UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

COMPARAÇÃO ENTRE ESTRATÉGIAS PARA PROGRAMAÇÃO INTEGRADA DE OPERAÇÕES ENVOLVENDO DECISÕES DE PROGRAMAÇÃO DE OPERAÇÕES DA PRODUÇÃO E ROTEIRIZAÇÃO DE VEÍCULOS

JULIANE FERNANDES CAETANO SOUZA

DISSERTAÇÃO DE MESTRADO

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

COMPARAÇÃO ENTRE ESTRATÉGIAS PARA PROGRAMAÇÃO INTEGRADA DE OPERAÇÕES ENVOLVENDO DECISÕES DE PROGRAMAÇÃO DE OPERAÇÕES DA PRODUÇÃO E ROTEIRIZAÇÃO DE VEÍCULOS

Juliane Fernandes Caetano Souza

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Produção.

Orientador: Prof. Dr. Roberto Fernandes Tavares Neto

SÃO CARLOS
2022

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Juliane Fernandes Caetano Souza, realizada em 03/02/2022.

Comissão Julgadora:

Prof. Dr. Roberto Fernandes Tavares Neto (UFSCar)

Prof. Dr. Fábio Molina da Silva (UFSCar)

Prof. Dr. Marcelo Seido Nagano (USP)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.
O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Engenharia de Produção.

AGRADECIMENTOS

Agradeço à minha família e amigos pelo apoio incondicional, especialmente aos meus pais, irmã e parentes mais próximos que, mesmo estando a alguns quilômetros de distância, se mantiveram constantes em suas manifestações de apoio e carinho.

Agradeço meu orientador por ter me guiado durante toda a construção do trabalho e também por todos seus momentos de paciência e compreensão.

Agradeço ao programa de Pós-graduação em Engenharia de Produção da UFSCar por possibilitar a realização dessa pesquisa.

RESUMO

A principal motivação do atual trabalho é o estudo de uma técnica de refinamento das soluções obtidas com a programação e sequenciamento de operações integradas de produção e distribuição. É possível afirmar que integrar essas operações e resolvê-las de forma única é importante para garantir a busca pela melhor solução para o sistema. O objetivo da pesquisa é comparar os resultados obtidos a partir de três formas de resolução: um modelo matemático resolvido de forma desacoplada, um modelo integrado e um modelo integrado que utiliza uma solução inicial do problema desacoplado. A análise foi realizada em problemas onde as principais características são: máquina única e único veículo para distribuição. Como objetivo, priorizou-se a minimização do tempo total de fluxo das ordens no sistema, normalmente relacionada pela literatura com o tempo total de atendimento de uma ordem e com níveis de estoque. Para atingir o objetivo utilizou-se de programação linear inteira mista e comparou-se execuções com dois tempos limites diferentes, 120 e 1.800 segundos. Uma das conclusões mais importantes foi que, com um tempo limite de execução relativamente baixo, 30 minutos, a utilização de uma solução inicial desacoplada no modelo integrado, em instâncias com maior número de tarefas, possui melhoria média de aproximadamente 11,15% nos resultados se comparado ao integrado e 18,24% se comparado ao modelo desacoplado.

Palavras Chave: Programação e sequenciamento produção e distribuição, Integração produção e distribuição, Otimização integrada, MIP-*and-refine*.

ABSTRACT

The main motivation of this paper is to study a solution refinement technique for scheduling and routing of integrated production and distribution problems. It is possible to affirm that solving these operations in an integrated form is important to guarantee the search for the best solution for the system. The objective of the research is to compare the results obtained from three types of resolution: a mathematical model solved in an uncopled form, an integrated model and an integrated model that uses an initial solution of the uncoupled problem. The main characteristics of the studied problems were: unique machine and one vehicle for distribution. As the objective function, was prioritized the minimization of the total flow time of the orders in the system, normally related by the literature with the total time of fulfillment of an order and with inventory levels. To achieve this goal, mixed integer linear programming was used and two different time limits were set, 120 and 1.800 seconds. One of the most important conclusions was that, with a relatively short time limit of execution, 30 minutes, the use of an uncoupled initial solution in an integrated model, in groups of highest number of jobs, provides an average improvement of approximately 11.15% in results compared to the integrated model and 18.24% of improvement when compared to the uncoupled form.

Keywords: Scheduling and routing, Integrated scheduling and routing, Integrated optimization, MIP-and-refine.

SUMÁRIO

1 INTRODUÇÃO

Segundo vários autores (por exemplo, SAWIK, 2016), é possível identificar duas atividades importantes que fazem parte de uma cadeia de suprimentos: programação e sequenciamento de operações de produção e distribuição.

Os fatores estratégicos das empresas como qualidade, confiabilidade, flexibilidade, rapidez e custos possuem parte de seu desempenho determinado pelo planejamento e controle da produção, sendo que este engloba a programação e sequenciamento das operações de produção e distribuição. Assim, os resultados obtidos com essas atividades são extremamente relevantes pois, por estarem ligados às dimensões estratégicas, influenciam consideravelmente no desempenho macro das empresas (MORAIS; MOCCELLIN, 2010).

A programação e sequenciamento da produção ou *scheduling* surgiu como uma área de pesquisa na década de 1950 e uma das publicações mais antigas é o trabalho de Johnson (1954). Desde então, o tema é explorado por estudiosos da área de pesquisa operacional, matemáticos e pesquisadores da área de planejamento e controle da produção. Nos últimos 67 anos uma significativa quantidade de estudos que tratam dos problemas de programação e sequenciamento de operações de produção foram explorados utilizando de diversas configurações de máquinas, diferentes restrições e variadas funções objetivo. Segundo a definição de Baker (1974), o problema de *scheduling*, em geral, requer decisões de sequenciamento e alocação de recursos e tais recursos podem ser ferramentas, materiais, máquinas, etc.

Uma parcela considerável dos problemas de *scheduling* ou programação e sequenciamento de operações da produção, podem ser considerados muito complexos de se resolver por apresentarem um elevado tempo computacional de resolução (LENSTRA; RINNOOY, 1979). De acordo com as características dos sistemas produtivos, como por exemplo mais de uma máquina por estágio produtivo, as diferentes capacidades das máquinas e diferentes habilidades da mão de obra, estas podem aumentar a complexidade dos problemas. Segundo Slack *et al*. (1999), os problemas de *scheduling* podem envolver inúmeros tipos de tarefas a serem programadas e, em muitos casos, é necessário administrar decisões que englobam a alocação de recursos simultâneos diversos.

Além do problema de produção, há as questões que envolvem a resolução de problemas de distribuição dos produtos para os clientes. Estes pertencem à classe de problemas que definem como será realizada a distribuição dos produtos a armazéns, varejistas e/ou clientes

finais. Nessa tarefa de distribuição, o operador logístico é responsável por continuamente tomar decisões sobre qual volume carregar em cada veículo, quantos veículos utilizar, para qual rota enviá-los, entre outras, buscando, principalmente, realizar a operação de maneira eficaz. Esse tipo de problema se enquadra na classe geral de problemas de roteamento de veículos (SIMCHI-LEVI; CHEN; BRAMEL, 2014) e uma de suas mais antigas publicações sobre o tema foi realizada por Dantzig e Ramser (1959) no qual explorava o roteamento de veículos considerando capacidade (do inglês, *Capacitated Vehicle Routing Problem* - CVRP).

Tais problemas, segundo Desrosiers *et al*. (1995), se tornam cada vez mais importantes e apresentam um crescimento acelerado nas pesquisas do tema devido ao fato de muitas empresas de manufatura, serviços e/ou transporte buscarem, além de reduzir seus custos de transporte, almejarem também a diferenciação nos serviços. Um exemplo de diferenciação nos serviços seria garantir uma entrega rápida e eficiente. A utilização de modelos matemáticos de otimização para solução dos problemas de roteirização de veículos (VRP) em operações de transporte de mercadorias vem sendo estudada há décadas e, devido a sua complexidade e importância, uma ampla gama de sistemas de distribuição têm sido explorados buscando-se novas estratégias resolução e também aprimorar estratégias já existentes. Nos últimos anos, devido aos avanços nas pesquisas da área e popularização de várias tecnologias, permitiu-se que um eficiente gerenciamento de suas frotas de veículos conseguisse promover uma considerável redução de custos, aumento de produtividade e que as empresas obtivessem um resultado positivo ainda maior neste âmbito de decisões. Um exemplo de problema que obteve redução de custos através da roteirização foi do serviço de entrega de um varejista online de grande porte no qual foi explorado por Kim e Bae (2016). O estudo utilizou do algoritmo de Dijkstra (1959) e uma análise comparativa de algoritmos meta-heurísticos para definir o número de veículos e de rotas otimizados. No algoritmo de Dijkstra o objetivo era escolher a sequência que minimizava o custo total da rota percorrida. A análise comparativa de um algoritmo genético versus um algoritmo de busca Tabu demonstrou que as rotas utilizando a busca Tabu foram melhores àquelas produzidas pelo algoritmo genético, obtendo uma média de 12,83% de menor custo. O estudo demonstrou que, devido à otimização das entregas, não só era possível reduzir os custos de distribuição para os operadores e agilizarem a entrega para os consumidores, mas também tinham o potencial de reduzir o valor agregado de emissões de carbono do varejista. Para analisar os resultados, estes foram comparados com dados de rotas existentes anteriormente e as rotas criadas utilizando o algoritmo de Dijkstra. Os resultados

indicaram que os custos de operação diminuíram em 14,6%, o tempo total de serviço em 42,2%, e as emissões de carbono em 13,7%.

Neste cenário, a utilização e aprimoramento de técnicas referente à modelos matemáticos que otimizem cenários dinâmicos de roteirização de veículos são valiosos para as empresas e são explorados através das pesquisas acadêmicas.

As abordagens clássicas dos problemas de produção e distribuição resolvem os mesmos de forma separada. Dessa forma, os subproblemas são resolvidos separadamente com sucessiva mescla das subsoluções encontradas e, assim, formam uma solução para o problema geral. No entanto, esta abordagem pode gerar planos mal coordenados e resultar em altos atrasos na cadeia de suprimentos (ULRICH, 2013).

Outra abordagem encontrada na literatura é a resolução dos dois subproblemas de forma única, ou seja, realizar um modelo integrado de produção e distribuição. O problema de programação integrada de produção e distribuição (IPDP) pode ser considerado um problema comum em muitos setores da indústria como, por exemplo, uma indústria têxtil que produz e distribui tecido para diversas confecções ou uma fábrica de peças de automóveis que, além de produzir, precisa decidir como melhor distribuir seus produtos entre os seus clientes. Uma solução ótima de um IPDP requer a resolução simultânea do *scheduling* da produção e do problema de roteamento de transporte. Os problemas integrados de produção e distribuição visam melhorar, tanto a programação sob à perspectiva da produção, quanto a programação da distribuição (GEISMAR *et al*., 2008). A abordagem IPDP é estudada por diversos autores como Thomas e Griffin (1996), Chang e Lee (2004), Chen (2010), Scholz-Reiter, Makuschewitz, Novaes, Frazzon e Lima (2011), Ullrich (2013), Reimann, Tavares Neto e Bogendorfer (2014), entre outros.

Uma dificuldade desta classe de problemas integrados é o fato de ser necessário coordenar planejamento de produção, controle de estoques e estratégias de transporte ao mesmo tempo em que se busca um resultado otimizado destas atividades (SIMCHI-LEVI *et al*., 2014). Nos trabalhos publicados por Thomas e Griffin (1996) e Scholz-Reiter, Makuschewitz, Novaes, Frazzon e Lima (2011) foram apontados alguns motivos pelos quais as empresas optam por uma abordagem não integrada das operações de produção e distribuição em detrimento de uma abordagem integrada. Um dos motivos apontados foi a complexidade da resolução de um problema integrado, na qual poderia ser considerada superior à resolução do problema não integrado. Devido a esse motivo, algumas empresas optam pela utilização de *buffers* entre as atividades de produção e distribuição como razão para reduzir a necessidade de integrar essas

funções da cadeia de suprimentos. Assim, esta abordagem de resolução separada pode ser considerada adequada se as interdependências entre os dois subproblemas de planejamento e distribuição podem ser evitadas por altos estoques intermediários. No entanto, no que se refere à prática, a necessidade de baixos estoques é mais comum (ULRICH, 2013).

Garey *et al*. (1976) apontaram que determinados problemas de natureza combinatória não são possíveis de se resolver otimamente pois não encontram uma solução em um tempo computacional aceitável. Essa característica desta classe de problemas foi explorada anteriormente nos trabalhos de Cook (1971) e de Karp (1972) onde concluíram que, se não é possível resolver algum destes problemas em tempo polinomial, todos os outros da mesma classe também não serão.

Os problemas de programação e sequenciamento de operações de produção e distribuição, mesmo isolados, podem ser considerados complexos de se resolver devido principalmente à natureza combinatória dos subproblemas, sendo estes classificados como *NP-Hard*. Devido a este fator, a integração destes subproblemas também os classifica como pertencentes à classe *NP-Hard* (EHM; FREITAG, 2016). Assim, existem diversas abordagens de resolução, por exemplo, modelos exatos de programação matemática ou meta-heurísticas para se resolver tais problemas. Os métodos exatos de resolução buscam a otimalidade da solução e possuem um determinado custo e tempo computacional para encontrá-la, podendo ser considerados muito elevados em alguns casos. Já os métodos heurísticos buscam balancear a qualidade da solução com o tempo e os requisitos computacionais necessários para obtê-la.

O avanço das técnicas computacionais para as decisões de *scheduling* e roteirização se apresentam como um fator fundamental para evolução das aplicações desta classe de problemas e buscam suportar melhores decisões no gerenciamento das operações de produção e distribuição das empresas (BODIN, 1990). As empresas que buscam evoluir no sentido de buscar a otimização destas decisões podem garantir importantes vantagens competitivas em relação às suas concorrentes.

Conforme a literatura sobre o tema de programação e sequenciamento de operações de produção e distribuição se expande, é possível concluir que integrar essas funções e resolvê-las de forma única é vital para garantir a busca pela melhor solução para a cadeia de suprimentos (SAWIK, 2016).

Em Ullrich (2013) integrou-se um problema de produção e distribuição de uma indústria de bens perecíveis buscando minimizar o atraso total. As principais características da produção e distribuição eram um ambiente de máquinas paralelas com opções de único ou

múltiplos veículos para distribuição. Os resultados da integração foram significativamente melhores do que os obtidos com as abordagens clássicas de resolução dos problemas, onde eram resolvidos de forma separada. Por exemplo, em uma configuração de um sistema produtivo com duas máquinas paralelas e que possuía dois veículos para a distribuição, a abordagem integrada obteve aproximadamente 3,2% de desvio médio da otimalidade enquanto os desvios das abordagens separadas de produção e distribuição variaram entre 13,8% à 18%.

Nesse contexto, o presente trabalho explora problemas de *scheduling* e roteirização e busca analisar os resultados obtidos com resoluções não integrada e integrada dos mesmos. Embora já seja claro que a solução não integrada pode não fornecer a solução ideal para a cadeia produtiva como um todo, a pesquisa atual busca analisar se essa solução pode ser utilizada para auxiliar a obter a solução ótima ou com um menor desvio da ótima para resolução do problema integrado. Dentro do contexto de uma classe de problemas de produção e distribuição, buscou-se analisar e comparar três tipos diferentes de resolução: desacoplada, integrada e um modelo integrado que utiliza de uma solução inicial do modelo desacoplado. Na abordagem desacoplada do atual trabalho considera-se que a programação e sequenciamento das etapas produção e distribuição são realizadas separadamente e de maneira complementar, ou seja, primeiro é resolvido o subproblema de produção e em seguida é resolvido o subproblema de distribuição. A utilização de uma solução inicial desacoplada no modelo integrado de produção e distribuição pode ser considerada como uma técnica que busca o refinamento da solução (FISCHETTI *et al*., 2013).

O modelo base utilizado na pesquisa foi extraído de um artigo já existente na literatura, no qual considerava a programação e sequenciamento de ambiente produtivo de máquinas paralelas e este foi adaptado para máquina única no subproblema de produção. Já na parte de distribuição, a característica de entrega por um único veículo do artigo base foi mantida. Os dados utilizados nos algoritmos foram gerados aleatoriamente e criou-se uma biblioteca que foram inseridas nos parâmetros do problema. Além disso, os algoritmos foram executados em dois tempos limites diferentes.

1.1 OBJETIVOS GERAIS E ESPECÍFICOS

A atual pesquisa tem como objetivo geral realizar um estudo comparativo entre as soluções obtidas através de problemas de produção-distribuição integrados e não integrados

e determinar se a solução desacoplada, obtida pelos modelos não integrados, pode ser utilizada como ponto de partida para a solução integrada.

As principais características dos três modelos utilizados na pesquisa são: máquina única no subproblema de produção e único veículo com capacidade limitada, sendo possível realizar múltiplas rotas, no subproblema de distribuição. Como critério de desempenho escolheu-se minimizar o tempo total de entrega dos pedidos aos clientes, ou seja, minimizar o tempo total de fluxo do sistema.

Para resolução do problema foi utilizado uma adaptação de um modelo matemático de programação inteira mista já existente na literatura.

Os objetivos específicos da pesquisa podem ser definidos como os seguintes:

- Implementar um modelo desacoplado de produção e distribuição, com base em um modelo já existente na literatura, para um sistema produtivo de máquina única com entrega por um único veículo;
- Implementar um modelo integrado desse problema produção-distribuição;
- Utilizar a solução desacoplada para iniciar o modelo integrado e verificar se há ganhos nos resultados do *solver*.
- Comparar os resultados obtidos com essas três abordagens;

1. 2 JUSTIFICATIVA E LACUNA DE PESQUISA

Em certos problemas de *scheduling* e roteirização, a integração dos subproblemas podem proporcionar uma melhoria média entre 5% e 20% nas soluções obtidas, em comparação com os resultados obtidos através de uma abordagem não integrada dos mesmos (MOONS *et al*., 2017). No entanto, a abordagem integrada desses subproblemas, utilizando de métodos exatos de resolução e sem nenhuma estratégia adicional, podem se mostrar infactíveis devido à complexidade matemática dos mesmos. Assim, se percebe a necessidade de analisar técnicas alternativas de resolução de problemas integrados. Uma dessas técnicas é fornecer ao *solver* uma solução inicial viável de um modelo de programação matemática, na esperança de que o software consiga melhorar os resultados obtidos e, idealmente, encontrar uma solução ótima para o problema.

Além disso, na literatura de *scheduling* e roteirização, especificamente sobre métodos de refinamento das soluções de modelos de programação linear inteira mista, ainda não é possível encontrar um grande número de publicações no tema. Ao buscar as palavras-chave "métodos de refinamento" juntamente com "*scheduling* e roteirização" em uma importante base científica, somente três publicações foram encontradas no que se refere à utilização de uma solução inicial para resolução do problema. Ainda assim, nenhuma delas utilizava de uma solução inicial do modelo desacoplado para resolução do problema integrado. A primeira publicação refere-se à Fischetti *et al*. (2013) e utilizou de um algoritmo *Greedy* que gerava uma lista de soluções iniciais utilizada na execução do MIP e a técnica do artigo foi chamada de MIP-*and*-*refin*e, ou então, MIP-*refining procedure*. A segunda publicação encontrada foi de Ferris e Liu (2016) onde um modelo de otimização não linear foi utilizado para gerar uma lista de soluções locais e estas foram inseridas como solução inicial do MIP considerado na pesquisa. Por último, em Gelareh *et al*. (2020), utilizou-se uma extensão do problema do Caixeiro Viajante e utilizou de uma heurística de *Variable Neighborhood Search* (VNS) como solução inicial do modelo MIP.

1.3 ESTRUTURA DO TRABALHO

A estrutura do trabalho possui os seguintes capítulos: o capítulo 2 apresenta a metodologia de pesquisa utilizada e em quais classificações da literatura ela se encontra. No capítulo 3 foi realizada a revisão de literatura e explorado o estado da arte do tema de pesquisa estudado. O capítulo 4 contemplou o desenvolvimento do trabalho e as aplicações realizadas. No capítulo 5 foram apresentados os resultados obtidos com a resolução dos modelos. No capítulo 6 foram realizadas as considerações finais e as possibilidades de pesquisas futuras.

2 METODOLOGIA

Um dos principais objetivos da pesquisa científica é a busca pela veracidade dos fatos e, o que a difere dos demais, é a sua característica de verificabilidade. Para isso, procura determinar operações mentais e técnicas que possibilitam essa verificação (GIL, 2008).

Com isso, é possível classificar a pesquisa científica de acordo com os procedimentos técnicos utilizados. Os procedimentos técnicos podem ser divididos em métodos qualitativos ou quantitativos.

