2
Problemas de Escalonamento

O escalonamento ou programação da produção é um dos problemas operacionais que visam planejar e controlar a execução das tarefas de produção e de serviços. Ele tem como finalidade determinar uma seqüência factível de processamento de um conjunto de operações por um conjunto de recursos ao longo de um intervalo de tempo, visando otimizar uma ou mais medidas de desempenho. Tais operações formam parte das tarefas ou pedidos de clientes por produtos ou serviços. Nesse problema podem existir ainda restrições de precedência entre as operações e de disponibilidade de recursos por operação.

As medidas de desempenho comumente usadas num problema de escalonamento pelo setor de planejamento operacional são:

a) *Makespan* (c_M): é a diferença de tempo entre o início e o fim de uma seqüência de trabalhos ou tarefas.
b) *Tempo de fluxo total* ($\bar{F}$): é o tempo total gasto por um conjunto de trabalhos ou tarefas, dado pelo somatório da conclusão de todos os trabalhos ou tarefas.
c) *Carga de trabalho total* (w_T): representa o tempo total de trabalho utilizado pelos recursos ou máquinas.
d) *Carga de trabalho máxima* (w_M): representa o tempo de trabalho máximo utilizado por algum recurso ou máquina.
e) *Balanceamento de trabalho* (w_B): é a variância da carga de trabalho de todos os recursos ou máquinas envolvidas na produção.
f) *Tempo de atraso* (L): é o tempo de atraso entre o término da fabricação de um grupo de produtos e a data de entrega
g) *Tempo de antecipação* (E): é o tempo de antecipação do término da fabricação de um grupo de produtos em relação à data de entrega,

Uma abordagem global dos problemas de escalonamento é apresentada por Gen *et al*. (2008, 2009), onde descrevem o *Integrated Manufacturing System* (IMS), como um ambiente de tecnologias de fabricação e processamento, que

consiste em engenharia de design, processo de planejamento, manufatura, gestão da qualidade, armazenamento e distribuição. A busca por melhoria na qualidade de decisão nesses aspectos dá origem a problemas complexos de otimização combinatória, sendo a maioria deles pertencentes à classe dos problemas NP-difícil. Exemplos de problemas normalmente encontrados em um IMS são mostrados na Figura 1.

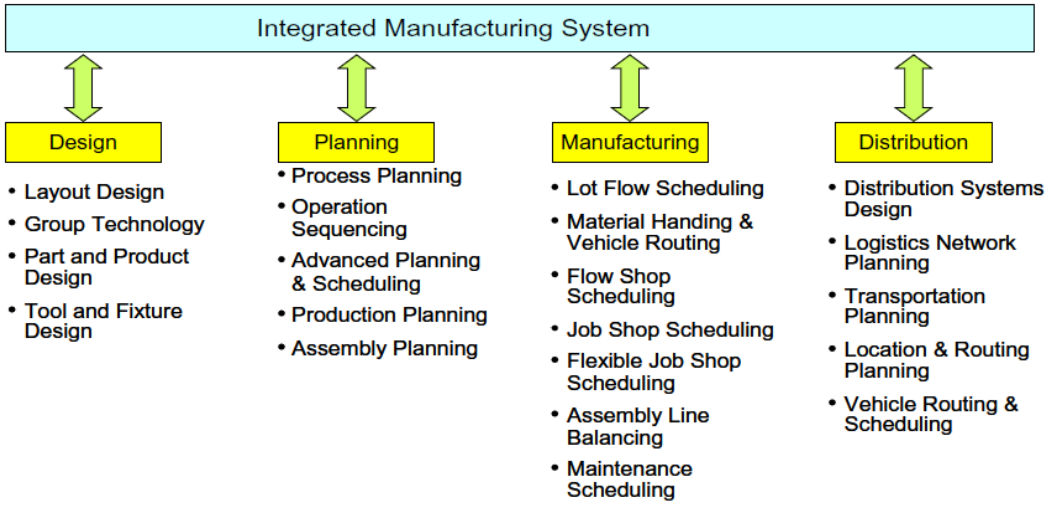


Figura 1 – Problemas comuns em um IMS
Fonte: Gen *et al.* (2009, p.781)

A seguir são descritos os problemas de escalonamento de interesse neste estudo: *flow shop problem* (FSP), *flexível job shop problem* (fSJP), *integrated resource selection and operation sequences problem* (iRS/OS) e *advanced planning and scheduling problem* (APS).

2.1
Escalonamentos de Tarefas Flow Shop

O FSP é um sistema de trabalho de K tarefas em N máquinas em série, onde cada tarefa tem que ser processada em cada uma das N máquinas. Todas as tarefas devem seguir o mesmo roteiro, ou seja, elas têm que ser processadas, primeiro na máquina 1, depois na máquina 2, e assim por diante. Após a conclusão de uma tarefa em uma máquina, a tarefa se junta à fila da próxima máquina.

Muitos esforços têm sido feitos no sentido de resolver o FSP envolvendo diferentes metodologias. Por exemplo, Widmer e Hertz (1989) propuseram um método heurístico para resolver o FSP com o objetivo de minimizar apenas o

makespan. Este método é composto de duas fases: a primeira fase considera uma seqüência inicial correspondendo a uma solução do problema do caixeiro viajante, e a segunda tenta melhorar essa solução usando técnicas de busca tabu. Já o trabalho de Ho (1993) apresenta uma heurística para minimizar apenas o tempo de fluxo médio para o FSP.

Considerando, agora, mais de um objetivo a ser otimizado no FSP, destacamos o trabalho de Ponnambalam *et al*. (2004). Seus autores propõem um algoritmo de busca evolutiva multiobjetivo, utilizando um algoritmo que resolve o problema do caixeiro viajante e um algoritmo genético. O algoritmo utiliza uma soma ponderada dos vários objetivos como função aptidão. Os pesos são gerados aleatoriamente a cada geração para permitir uma busca multi-direcional. As medidas de desempenho consideradas incluem minimizar o *makespan*, o tempo de fluxo médio e o tempo de máquina parada. Pasupathy *et al*. (2006) propuseram um algoritmo genético, com o objetivo de minimizar o *makespan* e o tempo de fluxo total, melhorando a implementação do algoritmo genético mediante técnicas de busca local. Esse algoritmo faz uso do princípio da não-dominância juntamente com uma métrica de aglomeração, visando superar a dificuldade do algoritmo em gerar uma fronteira eficiente.

2.1.1
Descrição do problema

Num ambiente *flow shop*, K tarefas devem ser processadas em um conjunto de N máquinas distintas, tendo a mesma ordem de processamento nas máquinas. O problema consiste em determinar uma seqüência de tarefas dentre $K!$ seqüências possíveis, que é mantida para todas as máquinas, de modo a otimizar uma determinada medida de desempenho da programação, associada geralmente ao fator tempo (BUZZO & MOCCELLIN, 2000).

O FSP é classificado como NP-difícil para a maioria dos problemas clássicos. Já são considerados NP-difícil os seguintes problemas: $F_2||\sum c_j$, um *flow shop* com duas máquinas em série com o objetivo de minimizar o somatório do tempo de término de todas as tarefas; $F_2||L_M$, um *flow shop* com duas máquinas em série com o objetivo de minimizar o máximo atraso; $F_3||c_M$, um

flow shop com três máquinas em série com o objetivo de minimizar o *makespan* (PINEDO, 2008).

As hipóteses consideradas no FSP são:

- Cada máquina está disponível continuamente, sem interrupções.
- Cada máquina pode processar apenas uma tarefa de cada vez.
- Cada tarefa pode ser processada por uma máquina de cada vez.
- Os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos.
- As tarefas têm a mesma data de liberação, a partir da qual, qualquer uma pode ser programada e executada.
- Os tempos de preparação das operações nas diversas máquinas são incluídos nos tempos de processamento.
- Uma vez iniciadas as operações nas diversas máquinas, elas não devem ser interrompidas.

2.1.2 Formulação do problema

Para formular o problema serão considerados os seguintes critérios de desempenho: *makespan* e o tempo de fluxo total. A otimização do *makespan* visa reduzir o tempo de conclusão das tarefas e resulta numa utilização eficiente dos recursos, enquanto que a otimização do tempo de fluxo reduz o número médio de tarefas em espera na fila (PASUPATHY *ET AL.*, 2006). Além disso, a otimização do tempo fluxo é importante, porque muitas das medidas de custo, como a rotação de produtos acabados dependem do tempo de fluxo (HO, 1995). Minimizar o tempo de fluxo requer que em média de conclusão de todas as tarefas seja a menor possível à custa de uma tarefa que demora um tempo maior em concluir, enquanto que minimizar o *makespan* requer que nenhuma tarefa demore muito tempo. Assim, minimizar o *makespan* implica em maximizar o tempo de fluxo (LIU *ET AL.*, 2010). Portanto, este problema deve ser tratado como um problema de otimização multiobjetivo, já que tem objetivos conflitantes.

Dado o tempo de processamento $p(k,m)$ da tarefa k na máquina m, para $k = 1, \dots, K$ e $m = 1, \dots, N$, dada uma seqüência de tarefas a serem

realizadas $\{J_1, J_2, \ldots, J_K\}$, então, o tempo de conclusão $c(J_k, m)$ da tarefa J_k na máquina m é dado da seguinte forma (REEVES, 1995):

$$c(J_1, 1) = p(J_1, 1)$$

$$c(J_k, 1) = c(J_{k-1}, 1) + p(J_k, 1), k = 2, \ldots, K$$

$$c(J_1, m) = c(J_1, m-1) + p(J_1, m), m = 2, \ldots, N$$

$$c(J_k, m) = max\{c(J_{k-1}, m), c(J_k, m-1)\} + p(J_k, m), k = 2, \ldots, K; m = 2, \ldots, N$$

Logo, o tempo de conclusão de todas as tarefas ou *makespan* é dado por:

$$c_M = c(J_K, N)$$

O tempo de fluxo total, denotado por $\bar{F}$, é definido como sendo o somatório do tempo total de processamento das K tarefas, dado por:

$$\bar{F} = \sum_{k=1}^{K} c(J_k, N)$$

2.1.3 Representação de soluções

Uma solução s^* de um problema de escalonamento *flow shop* é codificada por um vetor de tamanho K que é formado por uma permutação aleatória de $\{1, 2, \ldots, K\}$, correspondendo à ordem em que as tarefas serão executadas. Deste modo, a k-ésima posição do vetor representa a k-ésima tarefa a ser processada.

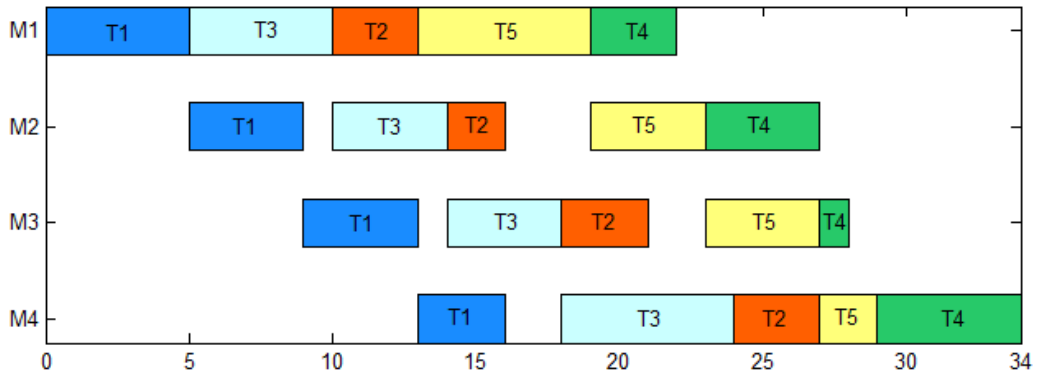


Figura 2 – Exemplo do FSP para $K = 5$ e $N = 4$

Na Figura 2, é apresentado um exemplo da representação de uma solução de um FSP para $K = 5$ e $N = 4$. A seqüência de tarefas é dada por $s^* = \{1, 3, 2, 5, 4\}$. Pode-se observar que a tarefa 1 é a primeira a ser inicialmente executada na máquina 1, depois na máquina 2, na máquina 3 e por último na máquina 4. No momento que a máquina 1 estiver liberada da tarefa 1, a tarefa 3 é executada, igualmente, no momento que a máquina 2 estiver liberada da tarefa 1, a tarefa 3 será processada, e assim por diante para todas as tarefas.

