Instituto Politécnico do Porto
Escola Superior de Estudos Industriais e de Gestão

Hugo Miguel Gomes Tavares

Estudo e Análise do Sequenciamento de Tarefas de Produção: *JOB SHOP SCHEDULING*

Dissertação de Mestrado
Mestrado em Engenharia e Gestão Industrial
Orientação: Professora Doutora Isabel Cristina Lopes
Coorientação: Professor Doutor Luís Pinto Ferreira

Vila do Conde, Dezembro de 2015

Instituto Politécnico do Porto
Escola Superior de Estudos Industriais e de Gestão

Hugo Miguel Gomes Tavares

Estudo e Análise do Sequenciamento de Tarefas de Produção: *JOB SHOP SCHEDULING*

Dissertação de Mestrado
Mestrado em Engenharia e Gestão Industrial
Orientação: Professora Doutora Isabel Cristina Lopes
Coorientação: Professor Doutor Luís Pinto Ferreira

Vila do Conde, Dezembro de 2015

Hugo Miguel Gomes Tavares

Estudo e Análise do Sequenciamento de Tarefas de Produção: *JOB SHOP SCHEDULING*

Dissertação de Mestrado

Mestrado em Engenharia e Gestão Industrial

Membros do Júri

Presidente

Professor Doutor Venceslau Manuel Magalhães Correia

Escola Superior de Estudos Industriais e de Gestão – Instituto Politécnico do Porto

Professora Doutora Isabel Cristina da Silva Lopes

Escola Superior de Estudos Industriais e de Gestão – Instituto Politécnico do Porto

Professora Doutora Eliana Oliveira da Costa e Silva

Escola Superior de Tecnologia e Gestão de Felgueiras – Instituto Politécnico do Porto

Vila do Conde, Dezembro de 2015

Agradecimentos

Gostaria de agradecer em primeiro lugar à minha orientadora, Professora Doutora Isabel Cristina Lopes pelo apoio, disponibilidade, motivação e confiança transmitida durante este estudo.

Ao meu coorientador, Professor Luís Pinto Ferreira pelo apoio e disponibilidade.

Aos meus colegas de mestrado, pela partilha de conhecimento ao longo destes dois anos.

À Escola Superior de Estudos Industriais e de Gestão, pelas condições proporcionadas para a realização deste mestrado.

À Liliana, que ao longo destes anos sempre me apoiou e deu forças estando sempre ao meu lado em todos os momentos.

Aos meus Pais e irmãos que sempre acreditaram em mim, pelos seus ensinamentos e conselhos transmitidos ao longo da vida.

E a todos que de certa forma contribuíram para a realização desta dissertação.

Resumo analitico

Esta dissertação apresenta um estudo sobre os problemas de sequenciamento de tarefas de produção do tipo *job shop scheduling.*

Os problemas de sequenciamento de tarefas de produção pretendem encontrar a melhor sequência para o processamento de uma lista de tarefas, o instante de início e término de cada tarefa e a afetação de máquinas para as tarefas.

Entre estes, encontram-se os problemas com máquinas paralelas, os problemas *job shop* e *flow shop*. As medidas de desempenho mais comuns são o *makespan* (instante de término da execução de todas as tarefas), o tempo de fluxo total, a soma dos atrasos (*tardiness*), o atraso máximo, o número de tarefas que são completadas após a data limite, entre outros.

Num problema do tipo *job shop*, as tarefas (*jobs*) consistem num conjunto de operações que têm de ser executadas numa máquina pré-determinada, obedecendo a um determinado sequenciamento com tempos pré-definidos. Estes ambientes permitem diferentes cenários de sequenciamento das tarefas. Normalmente, não são permitidas interrupções no processamento das tarefas (*preemption*) e pode ainda ser necessário considerar tempos de preparação dependentes da sequência (*sequence dependent setup times*) ou atribuir pesos (prioridades) diferentes em função da importância da tarefa ou do cliente.

Pretende-se o estudo dos modelos matemáticos existentes para várias variantes dos problemas de sequenciamento de tarefas do tipo *job shop* e a comparação dos resultados das diversas medidas de desempenho da produção.

Este trabalho contribui para demonstrar a importância que um bom sequenciamento da produção pode ter na sua eficiência e consequente impacto financeiro.

Palavras-chave: *job shop*; sequenciamento; sequenciamento de tarefas de produção; modelos matemáticos; modelos MILP; heurísticas.

Abstract

This paper presents a study on sequencing problems of the type job shop scheduling.

Production sequencing problems aim to find the best sequence for processing a task list, the starting and ending instants of each task, and the machine allocation for tasks.

Among these problems, there are the parallel machines, the job shop and flow shop problems. The most common performance measures are the makespan (completion time of all tasks), the total flow time, the tardiness (the sum of the delays), the maximum delay, the number of tasks that are completed after the deadline, among others.

In a job shop problem, the tasks (jobs) are a set of operations that must be performed at a predetermined machine, obeying a certain sequence, with preset processing times. These environments allow for different task sequencing scenarios. Typically, the processing tasks are not allowed to be interrupted (preemption), and may also be worth considering sequence dependent setup times, or assign different weights to the jobs depending on the size of the task or priority of the customer.

It is intended to study the existing mathematical models for the various variants of the job shop scheduling problems and to compare the results of the several performance measures.

This work contributes to demonstrate the importance that a good sequencing of production tasks can have on their efficiency and resulting financial impact.

Keywords: job shop; scheduling; sequencing production tasks; mathematical models; MILP models; heuristics.

Índice

Lista de figuras

Lista de tabelas

Glossário

C_i Instante de término do processamento da tarefa *i*.

Cmax *(makespan*) Instante final de processamento de todas as ordens de produção em todas as máquinas.

EDD "Earliest due date". Prioriza as operações por ordem crescente das datas de entrega.

FCFS "First come first served". Prioriza a primeira operação da fila de espera da máquina.

JSF (*job shop* flexível) Permite que uma tarefa possa ser processada por qualquer uma de um conjunto pré-definido de máquinas.

L_i (*lateness*) Diferença entre a data de término de uma tarefa e o prazo de término (atraso ou antecipação).

LPT "Longest processing time". Prioriza a OP com maior tempo de processamento.

MILP Programação linear inteira mista.

MIP Programação inteira mista.

MS "Minimum Slack". Prioriza a OP cujo tempo de folga até à data de entrega é menor.

NP - difícil Problema que cresce exponencialmente com a dimensão dos dados do problema.

OP Ordem de Produção.

SPT "Shortest processing time". Prioriza a OP com menor tempo de processamento.

T_i Atraso da tarefa *i*.

Tmax (*tardiness*) Maior atraso verificado em todas as tarefas.

TSP *Traveling Salesman Problem.* Problema do caixeiro-viajante.

Capítulo 1 - Introdução

1. Introdução

Nesta dissertação de mestrado é realizado um estudo e análise do sequenciamento de tarefas de produção do tipo *job shop scheduling*.

Pretende-se demonstrar que se pode obter ganhos temporais significativos, optando-se por um modelo de sequenciamento, ao invés de se produzir aleatóriamente ou pela convicção de quem conduz o trabalho. A utilização eficaz e eficiente dos recursos disponíveis proporciona uma série de vantagens que podem tornar uma empresa mais competitiva. A diminuição dos custos reduzindo gastos desnecessários, a diminuição do tempo necessário para concluir uma tarefa e o aumento da produtividade são alguns benefícios que um bom sequenciamento pode proporcionar.

1.1 Objetivo da dissertação

Pretende-se estudar diferentes metodologias para resolver problemas do tipo *job shop*. Inicialmente, ir-se-á aplicar um modelo de programação linear e testar a sua formulação recorrendo ao *solver* do Excel, percebendo a sua limitação. Posteriormente, ir-se-á testar essa formulação através de um servidor *online,* especifico para a resolução de problemas de programação linear.

Pretende-se estudar algumas heurísticas e meta-heuristicas testando-as através de um *software* didático denominado de LEKIN e ir-se-á comparar esses resultados, com os obtidos pelo modelo de programação linear.

Outro objetivo que se pretende alcançar é o de demonstrar que se pode recorrer a regras de despacho, também conhecidas como regras de prioridade, e obter boas soluções em tempo útil. Mas nem sempre a mesma regra é a melhor para todos os casos.

1.2 Metodologia

Na primeira fase desta dissertação será feito um estudo bibliográfico sobre o tema em questão. Numa segunda fase serão estudados alguns modelos matemáticos e numa terceira fase serão estudados alguns métodos heurísticos e

meta-heuristicos. Numa última fase, serão testados alguns modelos analisando e comparando os resultados obtidos.

1.3 Estrutura da dissertação

Esta dissertação foi estruturada de forma a permitir uma leitura fluída e clara, dividindo-se em 5 partes distintas.

No primeiro capítulo apresentam-se os objetivos que se pretendem atingir, bem como a estrutura da dissertação.

No segundo capítulo clarifica-se o que é um problema de escalonamento de tarefas de produção em ambiente *job shop*, apresentam-se as suas medidas de desempenho, tais como, o *makespan*, o tempo de fluxo total, o atraso máximo, a soma dos atrasos, a soma dos atrasos e avanços, o número de tarefas atrasadas e o *lateness*. Serão apresentadas outras variantes do problema, tais como *job shop* flexível, o *flow shop* e o *job shop* com tempos de preparação dependentes da sequência.

No terceiro capítulo apresentam-se várias metodologias para a resolução de problemas do tipo *job shop*, através de métodos exatos e aproximados. Dos modelos exatos, serão revistas as formulações matemáticas, as formulações por programação inteira e o *Branch and Bound*. Serão apresentados os modelos matemáticos de otimização com variáveis de tempo de conclusão da tarefa, com variáveis de rede, com variáveis de atribuição de posição e com variáveis de ordenação linear. Serão revistos diferentes tipos de sequenciamento, tais como o sequenciamento semi-ativo, ativo e não atrasado. Os métodos de aproximação apresentados estão divididos em métodos construtivos ou heurísticos e os meta-heuristicos. No final deste capítulo, serão apresentadas diferentes formas de representação gráfica, tais como o mapa de Gantt e o grafo disjuntivo.

No quarto capítulo descrevem-se as experiências computacionais realizadas, analisando e comparando os resultados obtidos. Numa primeira fase será apresentado o modelo matemático de programação linear, sendo testado e comparado com o método de *Shiffting Bottleneck*, para um problema de três tarefas a serem processadas em três máquinas. Numa segunda fase serão apresentadas e comparadas instâncias de teste, reconhecidas na literatura, com um grau de

complexidade maior, com o objetivo de testar o modelo matemático de programação linear e comparar com alguns métodos de aproximação.

No quinto capítulo apresentam-se as conclusões obtidas com a realização deste estudo e as perspetivas de desenvolvimento futuro.

Capítulo 2 - Problemas de Escalonamento de Tarefas de Produção

2. Problemas de escalonamento de tarefas de produção

Neste capítulo clarifica-se o que é um problema de escalonamento de tarefas de produção em ambiente *job shop*, apresentam-se as suas medidas de desempenho, tais como, o *makespan*, o tempo de fluxo total, o atraso máximo, a soma dos atrasos, a soma dos atrasos e avanços, o número de tarefas atrasadas e o *lateness*. Serão apresentadas outras variantes do problema, tais como *job shop* flexível, o *flow shop* e o *job shop* com tempos de preparação dependentes da sequência.

2.1 Introdução

No atual ambiente competitivo, o sequenciamento tornou-se numa necessidade para a sobrevivência no mercado. As empresas devem esforçar-se ao máximo para cumprir as datas acordadas com os seus clientes, caso contrário pode resultar numa perda significativa para a imagem da empresa (Pinedo, 2008). Além disso, um bom sequenciamento pode proporcionar ganhos temporais relevantes (*lead time*) para concluir um conjunto de tarefas de produção e isso pode originar ganhos financeiros significativos para uma empresa, não sendo necessário recorrer a horas extra, dias de descanso e/ou a mais recursos.

Um problema de sequenciamento da produção é a decisão a ser tomada sobre a ordem em que as tarefas serão executadas, sendo que a programação da produção envolve a consideração de uma série de elementos que disputam vários recursos por um período de tempo, recursos esses que possuem capacidade limitada, segundo Morton & Pentico (Cit. por Pacheco & Santoro, 1999).

Num problema de sequenciamento da produção o objetivo fundamental consiste em afetar os recursos de produção aos trabalhos ou operações que deverão ser realizadas, segundo uma determinada ordem, ou o procedimento contrário, isto é, atribuir as tarefas aos recursos.

O sequenciamento pode ser entendido como a atribuição de recursos no tempo para o processamento de um conjunto de tarefas (Baker, 1974).

Além disso, o problema de sequenciamento consiste em encontrar uma sequência de passagem das tarefas pelos recursos, correspondendo a um plano

exequível e ótimo em relação a um qualquer critério de otimização adotado (French, 1982). O sequenciamento é visto como a definição de tempos de início e de término e a atribuição dos recursos a cada tarefa de um dado conjunto, obedecendo às várias restrições dos recursos e ou tarefas, segundo Portman em 1997 (Cit. por Pereira, 2014). Esses recursos podem ser as máquinas num ambiente fabril, as pistas nos aeroportos, os quartos em hotéis, os professores numa escola.

Os objetivos da programação e sequenciamento de produção são aumentar a utilização de recursos, reduzir o *stock* em processo e o atraso na entrega dos trabalhos, segundo Martins (1993) (Cit. por MELO, ANTÔNIO & FILHO, 2006). Os principais critérios de otimização da produção são três: os tempos de processamento, as datas de entrega e os custos de armazenamento e utilização. O critério de otimização mais importante é a satisfação das datas de entrega, seguindo-se a maximização da utilização das máquinas do sistema, a minimização dos materiais em curso de fabrico e dos tempos de preparação e mudança de ferramenta e a maximização da produtividade (Smith et al., 1986).

Devido à complexidade exponencial o problema de sequenciamento *job shop* é incluído numa vasta gama de problemas numéricos intratáveis, referenciado como NP-difícil (Lawlar et al., 1993).

2.2 Problema *Job Shop*

Neste trabalho ir-se-á focar num problema de escalonamento de tarefas de produção designado por *job shop*.

Como podemos ver na figura 1, os sistemas do tipo *job shop* podem ser definidos como sendo o caso de *n* ordens de fabrico para processar em *m* máquinas, na qual cada ordem de fabrico possui um conjunto de operações (gama operatória) que poderão ser processadas obedecendo a uma ordem especifica ou podendo ser processadas aleatoriamente sem qualquer implicação no resultado final.

Sequenciar um processo *job shop* é a tarefa de atribuir cada operação a uma posição específica na escala temporal da respetiva máquina (Conway, 1967).

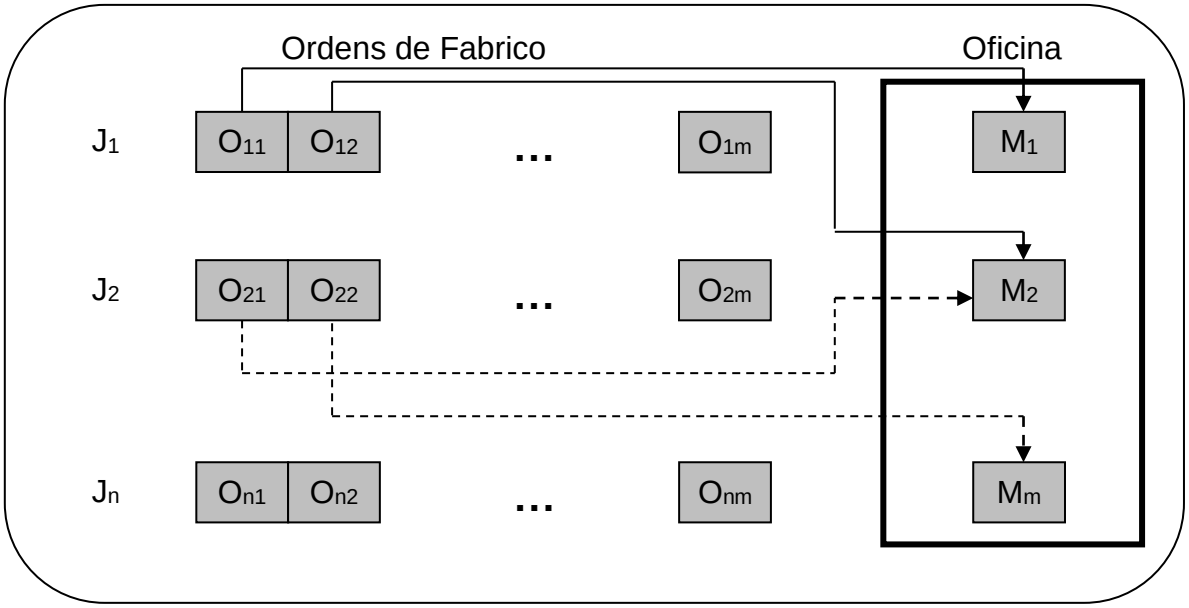


Figura 1 - Ilustração de um sequenciamento *job shop* (adaptado de Beirão, 1997).

Para a resolução dos problemas de sequenciamento são estabelecidos os seguintes pressupostos e condições (Beirão, 1997):

1. Todas as máquinas estão sempre disponíveis para fabrico e nunca se verificam avarias ou paragens.
2. Existe apenas uma máquina de cada tipo na oficina.
3. Uma operação só pode ser processada por uma máquina, não existindo caminhos alternativos.
4. No decorrer do fabrico de uma operação não pode haver interrupções, isto é, a operação tem de ser completada antes de se iniciar outra na mesma máquina.
5. Não podem existir sobreposições de operações pertencentes à mesma ordem de fabrico. Uma operação só pode ser processada após a conclusão da operação precedente.
6. Cada máquina apenas processa uma operação em cada momento.
7. As máquinas são os únicos recursos existentes.

Como exemplo de um sequenciamento *job shop* temos a Figura 1, sendo J_i as ordens de produção, em que *i* representa o número da ordem de produção de 1 até *n*, cada uma constituída por diferentes operações O_{ij}, em que *i* representa o número da ordem de produção e *j* a ordem com que devem ser processadas nas diferentes máquinas M_i, em que *i* representa o número da máquina de 1 até *m*.

O *job shop* é um problema NP-difícil, o que significa que um algoritmo ótimo para o resolver requer um número de passos que cresce exponencialmente com a dimensão dos dados do problema.

2.3 Medidas de desempenho do sistema

As medidas de desempenho permitem avaliar um determinado sequenciamento em função de um ou mais critérios.

As medidas de desempenho que são mais usadas na função objetivo de um problema de sequenciamento são o *makespan*, o tempo de fluxo total, o atraso máximo, a soma dos atrasos, a soma dos atrasos e avanços, o número de tarefas atrasadas e o *lateness* (Arenales, 2011). De seguida serão analisadas estas medidas.

2.3.1 Minimização do *makespan*

O *makespan* é simbolizado por C_{max} e corresponde ao instante final de processamento de todas as ordens de produção em todas as máquinas, ou ao tempo em que a última tarefa deixa o sistema.

Sendo C_i o instante de término de processamento da tarefa *i*, para *i=1, …, n*, então:

$$C_{\max} = \max\{C_i : i = 1, \ldots, n\} \quad (1)$$

Sob este ponto de vista, para o melhor desempenho do sistema, a função objetivo a usar é:

$$\min C_{max} \quad (2)$$

2.3.2 Minimização do tempo de fluxo total

O tempo de fluxo total é igual à soma dos tempos de término C_i de todas as tarefas.

Para esta medida de desempenho, pretende-se uma função objetivo do tipo:

$$\min \sum_{i=1}^{n} C_i \tag{3}$$

2.3.3 Minimização do atraso máximo (*tardiness*)

O atraso máximo, designado por T_{max}, corresponde à tarefa com maior diferença T_i entre o instante de término C_i e a data de entrega d_i, quando a tarefa é terminada após a data de entrega.

Neste contexto, tem-se:

$$T_{\max} \geq T_i \qquad i = 1, \dots, n \tag{4}$$

$$T_i \geq C_i - d_i \qquad i = 1, \dots, n \tag{5}$$

$$T_i \geq 0 \tag{6}$$

$$\text{ou seja, } T_i = \max\{0;\ C_i - d_i\} \tag{7}$$

Em que, T_i será igual ao maior valor positivo, não podendo ser menor que zero, pois isso significaria adianto em vez de atraso.

Logo a função objetivo a usar é:

$$\min T_{\max} \tag{8}$$

2.3.4 Minimização da soma dos atrasos

A função objetivo consiste em determinar a menor soma de todos os atrasos T_i, ou seja:

$$\min \sum_{i=1}^{n} T_i \tag{9}$$

2.3.5 Minimização da soma dos atrasos e avanços

A função objetivo consiste em determinar a menor soma de todos os atrasos T_i e avanços E_i, ou seja:

$$\min \sum_{i=1}^{n} (T_i + E_i) \tag{10}$$

Onde, os avanços E_i são tais que:

$$E_i \geq d_i - C_i \qquad i = 1, \ldots, n \tag{11}$$

$$E_i \geq 0 \tag{12}$$

ou seja, $E_i = \max\{0; d_i - C_i\}$ (13)

2.3.6 Minimização do número de tarefas atrasadas

A função objetivo consiste em minimizar o número de tarefas atrasadas e pode ser formulado da seguinte forma:

$$\min \sum_{i=1}^{n} y_i \tag{14}$$

onde,

$$y_i = \begin{cases} 1 & se\ a\ tarefa\ i\ está\ atrasada \\ 0 & caso\ contrário \end{cases}$$

Note-se que se $T_i > 0$, implica que $y_i = 1$.

2.3.7 Minimização do *lateness* máximo

O *lateness* (L_i) é a diferença entre a data de término de uma tarefa e o prazo de término. Se for positivo indica um atraso na entrega e se for negativo significa uma antecipação ao prazo de entrega.

Sendo $L_{max} = max\ L_i$, o *lateness* máximo, a formulação do problema pode ser apresentada da seguinte forma:

$$\min L_{\max} \tag{15}$$

$$L_{\max} \geq L_i \qquad i = 1, \ldots, n \tag{16}$$

$$L_i = C_i - d_i \qquad i = 1, \dots, n \tag{17}$$

Note-se que a diferença entre *o lateness* L_i e o *tardiness* T_i é que o *lateness* pode ser positivo ou negativo indicando um atraso ou uma antecipação, enquanto o T_i não pode ser negativo, isto é, ou tem atraso ou não tem atraso.

2.4 Variantes do *Job Shop*

Consoante o processo industrial em questão existem diversas variantes, devido às condicionantes próprias de cada processo industrial, tais como:

- *Job shop* com número distinto de operações por tarefa;

É o caso em que cada tarefa é processada por um subconjunto de *m* máquinas disponíveis.

- *Job shop* com interrupção (preemption)

Existem situações em que uma tarefa com maior prioridade torna-se disponível para processamento, neste caso a tarefa menos importante é interrompida.

- *Job shop* com instantes de disponibilidade distintos

Nem sempre as tarefas estão disponíveis para serem processadas no mesmo instante (*ready times*).

- *Job shop* flexível

Determinada operação pode ser executada por qualquer máquina, de um conjunto pré-definido de máquinas, permitindo vários roteiros para processamento de uma tarefa.

- *Job shop* com tempos de preparação dependentes da sequência

Há vários casos em que a sequência influencia os tempos de Setup, das máquinas, podendo-se ter ganhos ou perdas de tempo significativas.

- *Job shop* com *buffers*

Quando existem tarefas que para serem processadas têm tempos de espera, normalmente aguardam num *buffer* de capacidade limitada.

- *Job shop* com reentrada

Esta situação ocorre quando uma tarefa que já passou pelo ambiente de produção volta a entrar para um novo conjunto de operações.

- Restrições de precedência

Determinada tarefa só pode iniciar o seu processamento após o término de outras tarefas precedentes.

De seguida ir-se-á descrever mais pormenorizadamente algumas destas variantes.

2.4.1 Problema *Job Shop* Flexível

O *job shop* envolve um conjunto de tarefas, cada uma formada por uma sequência de operações. Cada uma dessas operações deve ser sequenciada numa máquina determinada previamente e com algum critério de otimização.

No *job shop* flexível, a partir de agora designado por JSF, cada operação pode ser processada em qualquer máquina de um conjunto predefinido de máquinas, daí a sua denominação de flexível. A flexibilidade do JSF faz com que uma tarefa possa ser processada por caminhos diferentes através das máquinas, aumentando consideravelmente o número de soluções possíveis.

Os JSF podem ser classificados com flexibilidade total ou parcial, em função do grau de flexibilidade das operações.Na flexibilidade total, cada operação pode ser processada por qualquer máquina, enquanto, na flexibilidade parcial, cada operação está associada a um subconjunto de máquinas.

Sendo considerado um problema NP-difícil, não são conhecidos algoritmos exatos que resolvam o JSF em tempo razoável. Por esse motivo, são usados métodos alternativos para gerar boas soluções em tempo aceitável (Melo et. al. 2012).

2.4.2 Problema *Flow Shop*

Nos problemas do tipo *flow shop*, todas as tarefas têm uma sequência semelhante para a realização das operações, ao longo das várias máquinas.

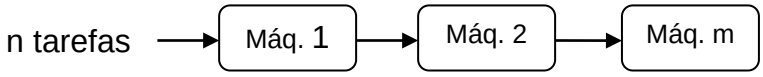


Figura 2 - Modelo *flow shop*.

O problema *flow shop* é um problema em que cada tarefa *i* consiste em *m* operações O_{ij} com tempos de processamento $p_{ij,}$ onde as operações O_{ij} devem ser processadas na máquina M_j, e há restrições de precedência da forma $O_{ij} \rightarrow O_{i,j+1}$, isto é, cada tarefa é processada primeiro na máquina 1, depois na máquina 2, depois na máquina 3, etc, como exemplifica a figura 2.

Poderão haver tarefas que não possuem a totalidade das *m* operações, sendo que nestes casos, a operação que não é necessário realizar será considerada com tempo de processamento nulo.

Qualquer uma das operações, quando iniciada, não poderá ser interrompida e todas as operações estarão disponíveis no seu instante inicial.

2.4.3 Problema *Job Shop* com tempos de preparação dependentes da sequência

Por norma, o tempo de *setup* de uma máquina abrange desde o final do processamento de uma tarefa até ao início da tarefa seguinte.

O tempo necessário para *setup* está relacionado com o grau de semelhança entre duas tarefas processadas sucessivamente numa mesma máquina. Isto é, se duas tarefas a serem processadas em sequência são semelhantes, o tempo necessário para o *setup* será relativamente menor. Se forem completamente diferentes, o tempo será proporcionalmente maior.

O tempo de intervalo em que a tarefa *j* ocupa a máquina é expresso por $S_{ij} + p_j$, onde *i* é a tarefa que precede *j* na sequência, S_{ij} é o tempo de *setup* necessário pela tarefa *j* após a tarefa *i* estar completada, e p_j é a quantidade de tempo necessário para completar a tarefa *j*.

Como exemplo, podemos considerar o planeamento de uma linha que produz quatro tipos de gasolina: *racing fuel, premium*, regular e sem chumbo. Num ciclo de produção completo, durante o qual um lote é dedicado a cada produto, a

quantidade de tempo improdutivo depende da sequência em que esses combustíveis são produzidos, ver Tabela 1.

Tabela 1 - Tempos de *setup*.