Segundo Gil (2008), no método qualitativo não há fórmulas exatas ou receitas predefinidas para orientar os pesquisadores no desenvolvimento da pesquisa. Já na abordagem quantitativa, de acordo com Bryman (1989), as principais características do método são: mensurabilidade, causalidade, generalização e replicação. A primeira característica, mensurabilidade, com a finalidade de testar uma ou mais hipóteses, deduzidas da teoria, define que um conjunto de variáveis de um problema precisa ser desenhado e estas devem ser mensuráveis. Esse processo pode ser denominado como operacionalização do problema. A partir disso, as variáveis podem ser medidas de forma a gerar dados e permitir obter conclusões sobre as hipóteses definidas inicialmente. Já a característica de causalidade procura explicar como as variáveis se comportam no sentido de provar a existência de relacionamento causal entre as variáveis da pesquisa. Ou seja, em muitas pesquisas, as hipóteses podem obter um relacionamento de causa e efeito entre a variável dependente (efeito) e as variáveis independentes (causas). A terceira principal característica, a generalização, não é uma preocupação exclusiva da abordagem quantitativa e trata da possibilidade de os resultados obtidos serem generalizados para além dos limites da pesquisa. Por fim, a possibilidade de replicação é um importante fator da pesquisa quantitativa porque garante que, ao replicar o desenvolvimento de uma pesquisa, o resultado será o mesmo, validando seus resultados e conclusões.

Ao se conduzir uma pesquisa baseada em modelos quantitativos, o seu conteúdo essencial deve contemplar o modelo teórico conceitual, o modelo científico do problema, a resolução, a prova da solução e a análise dos resultados obtidos (BERTRAND; FRANSOO, 2002).

Ao verificar as definições de pesquisa quantitativa e qualitativa, as características da atual pesquisa permitem que ela se enquadre na classe de pesquisa quantitativa. Assim, a programação de operações de produção e distribuição são exploradas

através de modelos quantitativos e estes buscam explicar ou representar parte das particularidades e/ou auxiliar nas tomadas de decisões de um problema real. (BERTRAND; FRANSOO, 2002).

Outra classificação utilizada em pesquisa científica é através da natureza de seus resultados (GANGA, 2012). Essa classificação possui as seguintes divisões: pesquisa básica ou pesquisa aplicada. A pesquisa básica busca gerar novos conhecimentos através da exploração de um determinado tema, sem ter, como premissa, uma aplicação prática. A pesquisa aplicada busca gerar novos conhecimentos tendo como objetivo aplicação prática e buscando solucionar um problema específico. Assim, segundo esta classificação, os as características da atual pesquisa se enquadram como uma pesquisa aplicada.

Além disso, é possível classificar a pesquisa quanto aos procedimentos técnicos utilizados. Os instrumentos de coleta de dados geralmente utilizados em pesquisas classificadas como quantitativas são os questionários (instrumentos da *survey*), dados aleatórios, modelagem e simulação e, por fim, dados empíricos, utilizados nos experimentos matemáticos (GANGA, 2012). Assim, a atual pesquisa segue a classificação de modelagem e simulação.

As pesquisas realizadas no âmbito do gerenciamento de operações de produção e distribuição, pertencem à área de pesquisa operacional. Na pesquisa operacional busca-se desenvolver e resolver problemas que envolvam modelos matemáticos e estatísticos com o objetivo de encontrar uma solução ótima ou próxima à ótima para determinado problema (BERTRAND; FRANSOO, 2002).

Segundo Bertrand e Fransoo (2002), focando-se na pesquisa quantitativa, é possível identificá-la e dividi-la em duas classes principais: empírica e axiomática. Na primeira classe sua principal motivação é obter medidas e descobertas que conduzem a um acréscimo, mudança ou alteração radical sobre o conhecimento de determinado assunto. Assim, nesta classe busca-se assegurar que exista coerência entre as observações, dados do problema, ações na realidade e o modelo desenhado para tal realidade. Já a pesquisa axiomática busca obter soluções para um modelo em questão e assegurar que tais soluções ajudem a esclarecer a estrutura do problema descrito no modelo. Portanto, diferentemente da pesquisa empírica, que se baseia em criar modelos que se adequem às relações causais de um problema real, a pesquisa axiomática explora o uso intensivo de métodos matemáticos buscando estudar um problema idealizado e formular teorias a partir dele.

A pesquisa axiomática pode ser dividida em normativa ou descritiva. A normativa desenvolve ações com a finalidade de obter resultados aprimorados e ou

diferenciados aos encontrados na literatura. Por exemplo, temos um conjunto considerável de pesquisas que se utilizam de variados modelos de programação matemática para a resolução problemas de dimensionamento de lotes ou roteamento de veículos buscando obter diferentes resultados aos já explorados na literatura. Já a pesquisa axiomática descritiva busca primordialmente analisar modelos quantitativos com o objetivo de entendê-lo e definir suas características (CAUCHICK *et al*., 2012). Um exemplo desta última categoria é uma pesquisa que analisa determinado modelo de programação matemática para diferentes situações e verifica quais os cenários em que ele se apresenta mais efetivo. Tendo isso, como o trabalho atual busca obter resultados aprimorados dos encontrados na literatura, este pode ser classificado como uma pesquisa axiomática normativa.

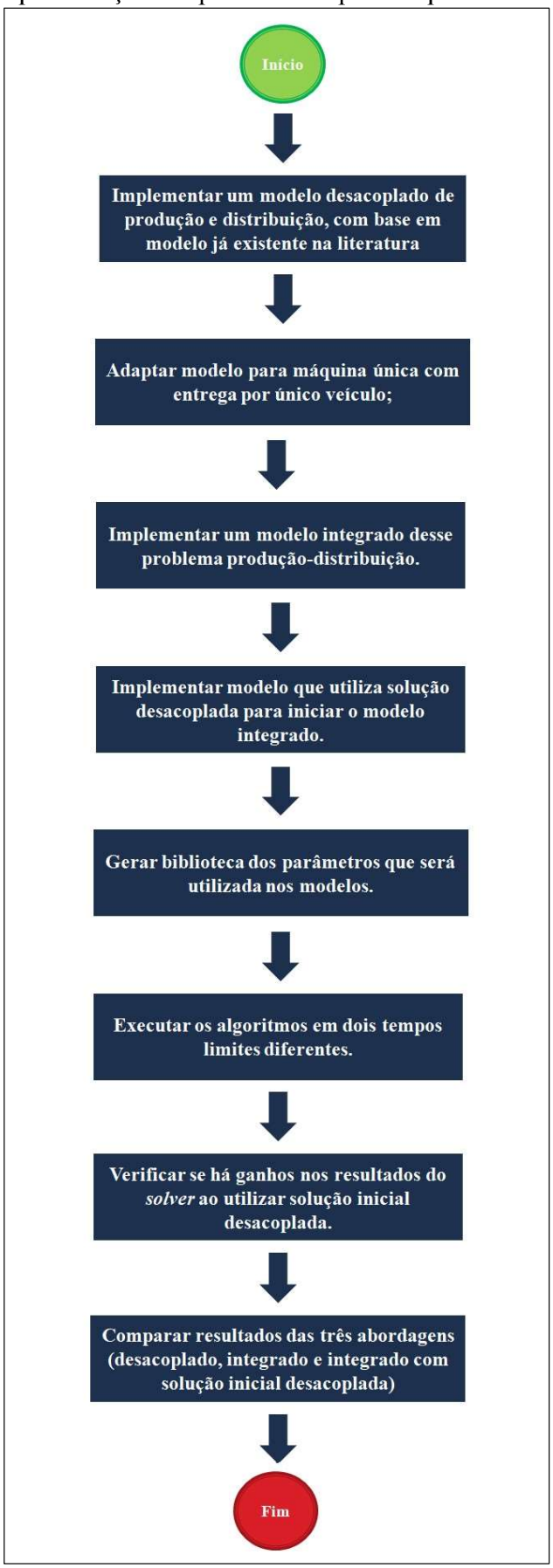
A metodologia de pesquisa inclui também as técnicas de coleta e análise de dados.). A análise dos dados pode ser dividida em análise quantitativa, por exemplo análise estatística, ou qualitativa, por exemplo análise de discurso. (GERHARDT; SILVEIRA, 2009).

Gil (2008) afirma que, nas etapas de análise de dados, boa parte das pesquisas possuem a etapa de análise estatística dos dados. Gerhardt e Silveira (2009) exemplificam sobre possíveis testes estatísiticos e dois deles referem-se à análises de médias e porcentagens de desvios médios. Ambos estão presentes na análise de dados da atual pesquisa e um exemplo de cada uma delas são: médias de tempo de execução dos algoritmos e análise do desempenho delas em relação ao método utilizado e número de tarefas consideradas e outra refere-se à análise das porcentagens de desvio da função objetivo.

Além disso, a metodologia indica como foi conduzida uma pesquisa, especificando suas etapas e os procedimentos que foram adotados (GERHARDT; SILVEIRA, 2009

As principais etapas da pesquisa atual podem ser definidas como as seguintes: pesquisar na literatura de programação e sequenciamento de produção e distribuição integrada um modelo matemático de programação inteira linear para resolução do problema da pesquisa, adaptação e implementação de um modelo matemático não integrado com base no modelo encontrado na literatura, em seguida implementar um modelo integrado do anterior e por último implementar um método que insere como ponto de partida uma solução inicial desacoplada para iniciar a resolução do modelo integrado. O passo a passo, de forma mais detalhada e específica, está descrito na Figura 2.1

Figura 2.1 Representação simplificada do passo a passo da atual pesquisa.



Fonte: Elaborado pelo autor.

Sobre a biblioteca de dados utilizada para os modelos, primeiramente foram definidos intervalos de valores para todos os parâmetros, em seguida os dados foram gerados aleatoriamente e, por fim, estes dados formaram um arquivo de biblioteca de parâmetros. Após isso, a biblioteca de dados foi salva como um arquivo base de entrada para a resolução de todos os modelos. Um exemplo de parâmetro da atual pesquisa é o número de tarefas a serem produzidas e entregues pelo problema de produção e distribuição considerado.

Os modelos foram executados em dois tempos limites. A primeira execução foi de 120 segundos e a segunda foi de 1.800 segundos. A justificativa para escolha destes dois tempos se deve ao fato de: no menor tempo, o objetivo era identificar se, mesmo em um tempo muito baixo, é possível encontrar soluções. A escolha do tempo de 1.800 segundos pode ser explicada por ser considerado um tempo razoável quando se considera em utilizar uma execução de um problema de produção-distribuição para tomada de decisão gerencial. Na realidade do dia-a-dia de uma indústria, por exemplo, a rapidez na obtenção de dados é essencial para tomada de decisões e escolha do melhor caminho possível a se seguir.

3 REVISÃO DA LITERATURA

Segundo Dutra e Erdmann (2006) o Planejamento e Controle da Produção (PCP) é um sistema de apoio à produção que busca planejar e controlar as operações produtivas de maneira eficaz para satisfazer os requisitos de volume, tempo e qualidade necessários para atender seus clientes.

O escopo do Planejamento e Controle da Produção pode ser dividido de acordo com os seus horizontes de planejamento, sendo estes: curto prazo, médio prazo e longo prazo. No horizonte de longo prazo, o PCP deve elaborar um Plano Mestre de Produção (PMP) com base nas previsões de vendas. No médio prazo, tendo estruturado o PMP, atua-se no nível tático e planeja o uso da capacidade de forma a atender às previsões de vendas e pedidos em carteira já confirmados. Por fim, no curto prazo, chamado de nível operacional, é realizado a programação e sequenciamento de operações de produção ou *scheduling* (MORAIS; MOCCELLIN, 2010).

No nível operacional, além de programar e sequenciar as ordens de produção nas máquinas, é possível realizar a programação e sequenciamento dessas ordens de produção no âmbito da distribuição para os clientes, decisões estas relacionadas à roteirização de veículos (SIMCHI-LEVI; CHEN; BRAMEL, 2014).

As decisões relacionadas à programação e sequenciamento das ordens de distribuição de produtos aos clientes podem ser consideradas como parte da área de Logística (BALLOU, 2014).

As principais decisões relacionadas à roteirização de veículos são: sobre quais clientes serão atendidos por cada veículo, para onde enviá-lo e qual rota utilizar se enquadram na classe geral de problemas relacionados ao roteamento de veículos (VRP – *Vehicle Routing Problem*) (SIMCHI-LEVI; CHEN; BRAMEL, 2014).

As operações de uma empresa podem ser visualizadas desde o início, com a compra dos materiais, até a entrega de bens e serviços aos clientes. Essa pode ser considerada como a visão de cadeia de suprimentos e inclui tanto operações de produção quanto de distribuição (BOWERSOX *et al*., 2014).

Considerando estes dois tipos de problemas, de programação de operações de produção e de distribuição, a abordagem clássica utiliza de resolução dos mesmos de forma desacoplada, ou seja, resolve-se separadamente o problema de produção e distribuição, mas sabe-se que é possível realizar a união dos mesmos e buscar resolvê-los de forma integrada (EHM; FREITAG, 2016).

Como o objetivo do trabalho é realizar um comparativo entre as resoluções de um modelo separado de produção e distribuição, um modelo integrado e um modelo integrado com solução inicial de um problema desacoplado, na revisão da literatura foram pesquisadas as publicações existentes sobre o tema e as palavras-chave utilizadas foram: *Integrated production and distribution* e MIP-*and-refine,* sendo utilizadas as citações exatas de ambos os grupos de palavras. Alguns tópicos de assuntos não relacionados ao tema foram excluídos do filtro de pesquisa e, aproximadamente, 100 publicações foram encontradas. Após essa seleção, algumas características similares com o problema atual como, máquina única ou paralela, somente um veículo para distribuição ou a função objetivo de minimização do tempo de fluxo total, foram consideradas para a escolha das citações utilizadas no trabalho. Além disso, o conteúdo da busca inicial foi base para a utilização de outras referências clássicas e também publicações mais recentes da literatura.

3.1 *SCHEDULING*

Em Baker (1974) o sequenciamento e programação da produção (*scheduling*) é definido como a ordenação e alocação de recursos para processar um conjunto de tarefas.

O problema de *scheduling* é onde busca-se definir a ordem (prioridade) de uma tarefa sobre um conjunto de tarefas que estão sendo processadas em um ou mais máquinas (NAWAZ *et al*., 1983). Esses problemas podem ser encontrados em diversas áreas e tipos de atividades e, além disso, podem envolver desde a alocação e sequenciamento de tarefas em um sistema com poucos recursos até um sistema mais complexo (GRAHAM, 1978).

O *scheduling* pode ser tratado como um fator importante do PCP, principalmente porque uma programação e sequenciamento de operações aplicado de maneira satisfatória pode trazer reais benefícios às empresas como, por exemplo, redução de custos. É possível obter redução de custos, por exemplo, em uma situação onde em determinado sistema produtivo o *scheduling* bem aplicado promoveu uma melhor porcentagem de utilização de um equipamento (GUPTA *et al*., 2011).

Pinedo (2008) considera que as tarefas, os recursos e os objetivos das empresas podem assumir diversas formas. Nos problemas de *scheduling* as tarefas podem ser operações em um processo produtivo, etapas de um projeto de construção, pontos de distribuição de energia, etc. Os recursos podem ser máquinas, mão de obra, unidades de processamento (computação), entre outros.

No processo de *scheduling*, para determinarmos quando as tarefas podem ser executadas, é preciso saber o tipo e a quantidade de cada recurso. Quando os recursos são determinados, pode se dizer que definimos as fronteiras do problema. Usualmente, cada tarefa é descrita em termos de informações como sua duração, o momento em que ela pode ser iniciada e o momento em que deve ser concluída. É preciso definir também as restrições que existem entre as tarefas. Além disso, encontrar uma solução ótima ou boa para esta classe de problemas pode ser uma questão complexa e utilizar-se de abordagens formais de resolução são fundamentais (BAKER; TRIETSCH, 2009).

Além das abordagens de solução escolhidas, algumas características dos problemas de *scheduling* como, por exemplo, o número de tarefas, número de estágios produtivos, número de máquinas paralelas por estágio, entre outras, podem ser consideradas fatores determinantes para os resultados (BRAH; LOO, 1999).

O *scheduling* envolve a escolha da tarefa a ser processada na sequência dentro de uma lista de tarefas ainda não alocadas, ou seja, envolve sequenciamento das ordens de produção. O sequenciamento, segundo Tubino (2007), pode ser realizado de acordo com diferentes ordens de prioridade. As principais formas de ordenação das tarefas de produção segundo o autor são:

- PEPS, primeiro que entra, primeiro que sai - (do inglês*, First In, First Out - FIFO)*: a prioridade é dada à primeira tarefa a ser inserida, devendo ser a primeira a ser produzida. Essa regra busca minimizar o tempo de permanência de cada tarefa no sistema;
- MTP, menor tempo de processamento (do inglês*, Shortest Processing Time - SPT)*: a ordenação escolhida define o menor tempo total de processamento. É classificado como ordem de tempo crescente e busca aumentar o fluxo de materiais;
- MDE, menor data de entrega (do inglês*, Earliest Due Date* - EDD*)*: as tarefas são processadas de acordo com a data mais próxima de entrega;
- IPI, índice de prioridade: as tarefas serão processadas de acordo com um índice de prioridade definido;
- ICR, índice crítico (do inglês*, Critical Ratio* - CR*)*: a prioridade é dada ao maior índice obtido pelo intervalo de tempo até a data de expiração dividido pelo tempo total restante da produção;

- IFO, índice de folga (do inglês, *Least Slack* - LS*)*: a prioridade é dada à menor folga entre a data de vencimento e o total tempo de processamento, buscando diminuir os atrasos;
- IFA, índice de falta: a ordem é definida de acordo com o menor valor obtido através da divisão da quantidade em estoque pela taxa de demanda.

A programação e sequenciamento da produção deve respeitar restrições de recursos humanos, financeiros, materiais e tecnológicos. As restrições tecnológicas são condicionadas principalmente pelo tipo de fluxo das operações (das tarefas) nas máquinas (NAGANO; MOCCELLIN, 2003).

Segundo Morais e Moccellin (2010), as classificações de tipos de fluxo de tarefas nas máquinas existentes podem ser divididas em: máquina única, máquinas paralelas, *flow shop*, *flow shop* com múltiplas máquinas, *job shop, job shop* com múltiplas máquinas e o*pen shop*.

Maccarthy e Liu (1993) descrevem os tipos de fluxo de tarefas nas máquinas como:

- máquina única: há apenas um estágio produtivo;
- máquinas paralelas: existe um único estágio com mais de uma máquina habilitada para realizar as tarefas;
- *flow shop*: existem vários estágios produtivos onde a sequência de operações é a mesma;
- *flow shop* com múltiplas máquinas: fluxo no qual cada estágio de produção possui um conjunto de máquinas paralelas.
- *job shop*: nesse fluxo, cada tarefa tem sua própria ordem de processamento dentre os estágios produtivos considerados;
- *job shop* com múltiplas máquinas: em cada estágio de produção existe um conjunto de máquinas paralelas;
- *open shop*: não há um fluxo definido para as tarefas a serem processadas nas máquinas;

A classe de problemas de *scheduling* possui a característica de buscar a otimização de seus resultados com base em um conjunto de indicadores de desempenho. Estes indicadores podem ter objetivos distintos, como por exemplo, a minimização do tempo total de

fluxo, a minimização do número de tarefas atrasadas, a minimização do *makespan (*o instante de término de processamento da última tarefa alocada), etc.

O tempo total de fluxo, quando todas as tarefas possuem o mesmo tempo de liberação, é a soma dos instantes de término de processamento de todas as tarefas, este é um indicador que mede o estoque em processamento (MORAIS *et al*., 2009).

Além disso, os problemas podem incluir um conjunto de restrições, por exemplo, *no-wait*, *no-idle*, priorização de clientes, entre outras.

Os modelos matemáticos podem ser considerados como uma abordagem para caracterizar e resolver problemas de otimização. Devido aos avanços obtidos na tecnologia e a evolução nas técnicas de resolução dos modelos matemáticos na obtenção de soluções rápidas, a utilização dessas técnicas como estratégia de solução se tornam cada vez mais viáveis (NADERI *et al*., 2009).