2.1.4
Geração de vizinhança para seqüências

Para cada solução s^* é definido $N(s^*)$, como o conjunto de movimentos aplicados para a solução s^*. $N(s^*)$ é chamado de vizinhança de s^*. Um movimento consiste em uma operação que torna s^* em s', uma nova solução (GLOVER, 1989; WIDMER & HERTZ, 1989; TAILLARD, 1990).

Taillard (1990) comparou vários métodos heurísticos para resolver o FSP, mostrando para este problema os métodos de geração de vizinhança, a eficiência na busca de soluções de qualidade e o tempo computacional:

1) Troca de duas tarefas adjacentes colocadas na k-ésima e $(k+1)$-ésima posição. Um movimento é definido completamente por k. O tamanho da vizinhança é $(K-1)$. Este tipo de movimento é ruim, tanto para a qualidade da solução e o tempo computacional.
2) Troca de duas tarefas colocadas na k-ésima e l-ésima posição. Um movimento é definido completamente por k e l. O tamanho da vizinhança é $\frac{1}{2}K(K-1)$. A avaliacao do *makespan* de toda a vizinhança é executado em tempo $O(K^3N)$. Este tipo de movimento tem complexidade alta e não é melhor que o próximo método para encontrar boas soluções. Esta vizinhança foi proposta por Widmer e Hertz (1989).
3) Remover a tarefa colocada na k-ésima posição e inseri-la na l-ésima posição. Um movimento é definido completamente por k e l. O tamanho da vizinhança é $(K-1)^2$. A avaliação de todos os *makespan* é executado em tempo $O(K^2N)$. Este tipo de vizinhança

produz movimentos eficientes, tanto para a qualidade da solução e o tempo computacional. Esta vizinhança foi proposta por Nawaz *et al*. (1983) como o método de inserção NEH.

2.2
Escalonamentos de Tarefas Job Shop Flexível

O fJSP é um dos problemas de escalonamento de produção mais difíceis de otimização combinatória. O problema *job shop* clássico consiste em encontrar uma ordenação adequada de um conjunto de tarefas em um conjunto de máquinas com o objetivo de minimizar um determinado critério, sujeito à restrição de que cada tarefa tem seu próprio roteiro de fabricação e diferentes entre si. O problema *job shop* (JSP) é classificado como NP-difícil (GAREY, JOHNSON & SETHI, 1976).

O fJSP é uma generalização do problema *job shop* e do ambiente de máquinas em paralelo, proporcionando uma aproximação maior à realidade atual no funcionamento das indústrias. Considera-se que existe pelo menos uma máquina com capacidade de realizar mais de uma operação. O fJSP consiste em atribuir uma operação para uma determinada máquina e obter de uma seqüência factível do conjunto de operações sobre o conjunto de máquinas, visando a otimização de alguma medida de desempenho.

As pesquisas sobre o fJSP são bastante abrangentes. Bruker e Schlie (1990) foram os primeiros em abordar este problema. Eles desenvolvem um algoritmo polinomial para resolver o fJSP com duas tarefas, na qual as máquinas são capazes de realizar uma operação no mesmo tempo de processamento. Para resolver o caso realista, com mais de duas tarefas, dois tipos de abordagens têm sido utilizadas: hierárquica e integrada.

Na abordagem hierárquica, a alocação de máquinas e o seqüenciamento de operações são tratados separadamente, enquanto que na abordagem integrada, estes não são diferenciados. Abordagens hierárquicas são baseadas na idéia de decompor o problema original em dois subproblemas (alocação e seqüenciamento) de modo a reduzir a sua complexidade. Brandimarte (1993) foi o primeiro a usar está decomposição para o fJSP. Ele resolveu o subproblema de alocação utilizando *dispatching rules* e depois resolveu o subproblema de

seqüenciamento usando busca tabu. Kacem, Hammadi, e Borne (2002a, 2002b), utilizaram um algoritmo genético multiobjetivo usando dois enfoques. O primeiro é um enfoque de localização segundo é um enfoque evolutivo (algoritmo genético), a fim de melhorar a solução. Zhang e Gen (2005) propuseram o *multistage operation-based genetic algorithm* para resolver o problema por programação dinâmica. Xia e Wu (2005) trataram o fJSP multiobjetivo, aplicando de forma combinada o *particle swarm optimization* (PSO) e *simulated annealing* (SA) como algoritmo de busca local. Gao *et al*. (2008), desenvolveram um novo enfoque para o algoritmo genético híbrido: *variable neighborhood descent*, a fim de explorar a capacidade da busca total do algoritmo genético e a capacidade da busca local. Zhang *et al*. (2009), propuseram um algoritmo híbrido multiobjetivo baseado no PSO e busca tabu. Recentemente, Moslehi e Manhnam (2011) apresentaram um novo enfoque baseado na hibridação do PSO e busca local para resolver o fJSP multiobjetivo.

Abordagens integradas são utilizadas considerando alocação e seqüenciamento ao mesmo tempo. Hurink, Jurisch e Thole (1994), propuseram uma heurística de busca tabu que acha a alocação e o seqüenciamento mediante dois tipos de movimentos. A abordagem integrada também foi usada por Dauzere-Peres e Paulli (1997). Eles definiram uma vizinhança e propuseram um procedimento de busca tabu. Mastrolilli e Gambardella (2002) melhoraram a técnica de busca tabu de Dauzere-Peres e apresentaram duas funções de vizinhança.

Neste estudo, um novo enfoque é proposto para resolver o fJSP multiobjetivo. Tem-se como objetivo minimizar simultaneamente as seguintes medidas de desempenho: *makespan*, carga de trabalho máxima e carga de trabalho total.

2.2.1 Descrição do problema

Dado um conjunto de K tarefas e um conjunto $M = \{M_1, M_2, \dots, M_N\}$ de N máquinas. Considere que cada tarefa k, $k = 1, \dots, K$, é composta por uma seqüência de J_k operações. Ainda considere que cada operação o_{ki} ($k = 1, 2, \dots, K;\ i = 1, 2, \dots, J_k$) da tarefa k tem que ser processado por alguma máquina

M_m dentro de um subconjunto de máquinas disponíveis M_{ki} de o_{ki}. A execução de cada operação requer que uma máquina seja selecionada dentro do subconjunto de máquinas disponíveis. O fJSP consiste em determinar a seqüência de operações e a alocação de máquinas, com a finalidade minimizar o *makespan* e balancear as cargas de trabalho das máquinas. Se houver $M_{ki} \subset M$ com pelo menos uma operação o_{ki}, dizemos que o problema é *parcialmente flexível*; se $M_{ki} = M$ para cada operação o_{ki}, dizemos que o problema é *totalmente flexível* (KACEM *ET AL.*, 2002a, 2002b).

As hipóteses consideradas no fJSP são:

- Cada operação não pode ser interrompida durante sua execução.
- Cada máquina pode executar no máximo uma operação ao mesmo tempo.
- A restrição de precedência de operações de uma tarefa é pré-especificada.
- Os tempos de *setup* ou transporte entre operações estão incluídos no tempo de processamento.
- As máquinas são independentes entre si.
- As tarefas são independentes entre si.

2.2.2 Formulação do problema

Para formular o modelo matemático do problema fJSP, que considera mais de uma medida de desempenho, serão apresentadas definições e notações dos conjuntos de índices, parâmetros e variáveis de decisão, a seguir (ZHANG *ET AL.*, 2009).

Índices

k, l: Índices das tarefas, $k, l = 1, 2, \dots, K$

i, j: Índices das operações, $i, j = 1, 2, \dots, J_k$

m, n: Índices das máquinas, $m, n = 1, 2, \dots, N$

Parâmetros

K: Número de tarefas

J_k: Número de operações da tarefa k

J: Número de operações

N: Número de máquinas

M_m: m-ésima máquina

o_{ki}: i-ésima operação da tarefa k

M_{ki}: Conjunto de máquinas disponíveis para a operação o_{ki}

p_{kim}: Tempo de processamento da operação o_{ki} na máquina m

Variáveis

$x_{kim} = \begin{cases} 1, \text{ se a máquina } m \text{ for selecionada para a operação } o_{ki} \\ 0, \text{ caso contrário} \end{cases}$

c_{ki}: Tempo de conclusão da operação o_{ki}

c_M: *Makespan*

w_M: Carga de trabalho máxima

w_T: Carga de trabalho total

w_m: Carga de trabalho da máquina m

$$w_m = \sum_{k=1}^{K} \sum_{i=1}^{J_k} p_{kim} x_{kim}$$

O modelo matemático do fJSP é formulado a seguir:

$$\min c_M = \max_{\forall k,i} \{c_{ki}\} \quad (1)$$

$$\min w_M = \max_{\forall m} \{w_m\} \quad (2)$$

$$\min w_T = \sum_{m=1}^{N} w_m \quad (3)$$

sujeito a:

$$c_{ki} - c_{k,i-1} \geq p_{kim} x_{kim} \;\; i = 2, \ldots, J_k, \forall k, m \quad (4)$$

$$[(c_{lj} - c_{ki} - p_{ljm}) x_{ljm} x_{kim} \geq 0] \vee [(c_{ki} - c_{lj} - p_{kim}) x_{ljm} x_{kim} \geq 0]$$
$$\forall (k,i), (l,j), m \quad (5)$$

$$\sum_{m \in M_{ki}} x_{kim} = 1 \; \forall k, i \tag{6}$$

$$c_{ki} \geq 0 \; \forall k, i \tag{7}$$

$$x_{kim} \in \{0,1\} \; \forall k, i, m \tag{8}$$

O objetivo em (1) representa a minimização do *makespan*. Os objetivos em (2) e (3) representam a minimização da carga de trabalho máxima e do total da carga de trabalho, respectivamente. Estes objetivos visam reduzir o tempo para concluir todas as tarefas e equilibrar as cargas de trabalho de todas as máquinas. As restrições em (4) asseguram as precedências estabelecidas entre as operações de uma tarefa. Já as restrições em (5) asseguram que cada máquina só pode executar uma operação de cada vez. A equação em (6) estabelece que a máquina selecionada deva pertencer ao conjunto de máquinas disponíveis de uma operação. A restrição em (7) impõem a condição de não negatividade, enquanto que a restrição em (8) definem $x_{kim} \; \forall k, i, m$ como variáveis binárias.

2.2.3
Representação de soluções

Uma solução de fJSP é representada por dois vetores: o vetor de seqüência de operações s^* e o vetor alocação de máquinas v^*.

A fim de exemplificar a construção dos vetores s^* e v^*, uma instância do problema fJSP com 3 tarefas e 4 máquinas será considerada. Os dados dessa instância são apresentados na Tabela 1.

Tabela 1 – Tempos de processamento de uma instância fJSP
com 3 tarefas e 4 máquinas

Tarefa	Índice	Operação	M_1	M_2	M_3	M_4
1	1	o_{11}	1	3	4	1
	2	o_{12}	3	8	2	1
	3	o_{13}	3	5	4	7
2	4	o_{21}	4	1	1	4
	5	o_{22}	2	3	9	3
	6	o_{23}	9	1	2	2
3	7	o_{31}	8	6	3	5
	8	o_{32}	4	5	8	1

Fonte: Zhang e Gen (2005, p.226)

A seguir, vamos mostrar em duas etapas a construção dos vetores s^* e v^*.

