos iguais a 1, obter-se-ia os mesmos resultados da aplicação da regra SPT.

Para isso, na barra de ferramentas superior em *Schedule* selecionou-se a opção *Rule e*, respetivamente SPT e CR. Da mesma forma descrita na aplicação das heurísticas, obtiveram-se os diagramas de Gantt que se encontram, respetivamente, nas Figuras 8 e 9.

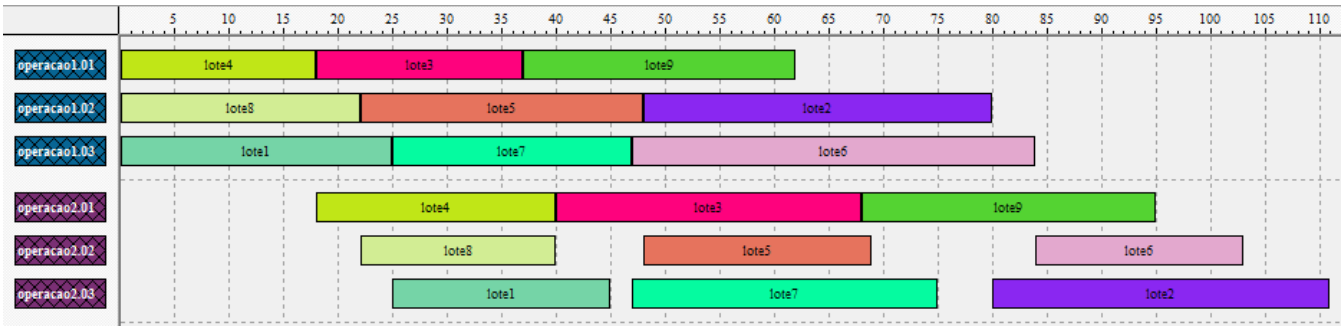

Figura 8 – Diagrama de Gantt obtido pela regra SPT.

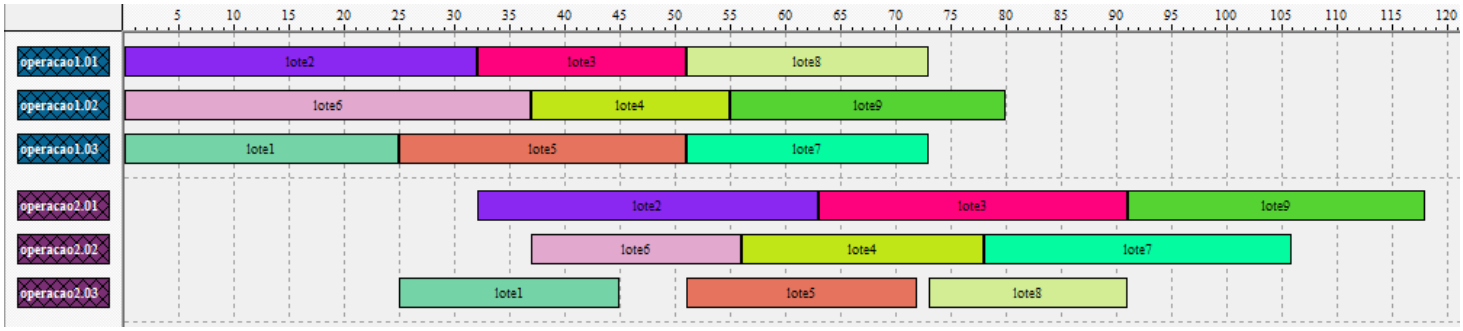

Figura 9 – Diagrama de Gantt obtido pela regra CR.

O desempenho de um algoritmo é julgado por valores de vários objetivos, assim como pela sua velocidade [12]. No sentido de obter esses valores, clicou-se em *Log Book window* na barra de ferramentas superior, donde se obtiveram os resultados para cada algoritmo usado, representados na Figura 10.

Schedule	Time	C_{max}	T_{max}	ΣU_j	ΣC_j	ΣT_j	$\Sigma w_j C_j$	$\Sigma w_j T_j$
CR	1	118	73	9	720	455	720	455
General SB Routine / Cmax	1	103	67	9	666	401	666	401
General SB Routine / Lmax	2	112	70	9	664	399	664	399
General SB Routine / sum(C)	1	108	93	9	646	381	646	381
General SB Routine / sum(T)	1	108	93	9	646	381	646	381
SPT	1	111	96	9	646	381	646	381

Figura 10 – Resultados das medidas de desempenho obtidas por aplicação dos diferentes métodos.

Time representa a velocidade do algoritmo, refletida pelo tempo de execução do algoritmo. C_{max} representa o *makespan*, ou seja, o tempo máximo de conclusão do trabalho; T_{max} representa o atraso máximo, ou seja, a maior parte positiva da diferença entre o tempo de conclusão (C_j) e a data de entrega (d_j). $\sum Uj$, $\sum Cj$ e $\sum Tj$ representam, respetivamente, o número total de lotes em atraso, o tempo de percurso total e o atraso total.

Por último, $\sum wjCj$ e $\sum wjTj$ correspondem ao tempo total de percurso pesado e ao atraso total pesado, respetivamente [12].

Note-se que, na literatura de escalonamento T_{max} (*maximum tardiness*) é muitas vezes referido como L_{max} (*maximum lateness*). Sendo que, embora o *lateness* de tarefas individuais não seja o mesmo que o seu *tardiness*, estes dois objetivos são praticamente equivalentes, $L_{max}=T_{max}$, a menos que todas as tarefas sejam iniciais [12].

Para obter os resultados através de gráficos, para as medidas de desempenho tempo total de produção ou *makespan*, atraso máximo e número de lotes atrasados, clicou-se na ferramenta *Select performance measures*, onde se selecionaram respetivamente estas medidas de desempenho.