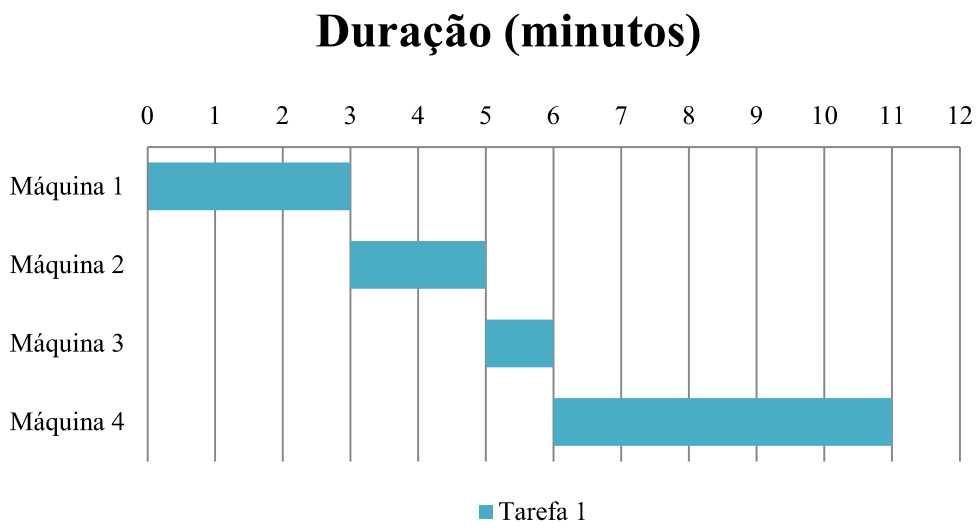
Um dos exemplos mais simples de modelo formal de representação de um problema de *scheduling* é o gráfico de Gantt. Em resumo, as atividades são alocadas ao longo do tempo, representado pelo eixo horizontal, e distribuídas nos recursos disponíveis, representados pelo eixo vertical (BAKER, 1974).

No grafico 3.1 é apresentado um exemplo simplista de utilização de representação de Gantt na alocação de 1 tarefa que deve ser processada em 4 máquinas em um ambiente *flow shop*. Os tempos de processamento deste exemplo estão descritos na tabela 3.1.

Tabela 3.1 Início e duração de uma tarefa (minutos).

Máquinas	**Tempo de processamento (minutos)**
Máquina 1	3
Máquina 2	2
Máquina 3	1
Máquina 4	5

Gráfico 3.1 Exemplo da representação de Gantt de 1 tarefa em 4 máquinas.



O exemplo da representação de Gantt no gráfico 3.1 demonstra o início da tarefa 1, na máquina 1, em um tempo zero e finaliza sua produção nesta máquina em 3 minutos. Na sequência ela inicia o processo na segunda máquina e, assim por diante, até a finalizar na quarta máquina. O *makespan* desse exemplo é de 11 minutos, ou seja, este é definido quando a última tarefa, neste caso única, finaliza sua produção na máquina 4.

A tentativa de aproximar os problemas de *scheduling* à realidade é um fator relevante no que se diz respeito à possibilidade de aumento da complexidade do problema. Em Naderi *et al*. (2009) os autores afirmaram que os fatores de tempos de *setup* e tempos de transporte entre as operações podem ser considerados como os dois mais realistas ao se desenvolver problemas de *scheduling* por serem os mais usuais.

3.1.1 Algoritmos clássicos de *scheduling*

Um algoritmo clássico e muito citado em diversos artigos na literatura de *scheduling* é o NEH de Nawaz *et al*. (1983). Este trata de um problema de sequenciamento *m* máquinas e *n* tarefas de *flow shop* e é uma heurística em que as tarefas com maior tempo de processamento total são priorizadas sobre tarefas com menor tempo, ou então, regra LTP (*Longest Processing Time)*. O objetivo final deste algoritmo é a minimização do *makespan*.

O algoritmo NEH de Nawaz *et al*. (1983) considera um tempo total de processamento decrescente para a regra de alocação de ordens de produção ou tarefas. Em cada estágio *i* são as definidas as posições relativas das tarefas que ainda não foram alocadas, assim, o *scheduling* é obtido tarefa após tarefa.

Os parâmetros e variáveis do algoritmo NEH estão elencados nas tabelas 3.1.1 e 3.1.2 e este foi desenvolvido da seguinte forma:

Tabela 3.1.1 Simbologia de parâmetros do algoritmo NEH de Nawaz *et al*. (1983).

Variável	Descrição
n	Número de tarefas
m	Número de máquinas
i	Índice para tarefas {1, 2, …, n}
j	Índice para máquinas {1, 2, …, m}
$t_{i,j}$	O tempo de processamento da tarefa i na máquina j

Tabela 3.1.2 Simbologia de variáveis do algoritmo NEH de Nawaz *et al*. (1983).

Variável	Descrição
T_i	Tempo total de processamento da tarefa i
C_{max}	Momento em que a última tarefa é concluída

O passo a passo da heurística NEH são os seguintes:

a) **passo 1:** Para cada tarefa aplicar a equação (3.1), onde, $t_{i,j}$ é o tempo de processamento da tarefa i na máquina j;

$$T_i = \sum_{j=1}^{m} t_{i,j} \qquad \forall\, 1 \leq i \leq n \qquad (3.1)$$

b) **passo 2:** Ordenar todas as tarefas obtidas no passo anterior segundo a regra de prioridade LPT;

c) **passo 3:** Escolher as duas tarefas da primeira e segunda posição obtidas no passo 2 e definir as sequências possíveis para essas duas tarefas. Feito isso, selecionar a sequência que minimize o *makepan* e definir i = 3. As sequências relativas definidas até o momento são fixas e não se alteram nas etapas subsequentes do algoritmo;

d) **passo 4:** Escolher a tarefa da i-ésima posição da lista gerada no passo 2. Inserir essa tarefa em todas as posições possíveis da sequência obtida até o momento, mantendo a sequência relativa entre as tarefas já sequenciadas. Mantenha a tarefa i na posição que minimiza o *makespan* da sequência parcial definida.

e) **passo 5:** Pare se $n = i$, do contrário, defina que $i = i + 1$ e retorne ao passo 4;

A função objetivo buscou minimizar C_{max}, ou seja, minimização do tempo de conclusão de todas as tarefas (*makespan*).

A validação do algoritmo de Nawaz *et al*. (1983) foi realizada através de um conjunto de instâncias geradas aleatoriamente, sendo que os tempos de processamento foram uniformemente distribuídos sobre um intervalo de 1 a 99. Foram resolvidas diversas instâncias de ambiente *flow shop* e os resultados obtidos foram comparados ao algoritmo CDS de Campbell *et al*. (1970). Nawaz *et al*. (1983) identificaram que o algoritmo obteve melhores resultados que o anterior considerando instâncias menores e, apesar de o algoritmo proposto não ter sido testado para instâncias muito maiores (m, n maior ou igual 100), provavelmente manteria uma boa performance nos resultados. Com uma exceção, onde o algoritmo CDS provavelmente superaria o de Nawaz *et al*. (1983), seria o caso em que o número de máquinas fosse muito maior que o número de tarefas, já que a eficácia do primeiro dependia do número de máquinas e o segundo dependia do número de tarefas.

Outro algoritmo clássico da literatura foi apresentado por Osman e Potts (1989) e considerou um *flow shop* permutacional com aplicação de *Simulated Annealing* (SA). No problema abordado, cada uma das n tarefas deveriam ser processadas nas m máquinas e os tempos de processamento p_{ij} da tarefa i na máquina j eram fornecidos. Não foram consideradas regras de prioridade entre as tarefas. A função objetivo buscava minimizar o *makespan*. *Simulated Annealing* é uma meta-heurística que faz uma analogia ao processo de recozimento da termodinâmica. Neste método, quanto maior for a temperatura T, maior é a componente aleatória incluída na próxima solução escolhida, ou seja, à medida que o algoritmo evolui, o valor de T é decrescido, e a solução começar a convergir para uma ótima local. *Simulated Annealing* é uma meta-heurística que tenta superar a desvantagem inerente aos métodos de decomposição e busca encontrar um mínimo local ao invés de um mínimo global.

3.1.2 Vertentes clássicas em problemas de *scheduling* utilizando modelos de programação inteira mista

Ao buscar a literatura clássica de *scheduling,* é possível dizer que há duas vertentes nas formulações de *scheduling* no que se refere a à modelos MILP's e possíveis

abordagens para a alocação das tarefas, sendo que estas são as formulações de Wagner (1959) e Manne (1960). A abordagem de Wagner (1959) desenvolveu sua formulação MILP voltada, principalmente, para o problema *job shop* e se baseia em designar tarefas à posições na sequência de processamento do sistema produtivo. Enquanto Manne (1960) desenvolveu um modelo MILP como um conjunto de restrições que buscam garantir e controlar a ordem de precedência entre as tarefas.

Inicialmente temos o modelo de Wagner (1959) voltado para ambiente produtivo *job shop*, considerando k máquinas e que permite tempo ocioso entre posições consecutivas. Ele utiliza de um problema clássico de alocação de tarefas à posições na sequência de produção. Seus índices, parâmetros e variáveis descritos nas tabelas 3.1.3 e 3.1.4.

Tabela 3.1.3 Simbologia de índices e parâmetros Wagner (1959).

Parâmetros	Descrição
n	Número de tarefas
m	Número de máquinas
i	Índice para tarefas {1, 2, …, n}
t, j	Índice para posição da tarefa {1, 2, …, n}
k	Índice para máquinas {1, 2, …, m}
p_{ik}	Representa o tempo de processamento da tarefa i máquina k
M	Um número relativamente grande

Tabela 3.1.4 Simbologia de variáveis de Wagner (1959).

Variáveis	Descrição
x_{ijk}	Binária. Representa 1 se i assume a posição j na máquina k , do contrário assume 0
h_{jk}	Representa o instante de início da tarefa j na máquina k
I_{1k}	Tempo de espera da máquina k entre o início da produção e o início da primeira tarefa
I_{jk}	Tempo de espera da máquina k entre o instante em que a tarefa na posição (j-1) é concluída e o instante de início da tarefa na posição j
C_{max}	Momento em que a última tarefa é concluída (*makespan*)

Na modelagem de Wagner (1959) buscou-se a minimização do *makespan*:

Minimizar C_{max} (3.2)

Sujeito às seguintes restrições:

$$\sum_{j=1}^{n} x_{ijk} = 1 \qquad \forall\ i \in \{1,2,\dots,n\},\ k \in \{1,2,\dots,m\} \tag{3.3}$$

$$\sum_{i=1}^{n} x_{ijk} = 1 \qquad \forall\ j \in \{1,2,\dots,n\},\ k \in \{1,2,\dots,m\} \tag{3.4}$$

$$h_{1k} = I_{1k} \qquad \forall\ k \in \{1,2,\dots,m\} \tag{3.5}$$

$$h_{jk} = \sum_{t=1}^{j} I_{tk} + \sum_{t=1}^{j-1} \sum_{i}^{n} p_{ik} x_{itk} \qquad \forall\ j \in \{2,3\dots,n\},\ k \in \{1,2\dots,m\} \tag{3.6}$$

$$\sum_{k=1}^{m} r_{ilk}(h_{jk}+p_{ik}) \leq \sum_{k=1}^{m} r_{i,(l+1),k} h_{wk} + M\left(1 - \sum_{k=1}^{m} r_{ilk} h_{ijk}\right) + M\left(1 - \sum_{k=1}^{m} r_{i,(l+1),k} x_{iwk}\right) \qquad \forall\ i,j,w \in \{1,2,\dots,n\},\ l \in \{1,2,\dots,m-1\} \tag{3.7}$$

$$h_{nk} + \sum_{i=1}^{n} p_{ik} x_{ink} \leq C_{max} \qquad \forall\ k \in \{1,2\dots,m\} \tag{3.8}$$

$$h_{jk}, I_{jk} \geq 0 \qquad \forall j \in \{1,2 \ldots, n\},\ k \in \{1,2 \ldots, m\} \tag{3.9}$$

$$x_{ijk} = 0 \text{ ou } 1 \qquad \forall i, j \in \{1,2 \ldots, n\},\ k \in \{1,2 \ldots, m\} \tag{3.10}$$

As equações (3.3) e (3.4) definem que, respectivamente, cada tarefa somente pode ser alocada uma vez em cada máquina e cada posição do sistema só pode conter uma tarefa alocada.

A equação (3.5) define que o início da tarefa na primeira posição alocada seja igual ao tempo de espera da máquina *k* entre o início da produção e o início da primeira tarefa. Na equação (3.6) temos que o instante de inicío de uma posição em uma máquina *k* seja igual à soma dos tempos de processamento das tarefas já alocadas e dos tempos ociosos. A equação (3.7) garante que uma tarefa *i* só inicie após o processamento da operação l na mesma máquina já estiver concluída. Por fim, na equação (3.8), obtemos o momento em que a última tarefa é concluída, que constitui o *makespan*.

Já o modelo de Manne (1960), desenvolvido para formulação *job shop*, considerando *k* máquinas e permitindo que não há, necessariamente, operações em todas as máquinas, utiliza de parâmetros *i*, *j* para definir a sequência de execução de tarefas. Seus índices, parâmetros e variáveis descritos nas tabelas 3.1.5 e 3.1.6.

Tabela 3.1.5 Simbologia de índices e parâmetros Manne (1960).

Parâmetros	Descrição
n	Número de tarefas
m	Número de máquinas
i,j	Índice para tarefas {1, 2, …, n}
k	Índice para máquinas {1, 2, …, m}
p_{ik}	Representa tempo de processamento da tarefa *i* na máquina *k*
r_{ilk}	Define se a da tarefa *i* possui a operação *l* na máquina *k*
M	Um número relativamente grande

Tabela 3.1.6 Simbologia de variáveis de Manne (1960).

Variáveis	Descrição
x_{ijk}	Binária. Representa 1 se i precede j em k
s_{ik}	Representa o instante de início da tarefa i na máquina k
C_{max}	Momento em que a última tarefa é concluída (*makespan*)

Na modelagem de Manne (1960) também se buscou a minimização *makespan,* C_{max} , na função objetivo.

$$\text{Minimizar } C_{max} \tag{3.11}$$

Sujeito às seguintes restrições:

$$\sum_{k=1}^{m} r_{ilk}\,(s_{ik} + p_{ik}) \leq \sum_{k=1}^{m} r_{i,l+1,k}\, s_{ik} \qquad \begin{matrix} \forall\ i \in \{1,2,\dots,n\} \\ l \in \{1,2,\dots,m-1\} \end{matrix} \tag{3.12}$$

$$(M + p_{jk})x_{ijk} + (s_{ik} - s_{jk}) \geq p_{jk} \qquad \begin{matrix} \forall\ 1 \leq i < j \leq n \\ k \in \{1,2,\dots,m\} \end{matrix} \tag{3.13}$$

$$(M + p_{ik})(1 - x_{ijk}) + (s_{jk} - s_{ik}) \geq p_{ik} \qquad \begin{matrix} \forall\ 1 \leq i < j \leq n \\ k \in \{1,2,\dots,m\} \end{matrix} \tag{3.14}$$

$$\sum_{k=1}^{m} r_{imk}(s_{ik} + p_{ik}) \leq C_{max} \qquad \forall\ i \in \{1,2\dots,n\} \tag{3.15}$$

$$s_{ik} \geq 0 \qquad \begin{matrix} \forall\ i \in \{1,2\dots,n\} \\ k \in \{1,2\dots,m\} \end{matrix} \tag{3.16}$$

$$x_{ijk} = 0 \text{ ou } 1 \qquad \begin{matrix} \forall\ i,j \in \{1,2\dots,n\} \\ k \in \{1,2\dots,m\} \end{matrix} \tag{3.17}$$

A equação (3.12) define a precedência da tarefa *i* em uma máquina *k* entre suas operações *l* e *l* + 1, garantindo que o instante de início da operação *l* + 1 seja maior ou igual ao instante de início da operação anterior mais o seu respectivo tempo de processamento.

As equações (3.13) e (3.14) definem as relações de precedência de todas as tarefas em uma máquina *k*. Por fim, na equação (3.15) é definido o *makespan* onde são somados os instantes de término da última operação de todas as tarefas.

Apesar disso de a abordagem escolhida para o MILP do atual trabalho utilizar a abordagem próxima à de Wagner (1959) para a alocação das tarefas, mais encontrada em trabalhos de sistemas *job shop*, esta também pode ser desenhada para ambientes *flow shop* e, em alguns possíveis casos de adaptação, também para sistemas de máquina única.

3.1.3 Algoritmo NEH e outras publicações relevantes que contemplam resolução MIP

O algoritmo NEH de Nawaz *et al*. (1983) é muito citado na literatura de *scheduling.* Como visto na seção 3.1.1, o NEH trata de um problema de sequenciamento *flow shop* e é uma heurística que segue a regra de maior tempo de processamento para ordenação dar tarefas, sendo que a função objetivo deste algoritmo é a de minimizar o *makespan*.

Uma publicação de resolução em ambiente produtivo *flow shop* que cita e faz um comparativo à resolução de NEH é o artigo de Nagano e Moccellin (2002). Os autores nomearam o método de resolução como N&M e foi desenvolvida uma nova heurística construtiva que também considera como objetivo final a minimização do *makespan*. O problema foi definido para *n* tarefas e *m* máquinas onde a sequência de processamento é a mesma para todas as tarefas.

No algoritmo de NEH, as tarefas com maiores tempos de processamento são priorizadas em todas as máquinas. Já a heurística N&M penaliza as prioridades definidas pelo NEH através de um *Lower Bound* vinculado ao tempo de espera de uma tarefa. Um dos principais resultados obtidos foi de que, para problemas contendo até dez máquinas e 100 tarefas a serem produzidas, o método N&M supera as médias dos resultados obtidos através da resolução NEH (NAGANO; MOCCELLIN, 2002).

Segundo Ruiz *et al*. (2008) há uma tendência de as pesquisas estarem voltadas à busca de métodos de solução de modelos construídos mais próximos da realidade. A complexidade de resolução de um problema de *scheduling* pode aumentar devido à tentativa de minimizar a lacuna entre a realidade e o problema modelado. Este fato foi retratado no artigo

de Ruiz *et al*. (2008) onde os autores buscaram desenvolver um modelo de programação inteira mista, ou então, *Mixed Integer Programming* (MIP) de um ambiente produtivo mais próximo da realidade onde é possível encontrar um elevado número de características e restrições simultâneas. Algumas destas eram: datas de liberação das ordens de produção nas máquinas, *setups* dependentes da sequência, disponibilidade de máquina, entre outras.

O artigo de Ruiz *et al*. (2008) propôs um problema que utilizou de MIP e algumas heurísticas para resolver o *scheduling* de *n* trabalhos em *m* estágios onde, em cada estágio, existia um número finito de máquinas independentes. Essas características classificavam o tipo de fluxo produtivo como *flow shop* híbrido. No problema havia a particularidade de que não era obrigatória a passagem por todos os estágios produtivos, que o classificava como um problema de *flow shop* híbrido flexível, ou então, *hybrid flexible flowline problem* (HFFL) e a função objetivo utilizada foi de minimização do *makespan*. Foram utilizadas algumas heurísticas como, por exemplo, uma adaptação do algoritmo NEH de Nawaz *et al*. (1983). Para instâncias maiores, o algoritmo NEH adaptado foi o que obteve melhor performance comparado ao MIP e demais heurísticas.

Ruiz *et al*. (2008) observou que uma parcela considerável dos trabalhos de *flow shop* híbrido utilizava de programação linear inteira mista (PLIM) como método de resolução dos problemas. Os problemas que envolvem MIP possuem a característica de que algumas de suas variáveis devem pertencer ao conjunto dos números inteiros (DAKIN, 1965).

Em Sadjadi *et al*. (2008) explorou-se três problemas gerais de *scheduling* em ambiente *flow shop* e utilizou de modelagem inteira mista para resolvê-los. Os três tipos de abordagem utilizados consideraram pesos para determinados parâmetros da função objetivo sendo que, na primeira abordagem, buscou-se minimização de atraso total e assumiu-se que haviam tempos de início específicos para as tarefas. A segunda abordagem buscou minimizar o *makespan* e considerou restrições relacionadas aos tempos de atraso. Por último, também foi utilizado o objetivo de minimizar o *makespan* e foram incluídas restrições de tempos de *setup* dependentes da sequência (do inglês, *Sequence Dependent Setup Times* - SDST).

Uma característica dos problemas estudados no artigo de Sadjadi *et al*. (2008) é a de que as três abordagens consideraram que não era permitido haver permutação entre as tarefas já alocadas e, em casos em que não se permite permutação, como demonstrado por Pugazhendhi *et al*. (2003), é possível que a complexidade de resolução do problema seja reduzida. Além disso, respeitando relações de precedência, não permitia tempo de espera entre tarefas. Essa última característica, segundo Sadjadi *et al*. (2008) é muito comum de ser

encontrada em problemas reais de produção. Por fim, a resolução dos modelos propostos apresentou resultados satisfatórios no que diz respeito à classe de problemas de *flow shop* híbrido onde não é permitido permutação de tarefas já alocadas e obteve resultados superiores se comparados às resoluções dos mesmos problemas masque permitiam tempo de espera entre as tarefas (SADJADI *et al*., 2008).