Etapa 1: Geração da seqüência de operações

O processo começa gerando o vetor de nível prioridade π correspondente ao conjunto de operações. Se J é o número total de operações, então, o vetor π é formado por uma permutação aleatória de $\{1, 2, \dots, J\}$. Um exemplo simples da representação de π é mostrado a seguir:

Índice j	1	2	3	4	5	6	7	8
Prioridade $\pi(j)$	8	4	7	2	3	6	5	1

O nível de prioridade é dado por: $1 > 2 > 3 > 4 > 5 > 6 > 7 > 8$, ou seja, a operação com prioridade 1 terá maior preferência no escalonamento que as operações com prioridade 2, 3, ..., 8; a operação com prioridade 2 terá maior preferência que as operações com prioridade 3, 4, ..., 8, e assim por diante. Usando a procedimento de seqüenciamento de operações, proposto por Liu *et al*. (2009) e apresentado no Anexo A, podemos obter uma seqüência factível $s^* = \{o_{21}, o_{22}, o_{31}, o_{32}, o_{23}, o_{11}, o_{12}, o_{13}\}$, a qual é escrita através de seus índices como $s^* = \{4, 5, 7, 8, 6, 1, 2, 3\}$, de acordo com a representação acima.

Etapa 2: Alocação de máquinas

Agora que a seqüência de operações foi fixada, podemos gerar o vetor v^* mediante o procedimento de alocação de máquinas, proposto por Zhang *et al.* (2006b) e apresentado no Anexo B.

Continuando com o exemplo anterior, a representação gráfica do procedimento de alocação de máquinas é mostrada na Figura 3.

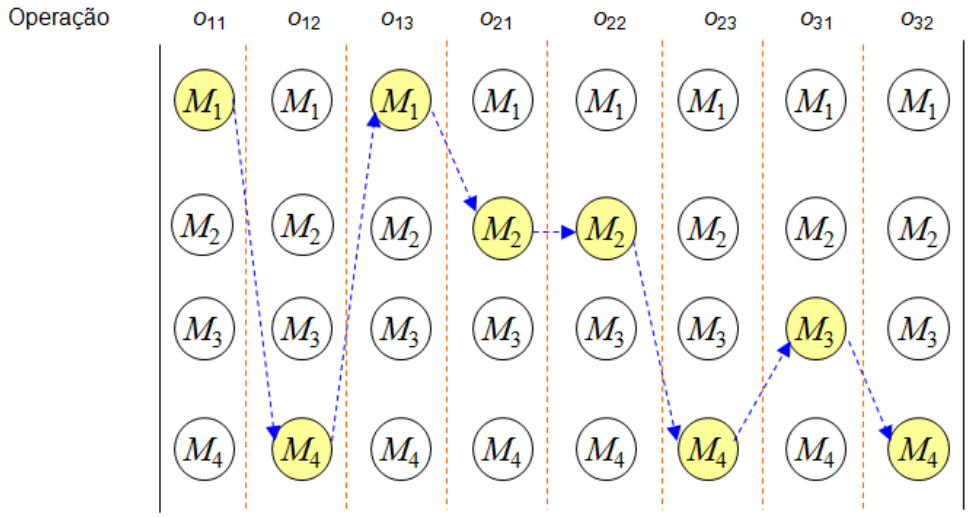


Figura 3 – Representação gráfica da alocação de máquinas do exemplo fJSP
Fonte: Zhang e Gen (2005, p.229)

Desta forma, uma solução é dada por:

	o_{21}	o_{22}	o_{31}	o_{32}	o_{23}	o_{11}	o_{12}	o_{13}
Seqüência s^*	4	5	7	8	6	1	2	3

Índice j	1	2	3	4	5	6	7	8
Máquina $v^*(j)$	1	4	1	2	2	4	3	4

Utilizando os dados da Tabela 1, podemos elaborar o diagrama de Gantt, apresentado na Figura 4.

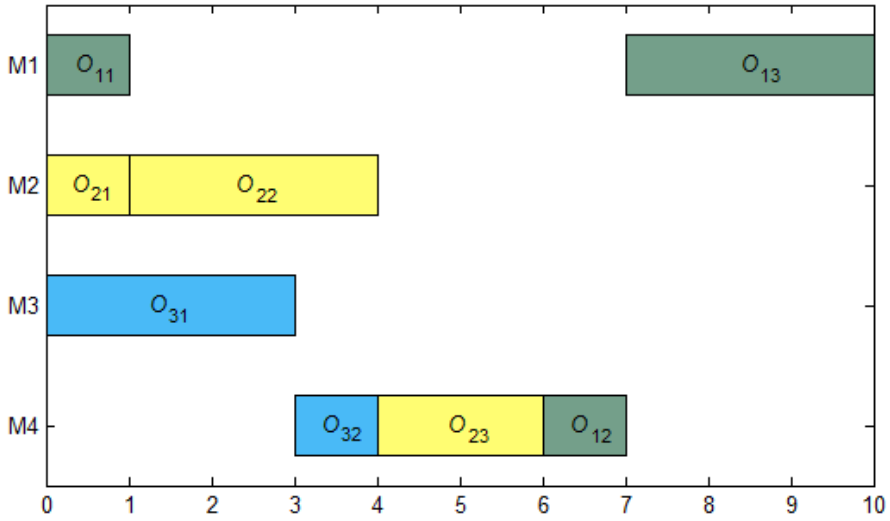


Figura 4 – Diagrama de Gantt do exemplo fJSP

2.2.4
Geração de vizinhança para máquinas

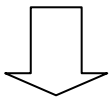
Se a máquina alocada da operação o_{ki} for m, então uma vizinhança pode ser gerada através da atribuição do conjunto de máquinas $n \in M_{ki}$ para realizar a operação o_{ki} tal que os tempos de processamento da operação o_{ki} nas máquinas n e m satisfaçam $p_{kin} \leq p_{kim}$, tendo em vista que um dos objetivos é minimizar a carga de trabalho total das máquinas.

2.2.5
Geração de vizinhança para seqüências

No algoritmo de busca local, o tipo de vizinhança pode influenciar fortemente o desempenho do algoritmo. Embora a escolha de uma vizinhança que contém um grande número de soluções candidatas aumenta a probabilidade de encontrar boas soluções, o tempo de computacional para encontrar vizinhos disponíveis também irá aumentar (XIA & WU, 2005). Um método simples para gerar soluções vizinhas é o método de troca entre duas operações de uma seqüência (similar ao método descrito na seção 2.1.4-2), que geram seqüências factíveis. Considerando a seqüência de operações do exemplo anterior, a vizinhança pelo método de troca é mostrada a seguir:

Seqüência s^*	4	5	7	8	6	1	2	3

Vizinhança s* soluções geradas pelo método de troca entre operações



4	7	5	8	6	1	2	3
4	5	7	6	8	1	2	3
4	5	7	1	6	8	2	3
4	5	7	8	1	6	2	3

Note que, se J é o número total de operações, então, o número de permutações é dado por: $\frac{1}{2}(J)\,(J-1)$, Em nosso exemplo de com $J = 8$ operações, o número total de permutações geradas de s^* é $\frac{1}{2}(8)(7) = 28$, porém somente 4 seqüências são factíveis.

2.3
Integrated Resource Selection and Operation Sequences Problem

O problema *integrated resource selection and operation sequences* (iRS/OS) é uma aproximação do processo de produção de um sistema de manufatura real. Nesse sistema, cada pedido está associado a um conjunto de operações, as quais não têm seqüência fixa e depende de restrições de precedência. O escalonamento do processo deve considerar os tamanhos de lote e de carga dos pedidos, tempos de transporte, disponibilidade e capacidade dos recursos (ZHANG *ET AL*., 2006b).

Durante os últimos anos, muitos pesquisadores têm realizado esforços na área de planejamento de processos integrados e escalonamento. Podemos destacar Tan (2000, 2004) que fez uma breve revisão da pesquisa do planejamento de processos integrados e escalonamento e discutiu a aplicabilidade de vários enfoques. Também propôs um modelo matemático para este problema. Kolisch e Hess (2000) que estudaram o problema de escalonamento múltiplo em grande escala para o processo de montagem, considerando recursos, área de montagem e restrições de disponibilidade de peças, onde os diferentes pedidos dos clientes precisam dos mesmos tipos de peças e a fabricação de peças deve levar em conta

as decisões de tamanho de lote. Eles desenvolveram três métodos heurísticos para resolver o problema; um método aleatório amostral e dois métodos baseados em busca tabu.

Nos trabalhos de Moon, Lee, Seo e Lee Young (2002b); Moon, Lee e Gen (2004a); Moon, Kim e Chen (2004b) e Moon e Seo (2005a, 2005b) são propostos algoritmos genéticos que podem resolver o problema iRS/OS. No entanto, a codificação do cromossomo só considera a informação da seqüência de operações, enquanto que para a alocação de recursos é utilizado uma heurística baseada no tempo de processamento mínimo, porém, pode não atingir uma solução ótima. Para evitar esta limitação Zhang *et al*. (2006b), propuseram um novo enfoque, o método *operation-based genetic algorithm* (moGA). Este enfoque define o cromossomo com dois vetores que contém tanto a informação da seqüência de operações e quanto a alocação de máquinas. Esse método separa cada operação em etapas, e em cada etapa uma máquina é escolhida dentro de um conjunto de máquinas disponíveis para alocação.

Neste estudo, um novo enfoque é proposto para resolver o problema multiobjetivo iRS/OS, tendo como objetivo minimizar simultaneamente o *makespan* e o balanceamento de carga de trabalho.

2.3.1 Descrição do problema

O problema iRS/OS é uma extensão dos problemas FSP, JSP e fJSP, e tem sido valioso para a pesquisa devido à sua semelhança com os sistemas de manufatura real. A Figura 5 apresenta um esquema para a extensão do problema iRS/OS.

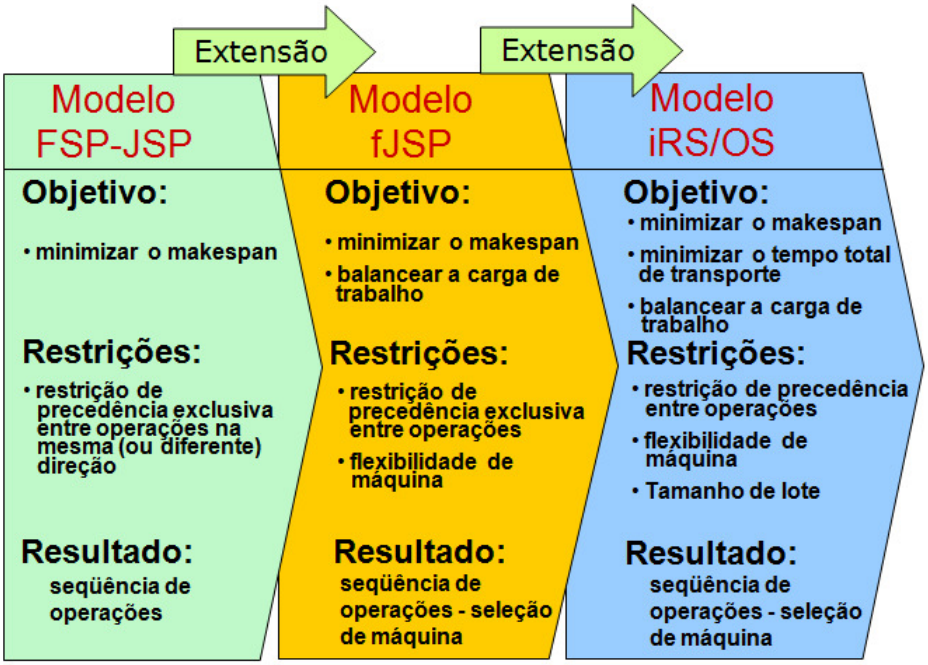


Figura 5 – Extensões ao problema iRS/OS
Fonte: Zhang *et al.* (2006b, p.387)

As hipóteses consideradas no problema iRS/OS são:

- Todos os pedidos chegam simultaneamente sem precedência.
- Todas as máquinas estão disponíveis no início.
- A carga de pedido é fixa para todas as máquinas, e as máquinas não podem parar até terminar de processar o tamanho de lote.
- Para o mesmo pedido, o processamento numa operação começa logo que a carga do pedido da operação predecessora tenha sido concluída e transferida.
- O tempo de transporte entre máquinas diferentes é considerado.
- O tempo de *setup* está incluído no tempo de processamento.

