Duas abordagens para a minimização do makespan em um sistema integrado flowshop-VRP

Guilherme Sproesser Ferreira
Departamento de Engenharia de Produção
Universidade Federal de São Carlos
Rod. Washington Luís km 235 - SP-310 - São Carlos
sproesser.f@gmail.com

Roberto F. Tavares Neto
Departamento de Engenharia de Produção
Universidade Federal de São Carlos
Rod. Washington Luís km 235 - SP-310 - São Carlos
tavares@dep.ufscar.br

RESUMO

Com a possibilidade de integração de sistemas de gestão, cada vez mais é possível a integração de diferentes atividades de planejamento de curto prazo em sistemas produtivos. Um exemplo é a integração entre *scheduling* e distribuição. Esse trabalho aborda um sistema produtivo composto de um ambiente *flowshop* com duas operações, cuja produção é entregue a clientes geograficamente dispersos através de um conjunto de veículos homogêneos. Foram propostas duas abordagens distintas: uma baseada na heurística *Iterated Greedy* e outra baseada em um modelo de programação matemática. Os resultados mostraram que o modelo matemático possui muita dificuldade em resolver esse problema mesmo para instâncias pequenas, sendo um indicativo de uma necessidade do uso de métodos híbridos para realizar essa tarefa.

PALAVRAS CHAVE. Problemas Integrados Produção Distribuição, Scheduling, VRP

Tópicos: AD & GP, L & T

ABSTRACT

With the increasing usage of enterprise-wide integrated management systems, it's now possible to implement integrated planning policies through distinct parts os a production system. One example is the integration between *scheduling* and logistics. This paper presents a problem where a production system is composed of a 2-machine flowshop and a distribution by a set of vehicles. Two approaches were proposed: the first one, a heuristic based on the Iterated Greedy technique and the second one, based on a mathematical programming model. Our results show that the exact approach finds much difficulty to solve this problem, indicating the need to develop hybrid methods to solve this task.

KEYWORDS. Integrated Production Distribution Problems, Scheduling, VRP

Paper topics: AD & GP, L & T

1. Introdução

O planejamento integrado das diversas funcionalidades de um sistema produtivo é fator chave para o sucesso de uma empresa Min e Zhou [2002]. Para isso é necessário determinar como otimizar um conjunto de indicadores de desempenho, normalmente considerados isoladamente. A literatura nos mostra que existem diferentes focos de como realizar essa integração, sendo classificadas no trabalho de Chen e Lee [2004] como: estratégia de compras, gestão de fornecimento, integração de logística e coordenação de cadeia de suprimento.

Na literatura científica que trata do planejamento integrado produção-distribuição (*Integrated Production-Distribution Problems* - IPDP), a produção é tratada na literatura através de duas principais frentes: a primeira, normalmente classificada como uma decisão de nível tático se foca no dimensionamento de lotes de produção e planejamento de custos de estoque; a segunda, se foca em decisões de curto prazo de sequenciamento e programação da produção - *scheduling* (nível operacional), Reimann et al. [2014].

A definição do nível de detalhamento do problema abordado pode permitir uma simplificação considerável. Como no caso do problema de roterização de veículos, que dependendo como os clientes estão distribuídos, como no trabalho de Lee e Chen [2001] ou a forma como é estruturado o canal de distribuição como em Li et al. [2005], podem gerar simplicações no momento do desenvolvimento de soluções alternativas. Inspirado na necessidade de se desenvolver conhecimento teórico nessa área a pesquisa foca na área de estudo do ambiente *flowshop*, como pode ser visto na próxima seção referente a revisão bibliográfica. Nos últimos anos a capacidade computacional avançou muito permitindo a implementação de algoritmos exatos como também heurísticas lidando com problemas mais complexos, como o planejamento integrado.

Assim, o objeto de estudo deste artigo foca no estudo de duas decisões simultâneas: (i) como sequenciar e programar um conjunto de ordens de serviço na manufatura; (ii) como realizar a distribuição, considerando que cada rota de entrega contempla várias ordens cujos pontos de entrega são separados por distâncias não desprezíveis e realizadas por um número não limitado de veículos de capacidade ilimitada. O ambiente produtivo é descrito como sendo um ambiente *flowshop*. A distribuição é entendida como sendo uma extensão do problema clássico do roteamento de veículos capacitados (*Capacitated Vehicle Routing Problem - CVRP*). O objetivo escolhido, tendo como base a literatura, é minimizar o makespan do sistema, ou seja, o momento em que a última rota retorna à origem.