Produto		(1)	(2)	(3)	(4)
Racing	(1)	---	30	50	90
Premium	(2)	40	---	20	80
Regular	(3)	30	30	---	60
S/Chumbo	(4)	20	15	10	---

A quantidade total de tempo de *setup* difere em cada uma das seis sequências distintas que incluem todos os quatro produtos, conforme mostra a Tabela 2.

Tabela 2 - Soma dos tempos de *setup*.

Sequência	Tempo de Setup
1-2-3-4-1	30 + 20 + 60 + 20 = 130
1-2-4-3-1	30 + 80 + 10 + 30 = 150
1-3-2-4-1	50 + 30 + 80 + 20 = 180
1-3-4-2-1	50 + 60 + 15 + 40 = 165
1-4-2-3-1	90 + 15 + 20 + 30 = 155
1-4-3-2-1	90 + 10 + 30 + 40 = 170

2.4.4 Outras variantes

Existem outros modelos, como demonstra a figura 3, tais como, modelo de máquina única, modelo de máquinas paralelas, *open shop*, *job shop* com múltiplas máquinas e *flow shop* com múltiplas máquinas.

O modelo de ***máquina única*** é o modelo mais simples, uma vez que as possibilidades de sequenciamento ficam condicionadas apenas às sequências das tarefas.

No modelo de ***máquinas paralelas*** estão disponíveis mais de uma máquina, normalmente idênticas, para as mesmas operações.

Nos modelos ***job shop e flow shop, com múltiplas máquinas***, existe em cada estágio de produção um conjunto de máquinas paralelas.

No modelo ***open shop*** não há fluxo definido ou devidamente especificado para as tarefas a serem processadas nas máquinas.

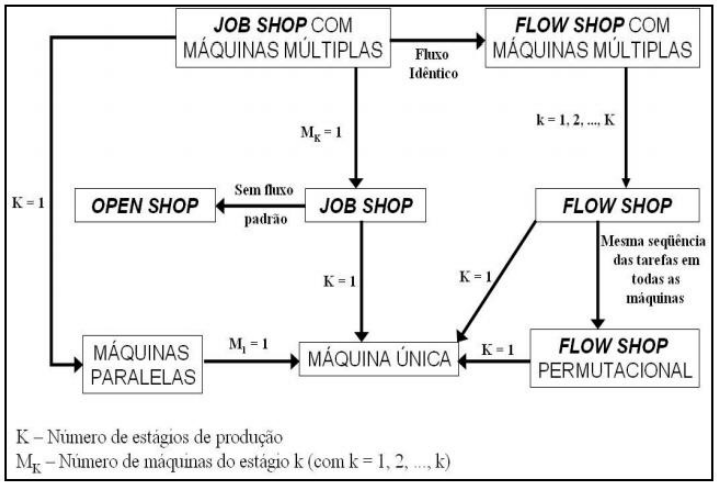


Figura 3 - Relação entre as classes de problemas de programação de operações em máquinas (MacCarthy & Liu, 1993).

2.5 Considerações finais

Neste capítulo apresentou-se o problema *job shop* e constatou-se que é um problema NP-dificil, ou seja, são problemas que não se conhecem um algoritmo de resolução eficiente devido à sua complexidade exponencial.

Foram apresentadas as medidas de desempenho mais usadas na função objetivo tal como, o *makespan* ou o tempo de fluxo de todas as tarefas. Concluiu-se que o *lateness* é diferente do *tardiness* na medida em que o *lateness* pode ser positivo ou negativo, representando um atraso ou uma antecipação ao prazo de entrega e que o *tardiness* ou é nulo ou positivo, indicando atraso ou não da tarefa.

Verificou-se que existem diversas variantes em função do contexto ou processo industrial, tais como, *job shop* com número distinto de operações por tarefa, *job shop* com interrupção permitindo priorizar uma tarefa mais importante, *job shop* flexível em que uma operação pode ser executada em mais de uma máquina, *job shop* com tempos de preparação dependentes da sequencia e outras variantes.

Capítulo 3 – Metodologias para a resolução do problema *job shop*

3. Metodologia para a resolução dos problemas *job shop*

Neste capítulo apresentam-se várias metodologias para a resolução de problemas do tipo *job shop*, através de métodos exatos e aproximados. Dos modelos exatos, serão revistas as formulações matemáticas, as formulações por programação inteira e o *Branch and Bound*. Serão apresentados os modelos matemáticos de otimização com variáveis de tempo de conclusão da tarefa, com variáveis de rede, com variáveis de atribuição de posição e com variáveis de ordenação linear. Serão revistos diferentes tipos de sequenciamento, tais como o sequenciamento semi-ativo, ativo e não atrasado. Os métodos de aproximação apresentados estão divididos em métodos construtivos ou heurísticos e os meta-heuristicos. No final deste capítulo, serão apresentadas diferentes formas de representação gráfica, tais como, o mapa de Gantt e o grafo disjuntivo.

3.1 Introdução

Ao longo dos tempos foram-se desenvolvendo novas metodologias para a resolução dos problemas *job shop*.

Em 1999, Jain & Meeran apresentaram um artigo contendo todas as metodologias de abordagem ao problema, como podemos ver na Figura 4 e Figura 5.

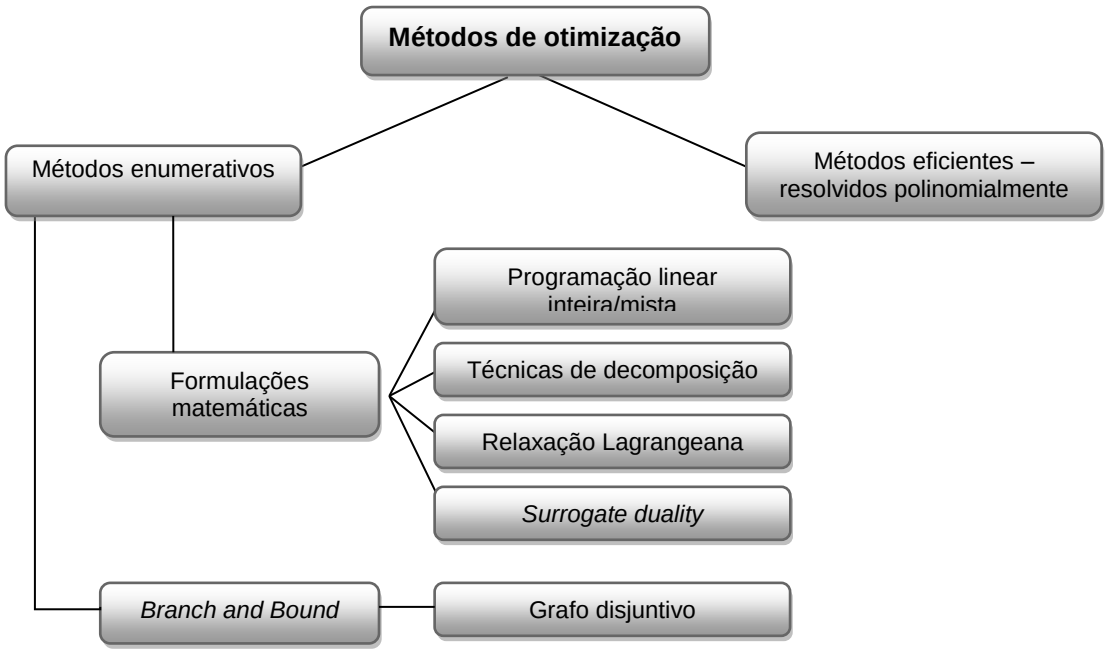


Figura 4 - Métodos de otimização (adapatado de Jain & Meeran, 1999).

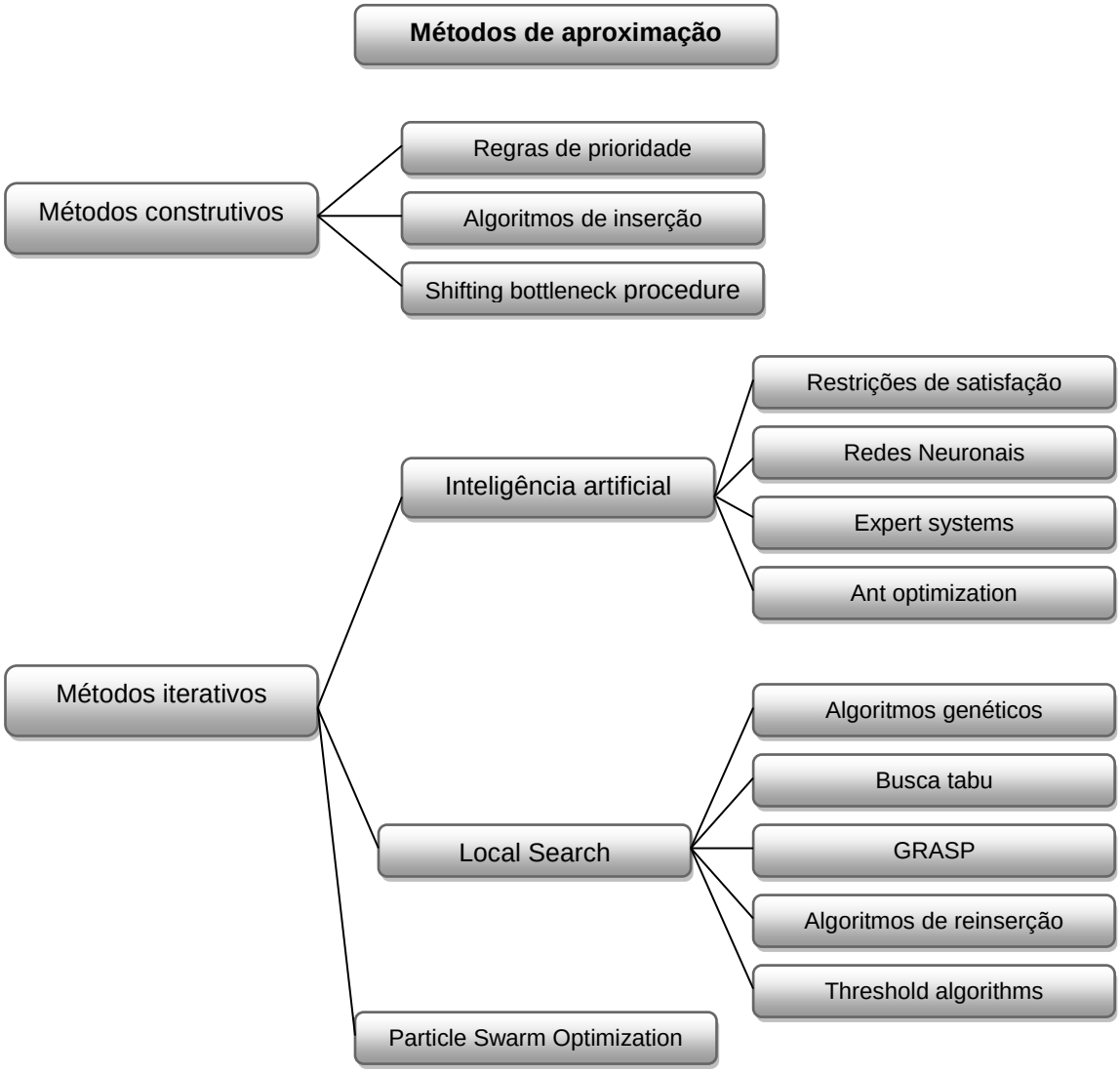


Figura 5 - Métodos de aproximação (adapatado de Jain & Meeran, 1999).

Os métodos de otimização, apresentados na figura 4, estão divididos em métodos enumerativos e métodos eficientes.

Entre os métodos enumerativos temos as formulações matemáticas, que podem ser resolvidas através da programação linear, técnicas de decomposição, relaxação Lagrangeana, *Surrogate duality* ou da árvore de pesquisa *Branch and Bound* e apresentados no grafo disjuntivo.

Os métodos de aproximação, apresentados na figura 5, estão divididos em métodos construtivos ou heurísticos, nomeadamente as regras de prioridade, algoritmos de inserção e o *Shifting Bottleneck Procedure,* e os métodos iterativos ou meta-heuristicos, através da inteligência artificial, *Local Search* e o *Particle Swarm Optimization*.

Nesta secção serão apresentados mais pormenorizadamente alguns destes métodos.

3.2 Métodos exatos

3.2.1 Formulações matemáticas

A programação matemática é um conjunto de técnicas para otimizar uma função cujas variáveis independentes estão sujeitas a restrições (French, 1982). Os problemas de sequenciamento podem ser resolvidos com técnicas de programação matemática.

A programação linear é um ramo da Matemática que estuda formas de resolver problemas de otimização cujas condições podem ser expressas por inequações lineares, isto é, inequações do primeiro grau.

Num problema de otimização o objetivo é encontrarmos a melhor solução, como por exemplo, a que dá menor prejuízo, maior lucro, a mais eficiente, etc.

Um problema de programação linear com apenas duas variáveis pode ser resolvido graficamente, representando as soluções de cada uma das inequações por um semiplano e de seguida procurando o ponto do polígono obtido (região admissível) que corresponde à solução ótima.

Um modelo de programação linear é composto por:

- Variáveis de decisão

$$x_1, \dots, x_n \tag{18}$$

- Função objetivo

$$Z = f(x_1, \dots, x_n) \tag{19}$$

- Restrições

$$R_k\ (x_1, \dots, x_n)\ \{\leq, =, \geq\}\ b_k \tag{20}$$

Num modelo de programação linear a função objetivo e as restrições são funções lineares das variáveis de decisão.

De uma forma genérica um modelo de programação linear consiste em determinar os valores das variáveis de decisão $x_1, x_2, \ldots, x_n$, que possibilitam a maximização ou minimização da função objetivo.

$$\max(\min) Z = c_1 x_1 + c_2 x_2 + \cdots c_n x_n \tag{21}$$

Sujeito a restrições funcionais do tipo:

$$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n \{\leq, =, \geq\} b_1 \tag{22}$$

$$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n \{\leq, =, \geq\} b_2 \tag{23}$$

$$\vdots$$

$$a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n \{\leq, =, \geq\} b_m \tag{24}$$

E as restrições de não negatividade:

$$x_1 \geq 0, \ldots, x_n \geq 0 \tag{25}$$

Tendo os seguintes parâmetros conhecidos:

Custos:

$$c_1, \ldots, c_n$$

Coeficientes técnicos:

$$a_{11}, \ldots, a_{mn}$$

Termos independentes:

$$b_1, \ldots, b_m$$

Quando se está a formular um problema de programação linear, deve-se definir as variáveis de decisão, a função objetivo, os recursos limitadores que devem ser satisfeitos (restrições).

Alguns exemplos onde se pode implementar um modelo de programação linear, para se obter a melhor decisão são:

- "*Mix*" de produção
- Misturas
- "*Scheduling*"
- Transportes
- Afetação

3.2.2 Formulações por programação inteira

A programação linear inteira para resolver este tipo de problemas é sugerida por alguns autores, (Wagner, 1959), (Conway et al., 1967), (Baker, 1974), (French, 1982) e consiste na formulação de um modelo matemático constituído por um conjunto de restrições, uma função objetivo e constituído por variáveis de decisão inteiras e binárias.

A resolução destes problemas só é possível em pequena escala com uma complexidade menor, uma vez que os atuais computadores ainda são bastante limitados para resolver problemas de grande escala, podendo demorar dias ou mesmo anos em muitos casos.

3.2.3 Branch and Bound

Os algoritmos de *Branch and Bound* tiveram a sua origem nos anos 60 com os trabalhos de Brooks & White (1965) e Ignall & Schrage (1965), tendo sido postos em prática nos problemas *job shop* por Balas (1969) (cit. por Jain & Meeran, 1998).

Este método baseia-se numa enumeração das soluções candidatas a solução ótima inteira de um problema, em que são efetuadas sucessivas partições ao longo da árvore de pesquisa considerando limites superiores e inferiores ao longo da enumeração, como exemplifica a figura 6.

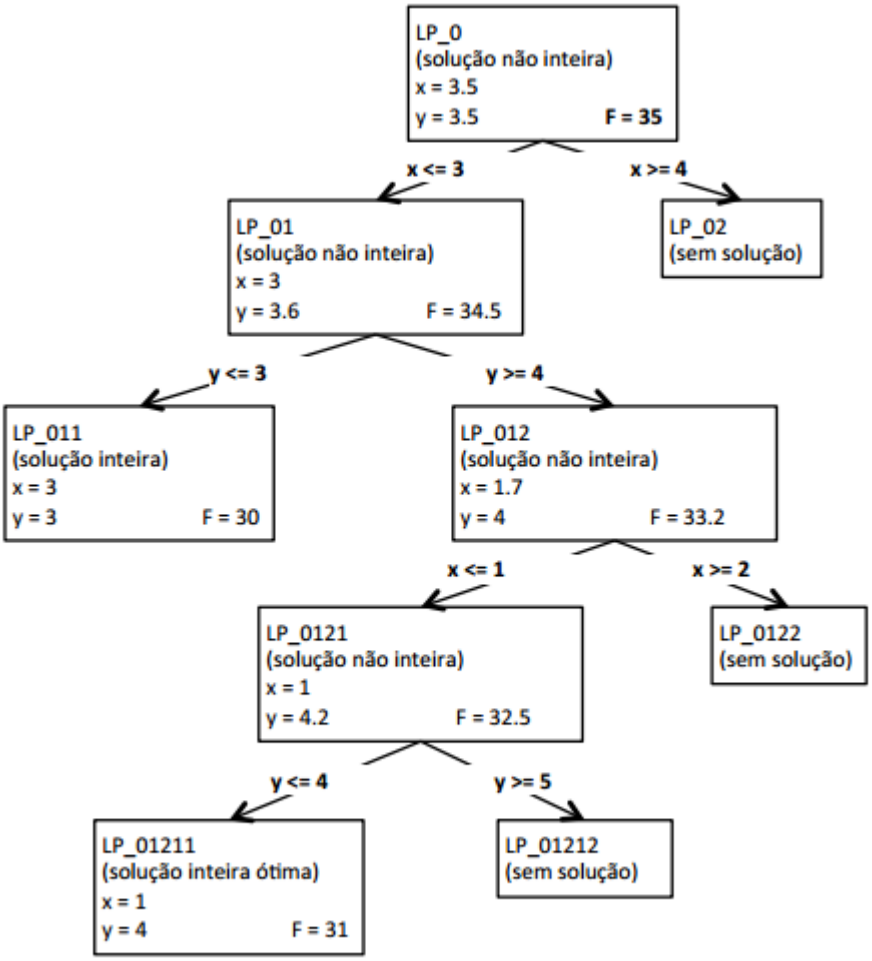


Figura 6 - Árvore de pesquisa do *Branch and Bound* (Carravila & Oliveira, 2012).

3.3 Modelos matemáticos para o problema *job shop*

Os problemas de sequenciamento podem ser resolvidos com técnicas de programação matemática. O modelo de programação linear inteira de Manne (1960), é uma das formas mais comuns de formulação matemática para o *job shop*.

A evolução da tecnologia tem permitido um desenvolvimento no estudo dos problemas de programação de tarefas no ambiente job shop, que até então não era possível com a mesma eficiência.

Os modelos para os problemas de sequenciamento podem ser classificados com base no que representam as variáveis de decisão: o tempo de conclusão da tarefa, atribuição da posição, ordenação linear entre as tarefas, variáveis indexadas pelo tempo e variáveis de rede. Existem vários tipos diferentes de variáveis de

decisão que foram usadas por investigadores para modelar e otimizar problemas de sequenciamento de produção. Unlu et al. (2010) analisou as diversas formulações de programação inteira mista em máquinas paralelas, sendo que algumas delas serão apresentadas e discutidas nas secções seguintes.

3.3.1 Modelo de otimização com variáveis de tempo de conclusão da tarefa

Pretende-se rever um modelo de programação linear para um *job shop* clássico, que se define por ser um ambiente de produção com *n* tarefas e *m* máquinas, onde cada tarefa pode ser processada nas *m* máquinas.

O tempo de conclusão da tarefa é uma métrica chave na avaliação da qualidade de uma programação de produção proposto. Na verdade, uma grande maioria das medidas de desempenho do problema de sequenciamento é uma função dos tempos de conclusão do trabalho. Variáveis de tempo de conclusão, que são por vezes referidas como "variáveis de data natural" (Queyranne & Schulz, 1994), foram utilizadas por Balas (1985) na sua formulação disjuntiva do problema de sequenciamento *job shop*. Mais tarde, a formulação inicial de Balas (1985) foi estudada por Queyranne e Wang (1991) e Queyranne (1993) (cit. por Unlu et al., 2010).

De seguida apresentam-se os parâmetros e a denominação de cada um deles, usados na formulação, admitindo que as *n* tarefas estão disponíveis para processamento no instante zero e que a interrupção de processamento de cada tarefa não é permitida.

p_{ik} – tempo de processamento da tarefa *i* na máquina *k*.

i(1), …, i(m) – roteiro de processamento da tarefa *i* nas *m* máquinas, ou seja, a sequência de máquinas em que as operações dessa tarefa são processadas.

M – número suficientemente grande.

Onde, as variáveis são:

C_{ik} – instante de término de processamento da tarefa *i* na máquina *k*.

$$x_{ijk} \begin{cases} 1 \; se \; a \; tarefa \; i \; precede \; a \; tarefa \; j \; na \; máquina \; k \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad caso \; contrário \end{cases}$$

A função objetivo consiste em minimizar o tempo de fluxo total das tarefas e pode ser formulada da seguinte forma:

$$\min \sum_{i=1}^{n} C_{i(m)} \tag{26}$$

Sujeito às seguintes restrições:

$$C_{i,i(1)} \geq p_{i,i(1)}\,, \qquad i = 1, \dots, n \tag{27}$$

$$C_{i,i(k+1)} \geq C_{i(k)} + p_{i,i(k+1)}\,, \qquad i = 1, \dots, n, \qquad k = 1, \dots, m-1 \tag{28}$$

$$C_{jk} \geq C_{ik} + p_{jk} - M\big(1 - x_{ijk}\big), \qquad i = 1, \dots, n, \qquad j = 1, \dots, n \;\; k = 1, \dots, m \tag{29}$$

$$C_{ik} \geq C_{jk} + p_{ik} - M x_{ijk}\,, \qquad i = 1, \dots, n, \qquad j = 1, \dots, n \;\; k = 1, \dots, m \tag{30}$$

A restrição (27) garante que a primeira operação de cada tarefa i seja terminada após o respetivo tempo de processamento.

A restrição (28) garante que a operação *k+1* da tarefa *i* seja concluída depois do término da operação *k* e do tempo de processamento da operação *k+1*.

As restrições (29) e (30) são disjuntivas indicando se na máquina *k*, a tarefa *i* precede a tarefa *j*, ou vice-versa. Por exemplo, se x_{ijk}= 1 a restrição (29) obriga a que o término da tarefa *j* seja posterior ao término da tarefa *i* mais o tempo de processamento da tarefa *j*, enquanto que a restrição (30) está desativada devido a *M* ter um valor arbitrariamente grande. Caso x_{ijk}= 0, de forma análoga a restrição (29) é desativada sendo apenas ativada a restrição (30).

3.3.2 Modelos de otimização com variáveis indexadas de tempo

Variáveis indexadas pelo tempo, que normalmente atribuem tarefas para períodos de tempo, são as variáveis de decisão indexadas por ambas as tarefas e dimensões de tempo que foram usadas numa variedade de problemas de sequenciamento de máquinas. Variáveis indexadas pelo tempo para o sequenciamento de máquinas foram introduzidas por Sousa e Wolsey (1992), que estudam o caso uma máquina única. Mais tarde, problemas diferentes numa variedade de ambientes de máquina única são estudados por Blazewicz et al.

(1991), Sousa & Wolsey (1992), Chan, Kaminsky, Muriel & Simchi-Levi (1995), Van den Akker, Hurkens e Savelsbergh (2000), Van den Akker, Van Hoesel e Savelsbergh (1999) & Šoric (2000) (cit. por Unlu et al. 2010).

Na formulação do modelo com variáveis indexadas pelo tempo, um horizonte de tempo é discretizado em períodos de tempo 1, …, *l*, onde *l* é um limite superior de tempo de término da última tarefa (*makespan*).

Sejam os conjuntos definidos da seguinte forma:

J conjunto de tarefas, *j=1, …, n*

μ conjunto de máquinas, *i=1,…, m*

T conjunto de tempos, *t=1, …,l*

E os parâmetros:

d_j data de entrega da tarefa *j*

r_j data de lançamento da tarefa *j*

w_j peso ou prioridade da tarefa *j*

p_j^i tempo de processamento da tarefa *j* na maquina *i*

l limite superior para o *makespan* (tempo total necessário para processar todas as tarefas)

Variáveis de decisão:

C_j Tempo de término da tarefa *j*

L_j *Lateness* da tarefa *j* ($L_j = C_j - d_j$)

T_j *Tardiness* da tarefa *j* ($T_j = max(0, C_j – d_j)$)

$$U_j = \begin{cases} 1 \; se \; a \; tarefa \; j \; está \; atrasada \\ 0 \qquad\qquad caso \; contrário \end{cases}$$

Onde,

$$x_{ij}^t = \begin{cases} 1 \; se \; a \; tarefa \; j \; inicia \; na \; máquina \; i \; no \; tempo \; t \\ 0 \qquad\qquad\qquad\qquad caso \; contrário \end{cases}$$

Sujeito às seguintes restrições:

$$\sum_{i\in\mu}\sum_{t=0}^{l-1} x_{ij}^{t} = 1 \;\; j \in J \tag{31}$$

$$\sum_{j\in J}\sum_{h=\max(0,t-p_j^i)}^{t-1} x_{ij}^{h} \leq 1 \;\; i \in \mu, \;\; t = 1, \dots, l \tag{32}$$

$$C_j \geq \sum_{i\in\mu}\sum_{t=0}^{l-1}(t + p_j^i)x_{ij}^{t} \;\; j \in J \tag{33}$$

A restrição (31) garante que uma tarefa não pode estar em mais de uma máquina no mesmo período de tempo.

A restrição (32) garante que uma tarefa só esteja em processamento numa máquina em qualquer momento.

Função objetivo para minimizar o tempo total de término ponderado:

$$\min \sum_{j\in J} w_j C_j \tag{34}$$

Sujeito às restrições (31) - (33).

Função objetivo para minimizar o total de atrasos com ponderação:

$$\min \sum_{j\in J} w_j T_j \tag{35}$$

Para a função objetivo, na equação (35) teremos que acrescentar a seguinte restrição às restrições (31) a (33):

$$T_j \geq C_j - d_j \;\; j \in J \tag{36}$$

Função objetivo para minimizar o maior atraso:

$$\min L_{\max} \tag{37}$$

Para a função objetivo na equação (37), a restrição seguinte deverá ser adicionada:

$$L_{\max} \geq C_j - d_j \quad j \in J \tag{38}$$

Sujeito às restrições (31) - (33) e (38).

Função objetivo para minimizar o número total de atrasos:

$$\min \sum_{i \in \mu} \sum_{j \epsilon J} \sum_{t=0}^{l-1} \frac{\max[0, t - d_j + p_j]}{\max[1, t - d_j + p_j]} x_{ij}^{t} \tag{39}$$

Sujeito às restrições (31) e (32).