Naderi *et al*. (2009) estudaram um caso de *scheduling* de ambiente *flow shop* onde o objetivo era minimizar o atraso total ponderado (do inglês, *Total Weighted Tardiness* - TWT), com diferentes pesos para as tarefas. Além de utilizarem MILP, para instâncias menores, foi utilizado uma metaheurística chamada de algoritmo eletromagnético (do inglês, *Eletromagnetism Algorithm* - EMA). O EMA se origina da teoria do Eletromagnetismo da Física, onde considera possíveis soluções do problema como partículas eletricamente carregadas que se espalham no espaço. A analogia se refere à utilização de mecanismos de atração e repulsão para mover as possíveis soluções em direção à otimização (KHALILI e TAVAKKOLI-MOGHADDAM, 2012). De maneira geral, os diferentes pesos nos atrasos representariam as partículas eletricamente carregadas onde se busca a minimização dessa carga total.

Tabela 3.1.7 Variações nas instâncias do problema de Naderi *et al*. (2009).

Parâmetro	**Variações**
n (nº de tarefas)	20; 50; 80; 120
M (nº de estágios)	2; 4; 8
m_i (nº máquinas por estágio)	Constante; Distribuição Uniforme: U(1,4)
$P_{j,i}$ (tempos de processamento)	U(1,99)
$R_{j,i}$ (tempos de *setup*)	U(1,25); U(1,50); U(1,99); U(1,125)
$ST_{k,j,i}$ (tempos de transporte)	U(1,30)

A tabela 3.1.7 define as variações dos parâmetros nas instâncias geradas para o problema.

Para comparar os resultados obtidos com os algoritmos foi utilizado um índice de porcentagem relativa (do inglês, *Relative Percentage Index* - RPI). No RPI a solução obtida é calculada com as possíveis e melhores soluções e gera um índice que varia de 0 à 100 sendo que quanto mais próximo de zero, melhor é a solução obtida.

O RPI é obtido através da equação 3.18.

$$\text{RPI} = \frac{Alg_{sol} - Min_{sol}}{Worst_{sol} - Min_{sol}} \times 100 \qquad (3.18)$$

As melhores e piores soluções são utilizadas na equação 3.2 (que são nomeadas Min_{sol} e $Worst_{sol}$, respectivamente) e Alg_{sol} representa o TWT obtido para um determinado algoritmo e instância. O critério de tempo computacional de parada foi definido como um tempo fixo de *n* multiplicado por *m* e multiplicado por 0,25 segundos.

O algoritmo EMA forneceu os melhores resultados entre esses algoritmos com RPI de aproximadamente 2,5%. Os algoritmos com pior desempenho em todos os subconjuntos foram os obtidos por MILP com RPI médio de aproximadamente 90% (NADERI *et al.*, 2009).

O trabalho de Ronconi e Birgin (2011) utilizou de MILP e explorou dois principais problemas: *flow shop* com *buffer* ilimitado e outro com zero *buffer*. Para *buffer* ilimitado explorou três variações dos modelos: um baseado na formulação de em Wagner (1959), outro baseado na formulação de Manne (1960) e por último na formulação de Wilson (1989). A função objetivo busca a minimização da antecipação e atraso das tarefas. Na tabela 3.1.9 é definida a simbologia de parâmetros utilizada no artigo e na tabela 3.1.8 são definidas as variáveis.

Tabela 3.1.8 Simbologia de parâmetros em Ronconi e Birgin (2011).

Parâmetros	Descrição
n	Número de tarefas
m	Número de máquinas
i	Índice para tarefas {1, 2, …, n}
k	Índice para máquinas {1, 2, …, m}
p_{ik}	Tempo de processamento da tarefa i na máquina k
d_i	Data de entrega da tarefa i

Tabela 3.1.9 Simbologia de variáveis de Ronconi e Birgin (2011).

Variáveis	Descrição
x_{ij}	Binária. Representa 1 se tarefa *j* é processada imediatamente *i*, do contrário assume 0
E_j	Contínua. Antecipação da tarefa *j*
T_j	Contínua. Atraso da tarefa *j*
C_{jm}	Contínua. Tempo para finalizar tarefa *j* na máquina *m*
$I_{j,k}$	Contínua. Tempo de ocioso entre as tarefas *j* e (*j* +1) na máquina *k*
W_{jk}	Contínua. Tempo de espera da tarefa *j* no *buffer* entre as máquinas *k* e (*k* + 1)

A modelagem do caso de *buffer* ilimitado, baseado no modelo de Wagner (1959), apresentou as seguintes equações:

$$\text{Minimizar } \sum_{j=1}^{n} E_j + T_j \tag{3.19}$$

Sujeito às seguintes restrições:

$$T_j \geq C_{jm} - \sum_{i=1}^{n} x_{ij} d_i \qquad \forall \{j = 1,2,\ldots,n\} \tag{3.20}$$

$$E_j \geq \sum_{i=1}^{n} x_{ij} d_i - C_{jm} \qquad \forall \{j = 1,2,\ldots,n\} \tag{3.21}$$

$$C_{1m} = \sum_{k=1}^{n-1} \left(\sum_{i=1}^{n} x_{i1}\, p_{ik} + W_{1k} \right) + \sum_{i=1}^{n} x_{i1}\, p_{im} \tag{3.22}$$

$$C_{jm} = C_{j-1,m} + I_{j-1,m} + \sum_{k=1}^{n} x_{ij}\, p_{im} \qquad \forall \{j = 2,3,\ldots,m\} \tag{3.23}$$

$$I_{jk} + \sum_{i=1}^{n} x_{i,j+1} p_{ik} + W_{j+1,k} = W_{j,k} + \sum_{i=1}^{n} x_{ij}\, p_{i,k+1} + I_{j,k+1} \qquad \forall \{j = 1,2,\ldots,n-1\} \; \{k = 1,2,\ldots,m-1\} \qquad (3.24)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall \{j = 1,\ldots,n\} \qquad (3.25)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall \{i = 1,\ldots,n\} \qquad (3.26)$$

A equação (3.19) refere-se à função objetivo do problema e esta buscava a minimização da antecipação e atraso das tarefas.

A restrição (3.20) determina o tempo individual de atraso de cada tarefa. A equação (3.21) é utilizada para identificar o adiantamento do início de cada tarefa. As restrições (3.22) e (3.23) definem os tempos de finalização de cada uma das tarefas na máquina *m*. A restrição (3.24) garante as relações de precedência necessárias para manter a consistência entre os tempos ociosos das máquinas e os seus tempos de atraso. Por último, as equações (3.25) e (3.26) garantem que cada tarefa possua apenas uma posição no *scheduling* e também busca certificar que cada posição esteja ocupada apenas por uma única tarefa (RONCONI; BIRGIN, 2011).

Para resolver o modelo de Ronconi e Birgin (2011), foi necessário buscar um valor adequado para o parâmetro *M* pois, em muitos casos, ele impedia o *solver* de encontrar soluções inteiras ótima sem um tempo computacional razoável. Este foi definido na equação (3.27).

$$M = 100 \sum_{j=1}^{n} \sum_{k=1}^{m} p_{j,k} \qquad (3.27)$$

Foram realizadas substituições de equações na resolução baseada na formulação de Manne (1960) buscando-se obter uma redução de variáveis binárias e, adicionalmente,

gerou-se uma inequação que representou um *lower* e um *upper bound* para uma das variáveis do problema. Se comparado aos demais modelos, o método *branch and bound* encontrou mais rapidamente uma solução ótima para o problema (RONCONI; BIRGIN, 2011) na resolução adaptada do modelo de Manne (1960). Com isso, os autores afirmaram que uma das possíveis explicações de se encontrar mais rapidamente a solução poderia ser realmente a redução destas variáveis binárias.

Em Haddadzade *et al*. (2014) foi explorado um problema de *scheduling* de operações de produção em um ambiente *job shop*. No problema, um conjunto de *n* partes de um produto deveriam ser processadas em *k* máquinas. Além disso, era preciso selecionar recursos específicos e sequenciar as operações buscando gerar uma sequência nas quais fossem satisfeitas as restrições de precedência entre as operações e, por fim, o objetivo de menor tempo de processamento total fosse alcançado. Para alcançar este objetivo um algoritmo híbrido de *Simulated Annealing* (SA) foi utilizado. Um diferencial importante dessa pesquisa era de que, primeiramente, um sistema de gerenciamento gerava todos os planos mestre de produção possíveis, independentemente dos recursos da fábrica. Em seguida, quatro planos próximos da otimalidade (*near-optimal*) são selecionados via algoritmo Dijkstra (1959) e dez cenários são gerados utilizando um método de amostragem específico. O objetivo era escolher a sequência que somava o menor tempo de processamento, ou seja, que representava o caminho mais curto.

3.2 ROTEIRIZAÇÃO

Além da programação e sequenciamento das operações de produção, uma grande parte das empresas possui a função de programar e sequenciar as tarefas de distribuição, representando, assim, problemas de roteirização de veículos.

As decisões sobre quanto carregar em cada veículo, para onde enviá-lo e qual rota utilizar, se enquadra na classe geral de problemas relacionados ao roteamento de veículos (VRP – *Vehicle Routing Problem*) ou roteirização de recursos logísticos (SIMCHI-LEVI *et al*., 2014). Segundo Gomes (2004) a roteirização está primordialmente ligada às decisões sobre a utilização de recursos logísticos e alocação de clientes a serem atendidos.

Em muitos sistemas de distribuição, além da data, da carga e local específico em que os materiais devem ser entregues, cada cliente pode especificar uma janela de tempo de entrega. As janelas de tempo são intervalos de tempo disponíveis para o atendimento da entrega/coleta de produtos. Outras características importantes em problemas práticos de

roteamento são: frota heterogênea e local de entrega ou coleta, armazéns (BODIN, 1990). Assim, a resolução de problemas de roteirização deve indicar a sequência e programação de entregas sem violar a capacidade do veículo, manter as restrições de janela de tempo, data e local definidos, ao mesmo tempo que, busca atender os clientes com decisões de distribuição otimizadas (SIMCHI-LEVI *et al*., 2014).

Além disso, segundo Bodin (1990), é possível que a função objetivo dos problemas de roteamento também contemple multicritérios, por exemplo, minimizar os custos de transporte e obedecer a restrições que aplicam penalidades ao não cumprimento das janelas de tempo.

Segundo Dantzig e Ramser (1959) o problema de roteamento de veículos pode ser considerado como uma generalização do clássico problema do Caixeiro Viajante, ou então, *Travelling Salesman Problem* (TSP) de Flood (1955). O problema do Caixeiro Viajante foi inicialmente exposto, em 1934, durante um seminário na Universidade de Princeton por Hassler Whitney e busca encontrar o menor tempo total de viagem, menor distância total, de um vendedor que deseja viajar de sua casa para cada uma das *n* cidades especificadas no problema e, em seguida, retornar o local inicial de partida (FLOOD, 1955).

Os problemas de roteamento de veículos podem ser matematicamente modelados como grafos, sendo estes compostos por um conjunto de nós representando clientes a serem atendidos e por um conjunto de arcos, que conectam os pares de nós, representando custos (monetários, tempo, distância, entre outros) de se mover entre os dois nós (LABADIE; PRINS, 2012).

Em Clarke e Wright (1964), foi tratado um problema de roteamento de uma frota de veículos, de capacidades variadas, que deveriam realizar entregas de um depósito central a um grande número clientes. A resolução do problema se baseou no clássico algoritmo de Dantzig e Ramser (1959), e o objetivo era minimizar a distância total percorrida. Este considera e analisa todos os pares de clientes e classifica as economias potenciais em ordem não crescente. Inicialmente cada cliente aparece separadamente em uma rota, em seguida a lista de pares de clientes possíveis é examinada e duas rotas são mescladas quando estas forem viáveis. Uma união de rotas é aceita somente se a redução de custo associada for não-negativa, mas, se o número de veículos for minimizado, uma união de rotas com valores negativos poderia ser considerada. As distâncias entre dois pontos eram fornecidas, o volume do material a ser transportado não variava e foi considerado uma viagem para cada entrega. A formulação mais recente foi 17% melhor que o algoritmo de Dantzig e Ramser (1959) nas instâncias testadas no

problema, que considerava 30 clientes. Uma possível explicação para este resultado foi a remoção de uma restrição do problema original, no primeiro dos *N* estágios. Essa restrição determinava que somente seriam selecionados clientes cuja carga combinada não excedesse um dado limitante. Isso foi considerado como motivo pois, com essa restrição, o algoritmo não permitia selecionar clientes que estavam muito distantes no primeiro estágio, o que se tornou positivo em certos casos analisados.

Outro algoritmo clássico da literatura é a heurística de Mole e Jameson (1976) no qual se baseou no problema de Clarke e Wright (1964) e foi definida como uma abordagem mais flexível que a anterior por considerar a combinação de diversas características como a possibilidade de múltiplas viagens pelo mesmo veículo, restrições de capacidade do veículo, restrições de distância máxima percorrida, número limitante de viagens por veículo e clientes prioritários. O método escolhido para resolução foi considerado computacionalmente eficiente por apresentar bons resultados e se apresentar flexível no estudo de sensibilidade relacionado à alterações na distância entre cidades.

Parte dos conceitos presentes na literatura de roteirização são utilizados em outros problemas de otimização. Por exemplo, em Wang *et al*. (2016) explorou-se um problema de uma fábrica de tabaco que produzia em batelada. A função objetivo definida para o problema era tal que buscava reduzir o custo total do sistema. Para a resolução do mesmo, foi utilizada uma adaptação do TSP. No problema, os lotes de tabaco que deveriam ser processados foram considerados como cidades e os tempos de processamento foram associados à distâncias entre essas cidades. Com isso, buscou-se uma permutação ideal dos lotes a serem produzidos e o custo de permutação foi considerado a dimensão de maior impacto nos resultados. Como o TSP é considerado um problema *NP-Hard*, o tempo de resolução aumentou exponencialmente quando o número de cidades aumentou, assim, um algoritmo genético foi utilizado para resolver o modelo. O Algoritmo Genético utilizado como base foi obtido no artigo de Al-Dulaimi e Ali (2008).

No artigo de Al-Dulaimi e Ali (2008) foi resolvido o problema do Caixeiro Viajante baseado no Algoritmo Genético, gerando o TSPGA (do inglês, *Traveling Salesman Problem based on Genetic Algorithm*). Para a resolução foi possível utilizar uma adaptação do TSP de forma que, no problema, o número de cidades *i* envolvidas foram parametrizados como letras do código genético. As distâncias entre as cidades foram calculadas ou consideradas como dados iniciais do problema. Um genoma aleatório inicial era gerado e neste eram aplicadas operações de mutações o que levava ao cálculo de uma nova solução possível que substituía a

solução inicial. Esse processo era repetido até que se encontrasse a solução final otimizada, ou seja, o sistema que minimizasse o caminho entre as cidades.

3.2.1 FORMULAÇÃO MATEMÁTICA DO VRP

O problema de roteamento de veículos considerando capacidade (do inglês, *Capacitated Vehicle Routing Problem - CVRP*) foi definido por Dantzig e Ramser (1959) e, a partir disso, encontram-se diversas heurísticas na literatura que podem ser utilizadas em problemas CVRP (LABADIE; PRINS, 2012).

A formulação matemática do VRP, apresentada por Dantzig e Ramser (1959), considera que todos os veículos possuem a mesma capacidade, C, e que um veículo poderia realizar de 1 à t entregas em cada viagem.

$$\sum_{i=1}^{t} q_i \leq C \tag{3.28}$$

$$\sum_{i=1}^{t+1} q_i > C \tag{3.29}$$

Nas equações (3.28) e (3.29) o somatório de produtos com volume q_i a serem entregues em uma viagem, com o número máximo de entregas t, deve ser menor ou igual à capacidade do veículo, C, e somatório de q_i com $t+1$ entregas devem ser maior que C.

Dado um número finito de rotas, temos que P_i (i = 1, 2 … n) representam as cidades, ou então, destinos e as distâncias entre os pontos P_i são representadas por d_{ij} (i, j = 0, 1 … n). P_0 é considerado o ponto final.

A variável binária x_{ij} representa as cidades visitadas e possuem valor igual à 1 se i precede j.

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \forall \{i = 1,2\ldots,n\} \tag{3.30}$$

A equação (3.30) garante as relações de precedências entre as cidades visitadas.

$$D = \sum_{i,j=0}^{n} d_{ij} x_{ij} \tag{3.31}$$

A equação (3.31) representa a distância total *D* percorrida e a função objetivo do problema busca a minimização do somatório das distâncias *D*.

3.2.2 *Scheduling* e roteirização

Tendo visto a literatura de *scheduling* e roteirização é possível encontrar artigos que abordam a resolução destes dois problemas de maneira complementar, ou seja, o problema de *scheduling* é resolvido e, em seguida, se resolve o problema de roteirização para distribuição da produção ou vice e versa.

Polacek *et al*. (2007) exploraram uma aplicação de *scheduling* e roteirização no setor de serviços onde considerava uma empresa multinacional do setor alimentício. Para resolução do problema foi aplicado a metaheurística *Variable Neighborhood Search* (VNS). Para resolver o problema, a metaheurística VNS foi adaptada para o problema de roteamento de veículos com janelas de tempo. A ideia básica dessa metaheurística é o conceito de mudança de vizinhança dentro de uma busca local. O problema apresentou várias restrições, como por exemplo, dias disponíveis para visitas a clientes, visitas repetidas a clientes e ao horário de trabalho dos vendedores. O problema também explorou alterações no cenário do problema aplicando mudanças nas características do negócio, sendo estas: penalização por variações no intervalo de tempo entre visitas consecutivas ao mesmo cliente, sem tempo livre de intervalo do funcionário, flexibilidade de horário do consumidor, flexibilidade de horário do funcionário e periodicidade de visitas. Embora a flexibilidade possa ser limitada em problemas de roteirização, em que os clientes podem ficar sem mercadorias, mostra-se que a abordagem flexível foi altamente benéfica para o problema. A flexibilidade do horário do funcionário, por exemplo, representou uma redução dos dias de visitação a todos os clientes, no mês, de 22 para 13 dias.

O planejamento integrado das diversas etapas do ciclo produtivo e a evolução de uma visão de cadeia de suprimentos vem se tornando um fator chave para o sucesso das empresas (MIN; ZHOU, 2002). A resolução de forma não integrada e sem efetiva coordenação

entre os dois subproblemas, podem não garantir, necessariamente, uma boa solução ou um resultado ótimo global (CHANG; LEE, 2004).

3.3 PROBLEMAS INTEGRADOS DE PRODUÇÃO E DISTRIBUIÇÃO

Considerando a literatura de problemas de programação e sequenciamento de operações de produção e roteirização, sabe-se que é possível realizar a união dos mesmos e buscar resolvê-los de forma integrada de modo a otimizar um único objetivo.

Problemas integrados de programação e sequenciamento de operações de produção e distribuição, em virtude da natureza combinatória de ambos, são considerados como problemas de resolução complexa, o que os classifica como pertencentes à classe *NP-Hard*. Devido a este motivo, a literatura sobre o tema busca, em grande parte, desenvolver métodos e modelos que são capazes de resolver tais problemas em tempo razoável (EHM; FREITAG, 2016).

Na literatura de PIPD's, problemas integrados de produção e distribuição (do inglês, *Integrated Production-Distribution Problems* - IPDP), existem duas categorias de problemas e estes são os chamados PPIPD (do inglês, *Integrated Scheduling of Production and Distribution Problems* – ISPDP), conhecido como Problemas de Programação Integrada de Produção, e PPIED (do inglês, *Integrated Scheduling of Production, Inventory and Distribution Problems* – ISPIDP) que são Problemas de Programação Integrada de Produção, Estoque e Distribuição (WANG *et al*., 2015).