Relativamente às medidas de desempenho tempo de percurso médio e atraso médio, estas não são diretamente obtidas do *Lekin*. Mas, sabendo o tempo de percurso total ($\sum Cj$) e o atraso total ($\sum Tj$) , esses valores podem ser obtidos pelo quociente destes últimos pelo número de lotes [9], neste caso 9. Assim, pegando nos valores obtidos para o tempo de percurso total e para o atraso total em todos os métodos, que se encontram na Figura 10, calculou-se o tempo de percurso médio e o atraso médio para cada método, numa folha de *Excel*. Depois, com estes valores, construiu-se, também no *Excel*, um gráfico de barras para cada medida de desempenho.

Os gráficos obtidos para cada medida de desempenho, encontram-se representados nas Figuras 11, 12, 13, 14 e 15.

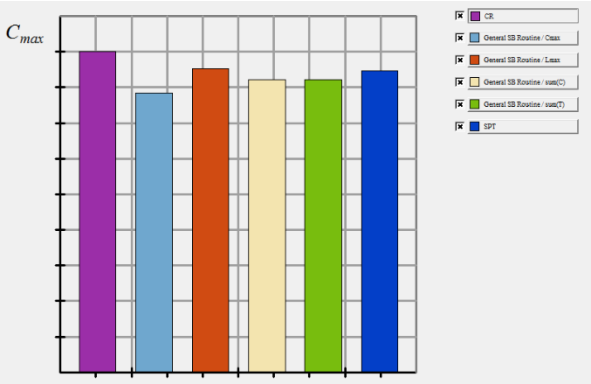

Figura 11 – Representação gráfica dos resultados obtidos para o *makespan* (C_{max}) em cada um dos algoritmos usados.

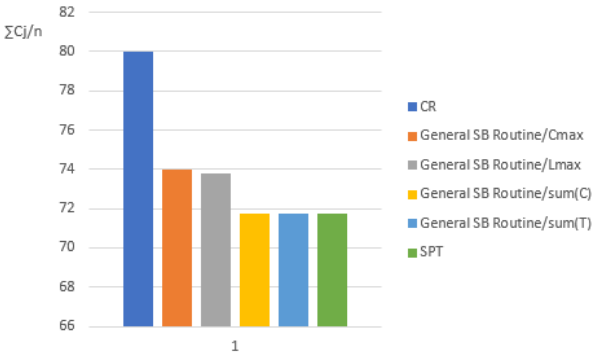

Figura 12 – Representação gráfica dos resultados obtidos para o tempo de percurso médio ($\frac{\sum Cj}{n}$) em cada um dos algoritmos usados.

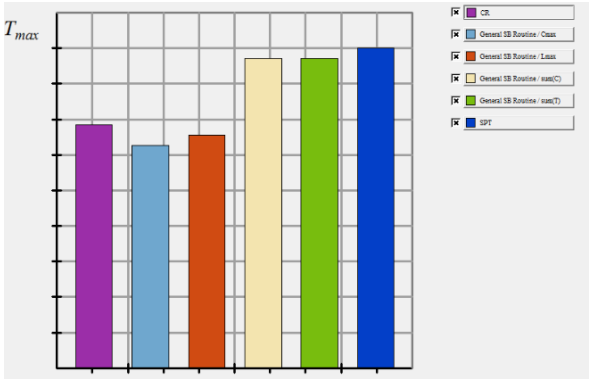

Figura 13 – Representação gráfica dos resultados obtidos para o atraso máximo (T_{max}) em cada um dos algoritmos usados.

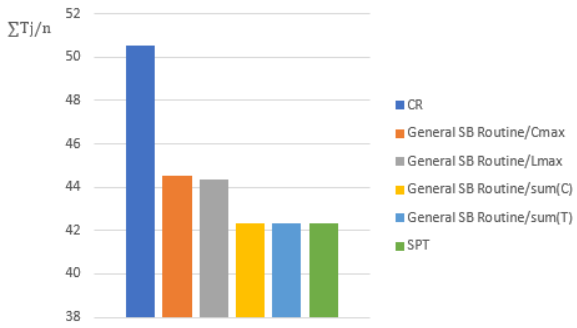

Figura 14 – Representação gráfica dos resultados obtidos para o atraso médio ($\frac{\sum Tj}{n}$) em cada um dos algoritmos usados.

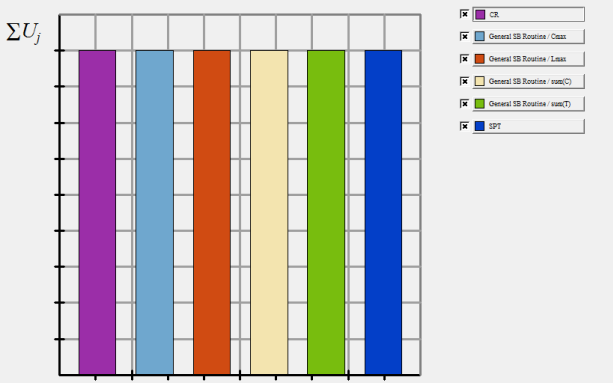

Figura 15 – Representação gráfica dos resultados obtidos para o número de lotes atrasados ($\sum Uj$) em cada um dos algoritmos usados.

2.2.Parte B – Problema em ambiente de oficina flexível

Nesta parte pretende-se fazer o memso que fora feito na parte anterior, mas resolvendo o problema de escalonamento da produção considerando-o em ambiente de oficina flexível (*Flexible Job Shop*, FJS).

Para isso, foi usado o sistema *Lekin*.