A seguir o problema iRS/OS será entendido com ajuda de um exemplo. A Figura 6 mostra dois tipos de materiais a serem processados. O primeiro pedido, Pedido 1, tem tamanho de lote de 60 unidades do Produto 1, e o segundo pedido, Pedido 2, tem tamanho de lote de 50 unidades do Produto 2. Para obter os produtos finais é preciso remover 7 componentes ou volumes dos dois materiais. A carga do pedido é 10 unidades, isto é, para o Pedido 1, o número de sub-lotes é de $60/10 = 6$ e para o Pedido 2 é de $50/10 = 5$ (em cada máquina o tamanho de

lote será processado em 10 unidades de cada vez). O plano de manufatura é mostrado na Tabela 2, a qual indica o tipo de cada operação e as máquinas disponíveis por operação.

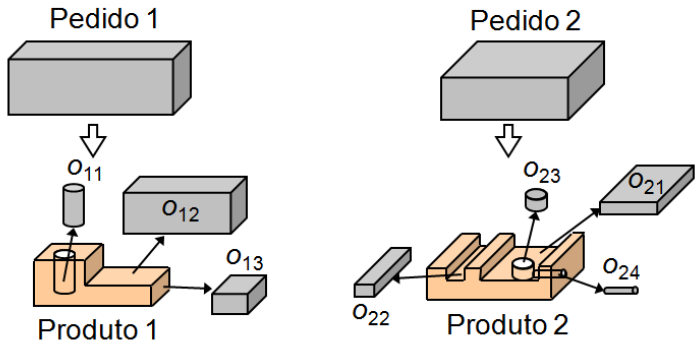


Figura 6 – Exemplo de um problema iRS/OS
Fonte: Zhang *et al.* (2006b, p.387)

Tabela 2 – Plano de manufatura do exemplo iRS/OS

Índice da operação (Pedido $\boldsymbol{k}$, Operação $\boldsymbol{o_{ki}}$)	Tipo de operação	Máquinas disponíveis
1 (1, $\boldsymbol{o_{11}}$)	Perfuração	M_1, M_4
2 (1, $\boldsymbol{o_{12}}$)	Fresagem	M_2, M_5
3 (1, $\boldsymbol{o_{13}}$)	Fresagem	M_2, M_3, M_5
4 (2, $\boldsymbol{o_{21}}$)	Fresagem	M_2, M_3
5 (2, $\boldsymbol{o_{22}}$)	Fresagem	M_3, M_5
6 (2, $\boldsymbol{o_{23}}$)	Fresagem	M_1, M_4, M_5
7 (2, $\boldsymbol{o_{24}}$)	Perfuração	M_1, M_2

Fonte: Zhang *et al.* (2006b, p.387)

A Figura 7 mostra as restrições de precedência das operações dos pedidos. As seqüências de operações são muitas e devem seguir as restrições de precedência para cada pedido. Por exemplo, para o Pedido 1, tanto $\{o_{11}, o_{12}, o_{13}\}$ quanto $\{o_{12}, o_{11}, o_{13}\}$ são seqüências factíveis, porém $\{o_{11}, o_{13}, o_{12}\}$ é infactível.

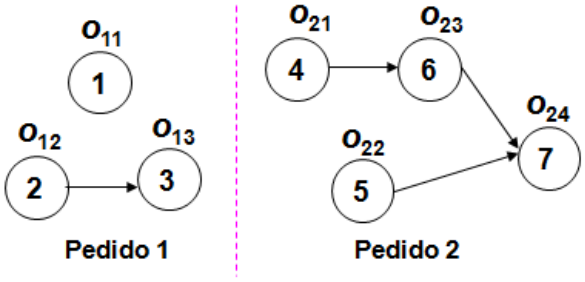


Figura 7 – Restrições de precedência do exemplo iRS/OS
Fonte: Zhang *et al.* (2006b, p.387)

A Tabela 3 mostra o tempo de processamento dado para cada operação. A Tabela 4 mostra o tempo de transporte dado entre as diferentes máquinas e a capacidade disponível L_m de cada máquina m.

Tabela 3 – Tempo de processamento das operações em máquinas diferentes do exemplo iRS/OS

Pedido k	Pedido 1			Pedido 2			
Operação o_{ki}	1	2	3	4	5	6	7
Máquina M_m	o_{11}	o_{12}	o_{13}	o_{21}	o_{22}	o_{23}	o_{24}
M_1	5	-	-	-	-	6	8
M_2	-	7	6	6	-	-	12
M_3	-	-	5	7	9	-	-
M_4	7	-	-	-	-	5	-
M_5	-	6	8	-	5	6	-

Fonte: Zhang *et al.* (2006b, p.388)

Tabela 4 – Tempo de transporte entre máquinas diferentes do exemplo iRS/OS

Máquina M_n / Máquina M_m	M_1	M_2	M_3	M_4	M_5	Capacidade L_m
M_1	-	7	27	26	17	1,500
M_2	19	-	15	19	10	1,500
M_3	5	17	-	36	27	1,500
M_4	8	20	4	-	30	1,500
M_5	18	18	14	5	-	1,500

Fonte: Zhang *et al.* (2006b, p.388)

Portanto, o problema iRS/OS pode ser definido como segue: dado um conjunto de K pedidos, de tamanho de lote q_k que devem ser processados por meio de J operações a serem realizadas em N máquinas conforme a disponibilidade. Deseja-se encontrar uma seqüência de operações dos pedidos, a alocação de máquinas e o escalonamento de todos os pedidos sobre as máquinas, de tal forma que se satisfaça as restrições de precedência e, além disso, seja ótima em relação à minimização do *makespan* e do balanceamento de carga de trabalho (ZHANG *ET AL.*, 2006b).

2.3.2 Formulação do problema

Para formular o modelo matemático do problema iRS/OS, que considera mais de uma medida de desempenho, serão apresentadas definições e notações dos conjuntos de índices, parâmetros e variáveis de decisão, a seguir (ZHANG *ET AL.*, 2006b).

Índices

k, l: Índices dos pedidos, $k, l = 1, 2, \dots, K$

i, j: Índices das operações, $i, j = 1, 2, \dots, J_k$

m, n: Índices das máquinas, $m, n = 1, 2, \dots, N$

Parâmetros

K: Número de pedidos

J_k: Número de operações por pedido k

J: Número de operações

N: Número de máquinas

o_{ki}: i-ésima operação do pedido k

r_{kij}: Restrição de precedência

$$r_{kij} = \begin{cases} 1, \text{ se } o_{ki} \text{ for predecessor de } o_{kj} \text{ no pedido } k \\ 0, \text{ caso contrário} \end{cases}$$

M_m: m-ésima máquina

A_m: Conjunto de operações que podem ser processadas pela máquina m

M_{ki}: Conjunto de máquinas disponíveis para a operação o_{ki}

q_k: Tamanho do lote do pedido k

p_{kim}: Tempo de processamento da operação o_{ki} na máquina m

L_m: Capacidade da máquina m

u_{kij}: Carga do pedido k da operação o_{ki} para a operação o_{kj}

t_{mn}: Tempo de transporte entre a máquina m e a máquina n

Variáveis

$$y_{kilj} = \begin{cases} 1, \text{ se } o_{ki} \text{ for realizada imediatamente antes de } o_{lj} \\ 0, \text{ caso contrário} \end{cases}$$

$$x_{kim} = \begin{cases} 1, \text{ se a máquina } m \text{ for selecionada para a operação } o_{ki} \\ 0, \text{ caso contrário} \end{cases}$$

s_{ki}: Tempo de início da operação o_{ki}

c_{ki}: Tempo de conclusão da operação o_{ki}

$$c_{ki} = s_{ki} + q_k \sum_{m=1}^{N} p_{kim} x_{kim}$$

c_M: *Makespan*

w_B: Balanceamento da carga de trabalho

w_m: Carga de trabalho da máquina m

$$w_m = \sum_{k=1}^{K} \sum_{i=1}^{J_k} q_k p_{kim} x_{kim}$$

$\overline{w}$: Carga de trabalho médio de todas as máquinas

$$\overline{w} = \frac{1}{N}\sum_{m=1}^{N} w_m$$

O modelo matemático do problema iRS/OS é formulado como:

$$\min c_M = \max_{\forall i,k}\{c_{ki}\} \quad (1)$$

$$\min w_B = \frac{1}{N}\sum_{m=1}^{N}(w_m - \overline{w})^2 \quad (2)$$

sujeito a:

$$(s_{lj} - c_{ki})x_{kim}x_{ljm}y_{kilj} \geq 0 \ \forall (k,i),(l,j),m \quad (3)$$

$$(s_{kj} - (s_{ki} + u_{kij}p_{kim} + t_{mn}))r_{kij}x_{kim}x_{kjn} \geq 0 \ \forall i,j,k,m,n \quad (4)$$

$$(c_{kj} - (c_{ki} + t_{mn} + u_{kij}p_{kjn}))r_{kij}x_{kim}x_{kjn} \geq 0 \ \forall i,j,k,m,n \quad (5)$$

$$\sum_{k=1}^{K}\sum_{i=1}^{J_k} q_k p_{kim}x_{kim} \leq L_m \ \forall m \quad (6)$$

$$r_{kij}y_{kjki} = 0 \ \forall i,j,k \quad (7)$$

$$y_{kiki} = 0 \ \forall i,j \quad (8)$$

$$y_{kilj} + y_{ljki} = 1 \ \forall (k,i),(l,j),(k,i) \neq (l,j) \quad (9)$$

$$\sum_{m=1}^{N} x_{kim} = 1 \ \forall i,k \quad (10)$$

$$x_{kim} = 0 \ \forall (k,i) \notin A_m \ \forall m \quad (11)$$

$$s_{ki} \geq 0 \ \forall i,k \quad (12)$$

$$y_{kilj} \in \{0,1\} \ \forall (k,i),(l,j) \quad (13)$$

$$x_{kim} \in \{0,1\} \ \forall i,k,m \quad (14)$$

Neste modelo, dois objetivos são considerados simultaneamente. O objetivo em (1) representa a minimização do *makespan* e o objetivo em (2) representa o balanceamento da carga de trabalho.

As restrições em (3) asseguram que qualquer máquina pode ser selecionada para realizar uma operação, desde que a operação predecessora já tenha sido completada, tal como mostra a Figura 8.

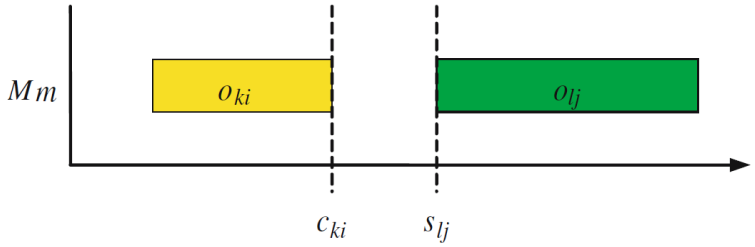


Figura 8 – Gráfico de precedência em uma mesma máquina
Fonte: Zhang *et al.* (2006b, p.389)

As restrições (4) e (5) garantem o transporte dos produtos intermediários entre etapas de produção. Estas restrições têm que ser satisfeitas simultaneamente para garantir que sejam executadas sem interrupção, como mostra a Figura 9. Nesta figura, $u_{kij}p_{kim}$ e $u_{kij}p_{kjn}$ representam tempos de processamento de um sub-lote ou carga do pedido.