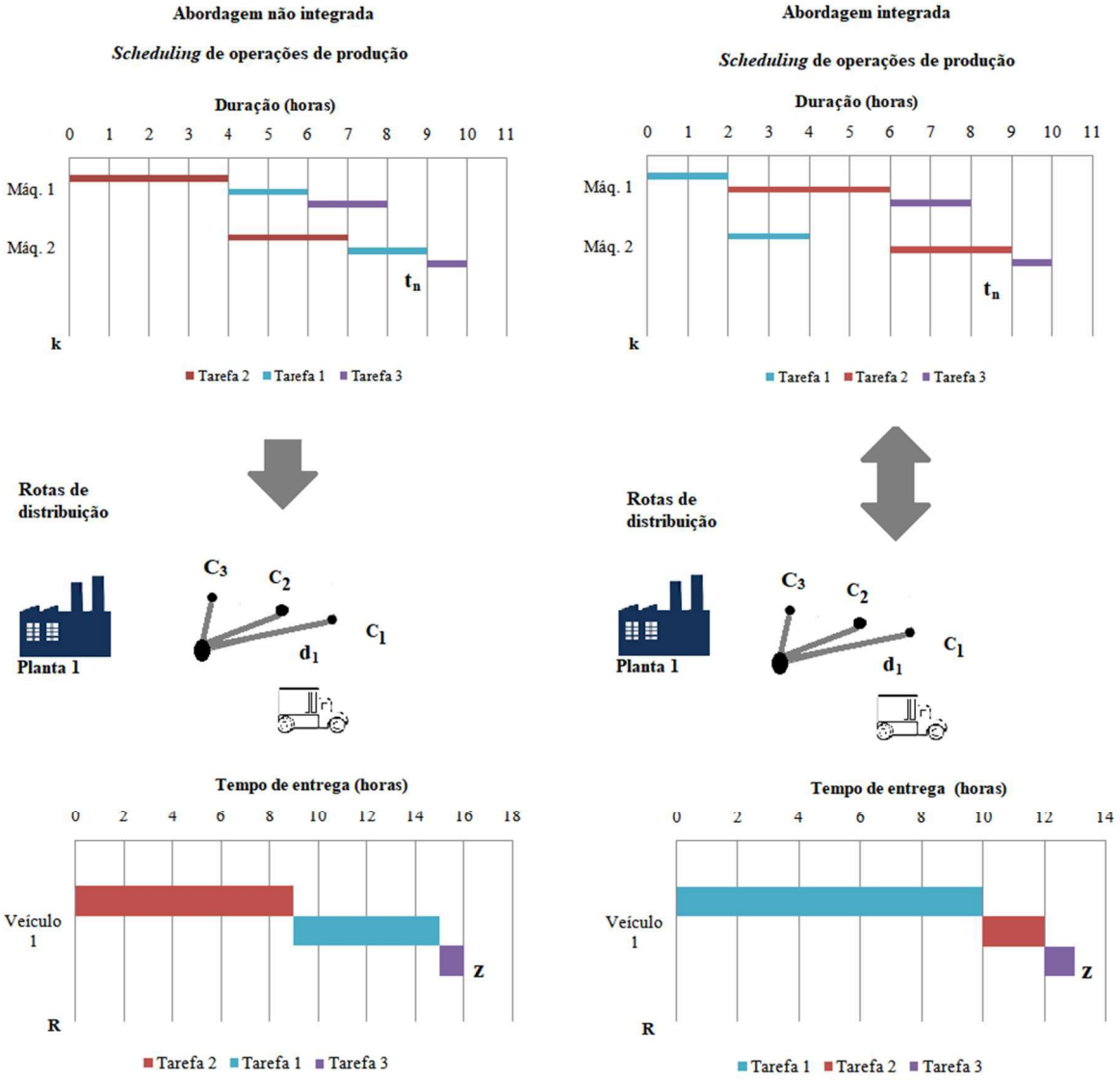
O primeiro, PPIPD, considera um limite de capacidade dos veículos e também de estoque zero (ou limitado), o que gera um estreitamento das decisões de produção e distribuição. Já o segundo, PPIPED, considera uma capacidade ilimitada de armazenagem em cada estágio produtivo e permite que o próprio sistema ajuste resultados de diferentes taxas de produção e distribuição (WANG *et al*., 2015).

Na literatura existente sobre a integração de problemas de produção e distribuição, um elemento-chave é a conexão existente entre cada subsistema. Para se obter a conexão entre os subproblemas de produção e distribuição, podem ser citadas duas estratégias diferentes normalmente relacionadas com uma determinada característica específica do material ou produto. Uma das estratégias permite que os produtos, ao finalizarem sua produção, sejam armazenados temporariamente em um estoque intermediário, com o objetivo de sincronizar produção e distribuição com diferentes taxas. Esta estratégia contempla os

problemas do tipo PPIED. Na segunda classe de problemas, PPIPD, existe uma restrição que impede que os produtos sejam armazenados em um estoque intermediário, sendo assim, os produtos precisam ser distribuídos imediatamente após a produção, o que compõe a segunda estratégia (WANG *et al*., 2015).

A figura 3.1 esquematiza as operações de planejamento e sequenciamento da produção e roteamento de distribuição e mostra uma abordagem não coordenada na primeira parte (parte esquerda) *versus* uma abordagem integrada. As máquinas, definidos por *k*, são representados no eixo *y* do gráfico representativo do *scheduling* de operações de produção e a duração das tarefas são representadas por t_n. Além disso, *n* representa a tarefa do pedido do cliente C_n, sendo que no exemplo da figura são 3 clientes a serem atendidos. Ou seja, os gráficos representam o *scheduling* desenhado para o problema em questão e as rotas de distribuição definidas. Há um veículo para distribuição e a capacidade de entrega é de um pedido por vez. A distância entre a planta fabril e os clientes é representada por d_i. Para o exemplo, foi utilizada a regra de alocação de maior tempo de processamento para o *scheduling* e, no integrado, alocação de acordo com maior tempo total. No exemplo da figura 3.2, analisando o tempo total final, *z*, o problema integrado tem um tempo total menor que considerando o *scheduling* somente como entrada para o problema de roteirização. A abordagem integrada dos problemas de produção e distribuição, ao serem resolvidos buscando a minimização ou maximização de um o mesmo objetivo final que a forma desacoplada, podem apresentar resultados superiores aos obtidos com a última.

Figura 3.1 Representação exemplificada de sistema de produção - distribuição não integrado e IPDP.



Fonte: Adaptado de Moons *et al*. (2017).

Ao estudar a literatura recente sobre o tema, os artigos contemplam diferentes focos de análise sobre problemas integrados de produção e distribuição (PIPDs).

Hao *et al*. (2015) exploraram uma versão simplificada do problema de programação de produção e distribuição integrada com vários fabricantes de maquinários industriais independentes, localizados em plantas diferentes. Nesse problema, uma plataforma central recebia um conjunto de pedidos e estes deveriam ser atribuídos aos fabricantes. Na fase de produção, foram consideradas máquinas paralelas em todos os fabricantes e a distribuição era realizada por veículos com capacidade limitada onde entregas parciais não eram permitidas. No problema busca-se maximizar o soma dos lucros de cada um dos fabricantes.

A função objetivo do modelo Hao *et al*. (2015) busca otimizar a soma total do lucro de cada fabricante, sob as condições de que o lucro de cada fabricante não deveria ser negativo e os pedidos deveriam ser concluídos antes do prazo de entrega, ou seja, não permitindo atrasos.

A resolução do modelo de programação inteira mista atingiu resultado ótimo quando o número de pedidos, o número de plantas (unidades de fabricação) e as datas limite de envio não possuíam grande volume e variedade. No caso, os autores reportaram que foi possível resolver o problema para um número de pedidos igual a 20, 3 plantas de fabricação e a data limite de entrega era de 1.000 dias. Isso porque o tempo necessário para o *solver* resolver o problema aumentou exponencialmente com o número de pedidos, número de fábricas e datas limite para o envio. Assim, uma sugestão de pesquisa futura foi explorar a resolução através de algoritmos heurísticos (HAO *et al*., 2015).

No problema explorado na publicação por Sawik (2016) foi estudado uma formulação de programação inteira mista para *scheduling* da produção e programação da distribuição em uma cadeia de suprimentos, na qual estava sujeita a riscos de ocorrerem rupturas e havia limites de atrasos. O problema busca atender dois objetivos que podem ser considerados conflitantes: a minimização do custo e a maximização do nível de serviço. Além disso, três métodos de distribuição dos produtos foram considerados: envio de cargas com pedidos de clientes diferentes, envio de única carga de pedidos de clientes diferentes e envio individual de cada pedido de um único cliente (imediatamente após sua finalização) (SAWIK, 2016).

Os resultados obtidos indicaram que, para todos os métodos de distribuição, quanto maior a distância entre locais de entrega, maior atraso dos cronogramas produção e distribuição esperados (SAWIK, 2016).

Além disso, uma observação do artigo de Sawik (2016) foi de que, para todos os tipos de métodos de envio de mercadorias, referindo-se aos diferentes tipos de recursos logísticos utilizados, o ramo orientado para o nível de serviço era mais diversificado que o ramo orientado para custos. Os resultados finais comprovaram que, para o objetivo de mínimo custo, o fornecedor mais barato era geralmente selecionado, mas também indicaram maiores atrasos na cadeia. Diferentemente do objetivo de nível máximo de serviço onde priorizava fornecedores mais caros e confiáveis. Outra observação interessante foi o fato de que, para o objetivo de nível máximo de serviço, a maior fração esperada de demanda atendida integralmente significou uma maior parte de demanda rejeitada. Isso significou que, para maximizar o nível de serviço

esperado, pedidos que não poderiam ser atendidos nas datas inicialmente solicitadas pelos clientes, eram rejeitados. Portanto, apesar de menores rupturas, quanto mais orientado para custos, maiores eram os atrasos da programação prevista de produção e distribuição. Por fim, foi identificado que as programações da produção foram melhores niveladas em redução de custo com envio de único lote e em um maior nível de serviço com um único envio de múltiplos lotes (SAWIK, 2016).

Na publicação de Cheng *et al*. (2015), o problema integrado analisado possuía, na parte de produção, tarefas em quantidades arbitrárias, tempos de processamento arbitrários e as máquinas de processamento em lotes possuíam capacidade finita. O tempo de processamento de um lote era considerado o maior tempo de processamento de todos os trabalhos do mesmo lote. No subproblema de distribuição, os veículos tinham capacidades iguais. Para resolver o problema da produção foi proposto a utilização do método aprimorado de colônia de formigas e um método heurístico para a parte de distribuição. As tarefas em batelada poderiam ser processadas juntas, desde que os volumes totais das tarefas da batelada não excedessem a capacidade da máquina. O objetivo do problema era minimizar o custo total de produção e distribuição (CHENG *et al*., 2015).

Uma característica importante do problema de Cheng *et al*. (2015) é que quando um lote estava sendo processado, nenhuma interrupção era permitida. Outro fator é que quando os trabalhos eram concluídos, estes eram entregues por veículos idênticos e com capacidades de transporte iguais. Os resultados apresentados mostraram que o desempenho da heurística era satisfatório e o tempo de execução não era superior a cinco segundos (considerando cerca de 200 tarefas). Por buscar resolver um problema de máquina única, o artigo sugeria pesquisas futuras onde as configurações da máquina são mais complexas. Isso porque tipos de fluxo como *flow shop* e *job shop* são mais frequentemente encontrados na prática.

Em Cheng *et al*. (2017), o artigo tem como objetivo resolver um problema de *scheduling* buscando minimizar os custos de produção, armazenamento e entrega. A parte de produção do problema era um ambiente de máquina única e o tipo de processamento era em lotes. Cada tarefa possui seu respectivo volume e tempo de processamento. As tarefas eram agrupadas e programadas na máquina de processamento em lote, desde que a carga total não excedesse a capacidade da máquina. O tempo de processamento do lote era o maior tempo de processamento entre todas as tarefas do lote. Na parte referente à distribuição, existia um único veículo para distribuir a produção e o custo de entrega era uma função linear do número de entregas. Se o veículo não estivesse disponível, os lotes finalizados seriam armazenados no

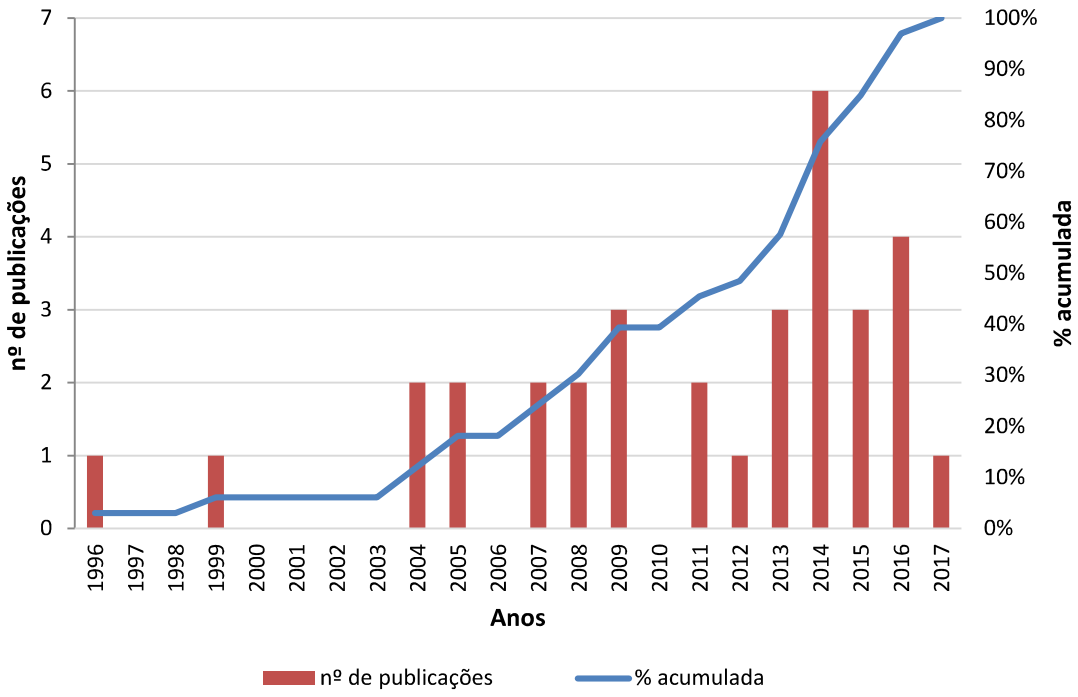
estoque e o custo de estoque era determinado como uma função do tempo de espera do lote. Nesta publicação concluiu-se que, se as tarefas tivessem todas o mesmo tamanho, o algoritmo era resolvido em tempo polinomial mesmo que os tempos de processamento fossem diversos (CHENG *et al*., 2017).

No artigo de Moons *et al*. (2017) foi realizada uma revisão da literatura em problemas integrados da produção e distribuição e as publicações do tema foram revisadas, discutidas e classificadas de acordo com as principais características dos modelos matemáticos e abordagens de solução para, principalmente, identificar direções para pesquisas futuras.

Segundo a pesquisa de Moons *et al*. (2017) sobre o estado da arte em PIPD's, um dos estudos mais antigos que utilizam problemas integrados de produção e distribuição é o artigo de Hurter e Van Buer (1996) que considera um ambiente de máquina única e explora a produção e distribuição de um jornal considerando apenas um recurso de impressão, mais de um veículo disponível para distribuição e o objetivo é a minimização do número de veículos utilizados. Sua revisão bibliográfica incluiu estudos que atendiam aos seguintes critérios de seleção: (1) problemas de produção e distribuição devem ser resolvidos por meio de uma abordagem integrada; (2) as operações de distribuição eram baseadas em decisões de roteamento de veículos; e, (3) problemas integrados enfocam o nível de decisão operacional (não considerando decisões no nível de decisão estratégica ou tática, por exemplo, decisões de dimensionamento de lotes). Para estreitar o escopo, apenas artigos escritos em inglês, publicados entre 1996 e 2016 e artigos que estavam disponíveis online em 2016 foram considerados na revisão de literatura. Em primeiro lugar, foram selecionados artigos publicados em periódicos com Fator de Impacto de pelo menos 1,0 no domínio de *Operations Research & Management Science* com as seguintes palavras-chave no título: *produção* ou *scheduling* em combinação com *entrega*, *distribuição, roteamento* ou *transporte*. Artigos adicionais considerando as mesmas palavras-chave foram coletados utilizando bases de dados técnico-científicas com acesso à periódicos eletrônicos, como Google Scholar, Web of Science e ProQuest. Os resultados da pesquisa foram ainda filtrados se incorporavam as palavras *integrado*, *sincronizado*, *coordenado* ou *combinado* nos artigos. Complementando, três artigos relevantes citados no estreitamento da pesquisa inicial também foram considerados. Assim, totalizou-se 33 publicações que obedeciam os critérios definidos. De acordo com a pesquisa, apesar de uma das publicações mais antigas no tema ter sido apresentada há cerca de 24 anos, como pode-se observar no gráfico da figura 3.2, no qual representa o número de artigos publicados ao longo dos anos de 1996 e 2017 no tema de PIPD, as pesquisas relacionadas ao

tema de programação e sequenciamento de operações de produção e distribuição obtiveram um elevado crescimento mais concentrado ao longo dos últimos 10 anos.

Figura 3.2 Demonstração do crescimento das publicações em PIPD's.



Fonte: Adaptado de Moons *et al*. (2017).

Em Moons *et al*. (2017) também foram citados os autores Thomas e Griffin (1996) e Scholz-reiter *et al*. (2011) e estes exploram razões pelas quais as empresas ainda preferiam uma abordagem não integrada para problemas de *scheduling* e roteirização. Segundo eles, uma das razões para esta afirmação era o fato de que, como diferentes departamentos de uma empresa ou mesmo diferentes empresas, como fornecedores de serviços logísticos, são os responsáveis por essas decisões, estes departamentos e níveis da cadeia precisariam estar integrados para que o *scheduling* da produção e distribuição obtivesse melhor desempenho de forma integrada. A segunda razão era de que os problemas individuais, como por exemplo, um problema de roteamento de veículos (PRV) do planejamento da distribuição, já é difícil de resolver sozinho, e integrá-lo com a produção engloba um maior desafio. A última razão, os estoques intermediários frequentemente utilizados entre as funções de produção e distribuição reduzem a necessidade de integrar essas funções da cadeia de suprimentos. No entanto, uma tendência crescente pode ser observada para reduzir esses estoques intermediários e para utilizar recursos de forma mais eficiente. Além disso, em alguns casos, a integração de problemas de

scheduling e roteirização pode gerar uma melhoria média entre 5% e 20% nas soluções obtidas em comparação com uma abordagem não integrada do mesmo problema (CHANG *et al*., 2014).

Outra observação interessante verificada por Moons *et al*. (2017) era de que as janelas de tempo são uma característica muito comum nas operações dos sistemas de distribuição, sendo que é possível observar que estas janelas de tempo estão consideradas em diversos problemas de distribuição na literatura publicada desde 2007. Já, referente às operações de produção, o estudo verificou que a maioria das publicações de problemas integrados exploravam problemas de produção em lotes e considerava *setup* entre lotes consecutivos.

Devapriya *et al*. (2017) exploraram um problema integrado da classe IPDP de um produto perecível que deveria ser produzido e distribuído antes de expirar sua data de validade. A função objetivo do modelo matemático buscava atingir o custo mínimo total do sistema. No problema foi considerado *batch processing* e um único veículo para a distribuição que lidava com entregas em locais geograficamente dispersos. O algoritmo contempla variáveis como o tamanho da frota e as rotas dos caminhões, sendo que estas estavam sujeitas a uma restrição no horizonte de planejamento devido justamente ao fato do material produzido possuir um tempo de vida limitado. Para resolver o problema, foram utilizadas heurísticas baseadas em algoritmos evolutivos e um modelo de programação inteira mista. Neste problema, à medida que o número de clientes aumentava, as heurísticas apresentavam melhor desempenho que o MIP. Para pesquisas futuras, no âmbito da distribuição, sugeriu-se incluir caminhões com capacidade variável, o que provavelmente produziria resultados que são mais encontrados na prática. Outra sugestão foi a de incluir uma maior variedade de produtos e cada um com diferentes datas de validade. Essas sugestões tendem a agregar considerável nível de complexidade ao problema.

Rafiei *et al*. (2018) estudaram um problema de planejamento integrado da produção e distribuição dentro de uma cadeia de suprimentos de quatro níveis, considerando duas funções objetivo. A primeira era minimizar o custo total da cadeia e a segunda era maximizar o nível de serviço. Uma característica interessante do problema era de que o mesmo foi modelado sob duas circunstâncias diferentes: uma em um mercado sem concorrência e outra em um mercado competitivo.

Para resolver o problema, foram desenvolvidos dois modelos matemáticos baseados em programação linear inteira mista. Um dos resultados obtidos foi de que, apesar de a concorrência melhorar o desempenho da cadeia em termos de qualidade, esta também podia

elevar os custos pelo mesmo motivo. Ou seja, para garantir melhores níveis de serviço, maiores custos deveriam ser empregados (RAFIEI *et al*., 2018).

O método de resolução utilizado no problema do artigo de Rafiei *et al*. (2018) foi uma abordagem multiobjetiva que considerava restrições que acrescentavam pesos de penalidade sobre a função objetivo, podendo ser chamada de *restrições elásticas*. Esse método pode ser considerado uma combinação de características do método das restrições e do método da soma ponderada que busca determinar soluções otimizadas da programação inteira multiobjetiva (EHRGOTT, 2006).