Mais uma vez, começou-se por introduzir os dados no sistema, pelo mesmo processo descrito para o ambiente em linha flexível, à exceção do tipo de estrutura a selecionar que foi *Flexible Job Shop* e a ordem de realização das operações a partir do lote 5, inclusive, que são por ordem inversa do fluxo normal de produção da configuração em linha flexível. Ou seja, a partir do lote 5 inicia-se o processo de produção com a operação 2 no grupo de máquinas B e logo após a operação 1 no grupo de máquinas A.

Da mesma forma que no ambiente em linha flexível, procedeu-se de seguida ao escalonamento, selecionando a opção *Schedule* na barra de ferramentas superior. Sendo que, é possível escolher todas as regras de prioridade e apenas uma heurística, a *General SB Routine.*

Começando por selecionar a heurística *General SB Routine*, para minimização do tempo total de produção (**C_{max}**), do atraso máximo (**L_{max}**), do tempo de percurso total (**sum(C)**) e do atraso total (**sum (T)**), obtiveram-se os diagramas de Gantt que se encontram respetivamente nas Figuras 16, 17, 18 e 19. Note-se que, mais uma vez não se selecionaram as funções objetivo, tempo de percurso total pesado e atraso total pesado, pelas mesmas razões descritas anteriormente

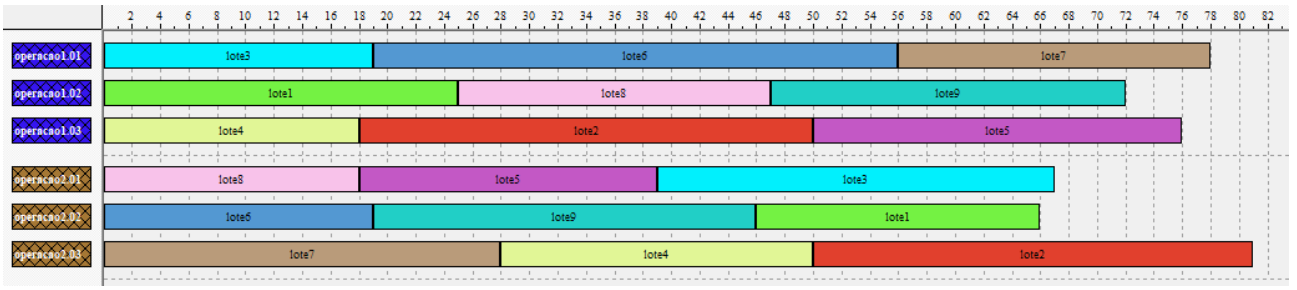

Figura 16 – Diagrama de Gantt obtido pela heurística *General SB Routine* para minimização do tempo total de produção.

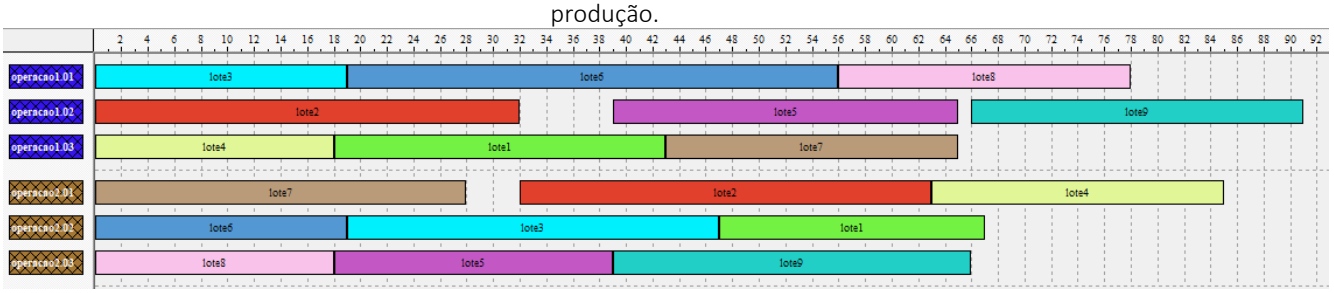

Figura 17 – Diagrama de Gantt obtido pela heurística *General SB Routine* para minimização do atraso máximo.

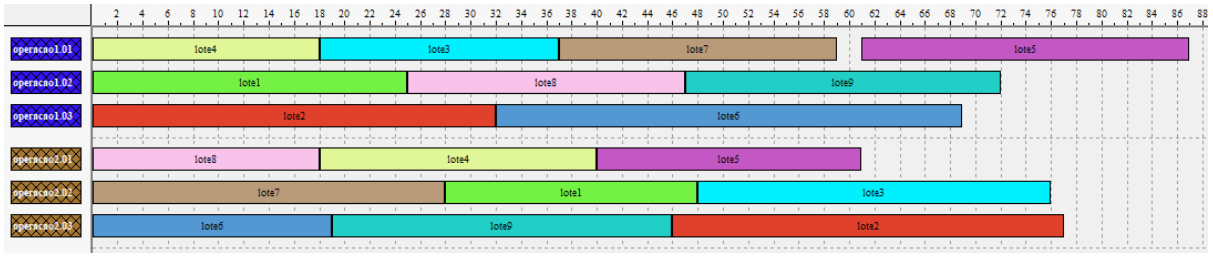

Figura 18 – Diagrama de Gantt obtido pela heurística *General SB Routine* para minimização do tempo de percurso total.