A abordagem estudada e implementada foi o *Iterated Greedy* (IG), sendo os pontos fortes do algoritmo sua simplicidade de programação quando comparado com outras meta-heurísticas, além de ter desempenho aceitável na resolução de problemas de *scheduling* publicado por Ruiz e Stützle [2007]. O objetivo do desenvolvimento do algoritmo é minimizar o tempo tota de fluxo (tempo entre o pedido do cliente e o recebimento do produto) de uma ordem no sistema, considerando que a ordem ”termina”quando chega ao cliente final. O algoritmo apresentou soluções para um ambiente *flowshop* com duas máquinas e distribuição dada por um número ilimitado de veículos com capacidade ilimitada, sendo o veículo habilitado a percorrer múltiplas rotas.

2. Revisão Bibliográfica

2.1. Programação em *flowshop*

O ambiente *flowshop* de manufatura é caracterizado por um fluxo de n tarefas que é processada em uma determinada ordem entre as m máquinas. Existem diversas variações de *flowshop*, mas para efeito de estudo da presente pesquisa, utilizaremos as características apresentadas na pesquisa apresentada por Cheng et al. [2015] que explora o ambiente de duas máquinas em regime *flowshop* com restrição de precedência, ou seja, uma tarefa só pode ir para a etapa seguinte após completar o primeiro processamento. Também chamado de Ambiente *Flowshop* Permutacional (PFSP).

Hamdi et al. [2015] utilizam o *Branch and Bound* (B&B) afim de minimizar o tempo de espera total entre as operações de uma tarefas a partir do uso de algumas regras de dominância

já conhecidas e o desenvolve uma melhora na regra de Kim [1993] que busca identificar *jobs* que podem ser colocadas no final do sequenciamento. Enquanto, Msakni et al. [2015] buscam minimizar o atraso relacionado a execução de cada tarefa, neste modelo leva-se em consideração o tempo de espera de cada ordem de serviço para a próxima máquina aumentando a aplicabilidade do modelo em situações reais.

Hnaien et al. [2015] consideraram o ambiente *flowshop* com duas máquinas em que a primeira máquina possui período de inviabilidade devido à manutenção preventiva. Os autores formularam duas versões de modelos de programação inteira mista (*Mixed-integer-programming*-MIP). Por último, os autores propuseram um método (B&B) e concluiram, nesse caso, que o algoritmo B&B é superior aos modelos lineares. Os resultados computacionais mostram que o modelo MIP foi capaz somente de resolver instâncias de teste de até vinte ordens de serviço, enquanto o B&B foi capaz de resolver instâncias de até setenta ordens, chegando em soluções ótimas e menor tempo computacional.

O artigo de An et al. [2016] foca no problema de sequenciamento em que as tarefas são processadas em um ambiente *flowshop* com duas máquinas, com limite de tempo de espera (depois de processada na primeira máquina a tarefa tem um tempo limite para começar a ser processada na segunda) e setup dependente do sequenciamento, modelo altamente aplicável para a indústria de semicondutores. O algoritmo B&B é proposto com objetivo de minimizar o *makespan*. Devido ao grande esforço computacional necessário para resolver o problema novas heurísticas, limites e regras de dominância são as principais contribuições do artigo.

A pesquisa de Cheng et al. [2015] explora o ambiente de duas máquinas em regime *flowshop* com restrição de precedência, ou seja, uma tarefa só pode ir para a etapa seguinte após completar o primeiro processamento. Esse tipo de modelo é aplicável no caso de hospitais no sequenciamento de atendimentos médicos. É proposto a utilização do algoritmo B&B para encontrar solução ótima e *Genetic Algorithm* (GA) e *Large-order-value* (LH) para encontrar uma solução aproximada. Conclui-se que o algoritmo GA possui melhor desempenho em comparação com os demais.

Nos estudos relatados por Ying [2015] os métodos *Simulated Annealing* SA e *Iterated Greedy* (IG) são utilizados para sequenciamento de tarefas em um ambiente de duas máquinas *flowshop* com o objetivo de prevenção contra tempos de processamento incertos (grande aplicabilidade em problemas reais). Os resultados experimentais demonstraram que os dois métodos são efetivos para resolução do problema, entretanto para problemas de maior escala o IG mostrou-se mais eficiente.