3.3.3 Modelos de otimização com variáveis de rede

Rede ou '"variáveis do caixeiro-viajante" (Queyranne & Schulz, 1994) inicialmente foram usados para modelar problemas de máquina única com tempos de processamento dependente da sequência, como foi mostrado assemelhar-se a um problema do caixeiro-viajante dependente do tempo (TSP) (*traveling salesman problem*) (Houck & Marcelo, 1976). Relações de precedência para o mesmo problema, estudados por Queyranne & Schulz (1994), proporcionam uma análise poliédrica do modelo. Blazewicz et al. (1991) apresentam uma formulação de rede baseada em Miller, Tucker & Zemlin (1960) para problemas de sequenciamento de máquina única, estendendo-se também a formulação para considerar datas/tempos de liberação das tarefas. Cakici & Mason (2007) apresentaram uma formulação de MIP para sequenciamento de máquinas paralelas, na presença de restrições de recursos.

As formulações do modelo baseado em redes têm-se mostrado bastante úteis para modelar uma grande variedade de problemas. Enquanto o modelo de rede para máquina única pode ser considerado como um TSP (*traveling salesman problem*), um problema de máquinas paralelas está relacionado com um problema de roteamento de veículos capacitado quando as tarefas a serem agendadas são modeladas como clientes (nódos) e as máquinas representam os veículos a serem encaminhados (Unlu et al., 2010).

Deste modo temos a seguinte variável binária,

$$x_{lj}^{i} = \begin{cases} 1 & se\ a\ tarefa\ l\ \text{é}\ processada\ imediatamente\ antes\ da\ tarefa\ j\ na\ máquina\ i \\ 0 & caso\ contrário \end{cases}$$

Função objetivo para minimizar o tempo total de término ponderado:

$$\min \sum_{j \in J} w_j C_j \tag{40}$$

Sujeito às seguintes restrições:

$$\sum_{i \in \mu} x_{l0}^{i} \leq 1 \;\; l \in J : l \neq 0 \tag{41}$$

$$\sum_{i \in \mu} x_{0j}^{i} \leq 1 \;\; j \in J : j \neq 0 \tag{42}$$

$$\sum_{l \in J} \sum_{i \in \mu} x_{lj}^{i} = 1 \;\; j \in J : j \neq 0, l \neq j \tag{43}$$

$$\sum_{j \in J} \sum_{i \in \mu} x_{jl}^{i} = 1 \;\; l \in J : j \neq 0, l \neq j \tag{44}$$

$$\sum_{l \in J : l \neq j} x_{lj}^{i} - \sum_{l \in J : l \neq j} x_{jl}^{i} = 0 \; j \in J : j \neq 0, i \in \mu \tag{45}$$

$$C_l - C_j + \left(M - r_j\right) x_{lj}^{i} \leq M - (r_j + p_j^{i}) \tag{46}$$

As restrições (41) e (42) garantem que no máximo uma tarefa é agendada antes e depois da tarefa zero, respetivamente.

As restrições (43) e (44) garantem que todas as tarefas são agendadas numa máquina.

A restrição (45) garante que cada tarefa tem apenas um antecessor e um sucessor.

3.3.4 Modelos de otimização com variáveis de atribuição de posição

Formulações de MIP (programação inteira mista) contendo variáveis de atribuição de posição especificam qual é a próxima tarefa em cada máquina. Por exemplo, num problema de máquina única, *n* operações são atribuídas a *n* posições temporais disponíveis. Lasserre e Queyranne (1992) consideram um problema de máquina única como um sistema que pode ser controlado em instantes discretos no tempo usando uma combinação de controlos discretos e contínuos. Além disso, o problema de minimizar o número total de tarefas em atraso numa única máquina foi estudado por Dauzère-Peres (1997) e Dauzère-Peres e Sevaux (2003) usando variáveis de atribuição de posição (cit. por Unlu et al., 2010).

Num modelo de atribuição da posição, as variáveis de decisão são definidas com base na noção que cada máquina tem um número fixo de posições em que as tarefas podem ser atribuídas. Estas posições constroem posições relativas específicas para todas as tarefas processadas na mesma máquina e, portanto, a sequência da tarefa na máquina.

Deste modo temos as seguintes variáveis binárias,

$$u_{jl}^{i} = \begin{cases} 1 & se\ a\ tarefa\ j\ \text{é}\ atribuída\ \text{à}\ posição\ l\ da\ máquina\ i \\ 0 & caso\ contrário \end{cases}$$

$y_l^i = tempo\ de\ término\ da\ tarefa\ que\ está\ na\ posição\ l\ da\ máquina\ i.$

Função objetivo para minimizar o tempo total de término ponderado:

$$\min \sum_{j \in J} w_j C_j \tag{47}$$

Sujeito às seguintes restrições:

$$\sum_{i \in \mu} \sum_{l \in J} u_{jl}^{i} = 1 \quad j \in J \tag{48}$$

$$\sum_{j \in J} u_{jl}^{i} \leq 1 \quad l \in J : i \in \mu \tag{49}$$

$$\gamma_l^i \geq \sum_{j \in J} p_j^i u_{j1}^i \;\; i \in \mu \tag{50}$$

$$\gamma_l^i \geq \gamma_{l-1}^i + \sum_{j \in J} p_j^i u_{jl}^i \;\; i \in \mu, l \in J: l \geq 2 \tag{51}$$

$$C_j \geq \gamma_l^i - M\left(1 - u_{jl}^i\right) \;\; j \in J, l \in J, i \in \mu \tag{52}$$

A restrição (48) garante que todas as tarefas são atribuídas a uma posição numa única máquina.

A restrição (49) garante que cada posição em cada máquina contém no máximo uma tarefa.

O tempo de término da tarefa na posição *l* de cada máquina é determinado pelas restrições (50) e (51).

Finalmente, o tempo de término da tarefa *j* é dado pela restrição (52).

3.3.5 Modelos de otimização com variáveis de ordenação linear

A ordenação linear de tarefas pode ser pensada como o conjunto de todas as permutações de sequência (soluções). Desta forma, é definida uma variável de ordenação linear binária - é geralmente igual a um, se uma determinada tarefa sucede a outra, desse modo, descrevendo as relações de precedência entre todas as tarefas. Variáveis de ordenação linear são também referidas como “variáveis de sequenciamento” (Pinedo 2002). Dyer e Wolsey (1990) usam a ordenação linear em combinação com as variáveis de tempo nos seus problemas de máquina única estudados com datas de lançamento. Além de Dyer e Wolsey (1990) desenvolvendo várias abordagens poliédricas, essas variáveis são também estudadas por Blazewicz et al. (1991), Nemhauser & Savelsbergh (1992) e Chudak & Hochbaum (1999) (cit. por Unlu et al., 2010).

O modelo de ordenação linear é baseado em variáveis de ordem linear binária em que,

$$\delta_{lj} = \begin{cases} 1 & se\ a\ tarefa\ l\ precede\ a\ tarefa\ j \\ 0 & caso\ contrário \end{cases}$$

$$Z_{li} = \begin{cases} 1 & se\ a\ tarefa\ l\ está\ posicionada\ na\ máquina\ i \\ 0 & caso\ contrário \end{cases}$$

$$y_{lj} = \begin{cases} 1 & se\ a\ tarefa\ l\ e\ j\ não\ são\ executadas\ na\ mesma\ máquina \\ 0 & caso\ contrário \end{cases}$$

Função objetivo para minimizar o tempo total de término ponderado:

$$\min \sum_{j \in J} w_j C_j \tag{53}$$

Sujeito às seguintes restrições:

$$\delta_{lj} + \delta_{jl} + y_{lj} = 1 \quad l \in J, j \in J, l < j \tag{54}$$

$$\delta_{lj} + \delta_{jk} + y_{kj} \leq 2 \quad l \in J, j \in J, k \in J, l < j < k \tag{55}$$

$$z_{li} + z_{ji} + y_{lj} \leq 2 \quad l \in J, j \in J, l < j, i \in \mu \tag{56}$$

$$\sum_{i} z_{li} = 1 \quad l \in J \tag{57}$$

$$C_j \geq p_j^i z_{ji} \quad j \in J, i \in \mu \tag{58}$$

$$C_j \geq C_l + p_j^i \left(\delta_{ij} + z_{li} + z_{ji} - 2\right) - M\left(1 - \delta_{lj}\right) \; l \in J, j \in J, i \in \mu \tag{59}$$

A restrição (54) garante que a tarefa *l* está posicionada antes da tarefa *j* ou vice-versa, desde que as duas tarefas sejam agendadas na mesma máquina.

A restrição (55) representa a restrição de transição que assegura a ordenação linear entre três tarefas.

A restrição (56) permite calcular a variável y_{lj}.

A restrição (57) garante que cada tarefa está posicionada numa máquina.

Finalmente, as restrições (58) e (59) permitem determinar o tempo de término da tarefa.

3.4 Métodos de aproximação

A inexistência de métodos de otimização eficientes para resolver problemas de maior complexidade, como o *job shop*, em tempo útil conduz à inevitável aplicação de técnicas heurísticas para a resolução destes problemas. As técnicas heurísticas são um método desenvolvido para encontrar soluções para um problema, de uma forma simples e rápida, não garantindo as melhores soluções, apenas soluções viáveis.

Segundo Guimarães (2007), apesar da diversidade dos métodos de solução criados e da evolução dos computadores, muitos destes problemas ainda são considerados de difícil solução, devido à sua natureza combinatória. Num ambiente de produção do tipo *job shop* com *m* máquinas e *n* tarefas, existem $(n!)^m$ possibilidades de se executar as tarefas. Além disso, ao longo do tempo, os modelos criados passaram a incorporar novas características e restrições presentes nos ambientes de produção, o que dificulta ainda mais a sua resolução.

Esta situação levou ao desenvolvimento de vários métodos heurísticos para encontrar boas soluções, entre os quais o algoritmo *shifting bottleneck*, pesquisa *tabu* e algoritmos genéticos.

Vários estudos têm sido realizados ao longo dos últimos anos, mas na realidade a utilização do senso comum é a melhor forma de sequenciar quando o cenário é complexo (Bedworth, 1987).

3.5 Tipos de sequenciamento

Destacam-se três tipos ou classes de sequenciamento:

- Semi-ativo
- Ativo
- Não atrasado

De forma a exemplificar os diferentes tipos de sequenciamento, será de seguida apresentado na Tabela 3, um problema constituído por duas ordens de produção a serem processadas em duas máquinas.

Tabela 3 - Problema de sequenciamento com 2 ordens de produção e 2 máquinas.

Ordem de Produção	1ª Operação		2ª Operação	
	Máquina	Tempo	Máquina	Tempo
OP 1	M2	4	M1	2
OP 2	M1	1	M2	3

Como já referido, o número de soluções de um problema de sequenciamento é dado por $(n!)^m$, desta forma o número de soluções para este problema é igual a quatro. Sendo que, apenas três soluções são possíveis, pois uma delas viola as restrições de precedência.

A Figura 7 mostra as três soluções possíveis.

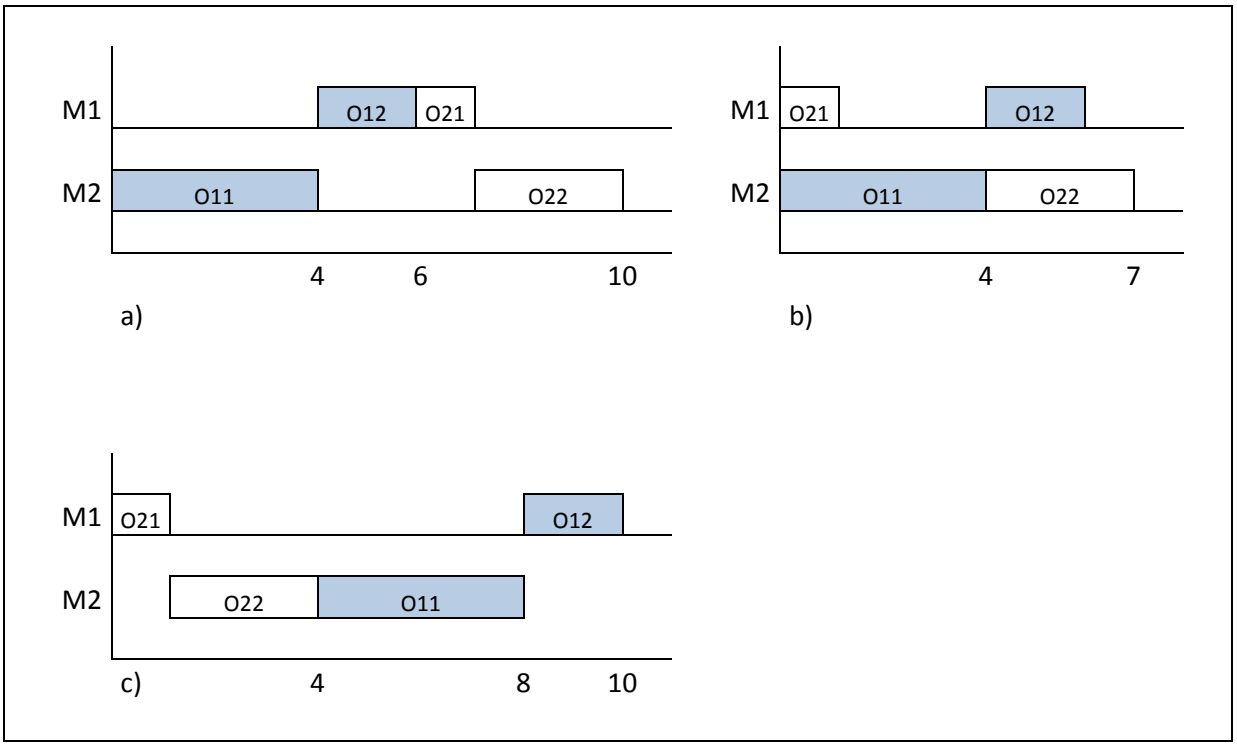


Figura 7 - Sequenciamentos possíveis para o problema.

3.5.1 Sequenciamento semi-ativo

Um sequenciamento é chamado de semi-ativo, se não tiver tempos de espera desnecessários. O sequenciamento semi-ativo garante que cada operação é iniciada o mais cedo possível, obedecendo às restrições de precedência. Desta forma, todos os sequenciamentos apresentados na Figura 7 são considerados semi-ativos.

3.5.2 Sequenciamento ativo

Num sequenciamento ativo nenhuma operação pode iniciar mais cedo sem provocar um atraso noutra operação ou sem violar as restrições de precedência. Os sequenciamentos ativos formam uma subclasse dos semi-ativos, ou seja, um sequenciamento ativo também é semi-ativo.

O sequenciamento a) da Figura 7 não é ativo, pois podemos mover a operação O_{21} para uma posição mais cedo sem provocar atrasos nas operações seguintes. Por sua vez, os sequenciamentos b) e c) são ativos, pois não é possível mover para a esquerda nenhuma operação sem provocar atrasos nas restantes operações.

3.5.3 Sequenciamento não atrasado

Neste tipo de sequenciamento, nenhuma máquina fica inativa quando pode iniciar o processamento de alguma operação. Este tipo de sequenciamentos formam uma subclasse dos ativos, ou seja, um sequenciamento não atrasado também é semi-ativo. O sequenciamento b) da Figura 7 é não atrasado, pois nenhuma máquina fica parada quando poderia processar outra operação (Beirão, 1997).

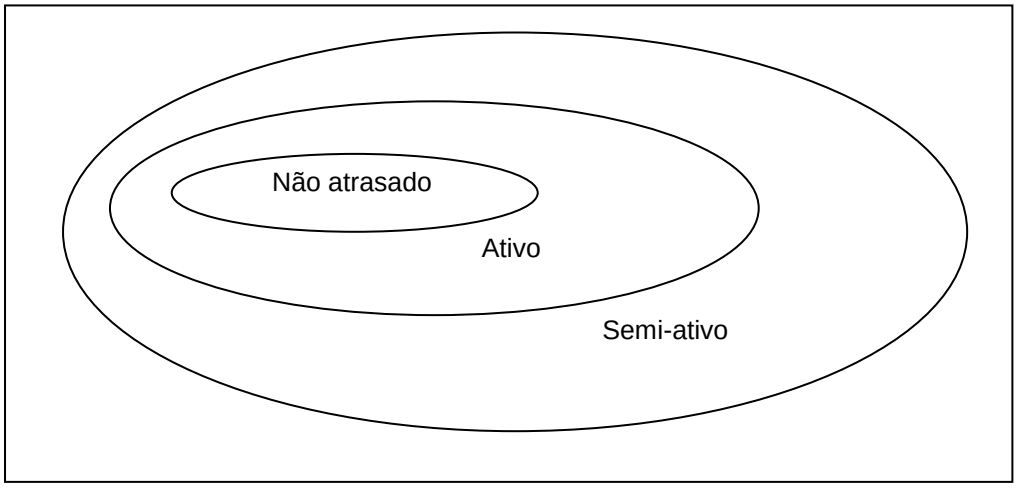


Figura 8 - Tipos de planos.

3.6 Métodos construtivos

3.6.1 Algoritmo de Johnson

A regra de Johnson é um algoritmo que minimiza o *leadtime* total de um conjunto de ordens processadas em dois recursos sucessivos (n tarefas em 2 recursos).

O problema pode ser resolvido da seguinte forma:

1. Encontrar o tempo de menor processamento de todas as tarefas.
2. Se o tempo de menor processamento ocorre na:
    a. Máquina 1 – Colocar a tarefa na primeira posição disponível da sequência (em situações de empate pode-se optar de forma aleatória). Seguir para o ponto 3.
    b. Máquina 2 – Colocar a tarefa na última posição disponível da sequência (em situações de empate pode-se optar de forma aleatória). Seguir para o ponto 3.
3. Remover a tarefa já sequenciada da lista de tarefas a sequenciar e recomeçar no ponto 1. Repetir até todas as tarefas estarem sequenciadas.

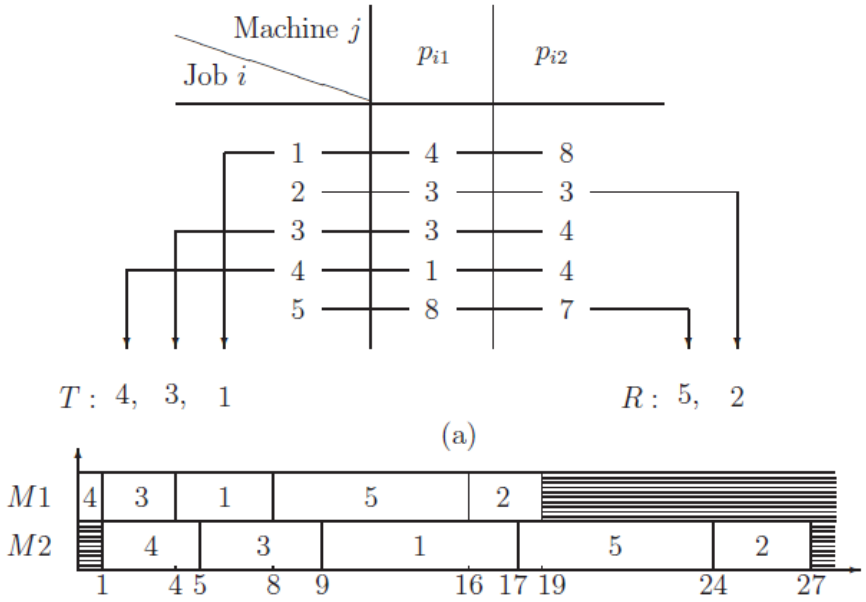


Figura 9 - Aplicação do algoritmo de Johnson.

3.6.2 Regras de prioridade

Contudo, as regras de prioridade são bastante antigas e utilizadas devido à fácil utilização e baixo tempo de processamento, sendo os primeiros trabalhos desenvolvidos por Jackson (1955), Smith (1956) e Giffler & Thompson (1960), onde este último serve de base a todas as regras de prioridade (Beirão,1997).

Na Tabela 4 são apresentadas algumas regras de prioridade mais comummente usadas.

Tabela 4 - Regras de prioridade.

Regra de prioridade	Descrição
PCO	"Preferred customer order". A ordem de produção (OP) de um cliente considerado prioritário é processada primeiro.
Aleatória	Seleciona aleatoriamente a operação para a máquina considerada.
SPT	"Shortest processing time". Prioriza a OP com menor tempo de processamento.
LPT	"Longest processing time". Prioriza a OP com maior tempo de processamento.
SOT	"Shortest operation time". Prioriza a operação com menor tempo de processamento na máquina considerada.
LOT	"Longest operation time". Prioriza a operação com maior tempo de processamento na máquina considerada.

SRPT	"Shortest remaining processing time". Prioriza a operação com o menor tempo restante de processamento da OP.
LRPT	"Longest remaining processing time". Prioriza a operação com maior tempo restante de processamento da OP.
FCFS	"First come first served". Prioriza a primeira operação da fila de espera da máquina.
LOS	"Longest operation sucessor". Prioriza a operação com maior tempo de processamento da operação subsequente.
SNRO	"Smallest number of remaining operations". Prioriza a operação com menor número de operações subsequentes.
LNRO	"Largest number of remaining operations". Prioriza a operação com maior número de operações subsequentes.
EDD	"Earliest due date". Prioriza as operações por ordem crescente das datas de entrega.
LWKR	"Least work remaining". Prioriza a OP com o menor valor da soma das durações das operações por realizar.
MWKR	"Most work remaining". Prioriza a OP com o maior valor da soma das durações das operações por realizar.
MS	"Minimum Slack". Prioriza a OP cujo tempo de folga até à data de entrega é menor.

3.6.3 Algoritmo de Giffler e Thomson

A heurística desenvolvida por Giffler e Thomson (1960) permite a geração de planos de sequenciamento do tipo ativos. Para descrever este algoritmo é necessário definir a notação e a terminologia que se irá utilizar. Deste modo, no algoritmo é sequenciado uma operação de cada vez. Uma operação pode ser sequenciada se todas as operações que a precedem na tarefa já foram sequenciadas. Se existirem *nm* operações o algoritmo terá *nm* iterações.

Na iteração *t*, tem-se a seguinte notação:

- P_t representa o sequenciamento parcial das (*t-1*) operações sequenciadas no estágio *t*.
- S_t representa o conjunto de operações sequenciáveis no estágio *t,* ou seja, todas as operações que têm de preceder as de S_t estão em P_t.
- σ_k representa o tempo mais cedo em que a operação O_k pertencente a S_t pode ser iniciada.
- Φ_k representa o tempo mais cedo em que a operação O_k pertencente a S_t pode ser concluída, ou seja, $\Phi_k = \sigma_k + p_k$ onde p_k é o tempo de processamento.

Os passos do algoritmo de Giffler e Thomson são os seguintes:

Passo 1 Seja $t = 1$ com $P_1 = \{\}$. S_1 será o conjunto de todas as operações sem predecessores, isto é, a primeira operação de cada ordem de produção.

Passo 2 Determinar $\Phi^* = \min \{ \Phi_k \}$ em $S_k \{ \sigma k \}$ e a máquina M^* na qual Φ^* ocorre.

Passo 3 Escolher uma operação o_j *em* S_t, que requeira a máquina M^* e em que $\sigma_j < \Phi^*$.

Passo 4 Mover para a próxima iteração realizando as operações.

(1) Adicionar o_j a P_t criando P_{t+1};

(2) Apagar o_j de S_t e criar S_{t+1} através da adição a S_t da operação que está imediatamente a seguir a o_j na ordem de produção. (excetua-se o caso em que o_j seja a última operação da ordem de produção);

(3) Incrementar t de 1.

Passo 5 Se existirem operações que ainda não foram sequenciadas, saltar para o passo 2, senão parar.

3.6.4 Algoritmo modificado de Giffler e Thomson

É possível alterar com facilidade o algoritmo de Giffler e Thomson de forma a gerar sequenciamentos não atrasados. O objetivo desta modificação consiste em que se houver uma operação para processamento e uma máquina disponível para o seu processamento, então procede-se ao inicio do fabrico, ou seja, existindo um recurso onde uma tarefa à espera pode ser processada nunca fica parado.

Os passos do algoritmo modificado de Giffler e Thomson são os seguintes:

Passo 1 Seja $t = 1$ com $P_1 = \{\}$. S_1 será o conjunto de todas as operações sem predecessores, isto é, a primeira operação de cada ordem de produção.

Passo 2 Determinar $\sigma^* = \min \{ \Phi_k \}$ em $S_k \{ \sigma k \}$ e a máquina M^* na qual σ^* ocorre. Se existir mais de um M^* no qual σ^* ocorre, escolher um deles aleatoriamente.

Passo 3 Escolher uma operação σ_j *em* S_t, que requeira a máquina M^* e em que $\sigma_j < \sigma^*$.

Passo 4 Mover para a próxima iteração realizando as operações.

(1) Adicionar o_j a P_t criando P_{t+1};

(2) Apagar o_j de S_t e criar S_{t+1} através da adição a S_t da operação que está imediatamente a seguir a o_j na ordem de produção. (excetua-se o caso em que o_j seja a última operação da ordem de produção);

(3) Incrementar *t* de 1.

Passo 5 Se existirem operações que ainda não foram sequenciadas, saltar para o passo 2, senão parar.

Como se pode observar no passo 3 de ambos os algoritmos, é escolhida uma operação do conjunto S_t, desta forma é necessário introduzir um mecanismo de escolha da operação baseado em alguma regra de prioridade de forma a permitir obter um sequenciamento ativo ou não atrasado.

3.6.5 Shifting Bottleneck Procedure

O método heurístico desenvolvido por Adams, Balas e Zawack (1988) é um método iterativo, baseado no grafo disjuntivo, que se fundamenta na decomposição do problema principal em sub-problemas de máquina única e cujo objetivo é ir melhorando a solução obtida. Em cada iteração determina-se as datas de disponibilidade e de entrega dos sub-problemas de forma a sequenciar as máquinas que formam o problema inicial.

Sendo *M* o conjunto de todas as máquinas do problema e $M' = \{ M_1 , \ldots, M_k \}$ o conjunto de todas as máquinas sequenciadas, desta forma pode-se considerar que $M \setminus M' = \{ M_{k+1}, \ldots, M_m \}$ é o conjunto de todas as máquinas ainda não sequenciadas.

Para cada máquina pertencente a $M \setminus M'$ é resolvido um problema de máquina única, baseando-se na data de disponibilidade e na data de entrega. Os valores das datas são definidos a partir do sequenciamento de operações nas máquinas do conjunto *M'.* A máquina cujo valor ótimo de *makespan* for superior é considerada a máquina crítica, ou seja, aquela que provoca o gargalo de produção. Após o sequenciamento da máquina crítica é resolvido um novo problema de máquina única, re-otimizando localmente todas as máquinas pertencentes a *M'*. O processo é repetido até que todas as máquinas de *M* estejam sequenciadas.

O funcionamento do algoritmo *shifting bottleneck* pode ser descrito da seguinte forma:

$M' = \{\}$

Passo 1 Encontrar a máquina *m* que provoca um gargalo entre as máquinas do conjunto $M \setminus M'$ e proceder ao seu sequenciamento ótimo.

$M' = M' \cup \{m\}$

Passo 2 Re-otimizar a sequência de cada máquina crítica $k \in M'$ mantendo as sequências das outras máquinas.

Se $M' \neq M$ voltar ao passo 1, senão parar.

3.7 Meta-heuristicas

As meta-heuristicas são um processo iterativo de pesquisa de melhores soluções na vizinhança, com o objetivo de encontrar um ótimo para o problema próximo do ótimo global.