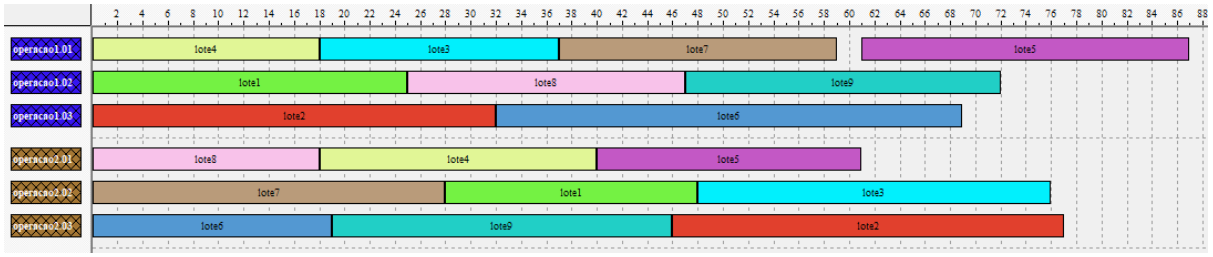

Figura 19 – Diagrama de Gantt obtido pela heurística *General SB Routine* para minimização do atraso total.

Mais uma vez, como se pretende resolver o problema de escalonamento da produção usando três métodos ou abordagens heurísticas, só se tendo obtido resultados a partir de um método, foram usadas também regras. Neste caso, foram aplicadas as regras LPT e SPT, tendo-se obtido os diagramas de Gantt que se encontram nas Figuras 20 e 21, respetivamente.

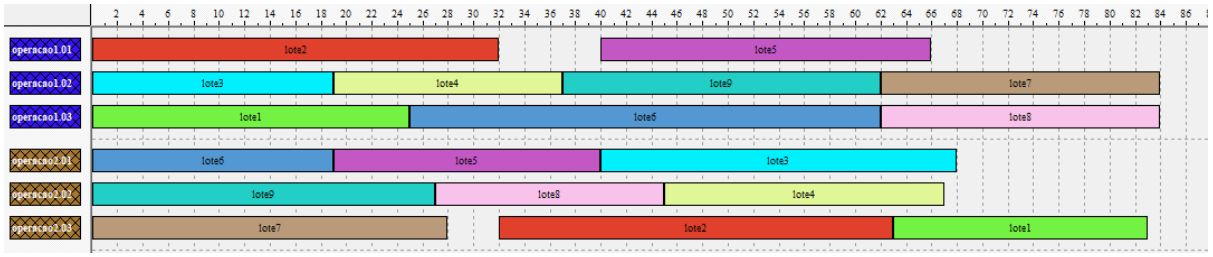

Figura 20 – Diagrama de Gantt obtido pela regra LPT.

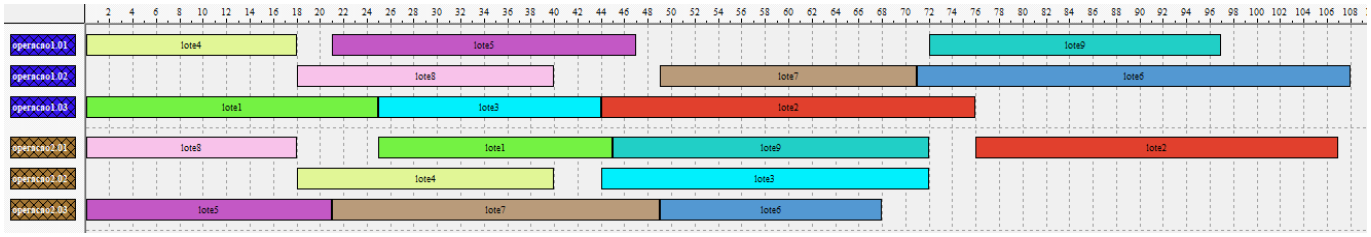

Figura 21 – Diagrama de Gantt obtido pela regra SPT.

Da mesma forma descrita para o ambiente em linha flexível, obtiveram-se os resultados das medidas de desempenho para cada algoritmo usado, que se encontram na Figura 22.

Schedule	Time	C_{max}	T_{max}	ΣU_j	ΣC_j	ΣT_j	$\Sigma w_j C_j$	$\Sigma w_j T_j$
General SB Routine / Cmax	1	81	66	9	593	328	593	328
General SB Routine / Lmax	1	91	50	9	617	352	617	352
General SB Routine / sum(C)	1	87	62	9	575	310	575	310
General SB Routine / sum(T)	1	87	62	9	575	310	575	310
LPT	1	84	63	9	639	374	639	374
SPT	1	108	92	9	627	362	627	362

Figura 22 – Resultados das medidas de desempenho obtidas para a heurística *General SB Routine* e para as regras de prioridade LPT e SPT.

Mais uma vez, através da ferramenta *Select performance measures* obtiveram-se os resultados graficamente para as medidas de desempenho. Os gráficos obtidos para cada medida de desempenho, encontram-se representados nas Figuras 23, 24, 25 e 26.

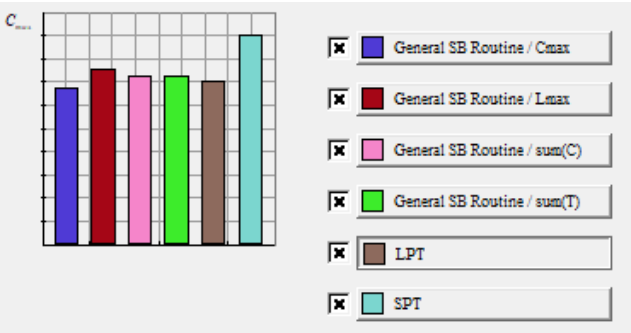

Figura 23 – Gráfico de barras do tempo total de produção (C_{max}) obtido para cada algoritmo usado.

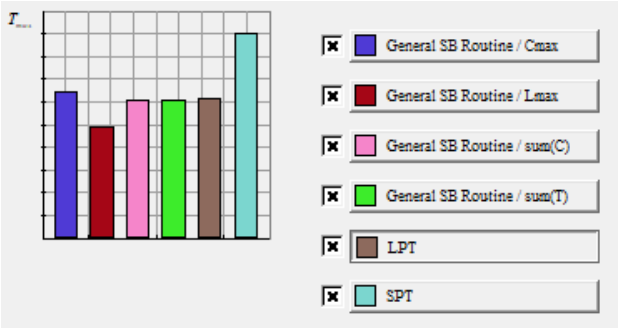

Figura 24 – Gráfico de barras do atraso máximo (T_{max}) obtido para cada algoritmo usado.