Shiau et al. [2015] levam em consideração o efeito da curva de aprendizado nos tempos de processamento do ambiente *flowshop* permutacional com aprendizado tendo aplicações na indústria química, refinarias de petróleo e fabricação de aço. O objetivo é minimizar o tempo total para completar as tarefas sem exceder o valor máximo de atraso de uma tarefa, os métodos utilizados são GA e B&B. Os dados experimentais evidenciaram o GA, com *crossover* uniforme e mutação em um ponto, como melhor opção, dentre os GA propostos, principalmente para problemas de maior porte. Enquanto o B&B resolveu problemas de até vinte tarefas em um intervalo de até uma hora.

Riahi e Kazemi [2016] propõem uma meta-heurística baseada no comportamento de uma colônia de formigas em busca de alimento (ACO), o modo de construção da solução como também o comportamento de atualização da trilha de feromônios são estudados para melhorar o algoritmo, é realizada uma hibridação entre ACO e com *Variable Neighborhood Search* (VNS), que mostrou melhores resultados. O objetivo da pesquisa é minimizar o *makespan* no *flowshop* sem espera, neste ambiente cada tarefa deve ser processada da primeira até a última máquina sem interrupção, sendo a sequência entre as máquinas única para todas as tarefas e cada máquina pode processar apenas uma tarefa por vez. Por meio da aplicação dos algoritmos estudados pode-se concluir que ao contrário da maioria dos estudos relacionados ao ACO o algortimo formulado neste artigo é de fácil implementação e replicação. A análise estatística mostrou que o híbrido ACO-SA é viável e

confiável apresentando resultados computacionais promissores.

Outros autores exploram o uso de modelos lineares quem podem ser resolvidos com a biblioteca CPLEX como Wang et al. [2016]. No primeiro modelo linear são definidas três variáveis binárias a primeira para identificar quais tarefas são aceitas no sequenciamento, a segunda é utilizada para definir o local em que essa tarefa é inserida no sequenciamento e a última variável que identifica qual tarefa deve ser processada após determinada tarefa, com o objetivo de otimizar o *makespan*. O segundo modelo linear é baseado em variáveis que mantém a informação a respeito dos tempos de finalização das tarefas, com o objetivo de diminuir o atraso total ponderado das tarefas. O último modelo aplicado é o B&B em que os limites inferiores são definidos a partir de 2 heurísticas escolhendo a que possui maior valor da função objetivo, a primeira heurística basicamente sequencia as tarefas de acordo com as datas de entrega priorizando as tarefas com prazos menores, já a segunda heurística utilizada de uma penalização por atrasos na receita total com a vendas dos produtos afim de otimizar a receita. Já o limite superior é definido a partir de dois algoritmos. O artigo conclui que os modelos lineares são capazes apenas de resolver problemas de pequeno porte, e que o método B&B é superior aos modelos lineares podendo resolver problemas de até vinte tarefas em menos de uma hora.

Ainda com foco no uso de programação matemática, Della Croce et al. [2000] propõem uma simplificação do modelo, que, após uma etapa de reordenação das tarefas usando a regra de *Johnson*, é reduzido para uma variação do problema da mochila (*knapsack problem*).

2.1.1. Algoritmo NEH (Nawaz-Enscore-Ham)

O algoritmo NEH apresentado no artigo de Nawaz et al. [1983] é utilizado para a construção da solução inicial do problema. Seguindo os seguintes passos:

1. Ordenar as tarefas em ordem decrescente de tempo total de processamento.

2. Montar duas possíveis subsequências com as duas tarefas com maior tempo total de processamento, escolher a com menor *makespan*.

3. Inserir as outras tarefas ordenadas decrescentemente na subsequência escolhida no passo anterior (2), inserir em todos os lugares possíveis e escolher a subsequência com menor makespan. Realizar este passo até que todas as tarefas sejam alocadas.

Seguido estes passos é priorizado o sequenciamento das tarefas com maior tempo de processamento somado nas duas máquinas. Ou seja, a partir de uma subsequência contendo as duas tarefas com maior tempo de processamento as demais tarefas são inseridas com o objetivo de alcançar o menor *makespan*.

2.2. Integração Produção-Distribuição

Indo além dos limites de um problema de *scheduling* clássico, nos últimos anos a coordenação entre as operações de produção e distribuição de produtos está cada vez mais importante para a gestão da cadeia de suprimentos. A partir de uma análise integrada é possível diminuir o tempo total dos produtos no sistema e ainda reduzir custos com estoque e distribuição por meio de uma melhor coordenação entre a logística de entrega e o *scheduling* da produção.