3.7.1 Tabu Search

O procedimento básico de busca na vizinhança também é conhecido por ser uma técnica de descida, uma vez que cada nova semente representa um valor mais baixo da função objetivo, assumindo que o objetivo é minimizar. Se tivéssemos de representar graficamente o valor da função objetivo para a semente em função do número de sementes, o gráfico seria uma função decrescente. Num problema de grande dimensão, a redução pode ser rápida nas fases inicias da pesquisa, mas muito lenta no fim da busca.

Um dos problemas dos procedimentos de busca na vizinhança é a sua tendência para se tornar “preso” em ótimos locais correndo-se o risco de não ser o ótimo global. Por vezes é preferível tentar uma nova semente mesmo sendo pior que a anterior de forma a escapar ao ótimo local e encontrar a solução ótima global. A flexibilidade de poder passar para uma solução pior é uma característica da busca tabu. A principal característica da busca tabu é uso de memória, guardando informação do processo de pesquisa de soluções. Usando a memória é possível

abandonar ótimos locais consentindo uma deterioração no valor da qualidade de uma solução ainda não visitada. O algoritmo verifica se essa solução já foi visitada recorrendo a uma lista tabu que guarda informação referente às soluções visitadas. A lista tabu consiste numa lista com o objetivo de guardar soluções dos últimos movimentos. Quando uma solução está na lista tabu esse movimento não é permitido, evitando ciclos e permitindo a exploração de outros espaços de soluções. A lista tabu tem uma dimensão fixa e quando fica cheia, o mais antigo é removido. O critério de paragem é baseado num número pré-definido de iterações, momento em que o algoritmo finaliza (Baker & Triestch, 2009).

O funcionamento do algoritmo pode ser descrito da seguinte forma:

Passo 1 Seja $k = 1$
Selecionar uma sequência inicial *S1* usando uma heurística.
Seja $S_0 = S_1$

Passo 2 Selecionar um candidato S_c da vizinhança de S_k.
Se o movimento $S_k \rightarrow S_c$ é proibido por qualquer mutação na lista tabu, então $S_{k+1} = S_k$ e avançar para o passo 3.
Se o movimento $S_k \rightarrow S_c$ não é proibido por qualquer mutação na lista tabu, então $S_{k+1} = S_c$ e colocar no topo da lista tabu. Descer todas as outras uma posição e remover a ultima da lista tabu.
Se $F(S_c) < F(S_0)$, então $S_0 = S_c$
Avançar para o passo 3.

Passo 3 Incrementar K em 1.
Se $k = N$ então PARAR, se não ir para o passo 2.

3.7.2 Simulated Annealing

A origem do *Simulated Annealing* está situada na década de 1980 nos trabalhos de Kirkpatrick et al. (1983) e Cerny (1985) e faz analogia com o processo de fundição (*annealing*) de um sólido. Este algoritmo surgiu com base no processo de arrefecimento do metal fundido, sendo o seu arrefecimento lento de forma a solidificar numa estrutura cristalina forte. Trata-se de um algoritmo estocástico e permite a degradação da solução atual através de movimentos para soluções de pior qualidade, de forma a escapar ao mínimo local e atrasar a convergência do algoritmo. Numa fase inicial do processo iterativo, serão aceites a maioria dos movimentos, ou seja, são aceites todas as substituições da solução atual por uma nova da sua vizinhança, escolhida aleatoriamente permitindo explorar o espaço das soluções. Com o andamento do algoritmo, a temperatura vai diminuindo gradualmente e vai se tornando cada vez mais seletivo na aceitação de novas soluções.

Ao contrário da busca tabu, este algoritmo não tem memória e por isso não utiliza nenhuma informação adquirida ao longo do processo de pesquisa. O processo inicia com uma solução inicial *S*, que pode ser obtida aleatoriamente ou por uma heurística. Por cada iteração é selecionada aleatoriamente uma solução S_c pertencente à vizinhança de S_k. Se esta solução for melhor que a anterior será aceite como nova solução e se for pior será aceite com base numa probabilidade.

$$P = exp\ \frac{F(S_k) - F(S_c)}{T_k} \tag{60}$$

O arrefecimento é essencial para o desempenho do algoritmo, sendo efetuado de k em k iterações. É aplicado um fator de redução de temperatura de modo a reduzir a temperatura de forma constante. À medida que o algoritmo avança, a probabilidade de aceitação de movimentos para piores soluções vai diminuindo, como exemplifica a figura 10 (Pereira, 2014).

$$T_{k+1} = \alpha T_k \tag{61}$$

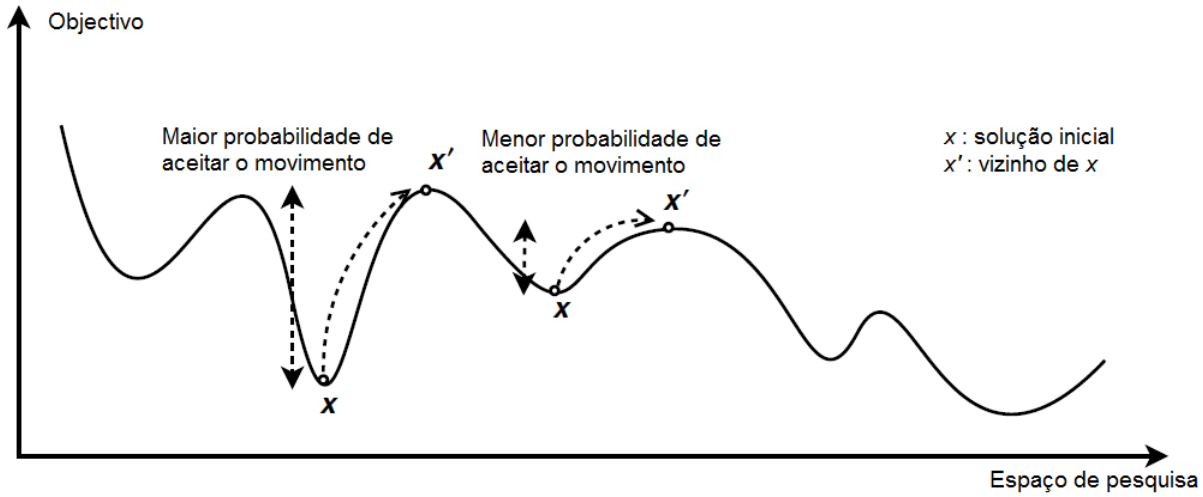


Figura 10 - *Simulated Annealing* a escapar aos ótimos locais (Pereira, 2014).

O funcionamento do algoritmo pode ser descrito da seguinte forma:

Passo 1 Seja $k = 1$ e selecionar T_1.
Selecionar uma sequência inicial S_1 usando uma heurística.
Seja $S_0 = S_1$

Passo 2 Selecionar um candidato S_c da vizinhança de S_k.
Se $F(S_0) < F(S_c) < F(S_k)$, então $S_{k+1} = S_c$ e avançar para o passo 3.
Se $F(S_c) < F(S_0)$, então $S_0 = S_{k+1} = S_c$ e avançar para o passo 3.
Se $F(S_c) > F(S_k)$ gerar um número aleatório U_k entre 0 e 1.
Se $U_k \leq P(S_k,S_c)$ então $S_{k+1} = S_c$ caso contrário $S_{k+1} = S_k$ e avançar para o passo 3.

Passo 3 Selecionar $T_{k+1} \leq T_k$
Incrementar k em 1.
Se $k = N$ então PARAR, caso contrário ir para o passo 2.

3.7.3 Algoritmos genéticos

Os algoritmos genéticos são uma analogia entre o processo de encontrar soluções ótimas e a teoria da evolução das espécies, de Charles Darwin (1859), baseando-se no princípio de sobrevivência dos indivíduos, mais aptos de uma população, tendo sido fundamentado por John Holland.

Segundo esta teoria a evolução genética ocorre nos cromossomas, sendo estes responsáveis pela codificação dos seres vivos. As melhores codificações são mais bem sucedidas, podendo ocorrer mutações nos cromossomas.

Os algoritmos genéticos buscam a melhor solução para os problemas de otimização através de um processo iterativo de busca da melhor solução. A busca tem origem a partir de uma população inicial que vai dar origem a uma nova, através da seleção dos melhores representantes. Cada nova iteração gera uma nova população com melhores soluções.

Um algoritmo genético pode ser representado segundo o fluxograma da figura 11.

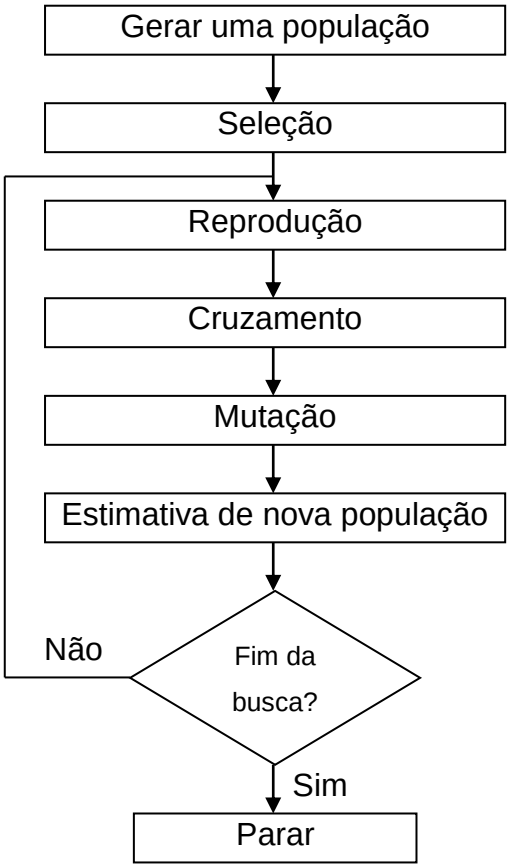


Figura 11 - Fluxograma representativo de um AG.

Os cromossomas podem ser representados por uma cadeia de bits (0 e 1), como exemplifica a figura 12. Estes cromossomas passam pela função *fitness* que é uma função de avaliação, representando a performance do cromossoma.

A técnica de seleção, cruzamento e mutação permitem criar filhos diferentes dos pais.

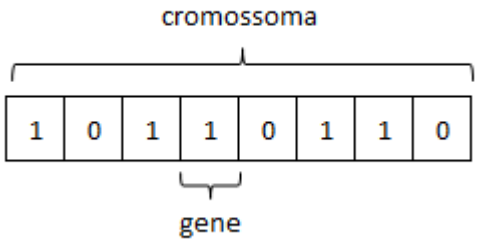


Figura 12 - Exemplo e identificação de um cromossoma.

Operadores genéticos

a) Seleção

A seleção num algoritmo genético permite escolher quais os elementos que vão entrar no processo de reprodução da nova população, sendo que os que apresentam um *fitness* mais elevado terão mais hipóteses de reprodução.

Cada indivíduo da população vai ocupar uma porção de uma roleta proporcional ao seu *fitness*, sendo que os mais aptos (maior *fitness*) ocuparão uma porção maior. Esta roleta será girada várias vezes em função do tamanho da população.

Seja:

12 na base 2 = 01100

$$x = a_4 * 2^4 + a_3 * 2^3 + a_2 * 2^2 + a_1 * 2^1 + a_0 * 2$$

$$x = 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2 = 12$$

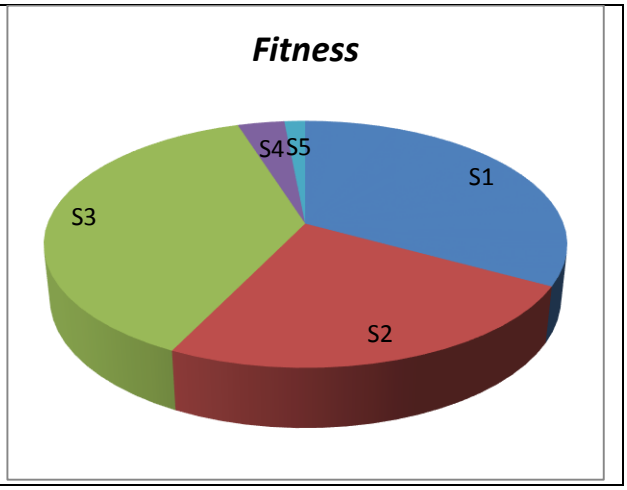
A função aptidão é dada por $f(x) = x^2$

Se x=8, então a aptidão $f(8) = 64$

A tabela 5 é um exemplo ilustrativo do algoritmo de seleção por roleta.

Tabela 5 - Indivíduos de uma população e a sua correspondente roleta de seleção.

	Cromossoma	x	$f(x)^2$	Aptidão relativa
S_1	10110	28	784	0,33
S_2	11000	24	576	0,24
S_3	11110	30	900	0,38
S_4	01001	9	81	0,03
S_5	00110	6	36	0,02



b) Cruzamento

O cruzamento consiste na combinação dos cromossomas dos pais para gerar os cromossomas dos filhos podendo ser de cruzamento num ponto ou de cruzamento uniforme.

O cruzamento num ponto consiste na divisão dos cromossomas selecionados num ponto aleatório da cadeia, sendo copiados uma parte de cada pai para gerar novos filhos.

A figura 13 exemplifica este método de cruzamento.

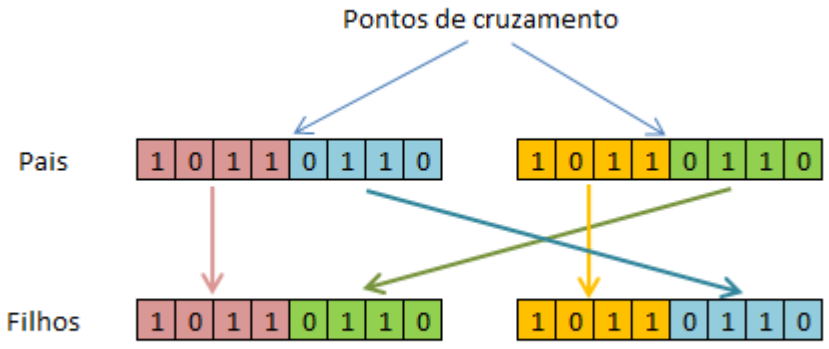


Figura 13 - Exemplo de um operador de cruzamento num ponto.

O cruzamento uniforme consiste na geração de cada gene do descendente copiando um dos genes dos pais em que, o gene é escolhido segundo uma máscara de cruzamento, gerada aleatoriamente. Percorre-se todas as posições da máscara observando os seus valores e quando o valor da posição for igual a um, o gene do

primeiro pai, referente à mesma posição da máscara, é copiado. Se o valor da máscara for igual a zero, então será copiado o gene do segundo pai. No fim do processo é gerado o novo descendente como exemplifica a tabela 6.

Tabela 6 - Exemplo de um operador de cruzamento uniforme.

Máscara de cruzamento	0101001
1º Pai	1101101
2º Pai	0001110
Descendente	0101111

c) Mutação

Este tipo de mutação é do tipo binário em que, gera uma probabilidade de mutação para cada bit do cromossoma.

A mutação inverte os valores dos bits e é aplicada com dada probabilidade, denominada taxa de mutação (≈ 1%), em cada um dos bits do cromossoma.

Considere-se a probabilidade de mutação igual a 0,001 e o cruzamento dos indivíduos 1 e 2;

Se x_i < 0,001 vai ocorrer mutação no gene i, caso contrário não ocorre mutação no gene i.

A figura 14, demonstra de uma forma genérica o processo de geração de uma nova população.

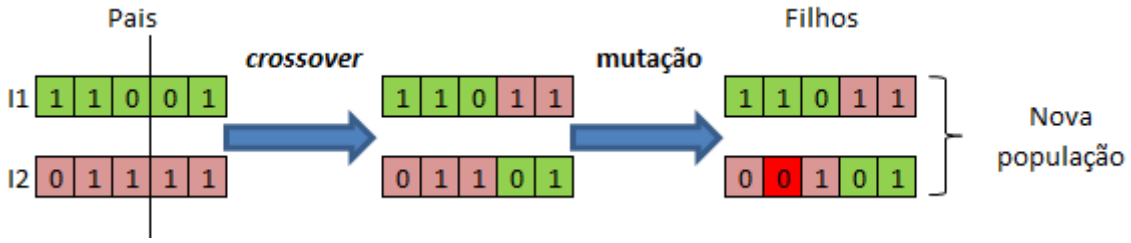


Figura 14 - Processo de geração de nova população.

Num algoritmo genético não existe a garantia de que o elemento mais apto passe para a geração seguinte. No entanto, o elitismo consiste na seleção obrigatória para a geração seguinte do indivíduo que apresenta a melhor aptidão.

3.8 Representação gráfica

Após obtenção de uma solução para o problema *job shop*, é interessante e mais legível apresenta-la através de um gráfico. As formas mais comuns de o fazer são através do diagrama de Gantt e do grafo disjuntivo.

3.8.1 O Mapa de Gantt

O mapa ou também conhecido como gráfico de Gantt foi introduzido por Henry Gantt nos inícios dos anos 1900, após ter apresentado um trabalho "A graphical daily balance in manufacturing" (Controle gráfico diário de produção), no qual descreveu um método gráfico de acompanhamento dos fluxos de produção. Esse método tornou-se no gráfico de Gantt, sendo bastante utilizado no sequenciamento da produção em ambientes do tipo *job shop*.

O gráfico de Gantt é de fácil interpretação, sendo constituído por dois eixos, em que o eixo das ordenadas representa os recursos e o eixo das abcissas representa o tempo, sendo as atividades representadas por retângulos, como se pode verificar na figura 15.

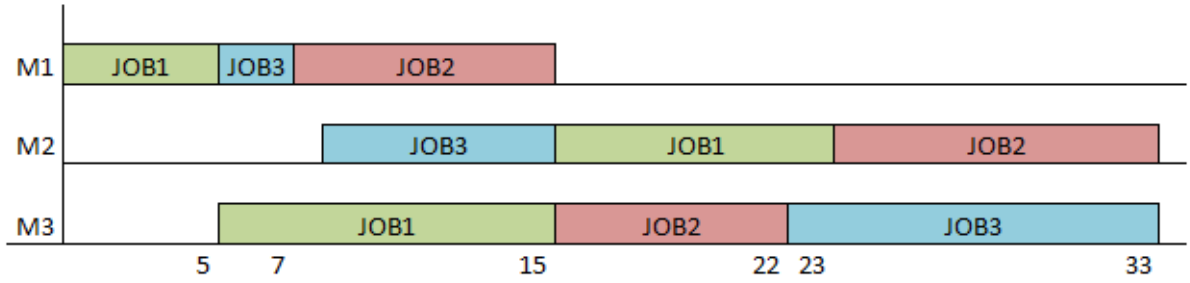


Figura 15 - Mapa de Gantt.

3.8.2 Grafo disjuntivo

O grafo disjuntivo foi proposto por Roy & Sussman (1964) e é um modelo de representação gráfica que permite descrever problemas de sequenciamento.

Para cada operação existe um nó. Adicionalmente existem dois nós (I e F) representando o início e o fim de um sequenciamento. O nó I corresponde à primeira

operação de cada ordem de produção e o nó F corresponde à última operação de cada ordem de produção, como se pode verificar na figura 16.

Chamamos sequenciamento a qualquer solução admissível do problema podendo ser representado através do grafo disjuntivo G = (O, A, E). Sendo:

- **"*O*"** o conjunto de nós, correspondendo cada nó a uma operação;
- **"A"** o conjunto de arcos conjuntivos, respeitando as relações de precedência;
- **"E"** o conjunto de todos os arcos disjuntivos. Os arcos disjuntivos não têm direção e representam o par de operações de diferentes tarefas a serem executadas na mesma máquina.

A Tabela 7 apresenta três tarefas com diferentes tempos de processamento e roteiro.

Tabela 7 - Tarefas e tempos de roteiro.

Tarefa	Roteiro	Tempo de processamento
1	M1 →M4→M3→M2	25, 7, 18, 15
2	M2→M3→M1→M4	10, 30, 7, 15
3	M4→M1→M2→M3	18, 22, 10, 7

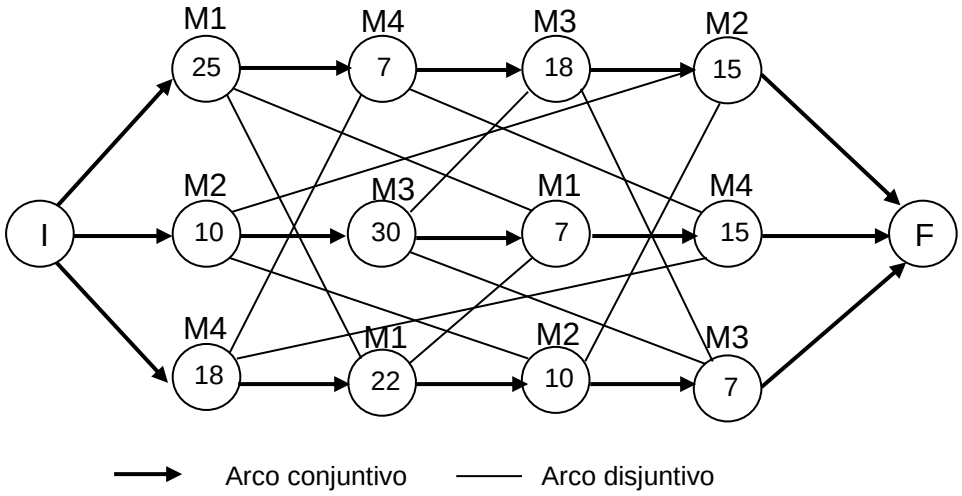


Figura 16 - Grafo disjuntivo.

O problema consiste em transformar os arcos disjuntivos em arcos com um único sentido.

O *makespan* é determinado pela soma dos tempos de processamento do caminho crítico, estando compreendido entre o início de processamento (nó I) até à conclusão de todas as tarefas em todas as máquinas (nó F).

Na Figura 17 é ilustrado o caminho crítico referente ao problema da tabela anterior. De notar que os arcos disjuntivos têm uma direção e o *makespan* está ilustrado pelas setas vermelhas.

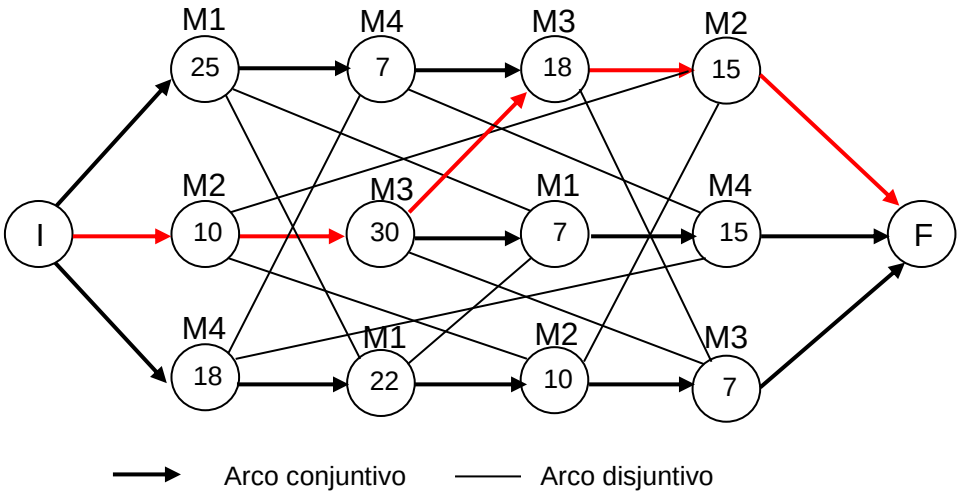


Figura 17 - Grafo disjuntivo e caminho crítico.

3.9 Considerações finais

Neste capítulo apresentou-se diversas metodologias de resolução dos problemas *job shop*, verificando-se que estes problemas podem ser resolvidos recorrendo a métodos exatos e a métodos aproximados.

Os métodos de aproximação podem dividir-se em métodos construtivos e em métodos iterativos ou meta-heuristicos. Nos métodos construtivos foram apresentados as regras de prioridade, algoritmos de inserção e o *Shifting Bottleneck Procedure*. Nos métodos iterativos ou meta-heuristicos foram apresentados o *Tabu Search*, o *Simulated Annealing* e os algoritmos genéticos.

Verificou-se que os resultados dos problemas podem ser apresentados graficamente, recorrendo ao mapa de Gantt ou através de um grafo disjuntivo.

Capítulo 4 – Experiências computacionais

4. Experiências computacionais

Nesta secção serão apresentados os modelos e métodos utilizados e comparados os diferentes resultados obtidos. Esta fase inicia-se com a elaboração de um problema para 3 tarefas em 3 máquinas e consequente resolução através do solver do Excel. Para determinação dos resultados, foi utilizado um computador portátil com as seguintes características: *AMD Athlon 64 x2 Dual Core TK-53 (1.7 GHz, 2 x 256KB cache)* com sistema operativo *Windows Vista Home Premium.*

Para a determinação da solução ótima dos problemas que se vão apresentar de seguida, será utilizado o solver Gurobi, através de um servidor online, designado NEOS *Server* e disponível no seguinte endereço:

http://www.neos-server.org/neos/solvers/milp:Gurobi/AMPL.html

O NEOS "Network Enabled Optimization System" é um serviço de internet disponível gratuitamente para resolver problemas numéricos de otimização. Desenvolvido pela Universidade de Wisconsin – Madison, fornece acesso a vários solucionadores que representam o estado da arte na área de otimização.

O AMPL "*A Mathematical Programming Language"* que é uma linguagem de modelação algébrica para descrever e resolver problemas de grande complexidade para cálculos matemáticos de grande escala, tendo sido desenvolvido por Robert Fourer, David Gay e Brian Kernighan. Para a resolução dos métodos de aproximação será utilizado o Lekin, que é um programa de sequenciamento de tarefas, utilizado como ferramenta educacional, e foi desenvolvido por Pinedo, Chao e Leung que está disponível no seguinte endereço:

http://community.stern.nyu.edu/om/software/lekin/

Serão comparados os resultados obtidos através do modelo matemático de programação linear com os obtidos através dos métodos de aproximação, tais como, *Shifting Bottleneck (SB), Local Search* e algumas regras de prioridade.

4.1 Apresentação do modelo matemático de programação linear

De seguida será apresentado o modelo de programação linear utilizado para determinar a solução ótima para o *job shop* clássico, que se define por ser um ambiente de produção com *n* tarefas e *m* máquinas, onde cada tarefa pode ser processada nas *m* máquinas. Os parâmetros e a denominação de cada um deles,

usados na formulação, admitindo que as *n* tarefas estão disponíveis para processamento no instante zero e que a interrupção de processamento de cada tarefa não é permitida são apresentados de seguida.

p_{ik} – tempo de processamento da tarefa *i* na máquina *k*.

i(1), …, i(m) – roteiro de processamento da tarefa *i* nas *m* máquinas, ou seja, a sequência de máquinas em que as operações dessa tarefa são processadas.

M – número suficientemente grande.

C_{ik} – instante de término de processamento da tarefa *i* na máquina *k*.