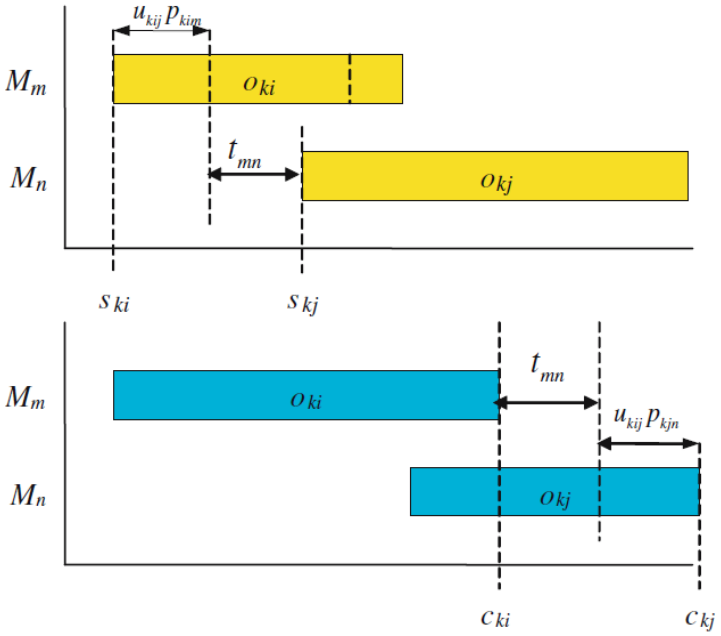


Figura 9 – Gráfico de restrições de precedência em uma mesma planta
Fonte: Zhang *et al.* (2006b, p.389)

As restrições em (6) restringem o processamento à capacidade disponíveis das máquinas. A restrição (7) garante que as restrições de precedência não sejam violadas. As restrições (8) e (9) asseguram a viabilidade das seqüências de operações, enquanto que (10) e (11) asseguram a viabilidade das alocações de máquinas. A restrição (12) impõe a condição de não negatividade para os tempos iniciais, e as restrições (13) e (14) definem $x_{kim}, \forall i,k,m,$ e y_{kilj}, $\forall (k,i),(l,j),$ como variáveis binárias.

2.3.3
Representação de soluções

Uma solução do problema iRS/OS é representada por dois vetores: o vetor de seqüência de operações s^* e o vetor de alocação de máquinas v^*.

A seguir, considerando o exemplo descrito na seção 2.3.1, vamos mostrar em duas etapas a construção dos vetores s^* e v^*

Etapa 1: Geração da seqüência de operações

Inicialmente é gerado o vetor de nível de prioridade π correspondente ao conjunto de operações. Se J é o número total de operações, então o vetor π é formado por uma permutação aleatória de $\{1, 2, \dots, J\}$.

Uma representação de π para o exemplo na seção 2.3.1 é mostrada a seguir:

Índice j	1	2	3	4	5	6	7
Prioridade $\pi(j)$	5	1	7	2	4	6	3

Utilizando o procedimento de seqüenciamento de operações, proposto por Liu *et al*. (2009) e apresentado no Anexo A, podemos obter a seqüência factível $s^* = \{o_{12}, o_{21}, o_{22}, o_{11}, o_{23}, o_{13}, o_{24}\}$, a qual é escrita através de seus índices como $s^* = \{2, 4, 5, 1, 6, 3, 7\}$, de acordo com a representação acima.

Etapa 2: Alocação de máquinas

Agora que a seqüência de operações foi fixada, podemos gerar o vetor v^* mediante o procedimento de Alocação de máquinas, proposto por Zhang *et al*. (2006b) e apresentado no Anexo B.

Continuando com o mesmo exemplo, a representação gráfica do procedimento de alocação de máquinas é mostrada na Figura 10.

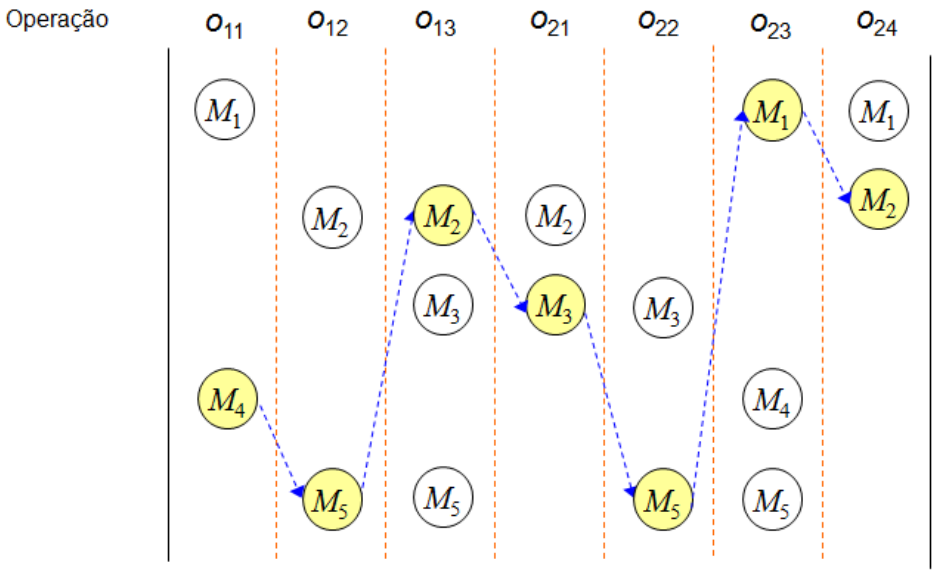


Figura 10 – Representação gráfica da alocação de máquinas do exemplo iRS/OS
Fonte: Zhang *et al.* (2006b, p.391)

Desta forma, a solução é dada por:

	o_{12}	o_{21}	o_{22}	o_{11}	o_{23}	o_{13}	o_{24}
Seqüência s^*	2	4	5	1	6	3	7

Índice j	1	2	3	4	5	6	7
Máquina $v(j)$	4	5	2	3	5	1	2

Utilizando os dados das Tabelas 3 e 4, podemos elaborar o diagrama de Gantt associado à solução do exemplo iRS/OS, apresentado na Figura 11.

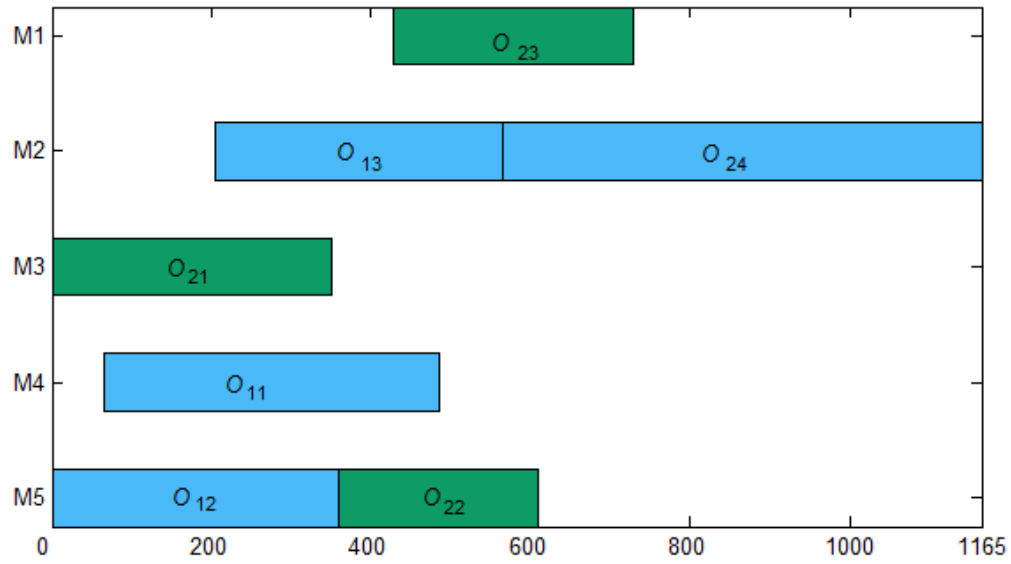


Figura 11 – Diagrama de Gantt do exemplo iRS/OS

2.3.4
Geração de vizinhança para máquinas

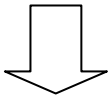
Este tipo de vizinhança pode ser gerado através da atribuição do conjunto de máquinas $m \in M_{ki}$ à operação o_{ki}, onde M_{ki} é o conjunto de máquinas disponíveis para realizar operação o_{ki}. Se J é o número total de operações e N é o número de máquinas, então, o número máximo de soluções vizinhas é dado por $J(N-1)$, já que o problema pode ser parcial ou totalmente flexível.

2.3.5
Geração de vizinhança para seqüências

Como no caso do fJSP, um método simples para gerar soluções vizinhas é o método de troca entre duas operações de uma seqüência (similar ao método descrito na seção 2.1.4-2), que gerem seqüências factíveis. Considerando a seqüência de operações do exemplo anterior, a vizinhança para seqüências pelo método de troca é mostrada no esquema a seguir:

Seqüência s^*	2	4	5	1	6	3	7

soluções geradas pelo método de troca entre operações



Vizinhança s*

4	2	5	1	6	3	7
5	4	2	1	6	3	7
1	4	5	2	6	3	7
2	5	4	1	6	3	7
2	1	5	4	6	3	7
2	4	1	5	6	3	7
2	4	6	1	5	3	7
2	4	3	1	6	5	7
2	4	5	6	1	3	7
2	4	5	3	6	1	7
2	4	5	1	3	6	7
2	4	5	1	6	7	3

Note que, se J é o número total de operações, então, o número de vizinhos é dado por $\frac{1}{2}(J)(J-1)$. Para o nosso exemplo com 7 operações ($J = 7$) o número total de permutações geradas a partir de s^* é $\frac{1}{2}(7)(6) = 21$, porém somente 12 seqüências são factíveis.

2.4
Advanced Planning and Scheduling Problem

Segundo Eck (2003), o *advanced planning and scheduling* (APS) é um sistema que serve como um guarda-chuva sobre a cadeia de suprimentos, permitindo extrair informação em tempo real, com a qual uma programação viável pode ser realizada, resultando em uma rápida e confiável resposta ao cliente. Assim, o APS pode ajudar a reduzir tempos de espera (*lead time*) e níveis de estoques, fornecer datas de entrega razoáveis e, além disso, reduzir custos na cadeia suprimento, incrementando sua eficiência. O APS tende a adotar uma visão

holística na tentativa de otimizar as operações em uma planta e também em todas as atividades desde o fornecedor até o cliente (ZHANG *ET AL*., 2006a).

Durante os últimos anos, o APS tem sido amplamente abordado por muitos pesquisadores, resultando em diversas propostas para resolver o problema APS no ambiente de produção *multi-plant chain* (MPC) e como apoio num ambiente *manufacturing supply chain* (MSC).

Num ambiente MPC, os processos industriais podem ser realizados em mais de uma planta com intercâmbio de informação e gestão coordenada, onde cada planta está equipada com recursos (máquinas), inclusive recursos flexíveis de capacidade limitada.