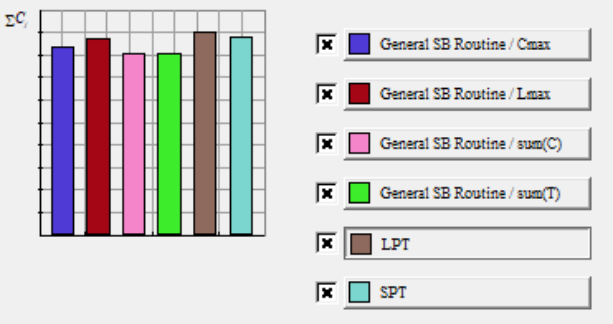

Figura 25 – Gráfico de barras do tempo de percurso total ($\sum Cj$)obtido para cada algoritmo usado.

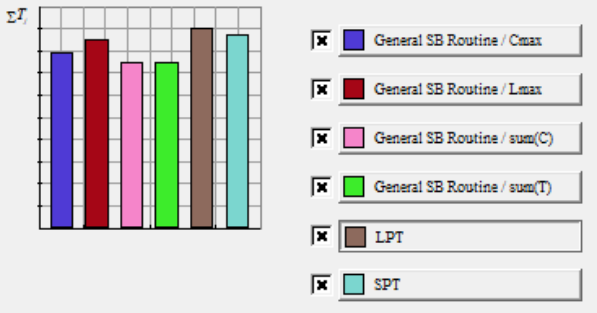

Figura 26 – Gráfico de barras do atraso total ($\sum Tj$) obtido para cada algoritmo usado.

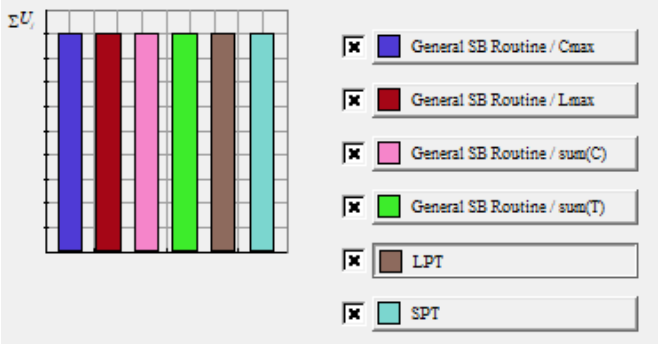

Figura 27 – Gráfico de barras do número de lotes atrasados ($\sum Uj$) obtido para cada algoritmo usado.

Contrariamente à parte A, nesta parte optou-se por obter também no *Lekin* os gráficos para as medidas de desempenho tempo de percurso total e atraso total. Pois, apesar de se querer obter o atraso médio e o tempo de percurso médio, os gráficos são proporcionais, no sentido em que a relação entre os vários métodos é a mesma, uma vez que para obter os valores médios, dividem-se os valores do tempo de percurso total e atraso total pelo número de lotes [9], que neste caso é 9.

No entanto, pegando nos valores do tempo de percurso total ($\sum Cj$) e do atraso total ($\sum Tj$) da Figura 22, calcularam-se os valores do tempo de percurso médio e do atraso médio para cada método usado, encontrando-se os resultados na Tabela 2.

Tabela 2 – Tempo de percurso médio e atraso médio calculados para cada método

Método/Medida de desempenho	***General SB Routine/C_{max}***	***General SB Routine/L_{max}***	***General SB Routine/sum(C)***	***General SB Routine/sum(T)***	**LPT**	**SPT**
Tempo de percurso médio (u.t.)	65.9	68.6	63.9	63.9	71	69.7
Atraso médio (u.t.)	36.4	39.1	34.4	34.4	41.6	40.2

2.3. Parte C – Métodos/abordagens de escalonamento da produção conhecidos e alternativos à resolução do problema

Além dos métodos que o *Lekin* possui para fazer o escalonamento existem outros, como os algoritmos de *Hodgson* e *Jonhson* para programação da produção em ambientes de linha de produção (*Flow Shop*) e o algoritmo de *Jackson*, para programação em ambientes de oficina de fabrico (*Job Shop*) [9].

O Algorimo de *Hodgson* é um procedimento heurístico que minimiza o número de entidades atrasadas [9].

Para isso, começa por determinar a solução inicial através da regra EDD, ou seja, sequencia o conjunto de entidades por ordem crescente da data de entrega. Além disso, cria 2 conjuntos, o conjunto E e o conjunto L, que inicialmente contêm todas as entidades, no caso do conjunto E e está vazio, no caso do conjunto L [9].

Depois, verifica se existem entidades no conjunto E em atraso. Sendo que, se não houver o algoritmo termina e o conjunto E é ótimo, mas se houver, identifica a primeira entidade de E que tem atraso. Supondo que a primeira entidade com atraso é a entidade k, o algoritmo identifica a entidade com maior tempo de processamento nas primeiras k entidades. Depois, tira essa entidade do conjunto E para o conjunto L. De seguida, recalcula os tempos de conclusão das entidades que restam no conjunto E e volta a verificar se existem entidades nesse conjunto em atraso, repetindo todo o processo até que este não tenha mais entidades em atraso [3].

No final, obtém-se a sequência de fabrico que minimiza o número de entidades em atraso. Esta é constituída pelas entidades do conjunto E pela ordem respetiva às quais se adicionam no final as entidades com atraso do conjunto L por qualquer ordem, pois isso não altera o número de tarefas com atraso [9].