Onde,

$$x_{ijk} = \begin{cases} 1 \; se \; a \; tarefa \; i \; precede \; a \; tarefa \; j \; na \; máquina \; k \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad caso \; contrário \end{cases}$$

Sujeito às seguintes restrições:

$$C_{i,i(1)} \geq p_{i,i(1)}\,, \qquad i = 1, \dots, n \tag{62}$$

$$C_{i,i(k+1)} \geq C_{i(k)} + p_{i,i(k+1)}\,, \qquad i = 1, \dots, n, \qquad k = 1, \dots, m-1 \tag{63}$$

$$C_{jk} \geq C_{ik} + p_{jk} - M\left(1 - x_{ijk}\right), \qquad i = 1, \dots, n, \qquad j = 1, \dots, n \;\; k = 1, \dots, m \tag{64}$$

$$C_{ik} \geq C_{jk} + p_{ik} - M x_{ijk}\,, \qquad i = 1, \dots, n, \qquad j = 1, \dots, n \;\; k = 1, \dots, m \tag{65}$$

$$C_{\max} \geq C_{i,i(m)} \tag{66}$$

$$T_{\max} \geq C_{i,i(m)} \tag{67}$$

$$C_{ik} \leq T_{i,i(m)} + d_{i,i(m)} \tag{68}$$

$$T_{i,i(m)} \leq M * U_{i,i(m)} \tag{69}$$

É possível elaborar diferentes modelos em detrimento da função objetivo pretendida, e dessa forma podemos apresentar os modelos seguintes.

Modelo 1: Minimizar o tempo de fluxo total das tarefas

$$\min \sum_{i=1}^{n} C_{i(m)}$$

Sujeito às restrições: (62), (63), (64) e (65).

Modelo 2: Minimizar o *makespan*

$$\min C_{\max}$$

Sujeito às restrições: (62), (63), (64), (65) e (66).

Modelo 3: Minimizar a soma dos atrasos

$$\min \sum_{i=1}^{n} T_{i(m)}$$

Sujeito às restrições: (62), (63), (64), (65) e (68).

Modelo 4: Minimizar o atraso máximo

$$\min T_{\max}$$

Sujeito às restrições: (62), (63), (64), (65), (67) e (68).

Modelo 5: Minimizar o nº de atrasos

$$\min \sum_{i=1}^{n} U_{i(m)}$$

Sujeito às restrições: (62), (63), (64), (65), (68) e (69).

Este modelo, com as possíveis variantes de função objetivo, foi implementado em AMPL e encontra-se no anexo A.

4.2 O problema de 3 tarefas em 3 máquinas

Iniciei esta parte prática, com um exercício de menor dimensão (3x3), testando a formulação no solver do Excel. Pude constatar que a partir daqui já não seria viável tentar formular recorrendo ao solver do Excel, devido à quantidade de restrições que aumenta consideravelmente.

De seguida são apresentados os dados e resultados obtidos para este problema:

Dados:

Sequenciamento:

Job 1 = M1 → M3 → M2

Job 2 = M1 → M3 → M2

Job 3 = M1 → M2 → M3

Tempo de processamento da tarefa *i* na máquina *k*:

p_{ik}	1	2	3
1	5	8	10
2	8	10	7
3	2	7	11

Tendo por objetivo o de minimizar a soma dos tempos de processamento de todas as tarefas na última operação, a função objetivo seria:

$$\min \sum_{i=1}^{n} C_{i(m)}$$

Resultados obtidos:

Instante de término de processamento da tarefa *i* na máquina *k*:

C_{ik}	1	2	3
1	5	23	15
2	15	33	22
3	7	15	33

Se a tarefa *i* precede a tarefa *j* na máquina *k*:

K=1			
x_{ijk}	1	2	3
1	-	1	1
2	0	-	0
3	0	1	-

K=2			
x_{ijk}	1	2	3
1	-	1	0
2	0	-	0
3	1	1	-

K=3			
x_{ijk}	1	2	3
1	-	1	1
2	0	-	1
3	0	0	-

Para esta função objetivo, minimização do tempo de fluxo total das tarefas, o melhor resultado obtido pelo solver do Excel é de 89 unidades de tempo, correspondendo à soma de C_{12}, C_{22}, e C_{33}.

Representação gráfica:

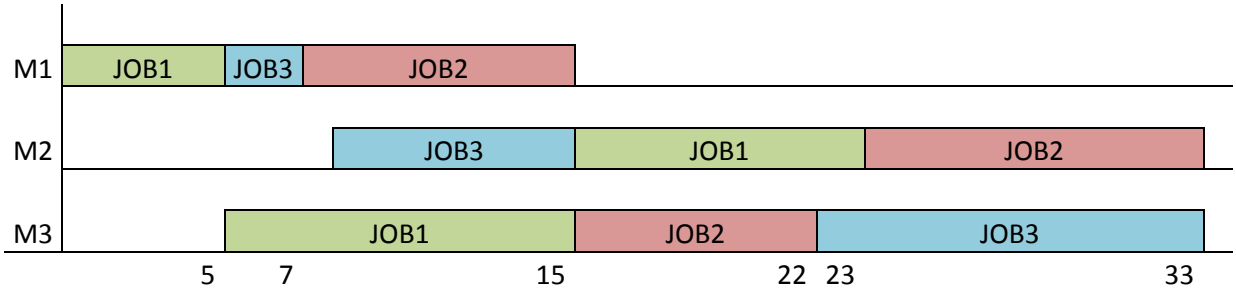


Figura 18 - Mapa de Gantt do problema 3x3 pelo modelo de otimização.

4.2.1 Shifting Bottleneck Procedure

Sendo:

M = conjunto de todas as máquinas

M' = conjunto de máquinas já sequenciadas

r_i = data de lançamento mais cedo

d_j = data de entrega mais tarde

C_j = tempo de processamento da tarefa j

$L_j = C_j - d_j$

$L_{max} = \max L_j$

$[r_j ; d_j]$

Inicialmente M' é igual a um conjunto vazio { }.

Passo 1: Encontrar o C_{max}

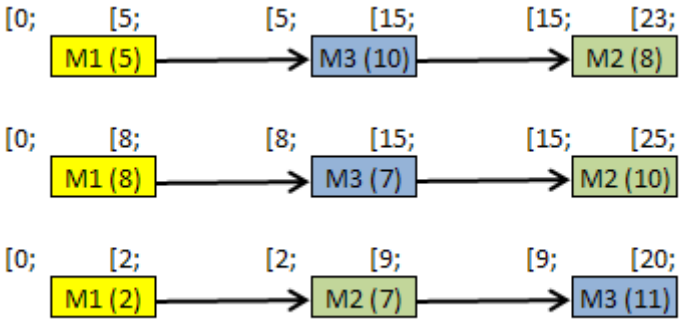


Figura 19 - Encontrar o Cmax, fase 1.

Depois de encontrar o C_{max}, que neste caso é igual a 25, coloca-se esse tempo no final da última operação de cada tarefa. Posto isto, reinicia-se o apuramento dos tempos do fim para o início em cada tarefa, subtraindo o tempo de processamento em cada máquina. Por exemplo, na tarefa 1 subtrai-se 8 u.t que corresponde ao tempo de processamento da tarefa na máquina M2, a 25 u.t correspondente ao Cmax atual. Na figura seguinte, poderemos ver o apuramento destes resultados.

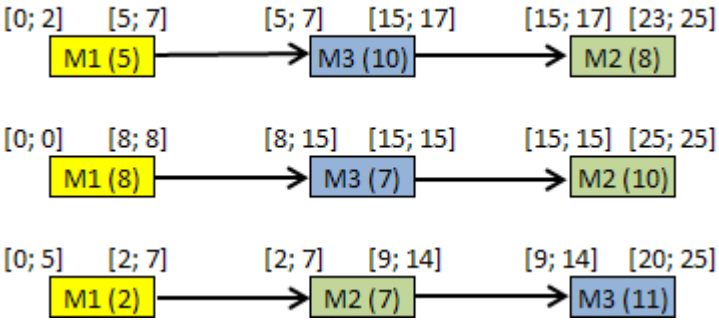


Figura 20 - Encontrar o Cmax, fase 2.

Passo 2: Re-otimizar o sequenciamento das máquinas

Usando as datas de lançamento e de entrega, minimizar o L_{max} em cada máquina.

M1	J1 (5) [0 ; 7] J2 (8) [0 ; 8] J3 (2) [0 ; 7]	J1 (5) J3 (2) J2 (8) 0 7 9 17	L(1) = 7-7= 0 L(2) = 9-7= 2 L(3) = 17-8= 9
M2	J1 (8) [15 ; 25] J2 (10) [15 ; 25] J3 (7) [2 ; 14]	J3 (7) J2 (10) J1 (8) 2 9 19 27	L(1) = 9-14= -5 L(2) = 19-25= -6 L(3) = 27-25= 2
M3	J1 (10) [5 ; 17] J2 (7) [8 ; 15] J3 (11) [9 ; 25]	J1 (10) J2 (7) J3 (11) 5 15 22 33	L(1)= 15-17= -2 L(2)= 22-15= 7 L(3)= 33-25= 8

Figura 21 - Re-otimizar o sequenciamento das máquinas, fase 1.

Escolhe-se a máquina com o maior L_{max}, que neste caso é a máquina M1 com L= 9.

M' = {M1}

M' ≠ M logo voltar ao PASSO 1.

O sequenciamento da máquina 1 foi encontrado e é igual a:

M1: J1→ J3 → J2

Passo 1: Encontrar o C_{max}

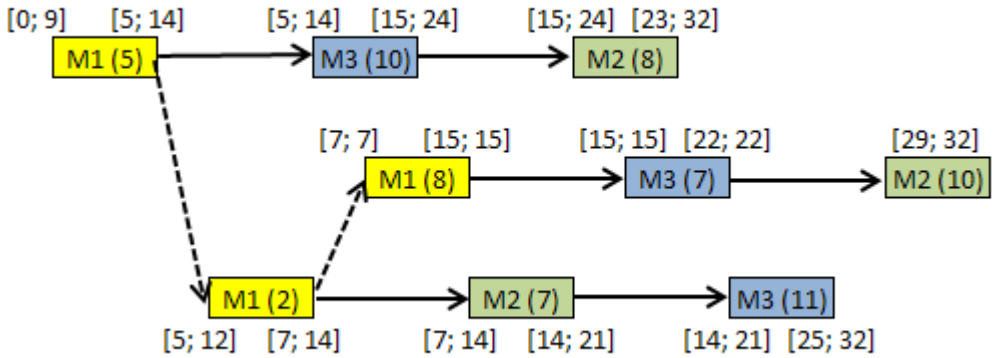


Figura 22 - Encontrar o Cmax, fase 3.

Passo 2: Re-otimizar o sequenciamento das máquinas

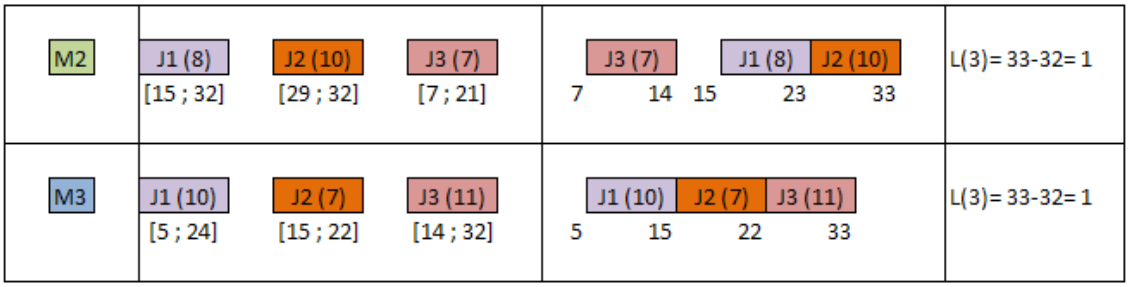


Figura 23 - Re-otimizar o sequenciamento das máquinas, fase 2.

Escolhe-se a máquina com o maior L_{max}, que neste caso é igual com L=1.

O sequenciamento da máquina 2 e 3 foi encontrado e é igual a:

M2: J3→ J1 → J2

M3: J1→ J2 → J3

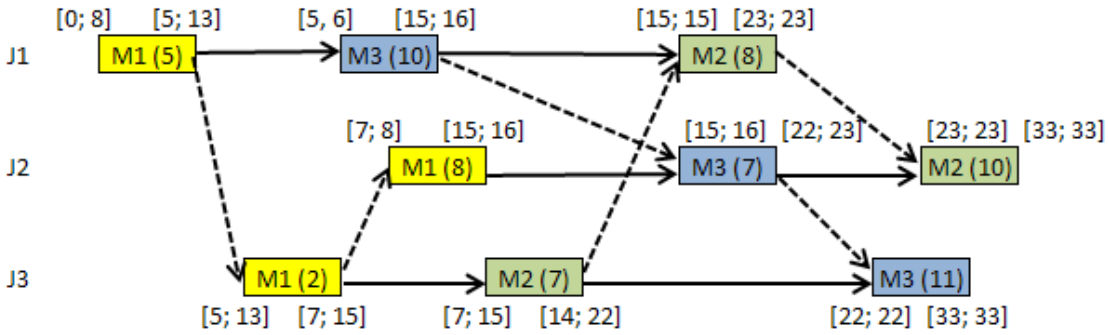


Figura 24 - Grafo disjuntivo do método *Shifting Bottleneck*.

M' = {M1, M2, M3}

M' = M logo PARAR.

Mapa de Gantt do problema:

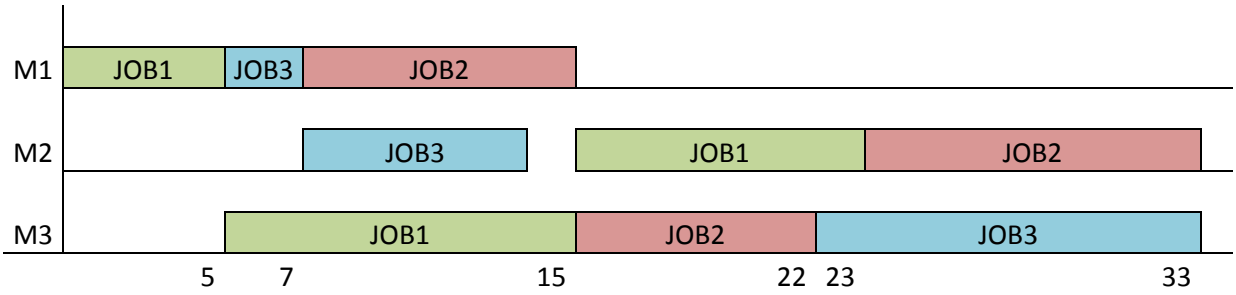


Figura 25 - Mapa de Gantt do problema 3x3 pelo método *Shifting Bottleneck*.

Podemos observar que a aplicação do método *Shifting Bottleneck (SB)*, produziu um resultado em que o sequenciamento é classificado como semi-ativo e não atrasado ao contrário do modelo de programação linear. Se observarmos a tarefa 3 na máquina 2, verificamos que no *SB* não existe tempos de espera desnecessários. A operação é iniciada o mais cedo possível, não ficando a máquina inativa quando pode iniciar o processamento da operação. No modelo de programação linear a máquina fica inativa quando poderia iniciar o processamento da tarefa 3.

Com um tempo de processamento de 35 segundos, o *solver* do Excel não é a melhor opção para resolver problemas com maior dimensão, pois demora muito tempo e a quantidade de restrições é demasiado grande para se elaborar numa folha de cálculo. A partir deste momento testarei algumas instâncias com diferentes métodos, sendo eles o método exato, permitindo obter a solução ótima e através de métodos aproximados, tais como, o *Shifting Bottleneck, Local Search* e algumas regras de prioridade apresentados na secção 3.5.2, a saber: SPT, LPT, EDD, FCFS e MS.

Serão testados dois conjuntos de instâncias. O primeiro é composto por duas instâncias, ft06 e ft10, desenvolvidas por Fisher & Thompson (1963) e o segundo grupo é composto por quatro instâncias, la01, la06, la11 e la21, desenvolvidas por Lawrence (1984). Estas instâncias são utilizadas para testes de desempenho, englobando diferentes níveis de dificuldade e de tamanho.

Tabela 8 - Instâncias de teste.

Instância	Ordens de Produção	Nº de Máquinas
ft06	6	6
ft10	10	10
la01	10	5
la06	15	5
la11	20	5
la21	15	10

4.3 Apresentação da instância ft06

Os *Jobs* representam as tarefas, *M* a máquina e *TP* o tempo de processamento na respetiva máquina. O sequenciamento de cada tarefa é definido pela ordem das máquinas representado na tabela, e a numeração das máquinas está compreendida entre zero e cinco.

Tabela 9 - Instância ft06.

Jobs	M	TP	M	TP	M	TP	M	TP	M	TP	M	TP
Job 1	2	1	0	3	1	6	3	7	5	3	4	6
Job 2	1	8	2	5	4	10	5	10	0	10	3	4
Job 3	2	5	3	4	5	8	0	9	1	1	4	7
Job 4	1	5	0	5	2	5	3	3	4	8	5	9
Job 5	2	9	1	3	4	5	5	4	0	3	3	1
Job 6	1	3	3	3	5	9	0	10	4	4	2	1

4.3.1 Aplicação dos modelos 1 e 2 – Tempo da tarefa

Modelo 1: Minimizar o tempo de fluxo total das tarefas

Modelo 2: Minimizar o *makespan*

Tabela 10 - Resultados para o tempo da tarefa.

					Regras de prioridade				
F.Obj	Mod.1	Mod.2	Shifting Bottleneck	Local Search	SPT	LPT	EDD	FCFS	MS
$min \sum C_i$	265	-	280	265	292	354	321	329	354
$min\ C_{max}$	-	55	59	55	73	67	63	65	67

A solução ótima para o modelo 1 é de 265 u.t. Através da busca local conseguimos obter o mesmo resultado da solução ótima, enquanto que através de uma regra de prioridade SPT é possível obter uma solução aproximada, mas não a melhor solução. Para o modelo 2 a solução ótima é de 55 u.t. e através da busca local conseguimos alcançar o mesmo resultado da solução ótima. Recorrendo a uma regra de prioridade a que menos se distancia da melhor solução é a EDD.

A implementação deste modelo em AMPL para esta instância encontra-se no anexo A e a respetiva solução obtida pelo solver Gurobi encontra-se no anexo B.

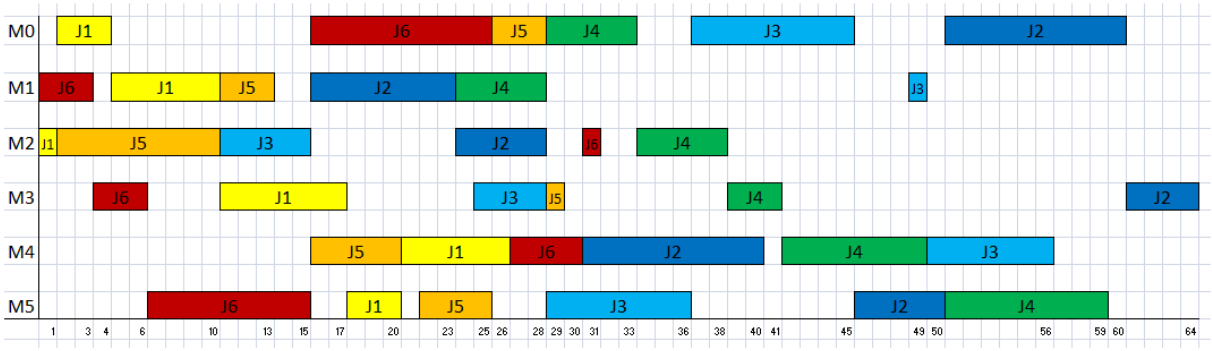


Figura 26 - Gantt da instância ft06, para o tempo de fluxo total.

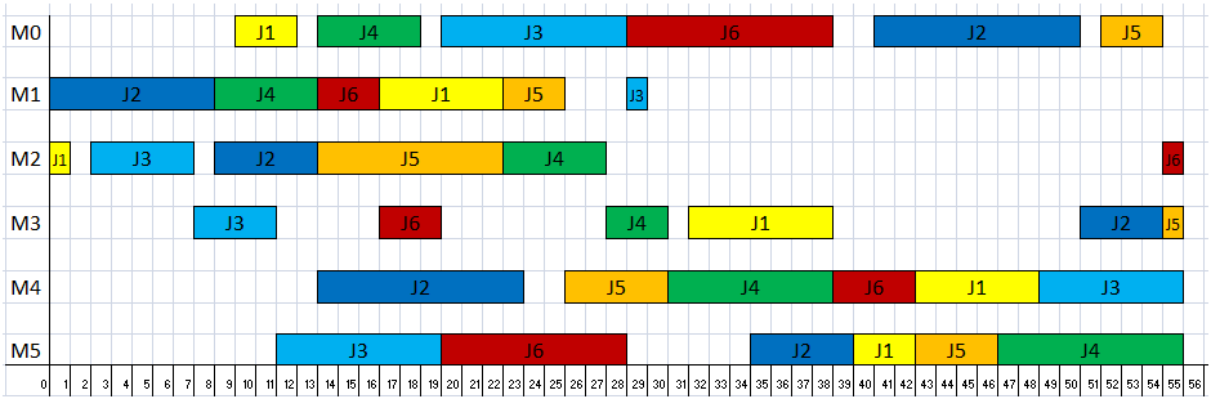


Figura 27 - Gantt da instância ft06, para o Cmax.

4.3.2 Aplicação dos modelos 3 e 4 – Atraso da tarefa

Modelo 3: Minimizar a soma dos atrasos

Modelo 4: Minimizar o atraso máximo

A partir deste momento vou introduzir mais duas variáveis (T_i e d_i). Sendo T_i o atraso da tarefa *i* e d_i a data de entrega da tarefa *i.*

O objetivo é de colocar um prazo para a entrega das encomendas e sequenciar em função das datas de entrega.

Tendo em conta os resultados obtidos até então, vou considerar um prazo de entrega de 54 u.t (unidades de tempo), igual para todas as ordens de fabrico i.

Tabela 11 - Resultados para o atraso da tarefa com d=54 u.t.

d=54 u.t					Regras de prioridade				
F.Obj	Mod.3	Mod.4	Shifting Bottleneck	Local Search	SPT	LPT	EDD	FCFS	MS
$min \sum T_i$	1	-	1	1	25	42	16	23	42
$min\ T_{max}$	-	1	1	1	19	13	9	11	13

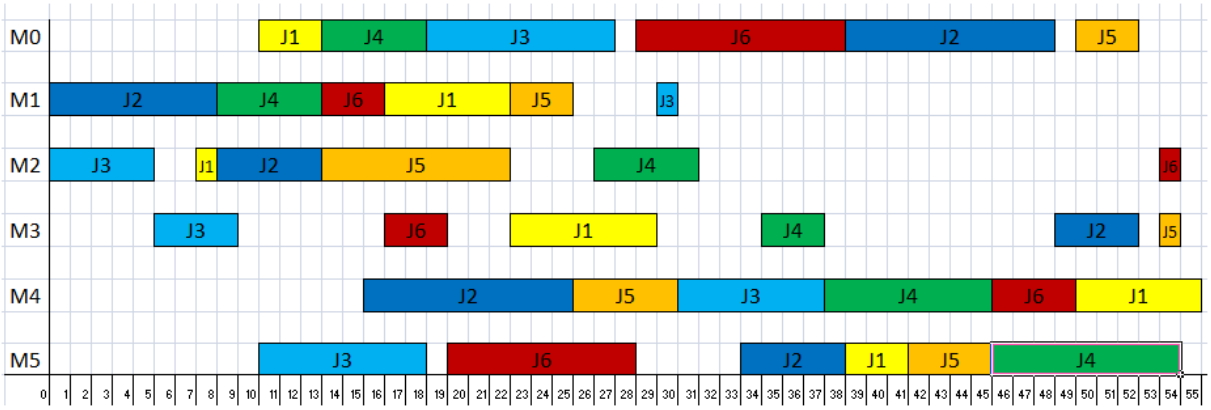


Figura 28 - Gantt da instância ft06, para a soma dos atrasos com d=54 u.t.

Observa-se que a soma dos atrasos é igual a 1 u.t.

Desta forma vou estabelecer um prazo de entrega mais apertado, de forma a verificar se existe mudanças significativas no sequenciamento.

Assim sendo, o novo prazo de entrega passará a ser igual a 50 u.t. para todas as ordens de fabrico.

Tabela 12 - Resultados para o atraso da tarefa com d=50 u.t.

d=50 u.t					Regras de prioridade				
F.Obj	Mod.3	Mod.4	Shifting Bottleneck	Local Search	SPT	LPT	EDD	FCFS	MS
$min \sum T_i$	14	-	14	14	37	59	32	39	59
$min\ T_{max}$	-	5	9	-	23	17	13	15	17

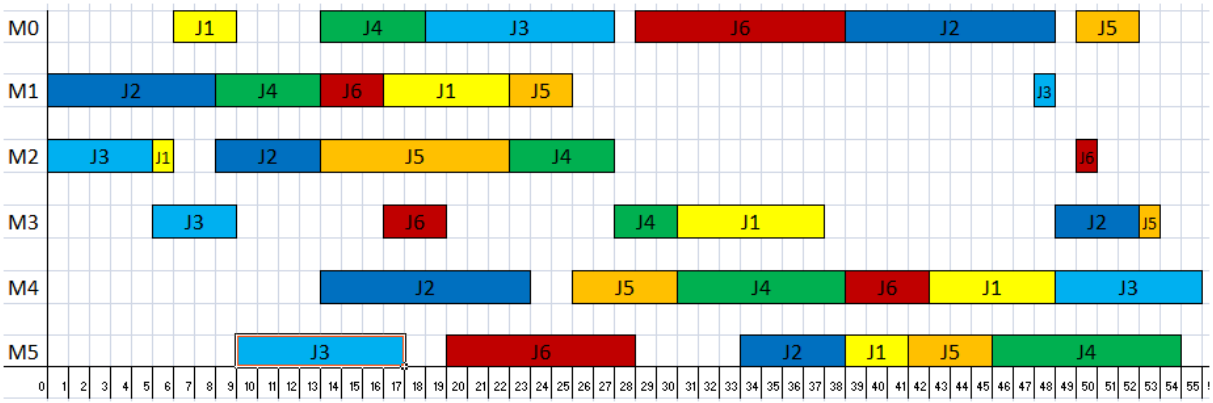


Figura 29 - Gantt da instância ft06, para a soma dos atrasos com d=50 u.t.

Tabela 13 - Comparação entre *deadline* igual a 54 e 50 u.t.

	d = 54 u.t.		d = 50 u.t.	
	INÍCIO	TÉRMINO	INÍCIO	TÉRMINO
J1	7	55	5	48
J2	0	52	0	52
J3	0	37	0	55
J4	8	54	8	54
J5	13	54	13	53
J6	13	54	13	50
Σ		306		312

Observa-se com esta alteração que o tempo de fluxo total das tarefas aumentou de 306 u.t para 312, de forma a minimizar a soma dos atrasos que é de 14 u.t. Constata-se ainda que, para satisfazer a restrição de data de entrega menor ou igual a 50 unidades de tempo, há um recuo na data de início da tarefa 1 culminando com uma data de término menor. As restantes tarefas iniciam no mesmo instante, sendo que as tarefas 2 e 4 terminam no mesmo instante e as tarefas 3, 5 e 6 terminam mais cedo (ver Tabela 13).