De acordo com Wang et al. [2015] os problemas de planejamento integrado podem ser classificados de duas formas: (i) considerando a integração entre produção e distribuição ou (ii) considerando a integração entre produção, estoque e distribuição. O primeiro grupo chamado de *Integrated Scheduling of Production and Distribution Problems* (ISPDP), geralmente aborda indústrias em que o produto final é distribuído imediatamente, como jornais e produtos químicos que não podem ser armazenados por muito tempo.

Já o segundo grupo denominado *Integrated Scheduling of Production, Inventory and Distribution Problems* (ISPIDP), o estoque é levado em consideração e muitas vezes usado para suavizar os impactos das mudanças da demanda.

Figura 1: Diagrama básico do problema abordado

3. Definição do Problema

Esta pesquisa aborda um (ISPIDP) com sequenciamento de tarefas em duas máquinas em sequência, em que a ordem das operações realizadas nas máquinas é igual para todas as tarefas e não pode ser alterada (PFSP). Além disso, a distribuição é feita com μ veículos de capacidade limitada.

Tabela 1: Símbolos usados nesse artigo

Símbolo	**Descrição**
i, j	Índices utilizados para definir a ordem de processamento
k	Índice usado para definir um veículo
ρ_{i0}, ρ_{i1}	Tempo de processamento da tarefa i na máquina 0 e 1
δ_{ij}	A distância entre i e j
y_{ij}	Variável binária que indica se uma ordem j é processada após uma ordem i ou não
x_{ijk}	Variável binária que indica se uma ordem j é entregue após uma ordem i pelo veículo k
$C0_i, C1_i$	Variável que indica o tempo de finalização da ordem i na máquina 0 e 1, respectivamente
S_k	Variável que indica quando o veículo k pode sair para entrega
D_i	Variável que indica quando a ordem i é entregue
z	Valor do *makespan*

3.1. Uma representação usando modelo matemático

O problema descrito anteriormente pode ser descrito através das equações 1-2.

$$z \geq D_i \qquad \forall \{i > 0 \qquad (1)$$

$$y_{ii} = 0 \qquad \forall \{i \qquad (2)$$

$$\sum_j y_{ij} = 1 \qquad \forall \{i \qquad (3)$$

$$\sum_i y_{ij} = 1 \qquad \forall \{j \qquad (4)$$

$$\sum_i y_{ij} = \sum_i y_{ji} \qquad \forall \{j \qquad (5)$$

$$C0_0 = 0 \qquad (6)$$

$$C1_0 = 0 \qquad (7)$$

$$C0_j \geq C0_i + \rho_{j0} - M \cdot (1 - y_{ij}) \qquad \forall \begin{cases} j > 0 \\ i \end{cases} \qquad (8)$$

$$C1_j \geq C1_i + \rho_{j1} - M \cdot (1 - y_{ij}) \qquad \forall \begin{cases} j > 0 \\ i \end{cases} \qquad (9)$$

$$C1_j \geq C0_i + \rho_{j1} \qquad \forall \{j > 0 \qquad (10)$$

$$S_k \geq C1_i - M \cdot (1 - x_{ijk}) \qquad \forall \begin{cases} i > 0 \\ j \\ k \end{cases} \qquad (11)$$

$$x_{iik} = 0 \qquad \forall \begin{cases} k \\ i \end{cases} \qquad (12)$$

$$\sum_i x_{0ik} = 1 \qquad \forall \{k \qquad (13)$$

$$\sum_i x_{i0k} = 1 \qquad \forall \{k \qquad (14)$$

$$\sum_{j,k} x_{jik} = 1 \qquad \forall \{i > 0 \qquad (15)$$

$$\sum_{i,k} x_{jik} = 1 \qquad \forall \{j > 0 \qquad (16)$$

$$\sum_i x_{ijk} = \sum_i x_{jik} \qquad \forall \begin{cases} j \\ k \end{cases} \qquad (17)$$

$$D_0 = 0 \qquad (18)$$

$$D_i \geq S_k + \delta_{0i} - M \cdot (1 - x_{0ik}) \qquad \forall \begin{cases} i > 0 \\ k \end{cases} \qquad (19)$$

$$D_j \geq D_i + \delta_{ij} - M \cdot (1 - x_{ijk}) \qquad \forall \begin{cases} i > 0 \\ k \end{cases} \qquad (20)$$

Nesse modelo, as restrições 2-5 e 12-17 garantem a factibilidade dos dados representados pelas variáveis x e y. O tempos de finalização na produção são estabelecidos pelas restrições 6-10. Os tempos de finalização da produção e os tempos de liberação dos veículos são relacionados nas equações 11. Os tempos de entrega de clientes são dados nas equações 18-20. O *makespan* do sistema é determinado pelas equações 1.