O algoritmo de *Johnson* é uma regra de sequenciamento dos trabalhos a executar numa linha [9] que encontra o menor caminho entre 2 pontos otimizando todos os recursos existentes na sua utilização. Para a sua execução, baseia-se em algumas regras, tais como os tempos de *setup* estão incluídos nos tempos de processamento e sequenciamento das tarefas, as etapas do processo são sucessivas e a etapa 2 é dependente da realização da etapa 1. Pode ser aplicado a qualquer processo, desde que os seus tempos de execução sejam conhecidos [7].

Em termos de processo, identifica sucessivamente as tarefas com menor tempo de processamento e sequencia-as por essa ordem conforme a operação ou máquina onde são realizadas. No final, obtém a sequência ótima de fabrico que minimiza o *makespan* [9].

Ao contrário do algoritmo de *Johnson* aplicável à programação de n trabalhos em 2 máquinas sob um esquema fixo, ou seja, os trabalhos seguem sempre a mesma ordem, a regra ou método de *Jackson* permite programar os trabalhos quando a sequência dos trabalhos é aleatória, ou seja, quando os trabalhos não seguem a mesma sequência [10]. Este método tem como objetivo a minimização do *makespan* em problemas de programação da produção em oficinas de fabrico com apenas 2 máquinas e qualquer número de entidades [11].

Neste contexto, o método *Jackson* considera várias etapas. Em primeiro lugar, classifica os trabalhos existentes em 4 grupos possíveis: aqueles que exigem apenas a máquina A (A), aqueles que exigem apenas a máquina B (B), aqueles que passam primeiro pela máquina A e depois pela B (AB) e, ainda aqueles que passam primeiro pela máquina B e depois pela A (BA). Em segundo lugar, ordena os trabalhos de (AB) e (BA) aplicando a regra

de *Johnson*. Depois, classifica os trabalhos de A e B arbitrariamente. Em quarto lugar, programa na máquina A primeiro os trabalhos de (AB), depois o trabalho em (A) e, por fim o trabalho em (BA). Por fim, programa na máquina B em primeiro lugar os trabalhos de (BA), depois os trabalhos em (B) e, finalmente, os trabalhos em (AB) [10].

Embora algumas técnicas sejam apropriadas para problemas de programação da produção específicos, uma grande parte dos problemas de programação da produção são problemas de optimização combinatória, pelo que também podem ser resolvidos por técnicas, heurísticas e algoritmos de uso geral para resolver essa classe de problemas. São exemplos, técnicas de *Branch and Bound*, técnicas de procura local e até as regras de prioridade [11].

As técnicas de *Branch and Bound* baseiam-se na enumeração inteligente das soluções candidatas a solução ótima inteira de um problema, efetuando sucessivas partições do espaço das soluções e cortando a árvore de pesquisa através da consideração de limites calculados ao longo da enumeração [3]. Este tipo de técnicas garante encontrar a solução óptima, mas para problemas de grandes dimensões o tempo necessário pode ser proibitivo [11].

As técnicas de procura local (NST – Técnica de Pesquisa na Vizinhança) permitem obter rapidamente, de forma simples e flexível, boas soluções que podem não ser ótimas, em alternativa a técnicas de optimização que apesar de garantirem soluções ótimas, podem ser impraticáveis devido ao tempo computacional necessário. Os elementos básicos, na aplicação desta abordagem à programação da produção são os conceitos de semente, de vizinhança e de mecanismo para geração de vizinhanças. O mecanismo de geração é um método que cria sistematicamente soluções relacionadas com uma solução semente [11].

Em termos de procedimento NST, começa-se por seleccionar uma solução semente avaliando o seu desempenho e, de seguida gerar e avaliar as soluções da vizinhança da semente. Sendo que, se nenhuma das soluções produzir melhor desempenho do que a semente, então a pesquisa termina. Caso contrário, selecciona-se a solução da vizinhança com melhor desempenho para ser a nova semente e repete-se o processo até ser atingido o critério de paragem definido. Note-se que é necessário especificar todos estes parâmetros, por exemplo, a forma como se gera a primeira semente, que pode ser pela regra SPT [11].

Uma regra de prioridade ou de despacho é uma regra que estabelece a prioridade com que são processadas as entidades que esperam numa máquina [11].

Existem muitas regras de prioridade, pelo menos várias centenas. Estas podem ser classificadas como estáticas, se o valor da prioridade é independente do instante de tempo, como é o caso da regra SPT pois o tempo de processamento não varia com o tempo ou, caso contrário como dinâmicas. Outra forma de classificação é como regras locais, se apenas consideram a informação relativa à fila de espera onde se encontra a entidade respectiva, como é exemplo a regra SPT ou, regras globais se consideram informação de todo o sistema produtivo, como é o caso da regra MWKR (*Most Work Remaining*) [11].

Nesta última regra, quando uma máquina fica livre, de todas as entidades na sua fila de espera tem maior prioridade aquela cujo somatório dos tempos de processamento das operações que ainda falta realizar nessa e nas próximas máquinas é maior, ou seja, é dada maior prioridade à entidade com mais conteúdo de trabalho por realizar [11]. Esta regra pode ser aplicada, por exemplo, na programação da produção em ambiente de oficina de fabrico clássica geral com duas ou mais máquinas [9].

Apesar da aplicação de regras de prioridade em ambientes industriais não se poder dizer que determinada regra domina as outras em termos de *makespan*, a regra com mais

sucesso tem sido frequentemente a regra MWKR e algumas das suas variações. No entanto, a regra SPT produz melhores resultados num pequeno número de situações [11].

As experiências demonstraram que a programação da produção em oficinas de fabrico baseadas em regras de prioridade é um método praticável para a obtenção de soluções subóptimas [11].