4.3.3 Aplicação do modelo 5 – Nº de atrasos

Modelo 5: Minimizar o número de atrasos

Tabela 14 - Resultados para o atraso da tarefa com d=50 u.t.

d = 50 u.t.			Regras de prioridade				
F.Obj	Mod.5	Shifting Bottleneck	SPT	LPT	EDD	FCFS	MS
$min \sum U_i$	1	1	3	5	4	4	5

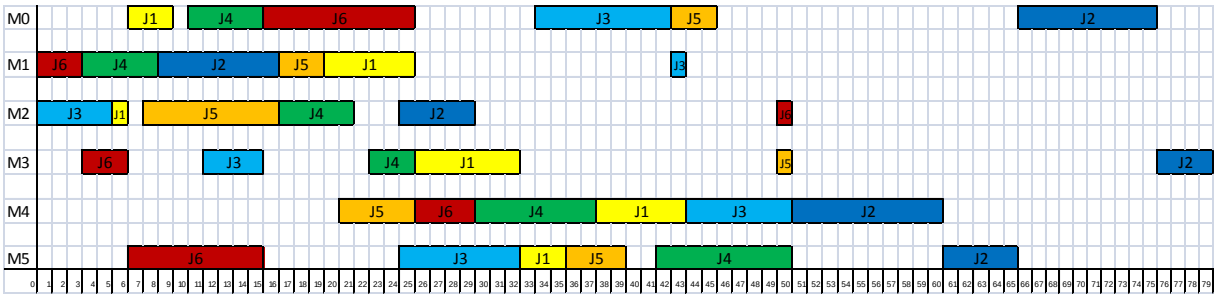


Figura 30 - Gantt da instância ft06, para a minimização dos atrasos com d=50u.t.

Através do modelo 5 conseguimos obter o melhor resultado para o objetivo de minimizar o número de atrasos. De facto, apenas se obtém um atraso, mas observa-se que a tarefa 2 terá um atraso grande de 29 u.t. em relação à data de entrega que deveria ser no máximo de 50 u.t.

De seguida apresentam-se duas tabelas que apresentam as melhores soluções para cada função objetivo, utilizando modelos de otimização e as soluções aproximadas, utilizando regras de prioridade, permitindo avaliar o comportamento dos restantes parâmetros (usando d=50).

Tabela 15 - Soluções ótimas para a instância ft06, com d=50 u.t.

	Soluções ótimas					
		$\sum_{i=1}^{n} C_{i(m)}$	C_{max}	$\sum_{i=1}^{n} T_i$	T_{max}	$Nº\ Atrasos$
F. Objetivo	$min \sum_{i=1}^{n} C_{i(m)}$	**265**	64	29	14	3
	$min\ C_{max}$	322	**55**	24	5	5
	$min \sum_{i=1}^{n} T_i$	312	55	**14**	5	4
	$min\ T_{max}$	306	55	19	**5**	4
	$Nº\ Atrasos$	322	79	29	29	**1**

Tabela 16 - Soluções aproximadas para a instância ft06, com d=50 u.t.

	Soluções aproximadas					
		$\sum_{i=1}^{n} C_{i(m)}$	C_{max}	$\sum_{i=1}^{n} T_i$	T_{max}	$Nº\ Atrasos$
Regras de prioridade	SPT	292	73	37	23	3
	LPT	354	67	59	17	5
	EDD	321	63	32	13	4
	$FCFS$	329	65	39	15	4
	MS	354	67	59	17	5

Podemos concluir que em função do objetivo pretendido podemos obter melhores ou piores soluções em função da regra de prioridade escolhida. Em cenários reais podemos simular com diferentes regras e optar por aquela que produzir melhores resultados para o objetivo pretendido.

Neste caso se pretendêssemos um cenário para minimizar a soma do tempo de fluxo de todas as tarefas e o número de atrasos, deveríamos optar pela regra *SPT.* Se por outro lado, o objetivo fosse o de minimizar a soma dos atrasos, o atraso máximo ou o makespan, então deveríamos de optar pela regra *EDD.*

4.3.4 Clientes prioritários

Outra situação que muitas vezes ocorre na indústria é a necessidade de atender prioritariamente determinado cliente em detrimento de outros, não podendo falhar com a data de entrega acordada. Para isso podemos atribuir diferentes "pesos" (w_j) em função da prioridade que cada tarefa deverá ter.

Supondo que cada tarefa corresponde a um cliente diferente, as datas acordadas para cada um deles, a prioridade de cada cliente, o tempo mínimo necessário e o nível de satisfação são apresentados na Tabela 187:

Cenário 1: Os clientes têm a mesma prioridade, no entanto o cliente 6 tem uma data de entrega igual a 30 u.t.

Tabela 17 - Parâmetros cenário 1, prioridades idênticas.

Parâmetro	Cliente 1	Cliente 2	Cliente 3	Cliente 4	Cliente 5	Cliente 6
Data entrega (u.t)	50	50	50	50	50	**30**
Tempo necessário	26	47	34	35	25	30
Prioridade	1	1	1	1	1	1
Data entrega real	36	53	56	58	54	31
Atraso	0	3	6	8	4	1
% atraso	0%	6%	12%	16%	8%	3%
Satisfação cliente						

Cenário 2: O Cliente 6 é prioridade máxima, não admitindo nenhum dia de atraso.

Tabela 18 - Parâmetros cenário 2, com diferentes prioridades.

Parâmetro	Cliente 1	Cliente 2	Cliente 3	Cliente 4	Cliente 5	Cliente 6
Data entrega (u.t)	50	50	50	50	50	**30**
Tempo necessário	26	47	34	35	25	30
Prioridade	1	1	1	1	1	20
Data entrega real	53	71	60	46	38	30
Atraso	3	21	10	0	0	0
% atraso	6%	42%	20%	0%	0%	0%
Satisfação cliente						

Podemos observar que apesar do tempo para executar a tarefa do cliente 6 ser igual à data de entrega acordada (30 u.t.), definindo prioridade máxima em que não pode atrasar nenhuma unidade de tempo (dias, horas, semanas, etc.) para esse cliente, conseguimos satisfazer a sua necessidade. No entanto, vamos prejudicar a qualidade do atraso para os outros clientes, sendo que o cliente 2, com um atraso de 42%, correspondendo a 21 unidades de tempo, ficará bastante insatisfeito correndo o risco de perda do cliente.

Já num cenário em que os clientes têm prioridades idênticas, a qualidade do atraso é consideravelmente melhor, em que a pior situação é para o cliente 4 com um atraso de 16%, correspondendo a 8 unidades de tempo.

Há que avaliar cada cenário, tentando perceber a importância de cada cliente e o que representam em termos financeiros para a empresa e a partir daí optar pelo cenário mais adequado.

Estes quadros permitem ao gestor tomar decisões mais ponderadas para o cenário que melhor satisfaça as suas necessidades em cada momento.

4.3.5 Outras instâncias

Em suma, apresenta-se de seguida um quadro com diferentes instâncias e os seus resultados, obtidos através dos diferentes modelos de otimização para obter a solução ótima e a comparação com os métodos aproximados.

Tabela 19 - Resultado das instâncias para o Cmax.

		min Cmax			Regras de prioridade				
Inst.	(nxm)	Solução ótima	Shifting Bottleneck	Local Search	**SPT**	**LPT**	**EDD**	**FCFS**	**MS**
ft06	6 x 6	55	59	55	73	67	63	65	67
ft10	10 x 10	930	1094	930	1338	1168	1246	1184	1168
la01	10 x 5	666	686	666	1122	752	865	772	752
la06	15 x 5	926	926	926	1475	926	1024	926	926
la11	20 x 5	1222	1235	1222	1802	1300	1272	1272	1300
la21	15 x 10	1046	1211	1051	1502	1266	1440	1265	1266

Tabela 20 - Tempos de processamento dos resultados das instâncias.

		min Cmax					
Inst.	(nxm)	Solução ótima	T.P(s)	Shifting Bottleneck	T.P(s)	Local Search	T.P(s)
ft06	6 x 6	55	10s	59	1s	55	2s
ft10	10 x 10	930	180s	1094	2s	930	200s
la01	10 x 5	666	25s	686	2s	666	2s
la06	15 x 5	926	4810s	926	2s	926	3s
la11	20 x 5	1222	5705s	1235	9s	1222	2s
la21	15 x 10	1046	28795s	1211	13s	1051	2400s

Podemos observar na Tabela 20, que de uma forma geral o modelo matemático de programação linear para obtenção de uma solução ótima, requer um tempo de processamento superior aos métodos aproximados de *Shifting Bottleneck* e *Local Search*. De referir que as regras de prioridade foram resolvidas em apenas um segundo cada uma delas. De salientar que para a instância ft10 e la21, conseguiu-se obter uma solução ótima, mas foi necessário gastar duzentos segundos e dois mil e quatrocentos segundos, respetivamente, quando se conseguiria obter uma solução aproximada em poucos segundos.

Podemos concluir que se for pretendido obter uma boa solução recorrendo a regras de prioridade, a melhor regra para um problema pode não ser a melhor para outro. Por exemplo, na instância ft06 a melhor regra de prioridade para o C_{max} é a EDD com 63 u.t., mas a melhor para a instância ft10 é a LPT ou a MS, como podemos observar na Tabela 19.

A solução obtida para as instâncias ft06, ft10, la01, la06, la11 e la21 encontram-se nos anexos B, C, D, E, F e G, respetivamente.

4.4 Considerações finais

Neste capítulo apresentou-se um problema de três tarefas em três máquinas, aplicando-se um modelo de programação linear, sendo resolvido recorrendo ao solver do Excel. Fez-se a sua representação gráfica através do mapa de Gantt e recorreu-se a um método de aproximação, o *Shifting Bottleneck Procedure (SB)* para obter uma boa solução que neste caso é muito semelhante ao resultado obtido pelo modelo de programação linear, tendo o *SB* produzido um sequenciamento semi-ativo e não atrasado.

De seguida foi testada a instância ft06 para diferentes modelos, sendo os seus resultados comparados com os resultados obtidos pelos métodos de aproximação *Shifting Bottleneck Procedure*, *Local Search* e as regras de prioridade.

Na indústria existe por vezes a necessidade de atender prioritariamente um cliente e daí ter-se apresentado dois cenários diferentes, comparando o impacto e as alterações geradas por essa priorização, medindo o nível de satisfação dos restantes clientes, num cenário hipotético.

Por último, comparou-se diferentes instâncias para os mesmos métodos e verificou-se que o melhor método para um problema pode não ser o melhor para outro.

Capítulo 5 – Conclusões e Trabalho Futuro

5. Conclusões e trabalho futuro

Após o estudo e análise de alguns problemas do tipo *job shop*, conseguiu-se demonstrar que um bom sequenciamento das tarefas de produção pode levar as empresas a obter ganhos de produção significativos.

Concluiu-se que, devido à sua complexidade exponencial, o problema *job shop* é um problema NP-dificil, não se conhecendo um algoritmo de resolução eficiente.

Consoante o processo industrial em questão, existem diversas variantes, devido às condicionantes próprias de cada processo industrial, tais como: *job shop* com número distinto de operações por tarefa, *job shop* com interrupção, *job shop* com instantes de disponibilidade distintos, *job shop* flexível, *job shop* com tempos de *setup* dependentes da sequência, *job shop* com *buffers*, *job shop* com reentrada e restrições de precedência.

Para a resolução de problemas do tipo *job shop* apresentaram-se diversas metodologias, nomeadamente métodos exatos e métodos aproximados, sendo que os métodos aproximados podem classificar-se em métodos construtivos e em métodos iterativos ou meta-heuristicos.

Compararam-se diferentes tipos de sequenciamento, destacando-se os ativos, os semi-ativos e os não atrasados. Num sequenciamento ativo nenhuma operação pode iniciar mais cedo sem provocar um atraso noutra operação ou sem violar as restrições de precedência. Os sequenciamentos ativos formam uma subclasse dos semi-ativos, ou seja, um sequenciamento ativo também é semi-ativo.

As medidas de desempenho mais utilizadas no *job shop* são o makespan, o tempo de fluxo total, o atraso máximo, a soma dos atrasos, o número de tarefas atrasadas e o atraso máximo. Aplicou-se um modelo matemático de programação linear inteira mista a instâncias conhecidas na literatura e compararam-se os resultados obtidos com as várias medidas de desempenho, para perceber bem a estrutura do problema e das suas variantes.

Os resultados dos problemas foram apresentados graficamente através do mapa de Gantt e através do grafo disjuntivo.

Aplicaram-se também diversas heurísticas de sequenciamento a estes problemas de teste. Demonstrou-se que para problemas de grande dimensão, a

utilização de métodos aproximados é a melhor opção, pois conseguem em tempo útil obter boas soluções.

Comprovou-se que nem sempre o melhor método de aproximação para um determinado problema é o melhor método para outro.

Por vezes, existe a necessidade de atribuir diferentes importâncias aos clientes, sendo necessário atender prioritariamente uns em detrimentos de outros.

O estudo prévio de vários cenários antes da tomada de decisão pode ajudar as empresas a decidir mais assertivamente em função da sua necessidade. Estes estudos permitem melhorar a eficiência da produção obtendo ganhos significativos, tais como a satisfação dos clientes, a diminuição de custos de não produção e o consequente aumentando da competitividade das empresas.

Numa perspetiva de desenvolvimento de trabalho futuro, poder-se-ia desenvolver um modelo considerando tempos de preparação de máquina (*setup times*). Para além do modelo matemático que foi considerado, poder-se-ia também aplicar os outros modelos de otimização apresentados na secção 3.2 tais como: variáveis indexadas de tempo, variáveis de rede, atribuição da posição, bem como testar e comparar outros métodos de aproximação heurísticos.

De futuro poder-se-á testar modelos e heurísticas para outros problemas semelhantes ao *job shop*, tais como os problemas de máquina única, modelo de máquinas paralelas, *open shop*, *job shop* com múltiplas máquinas e *flow shop* com múltiplas máquinas.

Capítulo 6 – Referências Bibliográficas

6. Referências bibliográficas

ARENALES, Marcos et al. – **Pesquisa Operacional.** Rio de Janeiro: Elsevier, 2011. ISBN 1397885352. Capítulo 3, Otimização discreta.

BEIRÃO, Nuno – **Sistema de Apoio à Decisão para Sequenciamento de Operações em Ambientes "*Job Shop*".** Porto: Universidade do Porto, Faculdade de Engenharia, 1997. Dissertação de Mestrado.

BAKER, Kenneth; TRIESTCH, Dan – **Principles of Sequencing and Scheduling.** New Jersey: Jonh Wiley and Sons,Inc., 2009. ISBN 9780470391655.

BAKER, Kenneth – **Introduction to Sequencing and Scheduling.** New York: Wiley, 1974.

BEDWORTH, David.; BAILEY, James. – **Integrated Production Control Systems, Management, Analysis, Design 2/E.** John Wiley and Sons, Inc, 1987.

CONWAY, R.W.; MAXWELL, W.L.; MILLER, L.W. – **Theory of Scheduling.** Mass: Addison Wesley, 1967. ISBN 0486428176. Capítulo 1, pág.5. Disponível em http://www.books.google.pt/books?id=qHXDAgAAQBAJ&pg=PR4&dq=Theory+of+Scheduling,+Addison+Wesley&hl=pt-PT&sa=X&ved=0CCwQ6AEwAGoVChMImNWcgPvlyAIVzEkaCh2v0wcn#v=onepage&q=Theory%20of%20Scheduling%2C%20Addison%20Wesley&f=false

GIFFLER, B.; THOMSON, G. – **Algorithms for Solving Production Scheduling Problems.** Operations Research, 1960.

GUIMARÃES, Kairon – **Escalonamento genético FJSP com tempo de configuração dependente da sequência.** Uberlândia: Universidade Federal de Uberlândia, 2007. Dissertação de Mestrado.

FRENCH, Simon – **Sequencing and Scheduling: An introduction to the Mathematics of the Job Shop.** Chichester: Ellis Horwood, 1982.

JACKSON, J. – **Scheduling a Production Line to Minimize Maximum Tardiness.** Los Angeles: Research Report, University of California, 1955.

LAWLER, E.L. et al. - "**Sequencing and scheduling: algorithms and complexity"**, in: S.C. Graves, A.H.G. Rinnooy Kan and P.H. Zipkin (eds.), Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory, Amsterdam, Elsevier, 1993.

MANNE, Alan – **On the Job Shop Scheduling Problem.** New Haven: Cowles Foundation, Yale University, 1960.

MELO, Janaina; VILLAR, Antônio; FILHO, Cosmo – **O posicionamento do planejamento e controle da produção – PCP em uma indústria alimentícia.** São Paulo: XIII SIMPEP, 2006. [Consult. 10 Out. 2014] Disponível em www. http://www.simpep.feb.unesp.br/anais/anais_13/artigos/863.pdf

MEERAN, Sheik; JAIN, Anant – **A State of the Art review of Job Shop Sheduling Techniques.** UK: University of Dundee, 1998.

MEERAN, Sheik; JAIN, Anant – **Deterministic Job Shop Scheduling: Past, present and future.** UK: University of Dundee, 1999.

PACHECO, Ricardo; SANTORO, Miguel – **Proposta de Classificação Hierarquizada dos Modelos de Solução para o Problema de *Job Shop Scheduling*.** São Paulo: Escola Politécnica da Universidade de São Paulo, 1999. [Consult. 10 Out. 2014] Disponível em www. http://www.scielo.br/pdf/gp/v6n1/a01v6n1.pdf

PEREIRA, Ivo – **Sistema Inteligente para Escalonamento Assistido por Aprendizagem.** Vila Real: Universidade de Trás-os-Montes e Alto Douro, 2014. [Consult. 10 Out. 2014] Disponível em www. https://repositorio.utad.pt/bitstream/10348/4250/1/phd_iaspereira.pdf

PINEDO, Michael L. – **Scheduling: Theory, Algorithms, and Systems.** New York: Springer Science + Business Media, LLC, 2008. ISBN 9780387789347. Preface, p.vii.

SMITH, M. L. et al. – **"Characteristics of U.S. Flexible Manufacturing Systems – A Survey, "** in Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications, K.E. Stecke and R. Suri (eds)**.** Amesterdam, Elsevier Science Publishers B.V., 1986. p.477-486.

SMITH, Wayne. – **Various optimizers for single stage production.** Naval Research Logistics Quarterly**,** 1956.

UNLU, Yasin; MASON, Scott – **Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems.** Fayetteville: University of Arkansas, 2010.

WAGNER, Harvey – **An integer linear-programming model for machine scheduling.** Naval Research Logistics Quarterly, 1959, vol 6. Pag. 131-140.

Anexos

Anexo A - Modelo AMPL

```
#*****************************
#            SETS
#*****************************

#lista de possiveis tarefas
set Jobs;

# Maquinas
set Maquinas;

# sequence (M1,M2,...,Ms) of repair operations that were established for
each job ;
set Sequencia{Jobs} within Maquinas ordered;

#*****************************
#            PARAMETERS
#*****************************

#tempos de processamento
param p{i in Jobs, k in Maquinas};

param d{i in Jobs};

param W{i in Jobs};

# large number
param M;

/*****************************************
 * Decision variables
 *****************************************/
var  y{i in Jobs, k in Maquinas} binary;

var  x{i in Jobs, j in Jobs, k in Maquinas : i<>j} binary;
            #x_{i j}=1 if operation i is performed before  operation j on
machine k;


var C{i in Jobs, k in Maquinas} >= 0;        #tempo de fim de cada tarefa;
```

```
var Cmax >=0;

var Tmax >=0;

var T{i in Jobs, k in Maquinas} >= 0;

/*****************************************
 * Objective function
 *****************************************/
      #n°atrasos

##minimize f_obj: (sum {i in Jobs} y[i,last(Sequencia[i])]);


##minimize f_obj: (sum {i in Jobs} C[i,last(Sequencia[i])]);
            #tempo de fim da ultima maquina onde passa cada job i

##minimize f_obj: Cmax;

minimize f_obj: (sum {i in Jobs} W[i] * T[i,last(Sequencia[i])]);
            #Soma dos atrasos

##minimize f_obj: Tmax;

/*****************************************
 * Constraints - Restrições
 *****************************************/
subject to yyyy {i in Jobs, k in Maquinas}:
      T[i,last(Sequencia[i])] - M * y[i, last(Sequencia[i])]<=0 ;

subject to  tarefa_Tmax {i in Jobs}:
       -Tmax + C[i,last(Sequencia[i])] <=0 ;

subject to  tarefa_Cmax {i in Jobs}:
       -Cmax + C[i,last(Sequencia[i])] <=0 ;
 subject to  somaatrasos {i in Jobs}:
      C[i,last(Sequencia[i])] - T[i,last(Sequencia[i])] - d[i] <=0 ;

  #1a restricao:
 subject to  tarefa_inicial {i in Jobs}:
```

```
        -C[i,first(Sequencia[i])] + p[i,first(Sequencia[i])] <=0 ;

#restricao2
subject to  tarefa_ordem_k_mais1 {
i in Jobs, a in Sequencia[i], s in Sequencia[i]: a <> last(Sequencia[i])
and s <> first(Sequencia[i]) and s=next(a)
                              } :
C[i,s] >= C[i,a] + p[i,s] ;

#s=operacao seguinte e a=operacao anterior

# restricao 3
      subject to binarias1 {i in Jobs, j in Jobs, k in Maquinas: i<>j}:
      -C[j,k] + C[i,k] + p[j,k] - M * (1 - x[i,j,k])<=0 ;

# restricao 4
subject to binarias2 {i in Jobs, j in Jobs, k in Maquinas: i<>j}:
      -C[i,k] + C[j,k] + p[i,k] - (M * x[i,j,k])<=0 ;

/*********************************************
 * DATA - Dados
 *********************************************/
data;
param M:= 100000;

# Jobs
set Jobs:= J1 J2 J3 J4 J5 J6;

#Sequencia{Jobs}

set Sequencia[J1] :=    2     0     1     3     5     4     ;
set Sequencia[J2] :=    1     2     4     5     0     3     ;
set Sequencia[J3] :=    2     3     5     0     1     4     ;
set Sequencia[J4] :=    1     0     2     3     4     5     ;
set Sequencia[J5] :=    2     1     4     5     0     3     ;
set Sequencia[J6] :=    1     3     5     0     4     2     ;
# Maquinas
set Maquinas:= 0 1 2 3 4 5 ;

#tempos de processamento
#param p{i in Jobs, k in Maquinas};
```

```
param d
 :=
      [J1] 50
      [J2] 50
      [J3] 50
      [J4] 50
      [J5] 50
      [J6] 30
;
param p
 :=
 [J1,*] 2   1   0   3   1   6   3   7   5   3   4   6
 [J2,*] 1   8   2   5   4   10  5   10  0   10  3   4
 [J3,*] 2   5   3   4   5   8   0   9   1   1   4   7
 [J4,*] 1   5   0   5   2   5   3   3   4   8   5   9
 [J5,*] 2   9   1   3   4   5   5   4   0   3   3   1
 [J6,*] 1   3   3   3   5   9   0   10  4   4   2   1
;
param W
      :=
      [J1] 1
      [J2] 1
      [J3] 1
      [J4] 1
      [J5] 1
      [J6] 1
      ;
/*******************************************
 * SOLUTION AND VISUALIZATION
 *******************************************/
#option solver gurobi;
#option show_stats 1;
solve;
display f_obj;
display C;
display x;
display p;
display d;
display T;
display y;
```