Alguns estudos sobre o MPC no planejamento e programação de produção têm sido realizados. Vercellis (1999) formulou um problema de programação inteira mista para o MPC, considerando como enfoques o plano mestre de produção e o problema de alocação, e propôs um método heurístico para resolvê-lo. Kolisch (2000) estudou a coordenação da fabricação e montagem com relação a limitações na capacidade, de tal forma a minimizar custos na cadeia de suprimentos. Thoney *et al*. (2002) desenvolveram uma heurística para a programação do MPC considerando transporte entre plantas, tamanho de lote e transporte interno e externo entre operações. Moon *et al*. (2002a) propuseram um modelo de integração de planejamento e programação, com o objetivo de minimizar o atraso total, através da alocação de recursos e seqüência de operações num MPC. Eles também desenvolveram um algoritmo genético para resolver o problema.

Moon *et al*. (2004b, 2005a) propuseram uma abordagem para o problema APS em um ambiente *multi-plant chain* (MPC) com o objetivo de minimizar o *makespan* e decidir a seqüência de operações com a alocação de máquinas, considerando restrições de precedência, flexibilidade das seqüências de operações e disponibilidade das máquinas. Segundo Zhang *et al*. (2006a), a próxima geração de sistemas APS terá que lidar com escalonamento ou programação em nível de transação, considerando requerimentos de recursos do MPC e *outsourcing*. Esses sistemas também terão que ser capazes de fornecer datas de entrega precisas em diversos locais. Ele formulou o problema APS como um sistema de manufatura flexível e apresentou o método *operation-based genetic algorithm* (moGA)

baseado em algoritmos genéticos para resolver o problema APS, minimizando o *makespan*. Moon e Seo (2005b) desenvolveram um algoritmo genético multiobjetivo com uma estratégia adaptativa, que calcula em cada geração a probabilidade de cruzamento e de mutação para melhorar o desempenho do algoritmo genético adaptativo. Os objetivos considerados por eles são: minimizar simultaneamente o *makespan* e o balanceamento das cargas de trabalho em um ambiente MPC.

Segundo Moon *et al*. (2006), muitas indústrias estão se desenvolvendo em cadeias globais, que abrangem vários locais de manufatura e vários fornecedores, e ainda entidades de *outsourcing*. Neste contexto, o *manufacturing supply chain* (MSC) tenta otimizar todo o sistema. Cada vez mais, as empresas estão sendo organizadas como cadeias de plantas. Por essa razão, as atividades de planejamento e programação são muito complexas, e devem ser consideradas na própria empresa e em toda a cadeia de suprimentos, a fim de obter produtos de alta qualidade com menor custo, menores níveis de estoque e altos níveis de desempenho. Para atingir a eficiência global, as empresas de manufatura estão migrando de processos de planejamento separados para processos de planejamentos integrados e coordenados. A Figura 12 mostra a relação entre as partes principais do problema APS e o MSC.

O APS em um ambiente *manufacturing supply chain* (MSC) foi proposto primeiro por Lee *et al*. (2002), que o definiu como segue: dado um conjunto de pedidos a serem processados por meio de um conjunto de operações a serem executadas em máquinas, deseja-se encontrar uma seqüência de operações, a alocação de máquinas e o escalonamento de todos os pedidos sobre as máquinas, de tal forma que se satisfaçam as restrições de precedência das operações, disponibilidade das máquinas, além disso, considere *outsourcing* (terceirização de operações ou aquisição de matérias-primas) e finalmente, seja ótima em relação à minimização de uma ou mais medidas de desempenho, de modo a assegurar o cumprimento dos prazos de entrega dos pedidos.

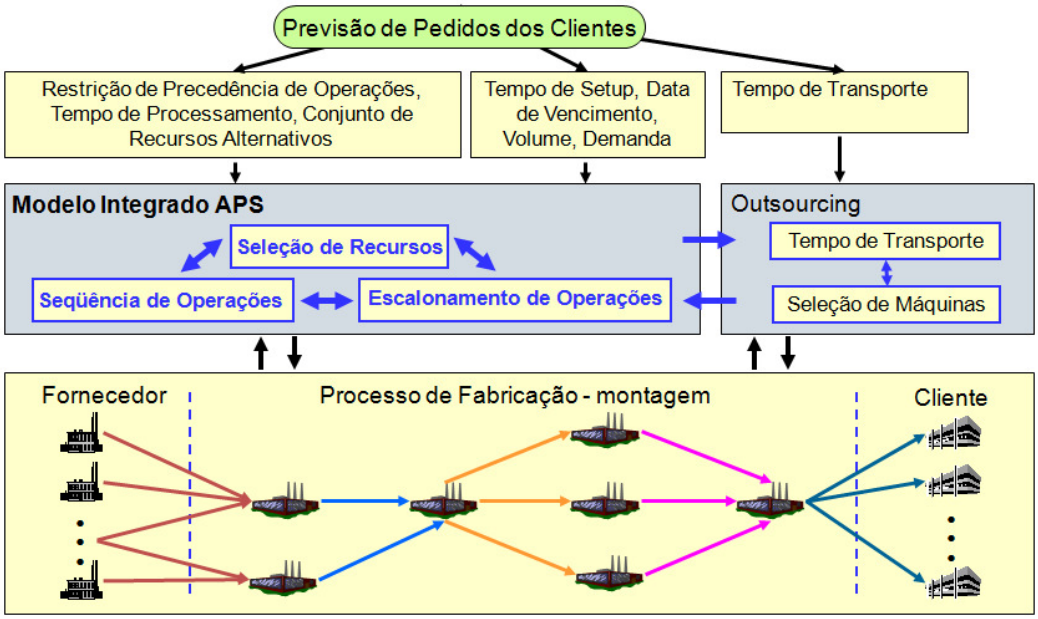


Figura 12 – Relação entre os problemas APS e MSC
Fonte: Zhang *et al.* (2006a, p.1846)

O problema APS no ambiente MSC foi estudado por Moon *et al*. (2006), que desenvolveram um algoritmo genético adaptativo para resolvê-lo. Yan *et al*. (2007) propuseram um algoritmo genético com um novo operador de mutação, baseado em perturbação e busca local, com a finalidade de minimizar o *makespan*, enquanto asseguram a restrição do vencimento de prazo de entrega com o auxílio de uma função aptidão que considera penalidade por atraso. Liu *et al*. (2009) apresentaram um enfoque para resolver o problema APS mediante um algoritmo genético multiobjetivo híbrido. Três objetivos foram simultaneamente minimizados: o *makespan*, o balanceamento da carga de trabalho e o tempo total de transporte entre máquinas. O algoritmo utiliza a estratégia de classificar as soluções *não-dominadas* e medir a aglomeração das soluções, junto com a busca local para melhorar as soluções *Pareto-ótimas*. Yang e Tang (2009) desenvolveram um algoritmo genético multiobjetivo com uma estratégia adaptativa. A minimização simultânea do tempo inativo das máquinas e da penalidade por atraso/antecipação foi considerada.

Neste estudo, uma nova abordagem é proposta para resolver o problema multiobjetivo APS num ambiente MPC. Consideram-se como objetivos simultâneos a minimização do *makespan* e o balanceamento de carga de trabalho.

2.4.1
Descrição do problema

Este problema é uma extensão de vários problemas importantes na área de escalonamento de produção. A Figura 13 mostra a relação do problema APS e outros problemas de escalonamento.

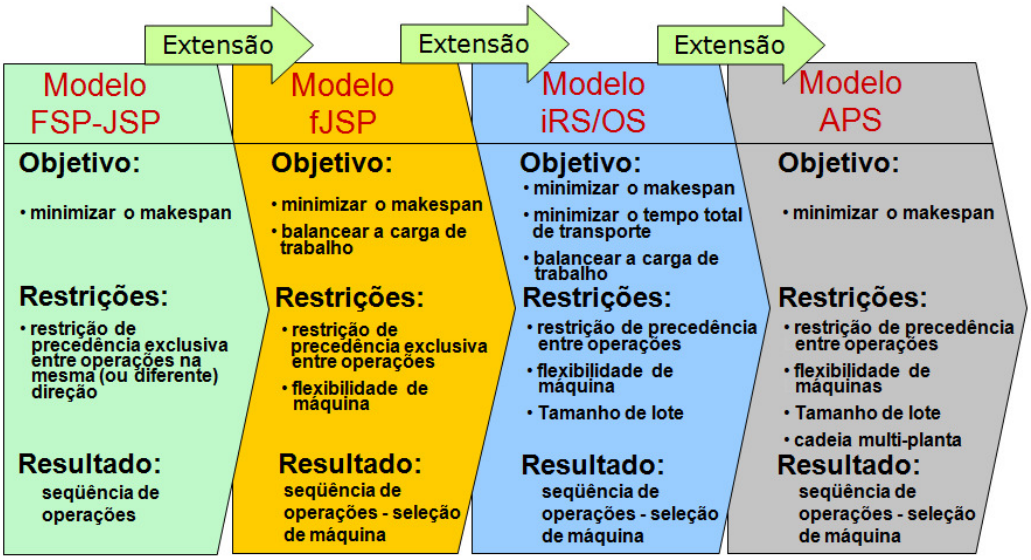


Figura 13 – Extensões ao problema APS
Fonte: Gen *et al.* (2009, p.800)

Num APS, o processo inicial registra os pedidos dos clientes. Para satisfazer os pedidos dos clientes, devem ser consideradas certas restrições, como tempos de *setup*, prazos de entrega e tempos de transporte. Deve-se ainda considerar a integração da alocação de recursos, a seqüência de operações e o escalonamento do problema, para minimizar o *makespan*. Além disso, em um ambiente MPC, um pedido, que representa uma seqüência de operações, pode ter uma de suas operações sendo realizada em outra planta, caso permitirem a carga de pedido e os dispositivos de transferência. Se for necessário o transporte entre plantas, o lote de transferência e o lote de processamento de produtos intermediários devem ser iguais (ZHANG *ET AL.*, 2006a).

As hipóteses consideradas no problema APS são:

- Todos os pedidos chegam simultaneamente sem precedência.
- Todas as máquinas estão disponíveis no início.
- A carga de pedido é fixa para todas as máquinas, e as máquinas não podem parar até terminar de processar o tamanho de lote.

- Para o mesmo pedido, o processamento numa operação começa logo que a carga do pedido da operação predecessora tenha sido concluída e transferida.
- Para o mesmo pedido em plantas diferentes, o processamento numa operação começa logo que o tamanho de lote da operação predecessora tenha sido concluída e transferida.
- O tempo de transporte entre máquinas diferentes é considerado.
- O tempo de *setup* entre operações é considerado.

A seguir, será apresentado um exemplo de uma instância do problema APS. A Figura 14 mostra dois tipos de materiais a serem processados, o Pedido 1 com tamanho de lote de 40 unidades e o Pedido 2 com tamanho de lote de 50 unidades. Repare que para obter ambos os produtos finais é preciso remover 10 volumes ou componentes dos dois materiais. A carga do pedido é de 10 unidades, isto é, para o Pedido 1, o número de sub-lotes é de $40/10 = 4$ e para o Pedido 2 é de $50/10 = 5$ (em cada máquina o tamanho de lote será processado em 10 unidades de cada vez). O plano de manufatura é mostrado na Tabela 5, a qual indica o tipo de cada operação e as máquinas disponíveis para realizar cada operação.