Note-se que, para outros ambientes outros métodos podem ser aplicados. Por exemplo, para programação em ambiente de operadores ou máquinas em paralelo, para trabalhos que permitem interrupção pode ser usado o algoritmo de *Mc Neughton* que minimiza o *makespan* [9].

Dada a diversidade de métodos de escalonamento da produção que existem, várias são as alternativas à resolução deste problema. No entanto, pode-se sugerir o algoritmo de *Hodgson e o* algoritmo de *Jonhson* aplicável a 3 máquinas para o problema em ambiente de linha flexível. Já o algoritmo de *Jackson* não é alternativa para o problema em ambiente de oficina flexível, uma vez que o problema tem mais de 2 máquinas. Além disso, técnicas de *Branch and Bound*, desde que o tempo de execução o permita e, técnicas de procura local e outras regras de prioridade como a MWKR, também podem ser uma alternativa.

3. Análise dos Resultados

Da análise dos diagramas de Gantt obtidos, tanto em ambiente de linha flexível como em ambiente de oficina flexível, respetivamente Figuras 4 a 9 e 16 a 21, verificou-se que os resultados obtidos por cada um dos métodos são diferentes, à exceção dos resultados obtidos pela heurística *General SB Routine* para minimização do tempo de percurso total e do atraso total.

No entanto, todos os resultados são válidos, uma vez que todos os lotes foram alocados a uma máquina em cada operação, realizando as operações pela sequência correta. Ou seja, realizam primeiro a operação 1 e só depois desta ter terminado a operação 2, no caso do ambiente em linha flexível e para os primeiros 4 lotes no ambiente em oficina flexível e, primeiro a operação 2 e só depois a operação 1, no caso dos restantes lotes em ambiente de oficina flexível.

Analisando os gráficos de barras de cada medida de desempenho em ambos os ambientes, Figuras 11 a 15 e 23 a 27, respetivamente, verifica-se que, em relação ao *makespan*, Figura 11 e 23, o seu menor valor foi obtido nos escalonamentos obtidos pela heurística *General SB Routine* quando se escolheu como função objetivo o *makespan*. Tal já era esperado, pois ao selecionar o *makespan* como função objetivo, a heurística determina a sequência de lotes em cada máquina de modo a minimizar o tempo total de produção. Note-se que, tal como era expectável o *makespan* obtido em todos os escalonamentos, corresponde ao instante de tempo em que o último lote acaba o processamento. Por exemplo, no escalonamento obtido para o problema em ambiente de oficina flexível pela heurística *General SB Routine* que minimiza o *makespan*, o makespan é de 81 u.t., valor que corresponde ao instante de tempo em que o último lote, o lote 2, termina o processamento na sua última operação, na operação 2.

Analogamente, em relação ao atraso máximo, Figuras 13 e 24, também se esperava que se obtivesse o menor valor nos escalonamentos obtidos pela heurística *General SB Routine* quando se escolheu como função objetivo o atraso máximo. No entanto, no ambiente de linha flexível tal não aconteceu, tendo sido o escalonamento obtido pela heurística *General SB Routine* com função objetivo *makespan*, que obteve o menor atraso máximo. Sendo o atraso de cada lote o tempo que o instante de tempo do fim do processamento excede a data de entrega, é possível perceber qual o lote ao qual corresponde o atraso máximo. Por exemplo, no escalonamento obtido para o problema em ambiente de oficina flexível pela heurística *General SB Routine* que minimiza o atraso máximo, o atraso máximo é de 50 u.t. que corresponde ao atraso do lote 4, pois apesar deste lote ter data de entrega de 35 acaba de ser processado no instante 85 (85-35=50).

Da mesma forma, em relação ao tempo de percurso total e ao atraso total e, consequentemente aos valores médios, respetivamente Figuras 25 e 26 e Tabela 2 e, 12 e 14, tal como era expectável os escalonamentos com menor valor destas medidas de desempenho, foram os obtidos pela heurística *General SB Routine* com função objetivo tempo de percurso total e atraso total. Sendo que, como os escalonamentos obtidos por esta heurística com as duas funções objetivo foram iguais, também os valores obtidos para estas medidas de desempenho, como para todas as outras são os mesmos. Note-se que, no ambiente em linha flexível, tanto o tempo de percurso médio como o atraso médio e, inerentemente o tempo de

percurso total e atraso total, como mostra a Figura 10, têm também o menor valor no escalonamento obtido pela regra SPT.

Por fim, das Figuras 15 e 27 verifica-se que em todos os métodos se obtiveram escalonamentos com o mesmo número de lotes atrasados, tanto em ambiente de linha flexível como em ambiente de oficina flexível. Concretamente, o número de lotes atrasados é 9, como se pode ver nas Figuras 10 e 22, pelo que em todos os escalonamentos obtidos todos os lotes apresentam atraso.

Relativamente às regras aplicadas em ambiente de linha flexível, a CR e a SPT, das Figuras 11, 12, 13 e 14, verifica-se que em todas as medidas de desempenho a regra SPT obteve melhores resultados. Sendo que, em relação aos valores de *makespan* e de atraso máximo, Figuras 11 e 13, estes são pouco inferiores aos obtidos pela regra *CR.* No entanto, para o tempo de atraso médio e para o atraso médio, Figuras 12 e 14, a diferença é mais acentuada.

Relativamente às regras aplicadas em ambiente de oficina flexível, a SPT e a LPT, verifica-se que em relação ao tempo total de produção e ao atraso máximo, Figuras 23 e 24 respetivamente, a regra LPT obteve melhores resultados, possuindo valores inferiores aos da regra SPT, diferença que é mais acentuada para o atraso máximo. Pois, contrariamente à regra SPT, a regra LPT inicia o processo tendo em conta os lotes com maior tempo de processamento, deixando os de menor tempo para última reposição, pelo que os lotes ficam menos tempo à espera de terminar o processo. Contrariamente, para as medidas de desempenho tempo de percurso total e atraso total, Figuras 25 e 26 respetivamente, ainda que a diferença seja muito pouca, é a regra SPT que obtém melhores resultados.