Anexo B - Instância FT06

```
************************************************************

   NEOS Server Version 5.0
   Job#     : 3771156
   Password : QNMfVZsi
   Solver   : milp:Gurobi:AMPL
   Start    : 2015-06-25 08:06:26
   End      : 2015-06-25 08:06:32
   Host     : NEOS HTCondor Pool

   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on neos-4.neos-server.org

Presolve eliminates 6 constraints.
Adjusted problem:
216 variables:
      180 binary variables
      36 linear variables
390 constraints, all linear; 1140 nonzeros
      390 inequality constraints
1 linear objective; 6 nonzeros.

Gurobi 5.5.0: threads=4
outlev=1
Optimize a model with 390 rows, 216 columns and 1140 nonzeros
Found heuristic solution: objective 333
Presolve time: 0.00s
```

```
Presolved: 390 rows, 216 columns, 1140 nonzeros
Variable types: 36 continuous, 180 integer (180 binary)

Root relaxation: objective 1.970000e+02, 116 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  197.00000    0   47  333.00000  197.00000  40.8%     -    0s
     0     0  197.00193    0   46  333.00000  197.00193  40.8%     -    0s
     0     0  205.55616    0   51  333.00000  205.55616  38.3%     -    0s
     0     0  209.93263    0   49  333.00000  209.93263  37.0%     -    0s
     0     0  216.03462    0   46  333.00000  216.03462  35.1%     -    0s
     0     0  216.03462    0   46  333.00000  216.03462  35.1%     -    0s
     0     0  219.31930    0   38  333.00000  219.31930  34.1%     -    0s
H    0     0                      325.0000000  219.31930  32.5%     -    0s
     0     0  219.31930    0   42  325.00000  219.31930  32.5%     -    0s
     0     0  219.35627    0   40  325.00000  219.35627  32.5%     -    0s
     0     0  219.35627    0   42  325.00000  219.35627  32.5%     -    0s
     0     0  219.35627    0   42  325.00000  219.35627  32.5%     -    0s
H    0     0                      323.0000000  219.35627  32.1%     -    0s
     0     0  219.35627    0   26  323.00000  219.35627  32.1%     -    0s
H    0     0                      313.0000000  219.35627  29.9%     -    0s
     0     0  219.35627    0   24  313.00000  219.35627  29.9%     -    0s
     0     0  219.35627    0   25  313.00000  219.35627  29.9%     -    0s
     0     0  219.35627    0   22  313.00000  219.35627  29.9%     -    0s
     0     0  219.35627    0   22  313.00000  219.35627  29.9%     -    0s
     0     0  219.35627    0   24  313.00000  219.35627  29.9%     -    0s
     0     0  219.35627    0   23  313.00000  219.35627  29.9%     -    0s
H    0     0                      310.0000000  219.35627  29.2%     -    0s
     0     0  219.35627    0   23  310.00000  219.35627  29.2%     -    0s
*   82    40              27      306.0000000  226.74163  25.9%   8.3    0s
*   87    40              30      305.0000000  226.74163  25.7%   8.0    0s
*   90    39              31      300.0000000  226.74163  24.4%   7.8    0s
*  182   107              26      294.0000000  227.40000  22.7%   7.0    0s
*  292   132              25      281.0000000  228.67222  18.6%   6.2    0s
* 1615   550              22      280.0000000  243.00000  13.2%   5.8    0s
* 2447   661              26      279.0000000  245.00000  12.2%   6.6    0s
* 2448   618              26      274.0000000  245.00000  10.6%   6.6    0s
H 2915   476                      267.0000000  245.00000  8.24%   7.1    0s
```

```
H 2969   402                      265.0000000  245.00000  7.55%   7.2    0s

Cutting planes:
  Learned: 1
  Gomory: 35
  Implied bound: 103
  MIR: 45

Explored 3652 nodes (29221 simplex iterations) in 0.79 seconds
Thread count was 4 (of 24 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 2.650000000000e+02, best bound 2.650000000000e+02, gap 0.0%
Optimize a model with 390 rows, 216 columns and 1140 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.970084e+02   0.000000e+00      0s
      43    2.6500000e+02   0.000000e+00   0.000000e+00      0s

Solved in 43 iterations and 0.00 seconds
Optimal objective  2.650000000e+02
Gurobi 5.5.0: optimal solution; objective 265
29221 simplex iterations
3652 branch-and-cut nodes
plus 43 simplex iterations for intbasis
f_obj = 265

C [*,*]
:    0    1    2    3    4    5     :=
J1    4   10    1   17   26   20
J2   60   23   28   64   40   50
J3   45   49   15   28   56   36
J4   33   28   38   41   49   59
J5   28   13   10   29   20   25
J6   25    3   31    6   30   15
;

x [J1,*,*] (tr)
:  J2  J3  J4  J5  J6    :=
0   1   1   1   1   1
1   1   1   1   1   0
2   1   1   1   1   1
```

```
3   1   1   1   1   0
4   1   1   1   0   1
5   1   1   1   1   0

 [J2,*,*] (tr)
:  J1  J3  J4  J5  J6    :=
0   0   0   0   0   0
1   0   1   1   0   0
2   0   0   1   0   1
3   0   0   0   0   0
4   0   1   1   0   0
5   0   0   1   0   0

 [J3,*,*] (tr)
:  J1  J2  J4  J5  J6    :=
0   0   1   0   0   0
1   0   0   0   0   0
2   0   1   1   0   1
3   0   1   1   1   0
4   0   0   0   0   0
5   0   1   1   0   0

 [J4,*,*] (tr)
:  J1  J2  J3  J5  J6    :=
0   0   1   1   0   0
1   0   0   1   0   0
2   0   0   0   0   0
3   0   1   0   0   0
4   0   0   1   0   0
5   0   0   0   0   0

 [J5,*,*] (tr)
:  J1  J2  J3  J4  J6    :=
0   0   1   1   1   0
1   0   1   1   1   0
2   0   1   1   1   1
3   0   1   0   1   0
4   1   1   1   1   1
5   0   1   1   1   0

 [J6,*,*] (tr)
```

```
:  J1  J2  J3  J4  J5    :=
0   0   1   1   1   1
1   1   1   1   1   1
2   0   0   0   1   0
3   1   1   1   1   1
4   0   1   1   1   0
5   1   1   1   1   1
;

p [*,*]
:    0    1   2   3   4    5     :=
J1    3   6   1   7    6    3
J2   10   8   5   4   10   10
J3    9   1   5   4    7    8
J4    5   5   5   3    8    9
J5    3   3   9   1    5    4
J6   10   3   1   3    4    9
;

Gurobi 5.5.0: threads=4
outlev=1
threads=4
outlev=1
Optimize a model with 390 rows, 216 columns and 1140 nonzeros
Found heuristic solution: objective 333
Presolve time: 0.00s
Presolved: 390 rows, 216 columns, 1140 nonzeros

Loaded MIP start with objective 265

Variable types: 36 continuous, 180 integer (180 binary)

Root relaxation: objective 1.970000e+02, 116 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |   Obj   Depth IntInf | Incumbent     BestBd    Gap | It/Node
Time

     0     0  197.00000    0   47  265.00000  197.00000  25.7%     -    0s
     0     0  197.00193    0   45  265.00000  197.00193  25.7%     -    0s
     0     0  205.55616    0   51  265.00000  205.55616  22.4%     -    0s
```

```
     0     0  209.93263    0   49  265.00000  209.93263  20.8%     -    0s
     0     0  217.83456    0   45  265.00000  217.83456  17.8%     -    0s
     0     0  219.31930    0   42  265.00000  219.31930  17.2%     -    0s
     0     0  219.35627    0   46  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   44  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   46  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   46  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   46  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   26  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   25  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   24  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   26  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   23  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   24  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   25  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   25  265.00000  219.35627  17.2%     -    0s
     0     0  219.35627    0   25  265.00000  219.35627  17.2%     -    0s

Cutting planes:
  Learned: 2
  Implied bound: 119
  MIR: 25

Explored 2812 nodes (16815 simplex iterations) in 0.30 seconds
Thread count was 4 (of 24 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 2.650000000000e+02, best bound 2.650000000000e+02, gap 0.0%
Optimize a model with 390 rows, 216 columns and 1140 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.970084e+02   0.000000e+00      0s
      43    2.6500000e+02   0.000000e+00   0.000000e+00      0s

Solved in 43 iterations and 0.00 seconds
Optimal objective  2.650000000e+02
Gurobi 5.5.0: optimal solution; objective 265
16815 simplex iterations
2812 branch-and-cut nodes
plus 43 simplex iterations for intbasis
```

Anexo C - Instância FT10

```
************************************************************

   NEOS Server Version 5.0
   Job#     : 3971629
   Password : iUTCqNYP
   Solver   : milp:Gurobi:AMPL
   Start    : 2015-11-08 10:53:17
   End      : 2015-11-08 10:55:22
   Host     : NEOS HTCondor Pool

   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on neos-3.neos-server.org

Presolve eliminates 10 constraints.
Adjusted problem:
1001 variables:
        900 binary variables
        101 linear variables
1900 constraints, all linear; 5600 nonzeros
        1900 inequality constraints
1 linear objective; 1 nonzero.

Gurobi 5.5.0: threads=4
outlev=1
Optimize a model with 1900 rows, 1001 columns and 5600 nonzeros
Presolve time: 0.02s
Presolved: 1900 rows, 1001 columns, 5600 nonzeros
Variable types: 101 continuous, 900 integer (900 binary)
Found heuristic solution: objective 3194.0000000
Found heuristic solution: objective 3185.0000000
Found heuristic solution: objective 3066.0000000

Root relaxation: objective 6.550000e+02, 564 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  655.00000    0  143 3066.00000  655.00000  78.6%     -    0s
H    0     0                    1286.0000000  655.00000  49.1%     -    0s
     0     0  681.00000    0  160 1286.00000  681.00000  47.0%     -    0s
H    0     0                    1267.0000000  681.00000  46.3%     -    0s
     0     0  681.00000    0  182 1267.00000  681.00000  46.3%     -    0s
H    0     0                    1252.0000000  681.00000  45.6%     -    0s
     0     0  681.00000    0  138 1252.00000  681.00000  45.6%     -    0s
H    0     0                    1240.0000000  681.00000  45.1%     -    0s
```

```
H     0     0                    1224.0000000  681.00000  44.4%      -     0s
H     0     0                    1137.0000000  681.00000  40.1%      -     0s
      0     0  681.00000    0  153 1137.00000  681.00000  40.1%      -     0s
H     0     0                    1135.0000000  681.00000  40.0%      -     0s
      0     0  681.00002    0  159 1135.00000  681.00002  40.0%      -     0s
      0     0  681.00002    0  161 1135.00000  681.00002  40.0%      -     0s
      0     0  681.00002    0  160 1135.00000  681.00002  40.0%      -     1s
      0     0  681.00002    0  158 1135.00000  681.00002  40.0%      -     1s
      0     0  681.00065    0  155 1135.00000  681.00065  40.0%      -     1s
      0     0  681.00112    0  142 1135.00000  681.00112  40.0%      -     1s
      0     0  681.01530    0  150 1135.00000  681.01530  40.0%      -     1s
      0     0  681.01530    0  150 1135.00000  681.01530  40.0%      -     1s
      0     0  681.01530    0  147 1135.00000  681.01530  40.0%      -     1s
      0     0  681.01530    0  149 1135.00000  681.01530  40.0%      -     1s
      0     0  681.01530    0  149 1135.00000  681.01530  40.0%      -     1s
      0     0  681.01530    0   76 1135.00000  681.01530  40.0%      -     1s
      0     0  681.82712    0   99 1135.00000  681.82712  39.9%      -     1s
      0     0  682.95122    0   97 1135.00000  682.95122  39.8%      -     1s
      0     0  685.16189    0  106 1135.00000  685.16189  39.6%      -     1s
      0     0  685.16189    0  117 1135.00000  685.16189  39.6%      -     1s
      0     0  685.16189    0   74 1135.00000  685.16189  39.6%      -     1s
      0     0  685.16189    0   84 1135.00000  685.16189  39.6%      -     1s
      0     0  685.16189    0   72 1135.00000  685.16189  39.6%      -     1s
      0     0  685.16189    0   86 1135.00000  685.16189  39.6%      -     1s
      0     0  685.16189    0   86 1135.00000  685.16189  39.6%      -     2s
      0     0  685.16189    0   85 1135.00000  685.16189  39.6%      -     2s
      0     2  685.16189    0   83 1135.00000  685.16189  39.6%      -     2s
H  500   441                    1116.0000000  721.00000  35.4%  19.9     2s
* 1084   896              95    1103.0000000  721.00000  34.6%  18.2     2s
H 1245   989                    1099.0000000  721.00000  34.4%  23.1     3s
H 1277   966                    1054.0000000  721.00000  31.6%  23.5     3s
H 1284   924                    1042.0000000  721.00000  30.8%  23.7     3s
H 1327   903                    1031.0000000  721.00000  30.1%  24.3     3s
H 1423   894                    1022.0000000  721.00000  29.5%  26.2     4s
* 1494   868              62    1021.0000000  721.00000  29.4%  27.3     4s
  1711   907  777.00000   22   96 1021.00000  721.00000  29.4%  30.0     5s
* 1808   884              58    1019.0000000  721.00000  29.2%  31.0     5s
* 2075   897              59    1017.0000000  724.00000  28.8%  33.5     5s
H 2262   945                    1004.0000000  724.00000  27.9%  34.9     6s
* 2433   932              55    1001.0000000  724.00000  27.7%  35.8     6s
* 3112  1042              62     989.0000000  735.00000  25.7%  37.2     7s
* 3261  1107              57     986.0000000  735.00000  25.5%  37.1     8s
* 3638  1243              39     984.0000000  746.00000  24.2%  39.5     8s
H 3698  1265                    983.0000000  746.00000  24.1%  39.8     8s
H 3717  1238                    971.0000000  746.00000  23.2%  39.9     9s
  3876  1288  827.00000   33   83  971.00000  746.00000  23.2%  41.1    10s
* 4337  1469              76     970.0000000  752.00000  22.5%  42.0    10s
H 4741  1601                    964.0000000  757.00000  21.5%  42.1    11s
H 4975  1605                    961.0000000  758.00000  21.1%  44.2    12s
* 5506  1724              51     959.0000000  760.00000  20.8%  47.5    14s
  5618  1755 infeasible   33       959.00000  760.00000  20.8%  48.3    15s
H 6275  1882                    956.0000000  762.00000  20.3%  52.2    17s
  6856  1969  900.00000   35   57  956.00000  766.00000  19.9%  55.8    20s
* 8232  2141              52     955.0000000  775.00000  18.8%  61.4    24s
* 8240  2137              53     953.0000000  775.00000  18.7%  61.4    24s
  8316  2147  902.00000   28  102  953.00000  775.00000  18.7%  61.7    25s
H 8545  2139                    948.0000000  776.00000  18.1%  62.7    26s
  9483  2248  802.00000   34   85  948.00000  779.00000  17.8%  65.8    30s
*10105  2310              41     946.0000000  781.00000  17.4%  67.5    32s
*10215  2283              45     945.0000000  781.00000  17.4%  67.4    32s
*10408  2287              44     944.0000000  782.96192  17.1%  67.9    33s
```

```
 10949  2280  847.00000   36   66  944.00000  786.00000  16.7%  69.5   35s
*12276  2308              58     941.0000000  799.00000  15.1%  72.0   39s
 12485  2352  861.00000   42   49  941.00000  799.00000  15.1%  71.8   40s
H12976  2347                     940.0000000  802.00000  14.7%  72.3   42s
H13047  2291                     930.0000000  802.96667  13.7%  72.4   42s
 13718  2171  865.00000   32   49  930.00000  812.00000  12.7%  73.7   45s
 15168  1542 infeasible   39       930.00000  844.57143  9.19%  76.7   50s

Cutting planes:
  Learned: 9
  Gomory: 97
  Implied bound: 320
  Clique: 1
  MIR: 307

Explored 17664 nodes (1321273 simplex iterations) in 54.60 seconds
Thread count was 4 (of 24 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 9.300000000000e+02, best bound 9.300000000000e+02, gap 0.0%
Optimize a model with 1900 rows, 1001 columns and 5600 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   5.109390e+03   0.000000e+00      0s
     178    9.3000000e+02   0.000000e+00   0.000000e+00      0s

Solved in 178 iterations and 0.01 seconds
Optimal objective  9.300000000e+02
Gurobi 5.5.0: optimal solution; objective 930
1321273 simplex iterations
17664 branch-and-cut nodes
plus 178 simplex iterations for intbasis
f_obj = 930

C [*,*]
:      0     1     2     3     4     5     6     7     8     9     :=
J1    105   523   532   568   617   628   717   777   845   929
J10   361   286   422   727   817   675   429   930   499   628
J2    148   700   314   672   430   813   763   885   930   441
J3    493   399   606   532   930   709   852   721   699   897
J4    256    81   179   825   355   890   364   505   420   847
J5    262   308   193   396   499   369   913   579   520   729
J6    408    86    84   233   505   138   494   530   281   361
J7    185   132   361   294   897   442   396   698   609   516
J8    348   445   224   904   567   535   655   813   727   777
J9     76   201   507   370   727   421   567   668   801   527
;

x [J1,*,*] (tr)
: J10  J2  J3  J4  J5  J6  J7  J8  J9    :=
0   1   1   1   1   1   1   1   1   0
1   0   1   0   0   0   0   0   0   0
2   0   0   1   0   0   0   0   0   0
3   1   1   0   1   0   0   0   1   0
4   1   0   1   0   0   0   1   0   1
5   1   1   1   1   0   0   0   0   0
6   0   1   1   0   1   0   0   0   0
7   1   1   0   0   0   0   0   1   0
8   0   1   0   0   0   0   0   0   0
9   0   0   0   0   0   0   0   0   0

 [J10,*,*] (tr)
```

```
:  J1  J2  J3  J4  J5  J6  J7  J8  J9    :=
0   0   0   1   0   0   1   0   0   0
1   1   1   1   0   1   0   0   1   0
2   1   0   1   0   0   0   0   0   1
3   0   0   0   1   0   0   0   1   0
4   0   0   1   0   0   0   1   0   0
5   0   1   1   1   0   0   0   0   0
6   1   1   1   0   1   1   0   1   1
7   0   0   0   0   0   0   0   0   0
8   1   1   1   0   1   0   1   1   1
9   1   0   1   1   1   0   0   1   0

 [J2,*,*] (tr)
:  J1 J10  J3  J4  J5  J6  J7  J8  J9    :=
0   0   1   1   1   1   1   1   1   0
1   0   0   0   0   0   0   0   0   0
2   1   1   1   0   0   0   1   0   1
3   0   1   0   1   0   0   0   1   0
4   1   1   1   0   1   1   1   1   1
5   0   0   0   1   0   0   0   0   0
6   0   0   1   0   1   0   0   0   0
7   0   1   0   0   0   0   0   0   0
8   0   0   0   0   0   0   0   0   0
9   1   1   1   1   1   0   1   1   1

 [J3,*,*] (tr)
:  J1 J10  J2  J4  J5  J6  J7  J8  J9    :=
0   0   0   0   0   0   0   0   0   0
1   1   0   1   0   0   0   0   1   0
2   0   0   0   0   0   0   0   0   0
3   1   1   1   1   0   0   0   1   0
4   0   0   0   0   0   0   0   0   0
5   0   0   1   1   0   0   0   0   0
6   0   0   0   0   1   0   0   0   0
7   1   1   1   0   0   0   0   1   0
8   1   0   1   0   0   0   0   1   1
9   1   0   0   0   0   0   0   0   0

 [J4,*,*] (tr)
:  J1 J10  J2  J3  J5  J6  J7  J8  J9    :=
0   0   1   0   1   1   1   0   1   0
1   1   1   1   1   1   1   1   1   1
2   1   1   1   1   1   0   1   1   1
3   0   0   0   0   0   0   0   1   0
4   1   1   1   1   1   1   1   1   1
5   0   0   0   0   0   0   0   0   0
6   1   1   1   1   1   1   1   1   1
7   1   1   1   1   1   1   1   1   1
8   1   1   1   1   1   0   1   1   1
9   1   0   0   1   0   0   0   0   0

 [J5,*,*] (tr)
:  J1 J10  J2  J3  J4  J6  J7  J8  J9    :=
0   0   1   0   1   0   1   0   1   0
1   1   0   1   1   0   0   0   1   0
2   1   1   1   1   0   0   1   1   1
3   1   1   1   1   1   0   0   1   0
4   1   1   0   1   0   1   1   1   1
5   1   1   1   1   1   0   1   1   1
6   0   0   0   0   0   0   0   0   0
7   1   1   1   1   0   0   1   1   1
```

```
8   1   0   1   1   0   0   1   1   1
9   1   0   0   1   1   0   0   1   0

 [J6,*,*] (tr)
:  J1 J10  J2  J3  J4  J5  J7  J8  J9    :=
0   0   0   0   1   0   0   0   0   0
1   1   1   1   1   0   1   1   1   1
2   1   1   1   1   1   1   1   1   1
3   1   1   1   1   1   1   1   1   1
4   1   1   0   1   0   0   1   1   1
5   1   1   1   1   1   1   1   1   1
6   1   0   1   1   0   1   0   1   1
7   1   1   1   1   0   1   1   1   1
8   1   1   1   1   1   1   1   1   1
9   1   1   1   1   1   1   1   1   1

 [J7,*,*] (tr)
:  J1 J10  J2  J3  J4  J5  J6  J8  J9    :=
0   0   1   0   1   1   1   1   1   0
1   1   1   1   1   0   1   0   1   1
2   1   1   0   1   0   0   0   0   1
3   1   1   1   1   1   1   0   1   1
4   0   0   0   1   0   0   0   0   0
5   1   1   1   1   1   0   0   1   0
6   1   1   1   1   0   1   1   1   1
7   1   1   1   1   0   0   0   1   0
8   1   0   1   1   0   0   0   1   1
9   1   1   0   1   1   1   0   1   1

 [J8,*,*] (tr)
:  J1 J10  J2  J3  J4  J5  J6  J7  J9    :=
0   0   1   0   1   0   0   1   0   0
1   1   0   1   0   0   0   0   0   0
2   1   1   1   1   0   0   0   1   1
3   0   0   0   0   0   0   0   0   0
4   1   1   0   1   0   0   0   1   1
5   1   1   1   1   1   0   0   0   0
6   1   0   1   1   0   1   0   0   0
7   0   1   1   0   0   0   0   0   0
8   1   0   1   0   0   0   0   0   1
9   1   0   0   1   1   0   0   0   0

 [J9,*,*] (tr)
:  J1 J10  J2  J3  J4  J5  J6  J7  J8    :=
0   1   1   1   1   1   1   1   1   1
1   1   1   1   1   0   1   0   0   1
2   1   0   0   1   0   0   0   0   0
3   1   1   1   1   1   1   0   0   1
4   0   1   0   1   0   0   0   1   0
5   1   1   1   1   1   0   0   1   1
6   1   0   1   1   0   1   0   0   1
7   1   1   1   1   0   0   0   1   1
8   1   0   1   0   0   0   0   0   0
9   1   1   0   1   1   1   0   0   1
;

p [*,*]
:     0    1    2    3    4    5    6    7    8    9     :=
J1    29   78    9   36   49   11   62   56   44   21
J10   13   85   61   52   90   47    7   45   64   76
J2    43   28   90   69   75   46   46   72   30   11
```

```
J3    85   91   74   39   33   10   89   12   90   45
J4    71   81   95   98   99   43    9   85   52   22
J5     6   22   14   26   69   61   53   49   21   72
J6    47    2   84   95    6   52   65   25   48   72
J7    37   46   13   61   55   21   32   30   89   32
J8    86   46   31   79   32   74   88   36   19   48
J9    76   69   85   76   26   51   40   89   74   11
;

Gurobi 5.5.0: threads=4
outlev=1
threads=4
outlev=1
Optimize a model with 1900 rows, 1001 columns and 5600 nonzeros
Presolve time: 0.01s
Presolved: 1900 rows, 1001 columns, 5600 nonzeros

Loaded MIP start with objective 930

Variable types: 101 continuous, 900 integer (900 binary)

Root relaxation: objective 6.550000e+02, 548 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  655.00000    0  166  930.00000  655.00000  29.6%     -    0s
     0     0  681.00000    0  144  930.00000  681.00000  26.8%     -    0s
     0     0  681.00000    0  146  930.00000  681.00000  26.8%     -    0s
     0     0  681.00000    0  172  930.00000  681.00000  26.8%     -    0s
H    0     0                       929.9999999  681.00000  26.8%     -    0s
     0     0  681.00000    0  156  930.00000  681.00000  26.8%     -    0s
     0     0  681.00000    0  166  930.00000  681.00000  26.8%     -    0s
     0     0  681.00000    0  139  930.00000  681.00000  26.8%     -    0s
     0     0  681.00000    0  126  930.00000  681.00000  26.8%     -    0s
     0     0  681.00282    0  136  930.00000  681.00282  26.8%     -    0s
     0     0  681.00406    0  140  930.00000  681.00406  26.8%     -    0s
     0     0  681.00406    0  139  930.00000  681.00406  26.8%     -    0s
     0     0  681.00406    0   78  930.00000  681.00406  26.8%     -    1s
     0     0  686.15717    0  107  930.00000  686.15717  26.2%     -    1s
     0     0  686.15717    0  100  930.00000  686.15717  26.2%     -    1s
     0     0  686.15717    0  102  930.00000  686.15717  26.2%     -    1s
     0     0  686.15717    0   86  930.00000  686.15717  26.2%     -    1s
     0     0  686.15717    0   90  930.00000  686.15717  26.2%     -    1s
     0     0  686.23107    0   92  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   91  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   96  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   95  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   94  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   84  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   97  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   98  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   89  930.00000  686.23107  26.2%     -    1s
     0     0  686.23107    0   86  930.00000  686.23107  26.2%     -    2s
     0     0  686.23107    0   85  930.00000  686.23107  26.2%     -    2s
     0     0  686.23107    0   85  930.00000  686.23107  26.2%     -    2s
     0     0  686.23107    0   85  930.00000  686.23107  26.2%     -    2s
     0     2  686.23107    0   85  930.00000  686.23107  26.2%     -    2s
  1497   745 infeasible   27       930.00000  727.00000  21.8%  32.4    5s
  2815   707 infeasible   36       930.00000  746.00000  19.8%  52.8   10s
```

```
  3800   691 infeasible   29       930.00000  753.00000  19.0%  64.4   15s
  5140   836 infeasible   35       930.00000  774.00000  16.8%  71.6   20s
  6377   967  880.00000   28   48  930.00000  780.00000  16.1%  75.1   25s
  7749   974  847.00000   28   61  930.00000  791.00000  14.9%  79.2   30s
  8935   956  869.00000   38   69  930.00000  803.00000  13.7%  80.6   35s
 10028   942  808.00000   29   88  930.00000  808.00000  13.1%  81.4   40s
 11450   751  856.00000   36   50  930.00000  822.00000  11.6%  83.1   45s
 12820   579  833.00000   22   63  930.00000  833.00000  10.4%  82.9   50s
 14641   164 infeasible   41       930.00000  869.00000  6.56%  82.1   55s

Cutting planes:
  Learned: 8
  Gomory: 71
  Implied bound: 409
  Clique: 6
  MIR: 229

Explored 15215 nodes (1254631 simplex iterations) in 56.26 seconds
Thread count was 4 (of 24 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 9.299999999067e+02, best bound 9.299999999067e+02, gap 0.0%
Optimize a model with 1900 rows, 1001 columns and 5600 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   5.109392e+03   0.000000e+00      0s
     182    9.3000000e+02   0.000000e+00   0.000000e+00      0s

Solved in 182 iterations and 0.01 seconds
Optimal objective  9.300000000e+02
Gurobi 5.5.0: optimal solution; objective 929.9999999
1254631 simplex iterations
15215 branch-and-cut nodes
plus 182 simplex iterations for intbasis
```