Rafiei *et al*., (2018) ainda realizaram uma análise de sensibilidade do problema. Nessa análise, descobriram que as variáveis como o nível de fluxo de material entre fornecedores, nível de fluxo de material entre unidades produtivas e o fluxo de produtos que eram transportados para os clientes, foram fatores que afetaram diretamente as variáveis de decisão do problema.

O artigo de Tavares e Nagano (2019) tratou de um problema integrado de produção e distribuição através de quatro diferentes abordagens de resolução. O ambiente produtivo era de máquinas paralelas e a distribuição dos produtos era realizada por um único veículo, no qual era permitido realizar várias rotas. As abordagens de resolução utilizadas no artigo foram de um problema inteiro misto (MIP), um Algoritmo Genético (GA), adaptado de Ulrich (2013), uma heurística construtiva e uma heurística de melhoria. O objetivo do problema integrado era minimizar o tempo total do sistema (*makespan*).

Nas tabelas 3.1 e 3.2 estão enunciados os parâmetros e variáveis utilizados para modelagem MIP do artigo de Tavares e Nagano (2019).

Tabela 3.3.1 Simbologia dos parâmetros de Tavares e Nagano (2019).

Símbolos	Descrição
i, j	Índices utilizados para identificar ordens de processamento
f	Índice utilizado para identificar máquinas
k	Índice utilizado para identificar uma rota
μ	Número de máquinas
ϱ_i	Tempo de processamento de i
k_{ij}	Representa o *setup* para alternar de i para j
σ_i	Representa o tamanho (volume) de i
p	Representa a capacidade do veículo na mesma unidade de σ_i
δ_{ij}	A distância entre i e j

Tabela 3.3.2 Simbologia das variáveis de Tavares e Nagano (2019).

Símbolos	Descrição
x_{ipf}	Variável binária que retorna 1 se a ordem de processamento *i* for produzida na posição *p* e na máquina *f*
w_{ijk}	Variável binária que retorna 1 se a ordem de processamento *j* utiliza a rota *k*, imediatamente após *i*
C_i	Representa o momento em que *i* é produzida
A_i	Representa o montante da capacidade alocada ao chegar ao cliente *i*
R_k	Representa o momento de liberação da rota *k*
D_i	Tempo de entrega do pedido ao cliente *i*
z	Representa o *makespan*
GP	Uma constante com alto valor relacionada ao sistema produtivo
$G\sigma$	Uma constante com alto valor relacionada à capacidade do veículo
GV	Uma constante com alto valor relacionada ao sistema de distribuição

A formulação matemática das restrições do problema de Tavares e Nagano (2019) estão descritas pelas equações (3.40) à (3.65). As equações que estão relacionadas às variáveis *x* são de (3.40) à (3.43). A equação (3.40) garante que a ordem zero somente seja alocada na posição zero, a equação (3.41) garante que uma ordem é produzida somente uma vez em todas as máquinas, a equação (3.42) define que uma posição de uma máquina somente terá uma ordem alocada e na equação (3.43) é garantido que uma sequência em qualquer máquina tenha posições não utilizadas somente após posições já alocadas.

$$x_{0pf} = 0 \qquad \forall \begin{matrix} p \\ f \end{matrix} \qquad (3.40)$$

$$\sum_{p,f} x_{ipf} = 1 \qquad \forall\, i > 0 \qquad (3.41)$$

$$\sum_{i} x_{ipf} \leq 1 \qquad \forall \begin{matrix} p \\ f \end{matrix} \qquad (3.42)$$

$$\sum_{j} x_{j(p+1)f} \leq \sum_{i} x_{ipf} \qquad \forall\, p > |P| \quad \forall\, f \qquad (3.43)$$

As equações (3.44) à (3.47) estão relacionadas à variável *C* que representa o momento em que uma ordem de processamento é produzida. O tempo de conclusão da ordem

0 é definido na equação (3.44). A Equação (3.45) define o valor do "*Big-M*", a Equação (3.46) define o tempo de conclusão da ordem alocada na primeira posição de cada máquina e os tempos de conclusão das posições restantes são definidos na equação (3.47).

$$C_0 = 0 \tag{3.44}$$

$$\mathrm{GP} = \sum_{\mathrm{i}} \rho_i + \sum_{\mathrm{i,j}} \kappa_{ij} \tag{3.45}$$

$$C_i \geq \rho_i + \kappa_{0i} - \mathrm{GP}(1 - x_{i0f}) \qquad \forall\, \mathrm{i},\ f \tag{3.46}$$

$$C_j \geq C_i + \rho_j + \kappa_{ij} - \mathrm{GP}(2 - x_{ipf} - x_{j(p+1)f}) \qquad \forall\, j > 0,\ i > 0,\ i \neq j,\ p < |P|,\ f \tag{3.47}$$

As equações (3.48) à (3.55) estão relacionadas à *w*, que representam as rotas a serem selecionadas. A equação (3.48) não permite rotas vazias entre duas já utilizadas, a Equação (3.49) não permite que um arco entre duas cidades seja utilizado mais de uma vez, a Equação (3.50) define que no máximo uma rota visite uma cidade, as equações (3.51) e (3.52) garantem que uma rota saia e chegue uma única vez no ponto zero, a equação (3.53) não permite que um veículo consiga sair e retornar à mesma cidade, a equação (3.54) faz com que, quando a rota *k* já foi utilizada, w_{00k} seja igual à zero e a equação (3.55) garante que um veículo que está saindo da cidade *i*, chegou à essa mesma cidade pela rota *k*.

$$w_{00(k-1)} \leq w_{00k} \qquad \forall\, k \tag{3.48}$$

$$\sum_{k} w_{ijk} \leq 1 \qquad \forall\, i > 0,\ j > 0 \tag{3.49}$$

$$\sum_{i,k} w_{ijk} \leq 1 \qquad \forall\, i,\ f \tag{3.50}$$

$$\sum_{j} w_{0jk} = 1 \qquad \forall\, k \tag{3.51}$$

$$\sum_{j} w_{j0k} = 1 \qquad \forall\, k \tag{3.52}$$

$$w_{jjk} = 0 \qquad \forall\, j > 0,\ k \tag{3.53}$$

$$w_{00k} + \sum_{j>0} w_{ijk} \leq 1 \qquad \forall\, k,\ i > 0 \tag{3.54}$$

$$\sum_{h \neq i} w_{ihk} = \sum_{h \neq i} w_{hik} \qquad \forall\, i,\ k \tag{3.55}$$

As equações (3.56) à (3.60) estão relacionadas às variáveis de capacidade do problema, representada por *A*. É definido a capacidade do ponto zero na equação (3.56), a Equação (3.57) calcula o "*Big-M*" relacionado à capacidade, a Equação (3.58) estabelece um limitante superior para capacidade do veículo, já as equações (3.59) e (3.60) estabelecem um limitante inferior e superior para a carga utilizada do veículo.

$$A_0 = 0 \tag{3.56}$$

$$\mathrm{G}\sigma = \sum_{\mathrm{i}} \sigma_i \tag{3.57}$$

$$A_i \leq \sigma_i \qquad \forall\, i > 0 \tag{3.58}$$

$$A_{\mathrm{i}} \geq A_j + \sigma_i - G\sigma.\left(1 - w_{ijk}\right) \qquad \forall\, i > 0,\ j \neq i,\ k \tag{3.59}$$

$$A_{\mathrm{i}} \leq \sum_{j>0} \sum_{h} \sigma_j\, w_{jhk} + G\sigma.\,(1 - w_{ihk}) \qquad \forall\, i > 0,\ k \tag{3.60}$$

As equações (3.61) à (3.64) estão relacionadas às variáveis de integração dos subproblemas de produção e distribuição. A rota *k* inicia somente após a produção de todos os pedidos desta rota e este fato é garantido pela equação (3.61), a equação (3.62) define o tempo de entrega dos pedidos que estão alocados na primeira posição das rotas, a equação (3.63) define o tempo de entrega dos demais pedidos da rota *k* e a equação (3.64) a rota seguinte somente pode iniciar após a finalização da anterior.

$$R_{\mathrm{k}} \geq C_i - GP.\left(1 - w_{ijk}\right) \qquad \forall\, i > 0,\ j,\ k \tag{3.61}$$

$$D_{\mathrm{i}} \geq R_{\mathrm{k}} + \delta_{0i} - GV.\left(1 - w_{0jk}\right) \qquad \begin{matrix} \forall\, i > 0 \\ j \\ k \end{matrix} \qquad (3.62)$$

$$D_{\mathrm{j}} \geq D_{\mathrm{i}} + \delta_{ij} - GV.\left(1 - \sum_{k} w_{ijk}\right) \qquad \begin{matrix} \forall\, i > 0 \\ j \neq i \end{matrix} \qquad (3.63)$$

$$R_{\mathrm{k}} \geq D_i + \delta_{0i} - GV.\left(1 - w_{i0(k-1)}\right) \qquad \begin{matrix} \forall\, i > 0 \\ k > 0 \end{matrix} \qquad (3.64)$$

Minimizando a equação (3.65), a mesma representa a função objetivo do problema: minimização do *makespan*.

$$\mathrm{z} \geq D_i + \delta_{i0} - M.(1 - x_{i0k}) \qquad \begin{matrix} \forall\, i > 0 \\ k \end{matrix} \qquad (3.65)$$

Os resultados obtidos com a aplicação de Tavares e Nagano (2019) foram analisados os resultados das diferentes abordagens e concluiu-se que a heurística de melhoria apresentou resultados superiores do que os demais métodos, mas, um fato relevante, foi de que, ao se considerar um número inferior de volume de tarefas, a heurística construtiva foi superada pelo Algoritmo Genético em nível significativo.

Após a análise da literatura, o modelo MIP de Tavares e Nagano (2019) foi o definido como base para construção do modelo utilizado no atual trabalho.

3.4 MIP-*AND-REFINE*

Na literatura de programação e sequenciamento de operações de produção e distribuição, buscando-se especificamente sobre métodos de refinamento das soluções de modelos MIP, no que se refere à utilização de solução inicial para resolução de problemas de *scheduling* e/ou roteirização,uma das publicações encontradas foi um artigo de Fischetti *et al*. (2013) e refere-se a um problema de programação das operações de dispositivos de energia. Esses dispositivos são como um conjunto de fontes e acumuladores de energia e o funcionamento destes é modelado em termos de uma determinada demanda de energia. O objetivo do problema era minimizar o valor final da conta, respeitando as restrições de duração, demanda de energia e preferências do usuário. Um modelo MIP e um algoritmo *Greedy* foram utilizados para a otimização. O algoritmo *Greedy* foi executado e este gerava uma lista de

soluções iniciais utilizada na execução do MIP, essa lista de soluções era considerada como um procedimento de refino do MIP e a técnica foi chamada de MIP-*and*-*refin*e, ou então, MIP-*refining procedure*. No artigo foram comparados o algoritmo *Greedy*, o MIP e a solução do MIP com solução inicial do *Greedy*. Os resultados computacionais confirmaram a viabilidade da abordagem MIP-*and*-*refin*e, em termos de qualidade e tempo necessário para gerar uma solução para as instâncias testadas. Ao final obteve-se uma melhoria média de 8,5% nos resultados se comparado ao modelo de segundo melhor desempenho, o *Greedy*. O tempo médio de execução do *Greedy* foi 23,41 segundos e do MIP-*and*-*refine* de 143,45 segundos.

Uma outra publicação encontrada foi de Ferris e Liu (2016) e, como o problema de Fischetti *et al*. (2013), explorou a otimização de um problema relacionado à demanda de energia. Um modelo de otimização não linear foi utilizado para gerar uma lista de soluções locais. Após isso, essa lista foi utilizada como solução inicial do MIP. O modelo foi testado em várias instâncias e dados, e gerou resultados otimizados para as operações, principalmente em instâncias de maior complexidade.

Em Gelareh *et al*. (2020) apresentou-se uma extensão do problema do Caixeiro Viajante no qual o objetivo era projetar um *tour* que proporcionasse o máximo lucro, respeitando as restrições definidas para os nós do problema. Uma formulação de programação linear inteira mista (MIP) foi proposta utilizando de uma heurística de *Variable Neighborhood Search* (VNS) como solução inicial do MIP. Foram comparados: VNS, VNS com solução inicial do MIP e MIP com aumento do limite do *gap* entre *Upper Bound* (UB) e a melhor solução encontrada. O MIP com solução inicial da heurística VNS obteve melhor desempenho em instâncias de menor tamanho mas, ainda assim, o método MIP com utilização de 1% no *gap* do CPLEX entre a função objetivo e o melhor UB encontrado foi o que obteve os melhores resultados.

Nos artigos de Ferris e Liu (2016) e Gelareh *et al*. (2020) a utilização de uma solução inicial é chamada de *mipstart parameter* para o modelo MIP.

4 DESENVOLVIMENTO

No presente trabalho de pesquisa, o tema principal é a exploração de uma técnica de refinamento das soluções obtidas com a programação e sequenciamento de operações integradas de produção e distribuição.

Apesar da abordagem não coordenada dos subproblemas ser considerada uma forma mais simplificada que se tratar de um mesmo problema integrado, a resolução integrada pode fornecer ganhos consideráveis aos resultados. A abordagem integrada utilizando-se de determinadas estratégias de resolução podem se mostrar infactíveis no problema integrado devido à complexidade matemática dos mesmos. Desta forma, fica claro a importância de se explorar diferentes técnicas de resolução de problemas integrados. Portanto, o foco deste estudo está em problemas de nível de operações integradas em que incluem tanto a programação e sequenciamento da produção e roteamento de veículos de forma integrada quanto desacoplada. Ou seja, a técnica explorada no atual trabalho busca combinar a relativa simplicidade do problema desacoplado, fornecendo ao *solver* uma solução inicial viável de um modelo de programação matemática resolvido separadamente e posterior utilização no problema integrado, combinado à comparação com a resolução somente na forma desacoplada e também na abordagem integrada, na busca de que o *software* consiga possivelmente melhorar os resultados obtidos e, idealmente, encontrar uma solução ótima para o problema.

Assim, a atual pesquisa foi realizada sob a ótica de comparar os resultados obtidos a partir de três formas de resolução de problemas PIPD's: um modelo matemático resolvido de forma desacoplada, um modelo integrado e um modelo integrado que utiliza uma solução inicial do problema desacoplado. Além disso, foram realizadas execuções com dois tempos limites diferentes.

Para desenvolver os algoritmos da atual pesquisa foram utilizadas adaptações do modelo matemático de programação inteira mista apresentado por Tavares e Nagano (2019). O modelo base foi delineado para um ambiente produtivo de máquinas paralelas e essa característica foi adaptada para máquina única. A característica de distribuição por único veículo foi mantida. Outra característica diferente do modelo base foi de que o mesmo considerava a função objetivo de minimização do *makespan* e esta pesquisa foi adaptada para minimização do tempo total de fluxo.

Os modelos indicados foram implementados em Python 3.5 e foi utilizada a biblioteca CPLEX 12.7.0.0.

Os dados de entrada utilizados no problema foram gerados aleatoriamente e formaram uma biblioteca de parâmetros.

4.1 MODELAGEM DO PROBLEMA

Tendo como base um modelo apresentado no capítulo de revisão bibliográfica, de Tavares e Nagano (2019), o modelo matemático integrado utilizado na pesquisa foi adaptado principalmente para a característica de máquina única no subproblema de produção e, além disso, não foram considerados tempos de *setup* entre as tarefas.

ÍNDICES, PARÂMETROS E VARIÁVEIS:

Inicialmente foram identificados os índices, parâmetros, variáveis, objetivos e restrições do problema. Os índices e parâmetros do subproblema de produção são os que estão descritos na tabela 4.1 e variáveis na tabela 4.2.

Tabela 4.1 Simbologia dos índices e parâmetros do subproblema de produção.

Símbolos	Descrição
i, j	Índices utilizados para identificar tarefas
p	Índice utilizado para identificar posição das tarefas
ϱi	Tempo de processamento da tarefa i

Tabela 4.2 Simbologia das variáveis do subproblema de produção.

Símbolos	Descrição
x_{ip}	Variável binária que retorna 1 se a tarefa i for produzida na posição p
C_i	Representa o momento em que a tarefa i é produzida

Os índices e parâmetros do subproblema de produção são os que estão descritos na tabela 4.3 e variáveis na tabela 4.4.

Tabela 4.3 Simbologia dos índices e parâmetros do subproblema de distribuição.

Símbolos	Descrição
i, j, h	Índices utilizados para identificar cidades
r	Índice utilizado para identificar rotas
σi	Define o volume da tarefa i
ψ	Capacidade total do veículo
δ_{ij}	Representa distância entre i e j

Tabela 4.4 Simbologia das variáveis do subproblema de distribuição.

Símbolos	Descrição
w_{ijr}	Variável binária que retorna 1 se a rota *r*, entre as cidades *i* e *j*, for utilizada
A_i	Representa o montante da capacidade alocada ao chegar ao cliente *i*
R_r	Representa o momento de liberação da rota *r*
D_i	Tempo de entrega do pedido ao cliente *i*
z	Representa o tempo de fluxo total do sistema

FUNÇÃO OBJETIVO:

A função objetivo do planejamento produção e distribuição é a minimização do tempo total de fluxo e está descrita na equação (4.1).

$$\text{Min } z = \sum_{i} Di \tag{4.1}$$

RESTRIÇÕES DO PROBLEMA:

As restrições do subproblema de produção do modelo integrado, adaptação do problema MIP de Tavares e Nagano (2019), é definido da seguinte forma e as equações relacionadas à variável binária x_{ip} são:

$$x_{0p} = 0 \qquad \forall\, p > 0 \tag{4.2}$$

$$\sum_{p} x_{ip} = 1 \qquad \forall\, i > 0 \tag{4.3}$$

$$\sum_{i} x_{ip} \leq 1 \qquad \forall\, p \tag{4.4}$$

$$\sum_{j} x_{j(p+1)} \leq \sum_{i} x_{ip} \qquad \forall\, p < |P| \tag{4.5}$$

A equação (4.2) busca garantir que a ordem 0, ou então *dummy order*, não possa ser alocada em nenhuma posição diferente da posição zero e a equação (4.3) tem o objetivo de assegurar que uma ordem só é produzida uma única vez em todas as posições. Já a equação (4.4) busca que se tenha, no máximo, uma ordem por posição e a equação (4.5) assegura que

somente possuam posições não utilizadas após posições em uso (TAVARES; NAGANO, 2019).

As equações relacionadas à variável contínua C_i são:

$$C_i \geq \varrho_i - GP.(1 - x_{i0}) \qquad \forall i \qquad (4.6)$$

$$C_j \geq C_i + \varrho_i - GP.\left(2 - x_{ip} - x_{j(p+1)}\right) \qquad \begin{matrix} \forall\, i \\ j > 0 \\ i \neq j \\ p < |P| \end{matrix} \qquad (4.7)$$

O subproblema de distribuição, adaptação do problema MIP de Tavares e Nagano (2019), é definido da seguinte forma. As equações relacionadas à variável binária w_{ijr} são:

$$w_{00(r-1)} \leq w_{00r} \qquad \forall\, r > 0 \qquad (4.8)$$

$$\sum_r w_{ijr} \leq 1 \qquad \begin{matrix} \forall\, i > 0 \\ j > 0 \end{matrix} \qquad (4.9)$$

A equação (4.8) assegura que não há rotas vazias entre duas rotas *i* e *j* e a equação (4.9) garante que uma rota entre duas cidades pode ser utilizada no máximo uma única vez por um recurso logístico.

$$\sum_j w_{0jr} = 1 \qquad \forall\, r \qquad (4.10)$$

$$\sum_j w_{j0r} = 1 \qquad \forall\, r \qquad (4.11)$$

$$w_{jjr} = 0 \qquad \begin{matrix} \forall\, j > 0 \\ r \end{matrix} \qquad (4.12)$$

$$\sum_{\mathrm{j},r} w_{ijr} = 1 \qquad \forall\, i > 0 \qquad (4.13)$$

$$w_{00r} + \sum_{j>0} w_{ijr} \leq 1 \qquad \begin{matrix} \forall\, i > 0 \\ r \end{matrix} \qquad (4.14)$$

$$\sum_{h \neq i} w_{ihr} = \sum_{h \neq i} w_{hir} \qquad \begin{matrix} \forall\, i > 0 \\ r \end{matrix} \qquad (4.15)$$

As equações (4.10) e (4.11) permitem que uma rota parta e chegue somente uma vez da origem. A equação (4.12) não permite que um veículo saia e retorne à mesma cidade. A equação (4.13) garante que somente uma rota visite determinada cidade. A equação (4.14) faz com que w_{00r} seja igual a 0 quando a rota r já foi utilizada. A equação (4.15) é utilizada para garantir que o arco que chega ao nó *i* está na mesma rota que o arco que sai do mesmo nó.