Note-se que, a eficiência de uma regra para minimizar o atraso nos trabalhos é dependente da quantidade ou carga de trabalhos na oficina, da forma como as datas de entrega são estabelecidas e do tempo de folga disponível para realizar os trabalhos.

Das Figuras 10 e 22, verificou-se que, para os mesmos métodos aplicados para resolver um problema de escalonamento da produção em ambiente de linha flexível e de oficina flexível, os valores das medidas de desempenho são superiores no ambiente em linha flexível. Assim, pode-se dizer que o ambiente em oficina flexível permite obter melhores escalonamentos.

4. Conclusões

Da realização deste trabalho, várias conclusões foram tiradas.

Em primeiro lugar, relativamente às heurísticas que o *Lekin* possui para resolver um problema de escalonamento da produção, concluiu-se que, dependendo do problema nem todas as heurísticas podem ser aplicadas. Além disso, concluiu-se que aplicando métodos diferentes ou o mesmo método, mas com funções objetivo diferentes ao mesmo problema obtêm-se, geralmente, escalonamentos diferentes, tendo sido uma exceção a heurística *General SB Routine* para a função objetivo tempo de percurso total e atraso total, donde se obtiveram os mesmos resultados.

Relativamente aos valores das medidas de desempenho, concluiu-se que, geralmente o método que obtém melhores resultados para uma dada medida de desempenho é o método que tem como objetivo a minimização dessa medida. Neste problema em concreto, foi uma exceção o atraso máximo no ambiente em linha flexível, para o qual se obteve que o escalonamento com menor atraso máximo, foi o obtido pela heurística *General SB Routine* com função objetivo *makespan*, em vez da função objetivo atraso máximo, como se esperava. Assim, o escalonamento obtido pela heurística *General SB Routine* com função objetivo *makespan* para o problema em ambiente de linha flexível é bom, no sentido em que minimiza o *makespan* e o atraso máximo.

Conclui-se então que, o melhor método para resolver um problema de escalonamento da produção irá depender do objetivo que se tem, ou seja, da medida de desempenho que se pretende minimizar.

Em relação às regras aplicadas a este problema de escalonamento, em ambiente de linha flexível concluiu-se que entre a regra SPT e CR, obtêm-se melhores resultados com a regra SPT. Já em ambiente em oficina flexível, concluiu-se que, entre a regra SPT e LPT, a que obtém melhores escalonamentos depende do objetivo que se pretende. Pois, enquanto a regra LPT obteve melhores resultados para o tempo total de produção e para o atraso máximo, a regra SPT obteve melhores resultados para as medidas de desempenho tempo de percurso total e atraso total. No entanto, como a diferença nestas últimas medidas de desempenho entre as duas regras é muito pequena, pode-se concluir que a regra LPT será melhor.

Além disso, também se concluiu que em todos os escalonamentos obtidos pelos diferentes métodos, os 9 lotes apresentam atraso e que neste problema o ambiente em oficina flexível permitiu obter melhores escalonamentos, pois para os mesmos métodos aplicados os valores das medidas de desempenho foram superiores no ambiente em linha flexível.

Por fim, concluiu-se que existe uma grande variedade de métodos de escalonamento da produção. Sendo que, o método escolhido para a resolução de um problema de escalonamento da produção depende sempre das características do problema, da capacidade da máquina que o processa e do tempo disponível de espera para encontrar uma solução.

Nesse sentido, neste problema outros métodos podiam ter sido aplicados alternativamente, como o algoritmo de *Hodgson e o* algoritmo de *Jonhson* aplicável a 3 máquinas para o problema em ambiente de linha flexível, técnicas de *Branch and Bound*, desde que o tempo de execução o permita e, técnicas de procura local e outras regras de prioridade como a MWKR.

5. Referências

[1] Silva, C. 2015.*Textos de Gestão da Produção Ver 04-2015.* Universidade do Minho.

[2] http://www.abepro.org.br/biblioteca/TN_STP_231_350_30213.pdf

[3] http://recipp.ipp.pt/bitstream/10400.22/8112/1/DM_JoseMarques_2015_MEEC.pdf

[4] http://web-static.stern.nyu.edu/om/faculty/pinedo/book3/index.html

[5] https://webcache.googleusercontent.com/search?q=cache:qtoPUd_-04oJ:https://www.engineeringresearch.org/index.php/GJRE/article/download/1316/1247+&cd=1&hl=pt-PT&ct=clnk&gl=pt

[6] http://www.densis.fee.unicamp.br/~franca/EA043/Transpa-Cap-6.pdf

[7] https://www.manufaturaemfoco.com.br/sequenciamento-de-tarefas-a-regra-de-johnson-e-suas-aplicacoes/

[8] https://repositorium.sdum.uminho.pt/bitstream/1822/7234/1/tese_LV.pdf

[9] Dias, J. 2018. *Apontamentos das aulas de Gestão da Produção.* Universidade do Minho.

[10] Gestión de Operaciones: https://www.gestiondeoperaciones.net/programacion-de-trabajos/aplicacion-de-la-regla-de-jackson-a-la-programacion-de-n-trabajos-en-2-maquinas/, consultado a 23/01/2018

[11] Carvalho, D.2000. *Capítulo IIII - Programação da produção.* Universidade do Minho

Disponível em: http://pessoais.dps.uminho.pt/jdac/apontamentos/Cap03_Program.pdf, consultado a 23/01/2018

[12] Help do sistema Lekin.