4. Abordagem do *Iterated Greedy* (IG)

A estratégia de solução adotada foi inspirada nas pesquisas de Ruiz e Stützle [2007] e Ruiz e Stützle [2008], o algoritmo IG basicamente constrói soluções a partir de duas fases, a fase de destruição e a fase de construção. Na fase de destruição alguns elementos são removidos da solução inicial aleatoriamente, posteriormente na fase de construção é aplicada uma heurística, neste casao a heurística NEH, até que a solução esteja completa novamente. Então é analisado o critério de aceitação da nova solução, caso a solução passe no critério ela será a nova solução encontrada para o problema. Na presente estratégia o critério de aceitação da nova solução é ela ter *makespan* menor

que a solução anterior. Os parâmetros que precisam ser estabelecidos para executar o algoritmo são o número de tarefas removidas a cada fase de destruição e o número de ciclos de destruição e construção.

Algoritmo 1: Algortimo Proposto

Algoritmo IG
A cada iteração na fase de destruição *n* tarefas são removidas da solução aleatóriamente, por exemplo (j_1 e j_2).
Após a destruição inicia-se a fase de construção em que é utilizado o Algoritmo NEH para inserir as tarefas removidas.
Algoritmo NEH para cada *tarefa restante* ($j_3, ..., j_n$) **hacer**
    Fixe a ordem das tarefas previamente sequenciadas e insira a tarefa inicialmente no sequenciamento do *scheduling* entre todas as possibilidades de posições para inserção, e posteriormente insira em todas as possibilidades de rotas, a posição final tanto no sequenciamento para produção como na rota de distribuição é a que resulta no menor *makespan*.
fin
fim Algoritmo NEH
fim Algoritmo IG

5. Resultados e Análises

Para analisar o algoritmo proposto na seção anterior, implementou-se o modelo matemático usando a biblioteca CPLEX 12.6 para Python3.4. O algoritmo proposto também foi implementado em Python3.4. Foram geradas 12 instâncias com 5 ordens de serviço e 12 instâncias com 10 ordens de serviço. Os parâmetros para geração das instâncias envolveram o tempo de processamento em cada máquina que pode variar aleatóriamente entre 1 e 100. Já as coordenadas dos pontos de demanda e o centro de distribuição foram gerandos aleatóriamente podendo suas coordenadas variarem de 0 a θ_D, para um mesmo número de ordens *n* os valores de θ_D foram [5, 10, 30], com as coordenas foi calculada a distância euclidiana ente os pontos. Além disso, para analisar a influência da distância entre o centro de distribuição e os clientes foi utilizado um fator de multiplicação θ_G assumindo os valores [1, 10, 20, 30] que multiplicou o a distância entre o centro de distribuição e cada um dos pontos de entrega. Por fim, foi determinado que os veículos para entrega estariam sempre disponíveis e sem limite de capacidade.

As instâncias de teste, assim como seu processo de geração, estão disponíveis on-line [1]. O mesmo conjunto de parâmetros foi usado no caso do IG e foram usadas as configurações padrão do CPLEX. Os resultados são mostrados na tabela 2. Nessa tabela, duas informações são mostradas: o *gap* da função objetivo ($gap = (valorIG - valorMIP)/valorMIP$) e os tempos computacionais médios.

Tabela 2: Resultados obtidos

	N=5	**N=10**
gap	24.48%	15.89%
Tempo médio IG [s]	0.0087	0.0593
Tempo médio MIP [s]	49.8281	600.2390

Percebe-se que, principalmente quando aumentamos o número de ordens, o IG conseguiu obter resultados mais próximos ao MIP. Embora o gap seja alto, percebe-se que o tempo computacional é muito menor no caso do IG. Uma análise dos dados nos permitiu perceber que a abordagem exata encontrou muito mais dificuldade em lidar com as instâncias maiores que o IG.

[1] http://www.dep.ufscar.br/tavares

6. Considerações finais

Esse trabalho apresentou duas abordagens para se resolver o problema de minimização de *makespan* em um sistema produtivo composto por um *flowshop* com duas operações e uma distribuição realizada por veículos homogêneos.