Equação relacionada à variável de capacidade contínua A_i:

$$A_j \geq A_i + \sigma j - G\sigma.\left(1 - w_{ijr}\right) \qquad \begin{matrix} \forall\, j > 0 \\ i \neq j \\ r \end{matrix} \qquad (4.16)$$

A equação (4.16) estabelece um limite inferior ao volume do veículo ao chegar no cliente *j*.

As equações finais relacionadas à integração dos subproblemas de produção e distribuição são:

$$\mathrm{R}_r \geq C_i - GP.\left(1 - w_{ijr}\right) \qquad \begin{matrix} \forall\, i > 0 \\ j \\ r \end{matrix} \qquad (4.17)$$

$$D_i \geq R_r + \delta_{0i} - GV.(1 - w_{0ir}) \qquad \begin{matrix} \forall\, i > 0 \\ r \end{matrix} \qquad (4.18)$$

$$D_j \geq D_i + \delta_{ij} - GV.\left(1 - \sum_r w_{ijr}\right) \qquad \begin{matrix} \forall\, i > 0 \\ j \neq i \end{matrix} \qquad (4.19)$$

$$\mathrm{R}_r \geq D_i + \delta_{i0} - GV.\left(1 - w_{i0(r-1)}\right) \qquad \begin{matrix} \forall\, i > 0 \\ r > 0 \end{matrix} \qquad (4.20)$$

A equação (4.17) busca garantir que a rota *r* somente possa ser utilizada se os pedidos dessa rota já foram produzidos. A equação (4.18) define o *start* dos pedidos localizados na primeira posição das rotas, a equação (4.19) é utilizada para definir o tempo de entrega das ordens remanescentes da rota e a equação (4.20) estabelece que a rota *r* só pode iniciar se a rota *r*-1 estiver finalizada.

As equações (4.21) à (4.24) correspondem às restrições para eliminação de sub-rotas de Miller-Tucker-Zemlin (MTZ - MILLER *et al*., 1960).

$$u_0 = 1 \qquad (4.21)$$

$$u_{\mathrm{i}} \geq 2 \qquad \forall\, i > 0 \qquad (4.22)$$

$$u_{\mathrm{i}} \leq \mathrm{i} \qquad \forall\, i > 0 \qquad (4.23)$$

$$u_{\mathrm{i}} - u_{\mathrm{j}} \leq (\mathrm{i} - 1)(1 - w_{\mathrm{ij0}}) \qquad \begin{matrix} \forall\, i > 0 \\ j > 0 \end{matrix} \qquad (4.24)$$

No modelo desacoplado, inicialmente foi resolvido o problema de produção e os valores de C*i* foram utilizados como dado de entrada do problema de distribuição. Ou seja, a minimização do tempo total de fluxo considerou inicialmente somente a produção e este *scheduling* foi utilizado como dados iniciais na soma dos tempos de processamento utilizados no modelo de roteirização. Assim, para reduzir tempo total de fluxo (ou estoque em processo) as tarefas do subproblema de produção foram ordenadas pela regra de prioridade SPT.

A utilização da ordenação SPT no problema de máquina única foi devido ao fato da mesma garantir a obtenção de resultado ótimo da situação desenhada, a qual objetiva a minimização do tempo total de fluxo. Este fato é garantido pela chamada lei de Smith (1956).

Após esta ordenação, foram calculados os tempos de liberação das tarefas, definindo quando estarão prontas para serem enviadas. Feito isso, estes dados foram inseridos como restrições no modelo VRP. Assim, os tempos de liberação das tarefas foram consideradas como dados de entrada para as equações (4.8) à (4.16) do modelo VRP.

Já no problema integrado, os subproblemas de produção e distribuição foram resolvidos no mesmo modelo e a minimização do tempo total de fluxo considerou ambos os subproblemas.

Feito isso, utilizou-se da solução do problema desacoplado como solução inicial do problema integrado e esta abordagem também foi comparada com as demais. O resumo do princípio de cada abordagem dos três modelos está descrito na figura 4.1.

Figura 4.1 Representação das diferenças entre os modelos das três abordagens.

Fonte: Elaborado pelo autor.

A comparação entre as três abordagens consideradas na pesquisa foi conduzida considerando três indicadores principais: tempo de execução dos algoritmos, valores obtidos com as funções objetivo e os valores de *lower bound*.

4.2 INSTÂNCIAS

O problema da pesquisa, contendo as características principais de máquina única e somente um veículo para distribuição, utilizou de dados gerados aleatoriamente para os seus parâmetros. Feito isso, eles eram fixos para as três execuções dos algoritmos (desacoplado, integrado e *mipstart*). Estes parâmetros estão elencados na tabela 4.5.

Tabela 4.5 Relação de parâmetros.

Parâmetros	**Descrição**	***Range***
n	número de tarefas	5;10;20;40
ϱ	duração das tarefas (tempos de processamento)	1-100
σ	tamanho da tarefa	1-10
d	distribuição espacial geográfica	10;20;30
δ	distância entre as cidades	1-10;1-20;1-30
ψ	capacidade do veículo	Máx$\{\sigma_{max}; 5*\sigma_i\}$

Os grupos de números de tarefas se dividiram em 5, 10, 20 e 40 tarefas. Para cada grupo existia três diferentes distribuições espaciais: 10, 20 e 30. Essa distribuição espacial representa a medida que delimita as possíveis distâncias das cidades, ou seja, para uma

quadrado de lado igual a 10 considera-se que o centro deste, é o ponto de distribuição dos produtos. Assim, em cada distribuição espacial, foram geradas 20 instâncias com diferentes tempos de processamento, tamanho (volume) das tarefas, distâncias entre as cidades e a capacidade do veículo. Ou seja, 20 instâncias com três tipos de distribuição geográfica e quatro grupos de tarefas geraram o total de 240 instâncias que foram executadas.

Os algoritmos foram aplicados para 240 instâncias de problemas de produção e distribuição utilizando-se de dados que variavam o número de tarefas, *n*, e, em seguida, a distribuição espacial geográfica para cada *n*.

Para os tempos de processamento foram utilizados números inteiros aleatórios de 1 a 100. O volume dos produtos foi considerado como uma distribuição aleatória de números inteiros de 1 a 10 e a capacidade do veículo utilizado para distribuição também foi aleatório com variação entre um valor numericamente igual ao volume máximo entre todos os produtos ou cinco vezes o mesmo. E, por fim, a distância entre as cidades foi calculada conforme a distância entre dois pontos que variavam aleatoriamente entre 1 e *d*, que representa o tamanho da distribuição espacial.

4.3 EXECUÇÃO

Foram utilizados dois limites para os tempos de execução dos algoritmos. Primeiramente utilizou-se de 2 minutos e, em seguida, 30 minutos.

Os tempos de execução foram divididos de forma que os problemas desacoplado e integrado obtinham o total de limite de 2 ou 30 minutos. Já no método *mipstart* o tempo para resolução do problema integrado que utilizava de uma solução inicial foi reduzido do problema desacoplado. Ou seja, por exemplo, se o tempo de execução do desacoplado foi de 10 minutos, o tempo disponível para o integrado foi de 20 minutos, buscando assim garantir um balanceamento entre os tempos de resolução disponíveis para os três tipos de resolução.

Resumidamente, explica-se como foi considerado o tempo de execução dos algoritmos com limite de tempo *t*:

1. Os modelos desacoplado e integrado foram executados por *t* segundos.

2. No modelo *mipstart*, o desacoplado foi executado por no máximo *t*/2 segundos, e depois o resultado obtido foi utlizado como solução inicial do modelo integrado. Este último rodou por *t*/2 segundos mais um possível tempo adicional que não tenha sido utilizado integralmente na parcela de *t*/2 definida para o problema desacoplado.

5 RESULTADOS

Nas três abordagens MILP's utilizadas na pesquisa buscou-se a minimização do tempo total fluxo de ordens executadas em um ambiente produtivo de máquina única e a distribuição realizada por um único veículo. O problema foi tratado e resolvido de forma não integrada (desacoplada). Em seguida, foi resolvido de forma integrada e, por fim, utilizou-se da solução do problema desacoplado como solução inicial do problema integrado (*mipstart*).

Este trabalho utilizou-se de um software para resolução do modelo de sequenciamento e programação de operações de produção e roteirização. Por ser vastamente utilizado nas pesquisas mais recentes sobre o tema, o software Python 3.7 com a biblioteca IBM ILOG CPLEX Interactive Optimizer 12.10.0.0.

Os modelos foram resolvidos em um computador geração i7 com 8 núcleos, 3.40GHz e 16GB of RAM, com Linux *Mint* 18.3.

A execução dos algoritmos indicou que os mesmos foram eficazes em encontrar soluções para os problemas e os resultados obtidos estão divididos em tempos de execução de 2 minutos e 30 minutos. Algumas análises são realizadas considerando separadamente estes dois tempos limites de execução e outras comparam as mesmas instâncias entre uma execução com 120 segundos e com um tempo maior para resolução.

Após a execução foram analisados os resultados obtidos e, para guiar a análise dos resultados, algumas perguntas principais sobre o desempenho dos modelos foram realizadas.

- ✓ **Pergunta 1:** Em que técnica de resolução não se conseguiu obter uma solução inicial?
- ✓ **Pergunta 2:** Quais os status (*codes*) obtidos por cada modelo?
- ✓ **Pergunta 3:** Como foi o resultado da FO entregue pelos modelos?
- ✓ **Pergunta 4:** Os modelos necessitaram do mesmo tempo computacional para serem executados?
- ✓ **Pergunta 5:** Entre o *mipstart* e o modelo integrado, qual conseguiu um melhor *Lower Bound*?

Tabela 5.1 Encontrou ou não solução inicial para o tempo de 120 segundos e 1.800 segundos.

n	Algoritmo	Conseguiu obter solução inicial (120 segundos)?	Conseguiu obter solução inicial (1.800 segundos)?
5	desacoplado	Sim (60); Não (0)	Sim (60); Não (0)
	integrado	Sim (60); Não (0)	Sim (60); Não (0)
	mipstart	Sim (60); Não (0)	Sim (60); Não (0)
10	desacoplado	Sim (60); Não (0)	Sim (60); Não (0)
	integrado	Sim (60); Não (0)	Sim (60); Não (0)
	mipstart	Sim (60); Não (0)	Sim (60); Não (0)
20	desacoplado	Sim (60); Não (0)	Sim (60); Não (0)
	integrado	Sim (55); Não (5)	Sim (58); Não (2)
	mipstart	Sim (60); Não (0)	Sim (60); Não (0)
40	desacoplado	Sim (0); Não (60)	Sim (3); Não (57)
	integrado	Sim (0); Não (60)	Sim (0); Não (60)
	mipstart	Sim (0); Não (60)	Sim (3); Não (57)

Na tabela 5.1 é possível visualizar que a maioria das instâncias conseguiu gerar alguma solução inicial. O total de instâncias que não geraram resultados foram 65 no tempo de execução de 120 segundos, sendo que, 5 dessas pertenciam ao modelo integrado e o *n* igual à 20. As demais instâncias que não encontraram solução inicial, 60, pertenciam ao grupo de instâncias com o maior número de tarefas, ou seja, 40. Portanto, na execução de 120 segundos, 175 obtiveram soluções representando aproximadamente 73% do total.

Pela tabela 5.1 visualiza-se também que o total de instâncias que encontraram uma solução inicial no tempo de execução de 1.800 segundos foi próximo ao obtido com o tempo limite de 120 segundos. O total de instâncias que não geraram resultados, em pelo menos um dos três modelos analisados, foram 59 e todas pertenciam ao grupo de *n* tarefas iguais à 40. Diferentemente da execução de 120 segundos, a execução de maior tempo encontrou soluções para três instâncias com 40 tarefas nos modelos desacoplado e *mipstart*. Apesar disso, mesmo com um tempo 15 vezes maior que a primeira execução, encontraram-se soluções para aproximadamente 75% das instâncias, apenas 2% mais eficiente que o tempo limite de 120 segundos. Com isso, a primeira pergunta proposta como guia da análise dos resultados na qual questionava se "em que técnica de resolução não se conseguiu obter uma solução inicial?" obtém a resposta afirmativa para ambas as execuções, de menor e maior tempo limite.

Tabela 5.2 Status das soluções obtidas (considerando limite de execução de 120 segundos).

Algoritmo	n	Provou otimalidade (101)	Excedeu o limite de execução (107)
desacoplado	5	60	0
	10	0	60
	20	0	60
	40	0	0
integrado	5	60	0
	10	0	60
	20	0	60
	40	0	0
mipstart	5	60	0
	10	0	60
	20	0	60
	40	0	0

O status 101 indica que se obteve a solução ótima inteira para a função objetivo. O status 107 indica que o tempo limite de execução foi excedido. Da tabela 5.2 é possível inferir que os três modelos atingiram a solução ótima para as instâncias com o número de tarefas, *n*, igual a cinco Todas as instâncias de 10 ou 20 tarefas, dos três modelos, excederam o tempo limite de execução.

Tabela 5.3 Abordagem que encontrou solução ótima (tempo de execução 1.800 segundos).

Algoritmo	n	Provou otimalidade (101)	Provou otimalidade – *gap* de 5%(102)	Excedeu limite de execução (107)	Memória excedida, mas solução encontrada (111)
desacoplado	5	60	0	0	0
	10	0	3	56	1
	20	0	0	59	1
	40	0	0	0	3
integrado	5	60	0	0	0
	10	0	0	60	0
	20	0	0	58	2
	40	0	0	0	0
mipstart	5	60	0	0	0
	10	0	0	60	0
	20	0	0	60	0
	40	0	0	3	0

Analisando as soluções obtidas com os modelos, na tabela 5.3, verificam-se quatro tipos *codes* da biblioteca IBM ILOG CPLEX, 101, 102, 107 e 111. Na execução de 1.800 obteve-se 2 *status* diferentes que não haviam ocorrido na execução de menor tempo. O status 101 representa que a execução encontrou a solução ótima inteira, o que ocorreu em todas as instâncias de n=5. O *code* 102 representa que encontrou a solução ótima dentro de um *gap* de 5% aceitável pelo CPLEX e ocorreu em 3 instâncias de 5 tarefas do modelo desacoplado. O status 107 indica que o tempo limite de execução foi excedido mas solução inteira existe e obteve-se esse *code* para quase todas instâncias de n iguais a 10 e 20. Já o código 111 indica que a memória do computador foi excedida, mas uma solução foi encontrada. Este status foi obtido em 1 instância de 10 tarefas e outra de 20 tarefas do modelo desacoplado. Assim, as tabelas 5.3 e 5.4 respondem a segunda pergunta de análise dos resultados: "Quais os status (*codes*) obtidos por cada modelo".

Tabela 5.4 Valores médios da F.O – tempo limite de 120 segundos.

n	**tamanho dist. espacial (*d*)**	**Desacoplado**	**Integrado**	***Mipstart***
5	10	385	385	385
	20	427	427	427
	30	440	438	438
	Todas instâncias com *n*=5	**417**	**417**	**417**
10	10	1.599	1.736	1.626
	20	1.919	2.043	1.925
	30	1.810	1.963	1.855
	Todas instâncias com *n*=10	**1.776**	**1.914**	**1.802**
20	10	12.809	12.107	11.974
	20	14.362	11.168	12.891
	30	15.218	12.889	13.221
	Todas instâncias com *n*=20	**14.129**	**12.055**	**12.695**

Tabela 5.5 Valores desvio padrão da F.O – tempo limite de 120 segundos.

n	tamanho dist. espacial (*d*)	Desacoplado	Integrado	*Mipstart*
5	10	148	148	148
	20	136	136	136
	30	86	88	88
	Todas instâncias com *n*=5	**123**	**124**	**124**
10	10	605	635	618
	20	522	560	527
	30	524	552	535
	Todas instâncias com *n*=10	**551**	**582**	**560**
20	10	3.044	1.877	2.400
	20	3.087	5.144	2.131
	30	4.043	4.816	2.204
	Todas instâncias com *n*=20	**3.391**	**3.945**	**2.245**

As tabelas 5.4 e 5.5 apresentam respectivamente os valores médios da função objetivo e desvios padrão das soluções obtidas comparando os resultados de mesmo número de tarefas, *n* e divididas entre sua distribuição espacial, *d*. Lembrando que, nesse grupo de tarefas, os tempos de processamento e as distâncias entre as cidades são diferentes entre si, o que explica parcialmente os desvios. Para o número de tarefas iguais a 5, os valores médios de funções objetivo que buscavam a minimização do tempo total de fluxo ficaram iguais entre todos os modelos, na média das distribuições espaciais, com o valor de 417. Para *n*=10, o modelo desacoplado foi o que apresentou o menor valor médio nos resultados, 1.776. Isso pode ter ocorrido porque, considerando o tempo de execução de 120 segundos, como o desacoplado faz uma ordenação das tarefas no subproblema de produção e, em seguida, resolve o subproblema de distribuição, esse fator pode ter representado uma vantagem na busca de uma boa solução dentro do tempo considerado, mesmo que não ótima, como visto na tabela 5.2.

Em instâncias de 10 tarefas, há um menor desvio padrão médio de 551 no modelo desacoplado em comparativo com o de 560 obtido pelo *mipstart* e o valor médio obtido para os resultados foi de 1.776 no desacoplado e de 1.802 no *mipstart*. Em *n* = 40 não se encontrou solução para estas instâncias.

Tabela 5.6 Valores médios da F.O– tempo limite de 1.800 segundos.

n	**tamanho dist. espacial (*d*)**	**Desacoplado**	**Integrado**	***Mipstart***
5	10	385	385	385
	20	427	427	427
	30	440	438	438
	Todas instâncias com *n*=5	**417**	**417**	**417**
10	10	1.553	1.611	1.553
	20	1.853	1.924	1.850
	30	1.764	1.803	1.762
	Todas instâncias com *n*=10	**1.723**	**1.779**	**1.722**
20	10	9.070	9.344	8.214
	20	10.127	9.620	9.028
	30	10.078	10.145	9.132
	Todas instâncias com *n*=20	**9.758**	**9.703**	**8.791**
40	10	62.093	0	58.826
	20	94.444	0	77.380
	30	0	0	0
	Todas instâncias com *n*=40	**78.268**	**0**	**68.103**

A tabela 5.6 apresenta os valores médios da função objetivo para o tempo limite igual à 1.800 segundos.

Tabela 5.7 Valores desvio padrão da F.O– tempo limite de 1.800 segundos.

n	tamanho dist. espacial (*d*)	Desacoplado	Integrado	*Mipstart*
5	10	148	148	148
	20	136	136	136
	30	86	88	88
	Todas instâncias com *n*=5	**123**	**124**	**124**
10	10	581	590	581
	20	493	513	494
	30	511	534	512
	Todas instâncias com *n*=10	**528**	**546**	**529**
20	10	2.027	1.597	1.664
	20	2.183	3.632	1.738
	30	1.519	1.548	1.537
	Todas instâncias com *n*=20	**1.910**	**2.259**	**1.646**
40	10	18.032	0	16.088
	20	0	148	0
	30	0	136	0
	Todas instâncias com *n*=40	**9.016**	**0**	**8.044**

A tabela 5.7 apresenta os valores médios dos desvios padrão das soluções obtidas para o tempo limite igual à 1.800 segundos. Para o número de tarefas iguais a 10, os valores médios de funções objetivo ficaram próximos entre si para todos os modelos, resultado melhor que o obtido na execução de 120 segundos. Para *n*=20, o modelo *mipstart* foi o que apresentou o menor valor médio da função objetivo nos resultados, 8.791.

Em instâncias de 40 tarefas, o desvio padrão obteve valor muito expressivo nos modelos desacoplado e *mipstart* porque somente três soluções de cada modelo foram encontradas e a dispersão entre elas foi considerável. O desvio padrão na distribuição espacial de volume 20 foi igual à zero pois somente uma solução foi encontrada neste grupo.

Tabela 5.8 Desvio relativo da solução (*GAP*) – limite de 120 segundos.