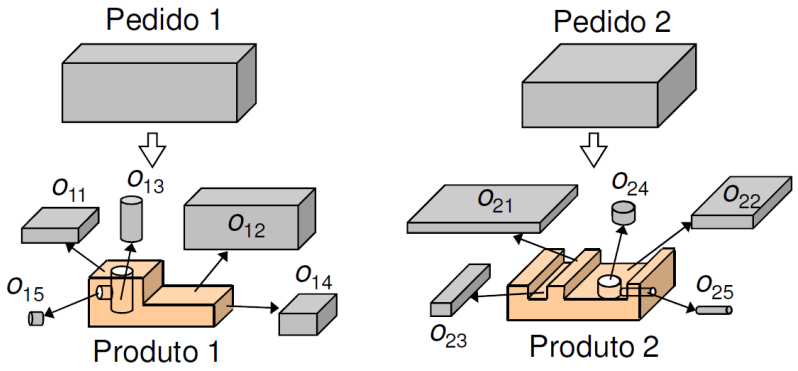


Figura 14 – Exemplo de um problema APS
Fonte: Zhang *et al.* (2006a, p.1842)

Tabela 5 – Plano de manufatura do exemplo APS

Índice da operação (Pedido $\boldsymbol{k}$, Operação $\boldsymbol{o_{ki}}$)	Tipo de operação	Máquinas disponíveis
1 (1, $\boldsymbol{o_{11}}$)	Fresagem	M_1, M_4
2 (1, $\boldsymbol{o_{12}}$)	Fresagem	M_1, M_4
3 (1, $\boldsymbol{o_{13}}$)	Perfuração	M_2, M_3, M_5
4 (1, $\boldsymbol{o_{14}}$)	Fresagem	M_1
5 (1, $\boldsymbol{o_{15}}$)	Perfuração	M_2
6 (2, $\boldsymbol{o_{21}}$)	Fresagem	M_1, M_2
7 (2, $\boldsymbol{o_{22}}$)	Fresagem	M_1, M_2
8 (2, $\boldsymbol{o_{23}}$)	Fresagem	M_3, M_5
9 (2, $\boldsymbol{o_{24}}$)	Perfuração	M_1, M_4, M_5
10 (2, $\boldsymbol{o_{25}}$)	Perfuração	M_1, M_2

Fonte: Zhang *et al.* (2006a, p.1842)

Na Figura 15, é mostrada a restrição de precedência dos pedidos. As seqüências de operações que podem ser geradas são múltiplas e devem cumprir as restrições de precedência de cada pedido. Por exemplo, para o Pedido 1, a seqüência de operações $\{o_{11}, o_{13}, o_{12}, o_{14}, o_{15}\}$ e $\{o_{11}, o_{12}, o_{13}, o_{14}, o_{15}\}$ são factíveis, enquanto que a seqüência $\{o_{13}, o_{11}, o_{12}, o_{14}, o_{15}\}$ é infactível.

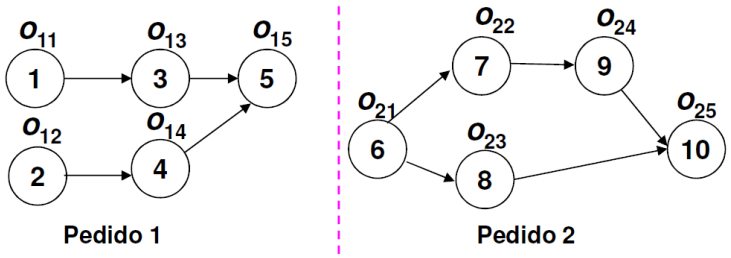


Figura 15 – Restrições de precedência do exemplo APS
Fonte: Zhang *et al.* (2006a, p.1842)

A Tabela 6 mostra a lista das máquinas disponíveis em cada planta e os tempos de processamento dados para realizar cada operação em cada planta. A Tabela 7 mostra o tempo de transporte dado entre as diferentes máquinas e a capacidade disponível L_m de cada máquina m.

Tabela 6 – Tempo de processamento das operações em máquinas diferentes do exemplo APS

	Pedido k	Pedido 1					Pedido 2				
	Operação o_{ki}	1	2	3	4	5	6	7	8	9	10
	Máquina M_m	o_{11}	o_{12}	o_{13}	o_{14}	o_{15}	o_{21}	o_{22}	o_{23}	o_{24}	o_{25}
Planta 1	M_1	7	7	-	6	-	3	8	-	10	6
	M_2	-	-	6	-	9	5	-	-	-	5
	M_3	-	-	5	-	-	-	12	5	-	-
Planta 2	M_4	5	6	-	-	-	-	-	-	10	-
	M_5	-	-	8	-	-	-	-	8	7	-

Fonte: Adaptado de Zhang *et al.* (2006a, p.1843)

Tabela 7 – Tempo de transporte entre máquinas diferentes do exemplo APS

		Planta 1		
Máquina M_n		M_1	M_2	M_3
Máquina M_m				
Planta 1	M_1	-	7	27
	M_2	19	-	15
	M_3	5	17	-
Capacidade L_m		1,500	1,500	1,500

		Planta 2	
Máquina M_n		M_4	M_5
Máquina M_m			
Planta 2	M_4	-	30
	M_5	5	-
Capacidade L_m		1,500	1,500

Fonte: Adaptado de Zhang *et al.* (2006a, p.1843)

Neste exemplo, o tempo de transporte entre máquinas de diferentes plantas é 100 e o lote de transferência entre plantas é igual ao tamanho de lote; não serão considerados tempos de *setup*.

Portanto, o problema APS pode ser definido como segue: dado um conjunto de K pedidos, com tamanho de lote q_k que devem ser processados por meio de J operações a serem realizadas em N máquinas conforme a disponibilidade em um ambiente com MPC. Deseja-se encontrar uma seqüência de operações dos pedidos, a alocação de máquinas e o escalonamento de todos os pedidos sobre as máquinas, de tal forma que se satisfaça as restrições de precedência e, além disso,

seja ótima em relação à minimização do *makespan* e do balanceamento de cargas de trabalho (ZHANG *ET AL.*, 2006a).

2.4.2 Formulação do problema

Para formular o modelo matemático do problema APS, que considera como medidas de desempenho a minimização do *makespan* e o balanceamento das cargas de trabalho serão apresentados as definições e notações dos conjuntos de índices, parâmetros e variáveis de decisão, a seguir (ZHANG *ET AL.*, 2006a).

Índices

k, l: Índices dos pedidos, $k, l = 1, 2, \dots, K$

i, j: Índices das operações, $i, j = 1, 2, \dots, J_k$

m, n: Índices das máquinas, $m, n = 1, 2, \dots, N$

d, e: Índices das plantas, $d, e = 1, 2, \dots, D$

Parâmetros

K: Número de pedidos

J_k: Número de operações por pedido k

J: Número de operações

N: Número de máquinas

D: Número de plantas

o_{ki}: i-ésima operação do pedido k

r_{kij}: Restrição de precedência entre operações

$$r_{kij} = \begin{cases} 1, \text{ se } o_{ki} \text{ for predecessor de } o_{kj} \text{ no pedido } k \\ 0, \text{ caso contrário} \end{cases}$$

M_m: m-ésima máquina

A_m: Conjunto de operações que podem ser processadas pela máquina m

M_{ki}: Conjunto de máquinas disponíveis para a operação o_{ki}

q_k: Tamanho de lote do pedido k

p_{kim}: Tempo de processamento da operação o_{ki} na máquina m

L_m: Capacidade da máquina m

u_{kij}: Carga do pedido k da operação o_{ki} para a operação o_{kj}

t^U_{kilj}: Tempo de setup entre a operação o_{ki} e a operação o_{lj}

t_{mn}: Tempo de transporte entre a máquina m e a máquina n

$$t_{mn} = \begin{cases} t^S_{mn}, \text{ se a máquina } m \text{ e } n \text{ estão na mesma planta} \\ t^D_{mn}, \text{ caso contrário} \end{cases}$$

b_{md}: Recursos da planta d

$$b_{md} = \begin{cases} 1, \text{ se a máquina } m \text{ pertence à planta } d \\ 0, \text{ caso contrário} \end{cases}$$

B_d: Conjunto de máquinas instaladas na planta d

Variáveis

$$y_{kilj} = \begin{cases} 1, \text{ se } o_{ki} \text{ for realizada imediatamente antes de } o_{lj} \\ 0, \text{ caso contrário} \end{cases}$$

$$x_{kim} = \begin{cases} 1, \text{ se a máquina } m \text{ for selecionada para a operação } o_{ki} \\ 0, \text{ caso contrário} \end{cases}$$

s_{ki}: Tempo de início da operação o_{ki}

c_{ki}: Tempo de conclusão da operação o_{ki}

$$c_{ki} = s_{ki} + q_k \sum_{m=1}^{N} p_{kim} x_{kim}$$

c_M: *Makespan*

w_B: Balanceamento da carga de trabalho

w_m: Carga de trabalho da máquina m

$$w_m = \sum_{k=1}^{K} \sum_{i=1}^{J_k} q_k p_{kim} x_{kim}$$

$\overline{w}$: Carga de trabalho médio de todas as máquinas

$$\overline{w} = \frac{1}{N} \sum_{m=1}^{N} w_m$$

O modelo matemático do problema APS é formulado a seguir:

$$\min c_M = \max_{\forall i,k}\{c_{ki}\} \quad (1)$$

$$\min w_B = \frac{1}{N}\sum_{m=1}^{N}(w_m - \overline{w})^2 \quad (2)$$

sujeito a:

$$(s_{lj} - (c_{ki} + t^U_{kilj}))x_{kim}x_{ljm}y_{kilj} \geq 0 \; \forall (k,i),(l,j),m \quad (3)$$

$$(s_{kj} - (s_{ki} + u_{kij}p_{kim} + t^S_{mn}))r_{kij}x_{kim}x_{kjn} \geq 0 \; \forall i,j,k,\forall m,n \in B_d,\forall d \quad (4)$$

$$(c_{kj} - (c_{ki} + t^S_{mn} + u_{kij}p_{kjn}))r_{kij}x_{kim}x_{kjn} \geq 0 \; \forall i,j,k,\forall m,n \in B_d,\forall d \quad (5)$$

$$(s_{kj} - (c_{ki} + t^D_{mn}))r_{kij}x_{kim}x_{kjn} \geq 0 \; \forall i,j,k,\forall m \in B_d,\forall n \in B_e,\forall d,e \quad (6)$$

$$\sum_{k=1}^{K}\sum_{i=1}^{J_k} q_k p_{kim}x_{kim} \leq L_m \; \forall m \quad (7)$$

$$r_{kij}y_{kjki} = 0 \; \forall i,j,k \quad (8)$$

$$y_{kiki} = 0 \; \forall i,j \quad (9)$$

$$y_{kilj} + y_{ljki} = 1 \; \forall (k,i),(l,j),(k,i) \neq (l,j) \quad (10)$$

$$\sum_{m=1}^{N} x_{kim} = 1 \; \forall i,k \quad (11)$$

$$x_{kim} = 0 \; \forall (k,i) \notin A_m \; \forall m \quad (12)$$

$$s_{ki} \geq 0 \; \forall i,k \quad (13)$$

$$y_{kilj} \in \{0,1\} \; \forall (k,i),(l,j) \quad (14)$$

$$x_{kim} \in \{0,1\} \; \forall i,k,m \quad (15)$$

Os objetivos a serem minimizados simultaneamente no problema APS são respectivamente o *makespan*, em (1), e o balanceamento de carga de trabalho, em (2).

A restrição (3) assegura que qualquer recurso (máquina) não pode ser selecionado por uma operação até que sua operação predecessora seja completada, e também que o tempo de *setup* seja considerado, tal como mostra a Figura 16.

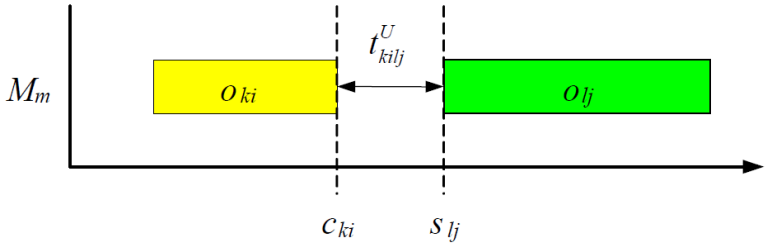


Figura 16 – Gráfico de precedência em uma mesma máquina
Fonte: Zhang *et al.* (2006a, p.1844)

As restrições (4) e (5) asseguram o transporte de produtos intermediários na mesma planta. Estas restrições devem ser satisfeitas simultaneamente para garantir que as operações sejam executadas em uma máquina sem interrupção, como mostra a Figura 17. Nesta figura, $u_{kij}p_{kim}$ e $u_{kij}p_{kjn}$ representam tempos de processamento de um sub-lote ou carga do pedido.