Anexo D - Instância LA01

```
************************************************************

   NEOS Server Version 5.0
   Job#     : 3971721
   Password : rdpwanxz
   Solver   : milp:Gurobi:AMPL
   Start    : 2015-11-08 12:37:02
   End      : 2015-11-08 12:37:22
   Host     : NEOS HTCondor Pool

   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on neos-3.neos-server.org

Presolve eliminates 10 constraints.
Adjusted problem:
501 variables:
        450 binary variables
        51 linear variables
950 constraints, all linear; 2800 nonzeros
        950 inequality constraints
1 linear objective; 1 nonzero.

Gurobi 5.5.0: threads=4
outlev=1
Optimize a model with 950 rows, 501 columns and 2800 nonzeros
Presolve time: 0.01s
Presolved: 950 rows, 501 columns, 2800 nonzeros
Variable types: 51 continuous, 450 integer (450 binary)
Found heuristic solution: objective 2443.0000000
Found heuristic solution: objective 2272.0000000
Found heuristic solution: objective 2263.0000000

Root relaxation: objective 4.130000e+02, 297 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  413.00000    0   81 2263.00000  413.00000  81.7%     -    0s
     0     0  463.00000    0   68 2263.00000  463.00000  79.5%     -    0s
H    0     0                     785.0000000  463.00000  41.0%     -    0s
     0     0  463.00000    0   68  785.00000  463.00000  41.0%     -    0s
     0     0  463.00000    0   62  785.00000  463.00000  41.0%     -    0s
     0     0  463.00000    0   35  785.00000  463.00000  41.0%     -    0s
H    0     0                     746.0000000  463.00000  37.9%     -    0s
H    0     0                     740.0000000  463.00000  37.4%     -    0s
```

```
     0     0  463.00000    0   62  740.00000  463.00000  37.4%     -    0s
H    0     0                    737.0000000  463.00000  37.2%     -    0s
     0     0  463.00000    0   50  737.00000  463.00000  37.2%     -    0s
     0     0  463.00000    0   39  737.00000  463.00000  37.2%     -    0s
H    0     0                    733.0000000  463.00000  36.8%     -    0s
     0     0  463.00000    0   38  733.00000  463.00000  36.8%     -    0s
H    0     0                    727.0000000  463.00000  36.3%     -    0s
H    0     0                    684.0000000  463.00000  32.3%     -    0s
     0     0  463.00000    0   32  684.00000  463.00000  32.3%     -    0s
H    0     0                    681.0000000  463.00000  32.0%     -    0s
H    0     0                    678.0000000  463.00000  31.7%     -    0s
     0     0  463.00000    0   62  678.00000  463.00000  31.7%     -    0s
     0     0  472.83331    0   45  678.00000  472.83331  30.3%     -    0s
H    0     0                    675.0000000  472.83331  30.0%     -    0s
     0     0  472.83331    0   58  675.00000  472.83331  30.0%     -    0s
     0     0  472.83331    0   33  675.00000  472.83331  30.0%     -    0s
     0     0  472.83331    0   42  675.00000  472.83331  30.0%     -    0s
H    0     0                    670.0000000  472.83331  29.4%     -    0s
     0     0  472.83331    0   50  670.00000  472.83331  29.4%     -    1s
     0     0  472.83331    0   47  670.00000  472.83331  29.4%     -    1s
     0     0  472.83331    0   26  670.00000  472.83331  29.4%     -    1s
H    0     0                    669.0000000  472.83331  29.3%     -    1s
     0     0  472.83331    0   50  669.00000  472.83331  29.3%     -    1s
     0     0  472.83331    0   26  669.00000  472.83331  29.3%     -    1s
     0     0  472.83331    0   36  669.00000  472.83331  29.3%     -    1s
     0     0  472.83331    0   46  669.00000  472.83331  29.3%     -    1s
     0     0  472.83331    0   50  669.00000  472.83331  29.3%     -    1s
     0     0  472.83331    0   31  669.00000  472.83331  29.3%     -    1s
     0     0  472.83331    0   36  669.00000  472.83331  29.3%     -    1s
H    0     0                    666.0000000  472.83331  29.0%     -    1s
     0     0  472.83331    0   35  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   30  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   32  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   39  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   30  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   30  666.00000  472.83331  29.0%     -    1s

Cutting planes:
  Implied bound: 1
  MIR: 3

Explored 22883 nodes (161491 simplex iterations) in 4.76 seconds
Thread count was 4 (of 24 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 6.660000000000e+02, best bound 6.660000000000e+02, gap 0.0%
Optimize a model with 950 rows, 501 columns and 2800 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.849252e+03   0.000000e+00      0s
      92    6.6600000e+02   0.000000e+00   0.000000e+00      0s

Solved in 92 iterations and 0.00 seconds
Optimal objective  6.660000000e+02
Gurobi 5.5.0: optimal solution; objective 666
161491 simplex iterations
22883 branch-and-cut nodes
plus 92 simplex iterations for intbasis
f_obj = 666

C [*,*]
:      0     1     2     3     4     :=
```

```
J1    121    21   433   399   249
J10   561   465   247   204    77
J2     21   666   525   344   448
J3    666   595   654   125   546
J4    267   212   499   604   432
J5    204   553   311   247   666
J6    465    86   204   666   353
J7    654   311   398    69   154
J8    327   368   161   527   629
J9    373   135   623    86   274
;

x [*,*,0]
:    J1 J10  J2  J3  J4  J5  J6  J7  J8  J9    :=
J1    .   1   0   1   1   1   1   1   1   1
J10   0   .   0   1   0   0   0   1   0   0
J2    1   1   .   1   1   1   1   1   1   1
J3    0   0   0   .   0   0   0   0   0   0
J4    0   1   0   1   .   0   1   1   1   1
J5    0   1   0   1   1   .   1   1   1   1
J6    0   1   0   1   0   0   .   1   0   0
J7    0   0   0   1   0   0   0   .   0   0
J8    0   1   0   1   0   0   1   1   .   1
J9    0   1   0   1   0   0   1   1   0   .

 [*,*,1]
:    J1 J10  J2  J3  J4  J5  J6  J7  J8  J9    :=
J1    .   1   1   1   1   1   1   1   1   1
J10   0   .   1   1   0   1   0   0   0   0
J2    0   0   .   0   0   0   0   0   0   0
J3    0   0   1   .   0   0   0   0   0   0
J4    0   1   1   1   .   1   0   1   1   0
J5    0   0   1   1   0   .   0   0   0   0
J6    0   1   1   1   1   1   .   1   1   1
J7    0   1   1   1   0   1   0   .   1   0
J8    0   1   1   1   0   1   0   0   .   0
J9    0   1   1   1   1   1   0   1   1   .

 [*,*,2]
:    J1 J10  J2  J3  J4  J5  J6  J7  J8  J9    :=
J1    .   0   1   1   1   0   0   0   0   1
J10   1   .   1   1   1   1   0   1   0   1
J2    0   0   .   1   0   0   0   0   0   1
J3    0   0   0   .   0   0   0   0   0   0
J4    0   0   1   1   .   0   0   0   0   1
J5    1   0   1   1   1   .   0   1   0   1
J6    1   1   1   1   1   1   .   1   0   1
J7    1   0   1   1   1   0   0   .   0   1
J8    1   1   1   1   1   1   1   1   .   1
J9    0   0   0   1   0   0   0   0   0   .

 [*,*,3]
:    J1 J10  J2  J3  J4  J5  J6  J7  J8  J9    :=
J1    .   0   0   0   1   0   1   0   1   0
J10   1   .   1   0   1   1   1   0   1   0
J2    1   0   .   0   1   0   1   0   1   0
J3    1   1   1   .   1   1   1   0   1   0
J4    0   0   0   0   .   0   1   0   0   0
J5    1   0   1   0   1   .   1   0   1   0
J6    0   0   0   0   0   0   .   0   0   0
J7    1   1   1   1   1   1   1   .   1   1
```

```
J8    0   0   0   0   1   0   1   0   .   0
J9    1   1   1   1   1   1   1   0   1   .

 [*,*,4]
:    J1 J10  J2  J3  J4  J5  J6  J7  J8  J9    :=
J1    .   0   1   1   1   1   1   0   1   1
J10   1   .   1   1   1   1   1   1   1   1
J2    0   0   .   1   0   1   0   0   1   0
J3    0   0   0   .   0   1   0   0   1   0
J4    0   0   1   1   .   1   0   0   1   0
J5    0   0   0   0   0   .   0   0   0   0
J6    0   0   1   1   1   1   .   0   1   0
J7    1   0   1   1   1   1   1   .   1   1
J8    0   0   0   0   0   1   0   0   .   0
J9    0   0   1   1   1   1   1   0   1   .
;

p [*,*]
:     0    1    2    3    4     :=
J1    53   21   34   55   95
J10   96   75   43   79   77
J2    21   71   26   52   16
J3    12   42   31   39   98
J4    55   77   66   77   79
J5    83   19   64   34   37
J6    92   54   43   62   79
J7    93   87   87   69   77
J8    60   41   38   24   83
J9    44   49   98   17   25
;

Gurobi 5.5.0: threads=4
outlev=1
threads=4
outlev=1
Optimize a model with 950 rows, 501 columns and 2800 nonzeros
Presolve time: 0.00s
Presolved: 950 rows, 501 columns, 2800 nonzeros

Loaded MIP start with objective 666

Variable types: 51 continuous, 450 integer (450 binary)

Root relaxation: objective 4.130000e+02, 294 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  413.00000    0   79  666.00000  413.00000  38.0%     -    0s
     0     0  463.00000    0   86  666.00000  463.00000  30.5%     -    0s
     0     0  463.00000    0   94  666.00000  463.00000  30.5%     -    0s
     0     0  463.00000    0   54  666.00000  463.00000  30.5%     -    0s
     0     0  463.00000    0   56  666.00000  463.00000  30.5%     -    0s
H    0     0                      665.9999999  463.00000  30.5%     -    0s
     0     0  463.00000    0   58  666.00000  463.00000  30.5%     -    0s
     0     0  463.00001    0   58  666.00000  463.00001  30.5%     -    0s
     0     0  463.01563    0   52  666.00000  463.01563  30.5%     -    0s
     0     0  463.01563    0   60  666.00000  463.01563  30.5%     -    0s
     0     0  463.01563    0   60  666.00000  463.01563  30.5%     -    0s
     0     0  463.01563    0   24  666.00000  463.01563  30.5%     -    0s
```

```
     0     0  463.01563    0   44  666.00000  463.01563  30.5%     -    0s
     0     0  466.79564    0   37  666.00000  466.79564  29.9%     -    0s
     0     0  472.83331    0   53  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   52  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   50  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   45  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   56  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   32  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   27  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   39  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   30  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   31  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   34  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   34  666.00000  472.83331  29.0%     -    0s
     0     0  472.83331    0   30  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   54  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   30  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   34  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   35  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   32  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   29  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   33  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   35  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   26  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   31  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   25  666.00000  472.83331  29.0%     -    1s
     0     0  472.83331    0   25  666.00000  472.83331  29.0%     -    1s
 29321  3688     cutoff   31       666.00000  625.00000  6.16%   6.2    5s

Cutting planes:
  Gomory: 2
  Implied bound: 2
  MIR: 9

Explored 46389 nodes (271503 simplex iterations) in 7.94 seconds
Thread count was 4 (of 24 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 6.660000000000e+02, best bound 6.660000000000e+02, gap 0.0%
Optimize a model with 950 rows, 501 columns and 2800 nonzeros
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.849252e+03   0.000000e+00      0s
      92    6.6600000e+02   0.000000e+00   0.000000e+00      0s

Solved in 92 iterations and 0.00 seconds
Optimal objective  6.660000000e+02
Gurobi 5.5.0: optimal solution; objective 666
271503 simplex iterations
46389 branch-and-cut nodes
plus 92 simplex iterations for intbasis
```

Anexo E - Instância LA06

```
************************************************************

   NEOS Server Version 5.0
   Job#     : 3971746
   Password : tzLjmJsq


   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
Job 3971746 has finished.
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on neos-3.neos-server.org

Presolve eliminates 15 constraints.
Adjusted problem:
1126 variables:
        1050 binary variables
        76 linear variables
2175 constraints, all linear; 6450 nonzeros
        2175 inequality constraints
1 linear objective; 1 nonzero.

Gurobi 5.5.0: threads=4
outlev=1
Optimize a model with 2175 rows, 1126 columns and 6450 nonzeros
Presolve time: 0.01s
Presolved: 2175 rows, 1126 columns, 6450 nonzeros
Variable types: 76 continuous, 1050 integer (1050 binary)
Found heuristic solution: objective 3341.0000000
Found heuristic solution: objective 2974.0000000
Found heuristic solution: objective 2968.0000000

Root relaxation: objective 4.130000e+02, 689 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  413.00000    0  192 2968.00000  413.00000  86.1%     -    0s
     0     0  463.00000    0  193 2968.00000  463.00000  84.4%     -    0s
H    0     0                    2442.0000000  463.00000  81.0%     -    0s
     0     0  463.00000    0  204 2442.00000  463.00000  81.0%     -    0s
     0     0  463.00000    0  153 2442.00000  463.00000  81.0%     -    0s
H    0     0                    1030.0000000  463.00000  55.0%     -    0s
H    0     0                    1014.0000000  463.00000  54.3%     -    0s
H    0     0                     977.0000000  463.00000  52.6%     -    0s
     0     0  463.00000    0  162  977.00000  463.00000  52.6%     -    0s
     0     0  463.00000    0  168  977.00000  463.00000  52.6%     -    0s
```

```
     0     0  463.00000    0  147  977.00000  463.00000  52.6%      -    0s
     0     0  463.00000    0  143  977.00000  463.00000  52.6%      -    0s
H    0     0                      968.0000000  463.00000  52.2%      -    0s
     0     0  463.00000    0  153  968.00000  463.00000  52.2%      -    0s
     0     0  463.00000    0  158  968.00000  463.00000  52.2%      -    0s
     0     0  463.00000    0  156  968.00000  463.00000  52.2%      -    0s
H    0     0                      949.0000000  463.00000  51.2%      -    1s
H    0     0                      926.0000000  463.00000  50.0%      -    1s
     0     0  463.00000    0   74  926.00000  463.00000  50.0%      -    1s
     0     0  463.00000    0  130  926.00000  463.00000  50.0%      -    1s
     0     0  463.00000    0   90  926.00000  463.00000  50.0%      -    1s
     0     0  466.27648    0   97  926.00000  466.27648  49.6%      -    1s
     0     0  468.80763    0  103  926.00000  468.80763  49.4%      -    1s
     0     0  468.80763    0  104  926.00000  468.80763  49.4%      -    1s
     0     0  468.80763    0   78  926.00000  468.80763  49.4%      -    1s
     0     0  468.80763    0   87  926.00000  468.80763  49.4%      -    1s
     0     0  468.80763    0   91  926.00000  468.80763  49.4%      -    1s
     0     0  468.80763    0   83  926.00000  468.80763  49.4%      -    2s
     0     0  468.80763    0   96  926.00000  468.80763  49.4%      -    2s
     0     0  468.80763    0   95  926.00000  468.80763  49.4%      -    2s
     0     2  468.80763    0   92  926.00000  468.80763  49.4%      -    2s
  1996  1426  543.00000   20   94  926.00000  507.29276  45.2%  19.8    5s
  8980  5782  768.00000   27   67  926.00000  517.00000  44.2%  19.2   10s
 14601  9543     cutoff   46       926.00000  530.00000  42.8%  20.7   15s
 19690 12202  539.00000   20   80  926.00000  533.00000  42.4%  22.8   20s

                                    (…)

 19085906 9709873  820.00000   46   51  926.00000  618.00000  33.3%   9.5
4760s
 19106007 9721905  820.00000   57   41  926.00000  618.00000  33.3%   9.5
4765s
 19132161 9737087  856.00000   55   37  926.00000  618.00000  33.3%   9.5
4770s
 19158274 9750682  910.00000   51   34  926.00000  618.00000  33.3%   9.5
4775s
 19185595 9766159  866.00000   65   42  926.00000  618.00000  33.3%   9.5
4780s
 19212795 9782064 infeasible   63       926.00000  618.00000  33.3%   9.5
4785s
 19239852 9797334  713.00000   54   57  926.00000  618.00000  33.3%   9.5
4790s
 19263751 9811085  873.00000   64   43  926.00000  618.00000  33.3%   9.5
4795s
 19290394 9825876  866.00000   68   41  926.00000  618.00000  33.3%   9.5
4800s
 19317331 9839743  821.00000   58   46  926.00000  618.00000  33.3%   9.5
4805s
 19344264 9854697 infeasible   55       926.00000  618.00000  33.3%   9.5
4810s
```

Anexo F - Instância LA11

```
************************************************************

   NEOS Server Version 5.0
   Job#     : 3977543
   Password : GJqkfiNE


   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
Job 3977543 has finished.
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on neos-5.neos-server.org

Presolve eliminates 20 constraints.
Adjusted problem:
2001 variables:
       1900 binary variables
       101 linear variables
3900 constraints, all linear; 11600 nonzeros
       3900 inequality constraints
1 linear objective; 1 nonzero.

Gurobi 5.5.0: threads=4
outlev=1
Optimize a model with 3900 rows, 2001 columns and 11600 nonzeros
Presolve time: 0.02s
Presolved: 3900 rows, 2001 columns, 11600 nonzeros
Variable types: 101 continuous, 1900 integer (1900 binary)
Found heuristic solution: objective 4118.0000000
Found heuristic solution: objective 3891.0000000
Found heuristic solution: objective 3890.0000000

Root relaxation: objective 4.130000e+02, 1275 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  413.00000    0  402 3890.00000  413.00000  89.4%     -    0s
     0     0  463.00000    0  368 3890.00000  463.00000  88.1%     -    0s
     0     0  463.00000    0  358 3890.00000  463.00000  88.1%     -    0s
     0     0  463.00000    0  367 3890.00000  463.00000  88.1%     -    0s
     0     0  463.00000    0  381 3890.00000  463.00000  88.1%     -    0s
H    0     0                    1471.0000000  463.00000  68.5%     -    0s
     0     0  463.00000    0  386 1471.00000  463.00000  68.5%     -    0s
H    0     0                    1280.0000000  463.00000  63.8%     -    1s
     0     0  463.00000    0  342 1280.00000  463.00000  63.8%     -    1s
     0     0  463.00000    0  335 1280.00000  463.00000  63.8%     -    1s
```

```
     0     0  463.00000    0  344 1280.00000  463.00000  63.8%      -    1s
     0     0  463.00000    0  343 1280.00000  463.00000  63.8%      -    1s
     0     0  463.00000    0  348 1280.00000  463.00000  63.8%      -    1s
     0     0  463.00000    0  348 1280.00000  463.00000  63.8%      -    2s
     0     0  463.00000    0  178 1280.00000  463.00000  63.8%      -    2s
H    0     0                    1263.0000000  463.00000  63.3%      -    2s
     0     0  464.62936    0  285 1263.00000  464.62936  63.2%      -    2s
     0     0  475.77529    0  263 1263.00000  475.77529  62.3%      -    3s
     0     0  475.77529    0  287 1263.00000  475.77529  62.3%      -    3s
     0     0  475.77529    0  235 1263.00000  475.77529  62.3%      -    3s
     0     0  475.77529    0  249 1263.00000  475.77529  62.3%      -    3s
     0     0  475.77529    0  198 1263.00000  475.77529  62.3%      -    3s
     0     0  475.77529    0  232 1263.00000  475.77529  62.3%      -    4s
     0     0  475.77529    0  253 1263.00000  475.77529  62.3%      -    4s
     0     0  475.77529    0  205 1263.00000  475.77529  62.3%      -    4s
     0     0  475.77529    0  180 1263.00000  475.77529  62.3%      -    5s
     0     0  475.77529    0  244 1263.00000  475.77529  62.3%      -    5s
     0     0  475.77529    0  218 1263.00000  475.77529  62.3%      -    5s
     0     0  475.77529    0  226 1263.00000  475.77529  62.3%      -    5s
     0     0  475.77529    0  249 1263.00000  475.77529  62.3%      -    5s
     0     0  475.77529    0  194 1263.00000  475.77529  62.3%      -    5s
     0     0  475.77529    0  196 1263.00000  475.77529  62.3%      -    6s
     0     0  475.77529    0  174 1263.00000  475.77529  62.3%      -    6s
     0     0  475.77529    0  174 1263.00000  475.77529  62.3%      -    6s
     0     0  475.77529    0  174 1263.00000  475.77529  62.3%      -    6s
     0     0  475.77529    0  205 1263.00000  475.77529  62.3%      -    6s
     0     0  475.77529    0  173 1263.00000  475.77529  62.3%      -    7s
     0     0  475.77529    0  218 1263.00000  475.77529  62.3%      -    7s
     0     0  475.77529    0  211 1263.00000  475.77529  62.3%      -    7s
     0     0  475.77529    0  162 1263.00000  475.77529  62.3%      -    7s
H    0     0                    1230.0000000  475.77529  61.3%      -    8s
H    0     0                    1229.0000000  475.77529  61.3%      -    8s
     0     0  475.77529    0  225 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  222 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  218 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  226 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  196 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  178 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  176 1229.00000  475.77529  61.3%      -    8s
     0     0  475.77529    0  198 1229.00000  475.77529  61.3%      -    9s
H    0     0                    1226.0000000  475.77529  61.2%      -    9s
     0     0  475.77529    0  200 1226.00000  475.77529  61.2%      -    9s
     0     0  475.77529    0  172 1226.00000  475.77529  61.2%      -    9s
     0     2  475.77529    0  172 1226.00000  475.77529  61.2%      -    9s
    87    87  552.00000   28  158 1226.00000  492.00000  59.9%  53.7   10s
H   90    87                    1222.0000000  492.00000  59.7%  52.8   10s
  1707  1506  567.00000   33  169 1222.00000  492.60076  59.7%  29.1   15s
  2809  2183  492.60076   20  293 1222.00000  492.60076  59.7%  26.4   20s
  5010  3249  930.00000   44   99 1222.00000  492.60076  59.7%  27.5   25s
  6491  4215  507.00000   24  237 1222.00000  492.60076  59.7%  28.9   30s
  8243  5488 1130.00000  147   29 1222.00000  492.60150  59.7%  29.7   35s
 10234  7032 1027.00000  100  222 1222.00000  492.61553  59.7%  28.4   40s
 10255  7046  796.00000   54  283 1222.00000  494.61948  59.5%  28.3   45s
 10291  7073  532.00000   35  139 1222.00000  509.00000  58.3%  33.3   50s
 12568  8386 infeasible  142      1222.00000  519.00000  57.5%  30.1   55s

                                    (…)

 12331928 8628334  718.00000   72  118 1222.00000  605.00000  50.5%  10.2
5670s
```

```
 12340424 8634161 1189.00000  106   51 1222.00000  605.00000  50.5%  10.2
5675s
 12349653 8640293 1106.00000   91   82 1222.00000  605.00000  50.5%  10.2
5680s
 12359175 8646836  842.00000   71  111 1222.00000  605.00000  50.5%  10.2
5685s
 12367074 8652135 1046.00000   86   79 1222.00000  605.00000  50.5%  10.2
5690s
 12374248 8657191  735.00000   67  126 1222.00000  605.00000  50.5%  10.2
5695s
 12381967 8662471 1106.00000   90   63 1222.00000  605.00000  50.5%  10.2
5700s
 12391111 8668876  796.00000   76  124 1222.00000  605.00000  50.5%  10.2
5705s
```

Anexo G - Instância LA21

```
************************************************************

          NEOS Server Version 5.0

   Disclaimer:

          This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
Job 3970906 has finished.
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on neos-4.neos-server.org

Presolve eliminates 15 constraints.
Adjusted problem:
2251 variables:
        2100 binary variables
        151 linear variables
4350 constraints, all linear; 12900 nonzeros
        4350 inequality constraints
1 linear objective; 1 nonzero.

       Gurobi 5.5.0: threads=4
outlev=1
Optimize a model with 4350 rows, 2251 columns and 12900 nonzeros
Presolve time: 0.03s
Presolved: 4350 rows, 2251 columns, 12900 nonzeros
Variable types: 151 continuous, 2100 integer (2100 binary)
Found heuristic solution: objective 5944.0000000
Found heuristic solution: objective 5931.0000000
Found heuristic solution: objective 5841.0000000

       Root  relaxation:  objective  7.170000e+02,  1245  iterations,  0.01
seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node
Time

     0     0  717.00000    0  221 5841.00000  717.00000  87.7%     -    0s
     0     0  717.00000    0  246 5841.00000  717.00000  87.7%     -    0s
     0     0  717.00000    0  212 5841.00000  717.00000  87.7%     -    0s
H    0     0                    1401.0000000  717.00000  48.8%     -    0s
H    0     0                    1294.0000000  717.00000  44.6%     -    0s
     0     0  717.00000    0  213 1294.00000  717.00000  44.6%     -    1s
     0     0  717.00000    0  212 1294.00000  717.00000  44.6%     -    1s
H    0     0                    1290.0000000  717.00000  44.4%     -    1s
H    0     0                    1280.0000000  717.00000  44.0%     -    1s
```

```
     0     0  717.00000    0  224 1280.00000  717.00000  44.0%     -    1s
H    0     0                    1218.0000000  717.00000  41.1%     -    1s
     0     0  717.00000    0  215 1218.00000  717.00000  41.1%     -    1s
     0     0  717.00000    0  214 1218.00000  717.00000  41.1%     -    1s
     0     0  717.19876    0  224 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  246 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  224 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  243 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  238 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  232 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  229 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  215 1218.00000  717.19876  41.1%     -    2s
     0     0  717.19876    0  224 1218.00000  717.19876  41.1%     -    3s
     0     0  717.19876    0  208 1218.00000  717.19876  41.1%     -    3s
     0     0  717.19876    0  201 1218.00000  717.19876  41.1%     -    3s
     0     0  717.19876    0  201 1218.00000  717.19876  41.1%     -    3s
     0     0  717.19876    0  204 1218.00000  717.19876  41.1%     -    3s
     0     0  717.19876    0  204 1218.00000  717.19876  41.1%     -    3s
     0     0  717.19876    0  128 1218.00000  717.19876  41.1%     -    4s
     0     0  719.00000    0  187 1218.00000  719.00000  41.0%     -    4s
     0     0  724.16998    0  200 1218.00000  724.16998  40.5%     -    5s
     0     0  724.16998    0  280 1218.00000  724.16998  40.5%     -    5s
     0     0  724.16998    0  111 1218.00000  724.16998  40.5%     -    5s
     0     0  724.16998    0  120 1218.00000  724.16998  40.5%     -    5s
H    0     0                    1201.0000000  724.16998  39.7%     -    5s
     0     0  724.16998    0  119 1201.00000  724.16998  39.7%     -    5s
     0     0  724.16998    0  118 1201.00000  724.16998  39.7%     -    6s
     0     0  724.16998    0  115 1201.00000  724.16998  39.7%     -    6s
     0     0  724.16998    0  114 1201.00000  724.16998  39.7%     -    6s
     0     0  724.16998    0  116 1201.00000  724.16998  39.7%     -    6s
     0     0  724.16998    0  116 1201.00000  724.16998  39.7%     -    6s
     0     2  724.16998    0  116 1201.00000  724.16998  39.7%     -    6s
  1061   946  767.00000   17  121 1201.00000  762.00000  36.6%  32.8   10s
H 1062   899                    1163.0000000  762.00000  34.5%  32.7   10s
  6142  3933  818.00000   35  128 1163.00000  777.00000  33.2%  19.8   16s
H 6143  3918                    1161.0000000  777.00000  33.1%  19.8   16s
* 8867  5817              78    1157.0000000  784.00000  32.2%  19.2   18s
* 8869  5815              79    1156.0000000  784.00000  32.2%  19.2   18s
 10565  7058  940.00000   33  129 1156.00000  786.00000  32.0%  19.5   20s
H18036 12007                    1153.0000000  789.00000  31.6%  18.8   24s
*18071 11984              72    1152.0000000  789.00000  31.5%  18.8   24s
 18790 12485 infeasible   71      1152.00000  789.00000  31.5%  18.7   25s
*22039 14549             123    1148.0000000  794.00000  30.8%  18.5   27s
 25189 16667  872.00000   33  138 1148.00000  802.00000  30.1%  18.6   30s
*28393 18712             136    1145.0000000  802.00000  30.0%  18.3   31s
 31683 20905 1072.00000   73   72 1145.00000  807.00000  29.5%  18.3   35s
*35444 23353             112    1144.0000000  808.00000  29.4%  18.5   37s
*35955 23616             108    1143.0000000  808.00000  29.3%  18.4   37s
 38998 25703  963.00000   35   99 1143.00000  810.00000  29.1%  18.4   40s
 40976 27092  941.00000   44  204 1143.00000  811.00000  29.0%  18.3   54s
 40980 27095  938.00000   42  193 1143.00000  811.00000  29.0%  18.3   55s
 40986 27099  941.00000   44  185 1143.00000  811.00000  29.0%  18.3   60s
 40993 27103  895.00000   39  205 1143.00000  811.00000  29.0%  18.3   65s
 41000 27108  956.00000   41  116 1143.00000  811.00000  29.0%  18.3   70s
H41117 25830                    1140.0000000  811.00000  28.9%  18.5   74s
 41294 25947  935.00000   66   92 1140.00000  811.00000  28.9%  18.5   75s
H41807 24812                    1138.0000000  811.00000  28.7%  18.6   76s
 45303 26211  885.00000   44  131 1138.00000  811.00000  28.7%  18.5   80s
 53037 29136 1063.00000   63   65 1138.00000  824.00000  27.6%  18.0   85s
H54200 28346                    1135.0000000  824.00000  27.4%  18.0   86s
*59189 29017              99    1133.0000000  824.00000  27.3%  17.9   89s
```

```
 59303 29052 infeasible   75       1133.00000  824.00000  27.3%  17.9   90s
H62199 28849                     1129.0000000  824.00000  27.0%  17.9   94s
                                      (…)

32211889 4264264  931.00000  130  103 1046.00000  931.00000  11.0%  15.2
28760s
 32217358 4265184 infeasible  138      1046.00000  931.00000  11.0%  15.2
28765s
 32223890 4266366  940.00000  126   95 1046.00000  931.00000  11.0%  15.2
28770s
 32229139 4268109  931.00000  117   91 1046.00000  931.00000  11.0%  15.2
28775s
 32235661 4269207 infeasible  134      1046.00000  931.00000  11.0%  15.2
28780s
 32241973 4270564 infeasible  147      1046.00000  931.00000  11.0%  15.2
28785s
 32248376 4271992 infeasible  153      1046.00000  931.00000  11.0%  15.2
28790s
 32255385 4273742 infeasible  154      1046.00000  931.00000  11.0%  15.2
28795s
```