O modelo matemático, embora tenha conseguido representar o problema, se mostrou com dificuldades em resolver problemas com 10 ordens de serviço (tamanho ainda considerado pequeno pela literatura). O algoritmo IG, por outro lado, não se mostrou tão sensível à essa variação.

Como perspectivas de estudo, são previstas três principais vertentes de peesquisa: a primeira, diz respeito a um estudo sobre os parâmetros mais adequados do algoritmo IG e das configurações do CPLEX para obter melhores resultados (inclusive considerando problemas de maior porte). A segunda vertente se foca na aplicação de técnicas que gerem um ganho de performance ao modelo matemático, permitindo assim seu uso em problemas com maior número de ordens. Por fim, espera-se que futuros trablahos incluam restrições comuns da literatura (como janelas de tempo, restrições de capacidade de veículo e armazenamento, etc).

Agradecimentos

Os autores desse artigo agradecem à FAPESP, que vem apoiando esse trabalho através do proc.2016/09080-5.

Referências

An, Y. J., Kim, Y. D., e Choi, S. W. (2016). Minimizing makespan in a two-machine flowshop with a limited waiting time constraint and sequence-dependent setup times. *Computers and Operation Research*, 71:127–136.

Chen, C.-L. e Lee, W.-C. (2004). Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *Computers & Chemical Engineering*, 28: 1131–1144.

Cheng, S.-R., Yin, Y., Wen, C.-H., Lin, W.-C., Wu, C.-C., e Liu, J. (2015). A two-machine flowshop scheduling problem with precedence constraint on two jobs. *Soft Computing Springer*, p. 1–13.

Della Croce, F., Gupta, J. N., e Tadei, R. (2000). Minimizing tardy jobs in a flowshop with common due date. *European Journal of Operational Research*, 120(2):375 – 381.

Hamdi, I., Oulamara, A., e Loukil, T. (2015). A branch and bound algorithm to minimise the total tardiness in the two-machine permutation flowshop scheduling problem with minimal time lags. *Int. J. Operational Research Elsevier*, 23(4):387–405.

Hnaien, F., Yalaoui, F., e Mhadhbi, A. (2015). Makespan minimization on a two-machine flowshop with an availability constraint on the first machine. *Int. J. Production Economics Elsevier*, 164: 95–104.

Kim, Y. D. (1993). A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. *Computers & Operations Research*, 20(4):391–401.

Lee, C.-Y. e Chen, Z.-L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling*, 4:3–24.

Li, C.-L., Vairaktarakis, G., e Lee, C.-Y. (2005). Machine scheduling with deliveries to multiple customer locations. *European Journal of Operational Research*, 164:39–51.

Min, H. e Zhou, G. (2002). Supply chain modeling: past, present and future. *Computers & Industrial Engineering*.

Msakni, M. K., Khallouli, W., Al-Salem, M., e Ladhari, T. (2015). Minimizing the total completion time in a two-machine flowshop problem with time delays. *Engineering Optimization*, 48:1164–1181.

Nawaz, M., Enscore, E. E. J., e Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *The International Journal of Management Science*, 11(1):91–95.

Reimann, M., Tavares Neto, R. F., e Bogendorfer, E. (2014). Joint optimization and vehicle routing problems: A review of existing strategies. *Pesquisa Operacional.*

Riahi, V. e Kazemi, M. (2016). A new hybrid ant colony algorithm for scheduling of no-wait flowshop. *Operational Research Springer*, p. 1–20.

Ruiz, R. e Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049.

Ruiz, R. e Stützle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187:1143–1159.

Shiau, R.-Y., Tsai, M.-S., Lee, W.-C., e Cheng, T. (2015). Two-agent two-machine flowshop scheduling with learning effects to minimize the total completion time. *Computers & Industrial Engineering*, 87:580–589.

Wang, D. Y., Grunder, O., e El Moudni, A. (2015). Integrated scheduling of production and distribution operations: A review. *International Journal of Industrial and Systems Engineering*, 19 (1):94 – 122.

Wang, X., Xie, X., e Cheng, T. (2016). Order acceptance and scheduling in a two-machine flowshop. *Operational Research Springer*, p. 1–20.

Ying, K.-C. (2015). Scheduling the two-machine flowshop to hedge against processing time uncertainty. *Operational Research Society*, 66:1413–1425.