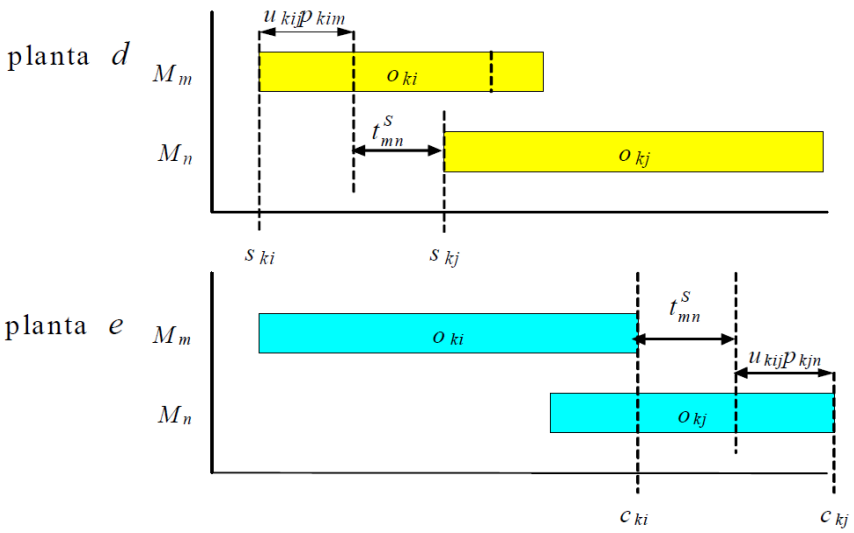


Figura 17 – Gráfico de precedência em uma mesma planta
Fonte: Zhang *et al.* (2006a, p.1844)

A restrição (6) restringe o transporte entre plantas diferentes. Assegura que uma operação não possa ser realizada em outra planta até o tamanho de lote da operação predecessora não esteja concluído, como mostra a Figura 18.

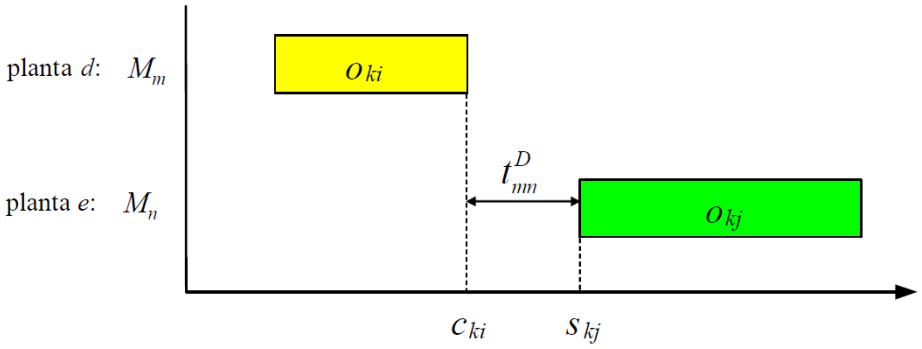


Figura 18 – Gráfico de precedência entre diferentes plantas
Fonte: Zhang *et al.* (2006a, p.1844)

A restrição (7) restringe a capacidade de produção disponível de cada máquina. A restrição (8) garante que a restrição de precedência entre operações não seja violada. As restrições (9) e (10) asseguram a viabilidade da seqüência de operações. Já as restrições (11) e (12) asseguram a viabilidade da alocação de máquinas. A restrição (13) impõe a condição de não negatividade à $s_{ki}, \forall i, k$, enquanto que as restrições em (14) e (15) definem $x_{kim}, \forall i, k, m,$ e $y_{kilj}, \forall (k,i),(l,j)$, como variáveis binárias.

2.4.3 Representação de soluções

Uma solução de APS é representada por dois vetores: o vetor de seqüência de operações s^* e o vetor de alocação de máquinas v^*.

Considerando o exemplo descrito na seção 2.4.1, a construção em duas etapas dos vetores s^* e v^* é ilustrada a seguir.

Etapa 1: Geração da seqüência de operações

O processo começa gerando o vetor de nível de prioridade π correspondente ao conjunto de operações. Se J é o número total de operações, então, o vetor π é formado pela permutação aleatória de $\{1, 2, \ldots, J\}$. A seguir é mostrado um exemplo simples da representação de π:

Índice j	1	2	3	4	5	6	7	8	9	10
Prioridade $\pi(j)$	5	1	7	9	4	6	3	8	2	10

Utilizando o procedimento de seqüenciamento de operações, proposto por Liu *et al*. (2009) e apresentado no Anexo A. podemos obter a seqüência factível $s^* = \{o_{12}, o_{11}, o_{21}, o_{22}, o_{24}, o_{13}, o_{23}, o_{14}, o_{15}, o_{25}\}$, a qual é escrita através de seus índices como $s^* = \{2, 1, 6, 7, 9, 3, 8, 4, 5, 10\}$, de acordo com a representação acima.

Etapa 2: Alocação de máquinas

Agora que a seqüência de operações foi gerada, podemos estabelecer o vetor v^* mediante o procedimento de alocação de máquinas, proposto por Zhang *et al*. (2006b) e apresentado no Anexo B.

Continuando com o exemplo, a representação gráfica do procedimento de alocação de máquinas é mostrada na Figura 19.

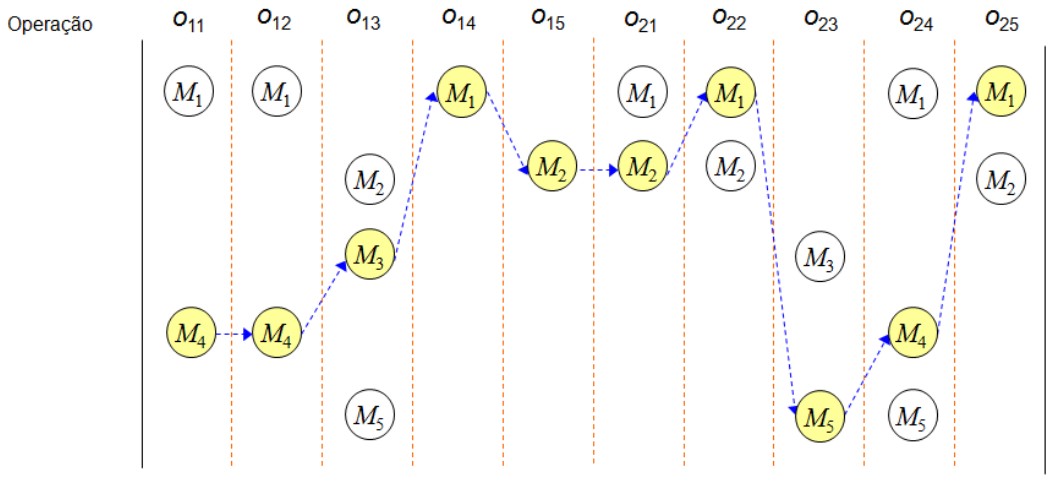


Figura 19 – Representação gráfica da alocação de máquinas do exemplo APS
Fonte: Zhang *et al*. (2006a, p.1845)

Finalmente, a solução factível é dada por:

	o_{12}	o_{11}	o_{21}	o_{22}	o_{24}	o_{13}	o_{23}	o_{14}	o_{15}	o_{25}
Seqüência s^*	2	1	6	7	9	3	8	4	5	10

Índice j	1	2	3	4	5	6	7	8	9	10
Máquina $v^*(j)$	4	4	3	1	2	2	1	5	4	1

Utilizando os dados das Tabelas 6 e 7, podemos elaborar o diagrama de Gantt, mostrado na Figura 20.

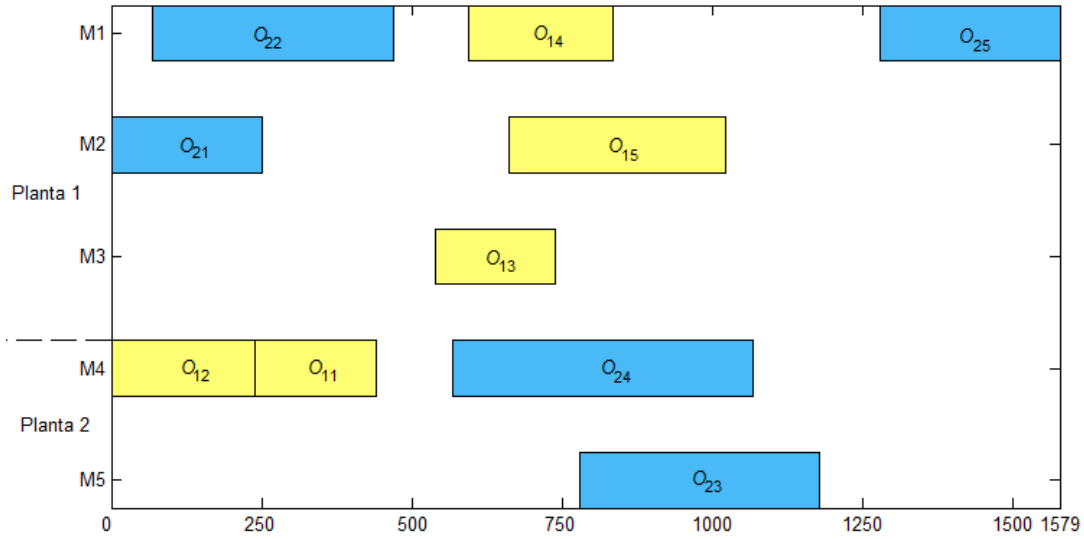


Figura 20 – Diagrama de Gantt do exemplo APS

2.4.4
Geração de vizinhança para máquinas

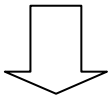
Este tipo de vizinhança pode ser gerado atribuindo-se o conjunto de máquinas disponíveis $m \in M_{ki}$ para a operação o_{ki}. Se J é o número total de operações e N é o número de máquinas, então, o número máximo de soluções vizinhas é dado por: $J(N-1)$, já que o problema pode ser parcial ou totalmente flexível.

2.4.5
Geração de vizinhança para seqüências

Como no caso dos problemas fJSP e iRS/OS, um método simples para gerar soluções vizinhas para o problema APS é o método de troca entre duas operações de uma seqüência, que gerem seqüências factíveis. Considerando a seqüência de operações do exemplo anterior, a vizinhança gerada pelo método de troca de operações é mostrada a seguir:

Seqüência s^*	2	1	6	7	9	3	8	4	5	10

soluções geradas pelo
método de troca entre operações



Vizinhança s*

1	2	6	7	9	3	8	4	5	10
6	1	2	7	9	3	8	4	5	10
2	6	1	7	9	3	8	4	5	10
2	1	6	7	3	9	8	4	5	10
2	1	6	7	8	3	9	4	5	10
2	1	6	7	4	3	8	9	5	10
2	1	6	7	9	8	3	4	5	10
2	1	6	7	9	4	8	3	5	10
2	1	6	7	9	3	4	8	5	10
2	1	6	7	9	3	8	4	10	5

Note que, se J é o número total de operações, então, o número soluções geradas é dada por: $\frac{1}{2}(J)(J-1)$. Para esse exemplo com 10 operações ($J = 10$) o número total de permutações que podem ser geradas a partir de s^* é $\frac{1}{2}(10)(9) = 45$, porém somente 10 seqüências são factíveis.