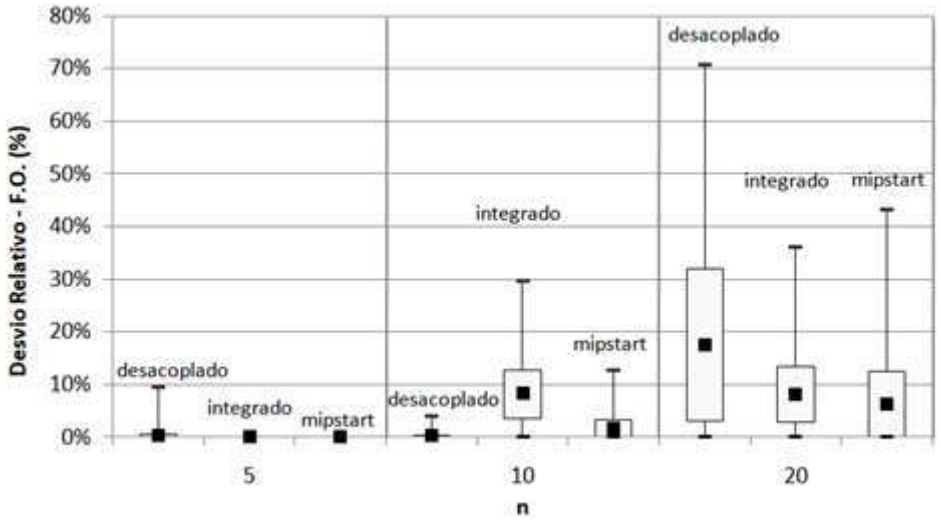
n	parâmetro	desacoplado	integrado	*mipstart*
5	máximo	9,34%	0,00%	0,00%
	Q3	0,48%	0,00%	0,00%
	média	0,24%	0,00%	0,00%
	mediana	0,00%	0,00%	0,00%
	Q1	0,00%	0,00%	0,00%
	mínimo	0,00%	0,00%	0,00%
10	máximo	3,82%	29,68%	12,55%
	Q3	0,35%	12,82%	3,27%
	média	0,18%	8,23%	1,64%
	mediana	0,00%	7,09%	0,00%
	Q1	0,00%	3,64%	0,00%
	mínimo	0,00%	0,00%	0,00%
20	máximo	70,77%	36,15%	43,31%
	Q3	32,05%	13,50%	12,46%
	média	17,53%	8,17%	6,24%
	mediana	10,36%	6,33%	0,47%
	Q1	3,02%	2,84%	0,02%
	mínimo	0,00%	0,00%	0,00%

Com a tabela 5.8 é calculado a dispersão relativa dos resultados em comparação à menor solução obtida. Na tabela é possível observar que, para o número de instâncias igual a cinco tarefas, ao se calcular a média do desvio relativo, estas são muito próximos ou iguais a zero, ou seja, todas as médias dos valores obtidos nas funções objetivo dos três modelos obtiveram resultados iguais por aproximação, como visto na tabela 5.6, representado aproximadamente 417,0 e também e, como visto na tabela 5.2, todas as instâncias de *n* iguais à 5 atingiram a sua solução ótima. É importante observar que, apesar de pouco considerável, há um desvio médio de 0,24% nas soluções do modelo desacoplado em relação à ótima possível e o que é mais interessante nessa observação é a visualização do fato de que o resultado ótimo para o modelo desacoplado não é, necessariamente, a mesma solução ótima do modelo integrado. Já para o número de instâncias igual a 10 observa-se que os modelos desacoplado e *mipstart* obtiveram um desempenho superior se comparado ao integrado, com melhor resultado médio obtido no modelo desacoplado.

Ao analisar as soluções de *n* igual 10, é possível observar que a solução inicial desacoplada representou um efeito positivo para estes casos e fez uma diferença de cerca de 6,59% de melhoria média nos valores obtidos para os resultados no *mipstart* em relação aos do

modelo integrado. Para este mesmo número de tarefas, o desacoplado obteve melhor resultado e superou o modelo *mipstart* em aproximadamente 1,5%. O *mipstart* utiliza como ponto de partida uma solução inicial do desacoplado e este último pode apresentar um resultado final mais otimizado que o modelo *mipstart* devido ao tempo limite de execução. Ou seja, no modelo desacoplado está definido um tempo *t* para executar e encontrar uma determinada solução, no modelo *mipstart* ele possui o tempo limite de *t*/2 para executar o desacoplado e obter a solução inicial antes de partir para o próximo passo. O restante do tempo é utilizado para executar o modelo integrado. Assim, considerando este tempo limite de 120 segundos, o desacoplado pode apresentar soluções com valores de função objetivo menores que o modelo *mipstart*, portanto, podem obter soluções melhores.

Gráfico 5.1 Desvio relativo (*GAP*) da solução – limite de execução 120 segundos.



A tabela 5.7 foi utilizada como base para a construção do gráfico *box-plot* do desvio relativo para o valor mínimo de solução para as suas respectivas instâncias de 2 minutos, gráfico 5.1. Ou seja, compara a solução ao menor valor obtido para a função objetivo da mesma instância entre os três modelos.

O gráfico 5.1 demonstra o desvio relativo com base em diferentes valores de *n*, número de tarefas. Pelo gráfico podemos observar que há um grande desvio das soluções do modelo desacoplado em comparativo com os outros dois em instâncias de *n* maiores, no caso, *n* igual à 20.

No modelo desacoplado os desvios relativos para *n* igual a 5 e 10 foram de respectivamente, 0,24% e 0,18%. Em *n* igual a 20 esse número de desvio aumentou consideravelmente para 17,53% em comparação com os valores de 8,17% do modelo integrado

e 6,24% do modelo *mipstart*. Com isso, é possível observar que, para instâncias maiores, o modelo *mipstart* representou uma melhoria de 11,29% nos resultados médios obtidos pela F.O em relação ao modelo desacoplado e de 1,93% em relação ao modelo integrado.

Tabela 5.9 Desvio relativo da solução (*GAP*) – limite de 1.800 segundos.

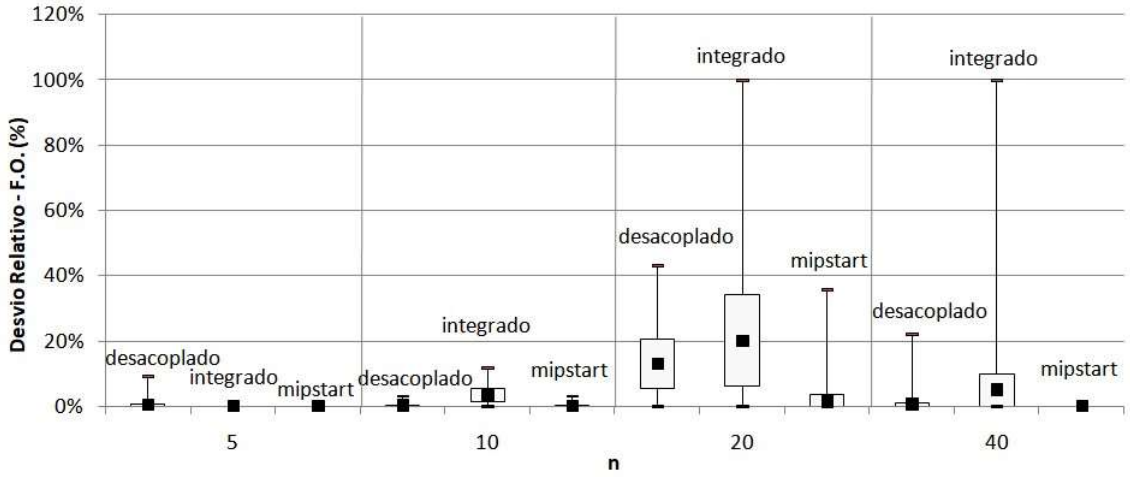
n	parâmetro	desacoplado	integrado	*mipstart*
5	máximo	9,34%	0,00%	0,00%
	Q3	0,48%	0,00%	0,00%
	média	0,24%	0,00%	0,00%
	mediana	0,00%	0,00%	0,00%
	Q1	0,00%	0,00%	0,00%
	mínimo	0,00%	0,00%	0,00%
10	máximo	3,14%	11,83%	3,14%
	Q3	0,31%	5,37%	0,10%
	média	0,15%	3,40%	0,05%
	mediana	0,00%	3,39%	0,00%
	Q1	0,00%	1,43%	0,00%
	mínimo	0,00%	0,00%	0,00%
20	máximo	43,15%	100,00%	35,83%
	Q3	20,41%	34,12%	3,64%
	média	12,97%	20,06%	1,82%
	mediana	12,09%	16,81%	0,00%
	Q1	5,53%	6,01%	0,00%
	mínimo	0,00%	0,00%	0,00%
40	máximo	22,05%	100,00%	0,00%
	Q3	1,09%	10,00%	0,00%
	média	0,54%	5,00%	0,00%
	mediana	0,00%	0,00%	0,00%
	Q1	0,00%	0,00%	0,00%
	mínimo	0,00%	0,00%	0,00%

Na tabela 5.9 verifica-se a dispersão relativa dos resultados em comparação à menor solução encontrada. Para o número de instâncias igual a cinco tarefas, os desvios foram muito próximos ou iguais a zero, como visto na tabela 5.8. Em comparação com a execução de limite igual a 120 segundos, para o número de instâncias de 10 tarefas, observa-se que os modelos integrado e *mipstart* obtiveram um desempenho consideravelmente superior em um tempo de 30 minutos. O modelo integrado obteve aproximadamente 8% de desvio do menor valor obtido da função objetivo no tempo de 2 minutos e de 3,4% de desvio na segunda execução. O modelo *mipstart* obteve 1,64% de desvio na média da primeira execução e 0,05%

de desvio no tempo de 1.800 segundos. Além disso, ainda considerando instâncias de 10 tarefas, o desacoplado obteve desempenho superior para o menor tempo de execução, mas ao considerar 30 minutos, apesar de uma pequena diferença, o *mipstart* obteve um melhor resultado. No maior tempo, o desacoplado atingiu um desvio médio de 0,15%.

Ao analisar as soluções de *n* igual 20, o modelo que obteve pior desempenho foi o integrado. O valor médio do desvio foi de cerca de 20%. O menor desvio foi do modelo *mipstart*. Para o grupo de 40 tarefas, considerando este tempo limite de 1.800 segundos, o *mipstart* também obteve o menor desvio, sendo igual à zero, mas para estas instâncias somente três das 60 possíveis encontraram solução.

Gráfico 5.2 Desvio relativo (*GAP*) da solução – limite de execução 1.800 segundos.



A tabela 5.8 foi a base para construir o gráfico 5.2 *box-plot* do desvio relativo da solução e este apresenta os desvios relativos com base em diferentes valores de *n*, número de tarefas. Pelo gráfico é possível verificar que o maior desvio das soluções foi obtido no modelo integrado para *n* igual à 20.

Outra observação importante é em relação ao desempenho do *mipstart*. Para *n* igual a 5, ele e o modelo integrado obtiveram, juntamente, o melhor desempenho e atingiram a solução ótima para todas instâncias. Além disso, no tempo de execução de 1.800 segundos, o *mipstart* obteve o melhor desempenho em todas as demais instâncias, 10, 20 e 40 tarefas. Em *n* igual 20 os desvios relativos médios do modelo desacoplado, integrado e *mipstart* foram de respectivamente, 12,97%, 20,06% e 1,82%. Assim, para instâncias maiores, o modelo *mipstart* representou uma melhoria média de 11,15% nos desvios relativos da função objetivo em relação ao segundo melhor. O valor médio obtido de melhoria foi consideravelmente superior ao obtido na execução de 120 segundos, comparado ao modelo integrado.

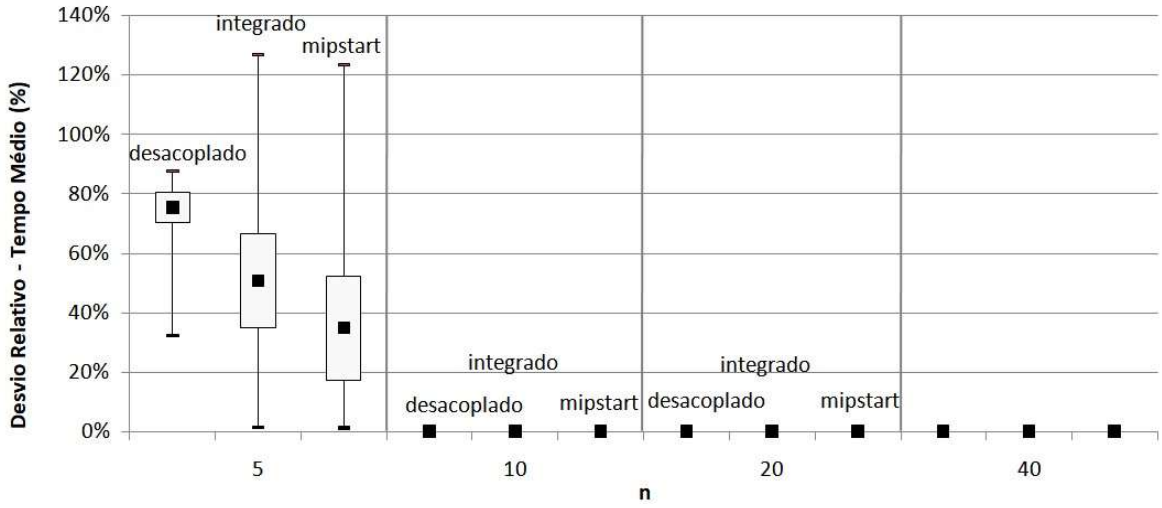
Tabela 5.10 Tempo médio de execução e desvio padrão – 120 segundos.

N	desacoplado		integrado		*mipstart*	
	tempo	D.P	tempo	D.P	tempo	D.P
5	0,04	0,01	0,28	0,09	0,24	0,06
10	120,06	0,12	120,03	0,06	120,01	0,02
20	120,02	0,02	120,05	0,06	120,05	0,05

A tabela 5.10 demonstra os resultados de tempos médios de execução para o limite de 120 segundos e o seus respectivos desvios padrão. Para o cálculo do desvio foram considerados os tempos médios de execução dos modelos nas instâncias com número de tarefas, *n*, iguais e este representa a dispersão dos tempos obtidos para o grupo. Para os três modelos executados e com *n* igual a cinco tarefas, os tempos médios foram abaixo de 1,0 segundo para cada instância. Já os demais modelos atingiram o limite de 120 segundos nas demais execuções. É possível visualizar que não houve desvio padrão considerável dentre todos os resultados obtidos sendo que o maior valor atingido foi de 0,12.

O maior desvio padrão de tempos de resolução ocorreu para o desacoplado e para instâncias com 10 tarefas. Um fator para explicar o desvio padrão sendo maior para tarefas iguais a 10 é de que, mesmo considerando tempos baixos de execução, o desacoplado atingiu um tempo médio de 120,06 segundos, comparados com 120,03 do integrado e 120,01 do *mipstart*.

Gráfico 5.3 Desvio relativo dos tempos computacionais – limite de 120 segundos.



O desvio relativo médio do tempo, no limite de 120 segundos, está descrito no gráfico 5.3 e este representa o *gap* entre os dados de tempo obtidos pelo modelo analisado e as

médias dos tempos dos três modelos considerando o mesmo grupo de número de tarefas, *n*. É importante analisar o gráfico 5.4 juntamente com a tabela 5.9, onde verificamos que os tempos de execução estão próximos entre os modelos de mesmo número de tarefas, pois o desvio padrão relativo pode representar uma porcentagem considerável se o intervalo de tempo analisado for pequeno, mas em número absoluto não é tão relevante. É o que ocorre, por exemplo, para *n* igual à 5 no modelo integrado, todos os valores ficaram abaixo de 1 segundo, mas houve uma diferença média de aproximadamente 50% entre 0 e 1 segundo do modelo integrado em relação às médias dos tempos médios de execução dos três modelos. Em valor absoluto, o modelo integrado obteve média de execução de 0,28 segundos e a média de todos os modelos foi de 0,19 segundos.

Tabela 5.11 Tempo médio de execução e desvio padrão – 1.800 segundos.

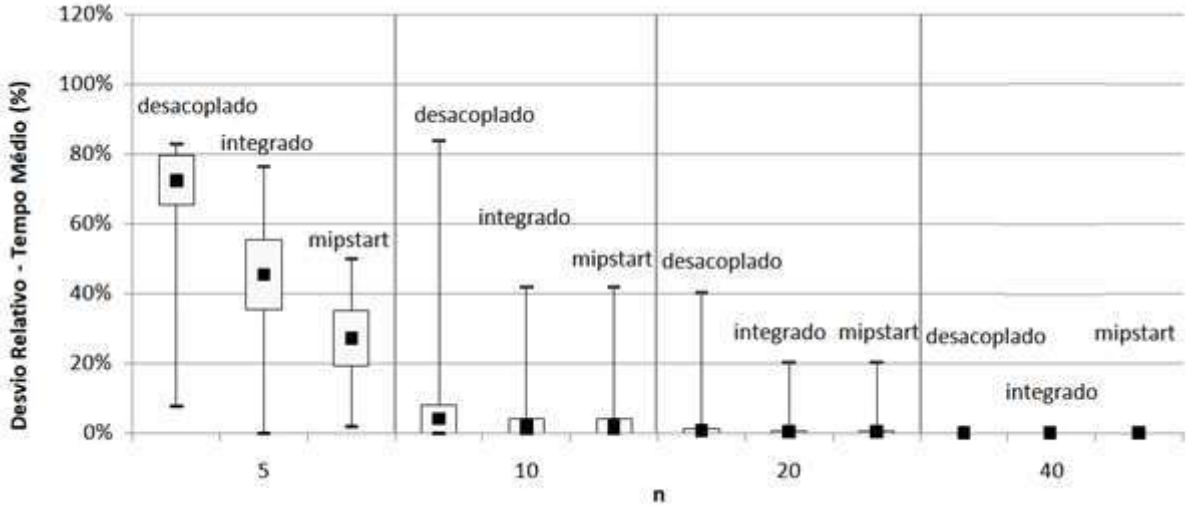
n	desacoplado		integrado		*mipstart*	
	tempo	D.P	tempo	D.P	tempo	D.P
5	0,05	0,02	0,27	0,07	0,23	0,05
10	1.718,09	323,26	1.800,71	0,58	1.800,45	0,36
20	1.784,92	117,29	1.800,08	0,08	1.800,10	0,11
40	1.800,04	0,03	1.800,00	0,00	1.800,21	0,27

A tabela 5.11 apresenta os resultados de tempos médios de execução para o limite de 1.800 segundos e seus respectivos desvios padrão. Como esperado, igualmente ao tempo limite de 120 segundos, para os três modelos executados e com *n* igual a cinco tarefas, os tempos médios permaneceram abaixo de 1,0 segundo.

É possível observar que os maiores desvios padrão dentre todos os resultados obtidos ocorreu para as instâncias de *n* iguais a 10 e 20 tarefas do modelo desacoplado e os valores atingidos foram de 323,26 e 117,29, respectivamente. Isso ocorreu porque alguns tempos obtidos para este modelo ficaram abaixo 1.800 segundos, tempo limite, assim, as médias dos tempos foram de respectivamente, 1.718,09 e 1.784,92.

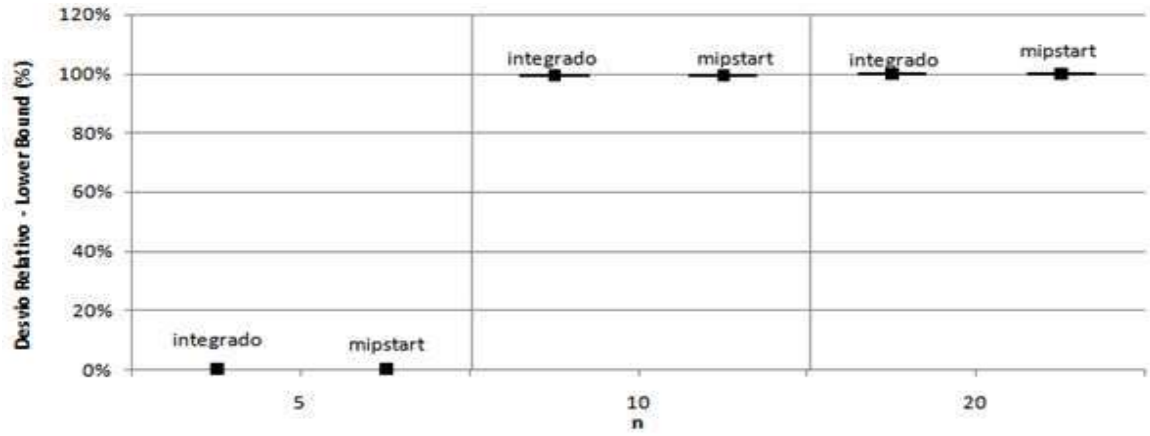
O tempo médio para instâncias de 40 tarefas atingiu o limite permitido, mas encontrou soluções para três iterações dos problemas desacoplado e *mipstart*, exceto para o modelo integrado que não encontrou nenhuma solução para este grupo de instâncias.

Gráfico 5.4 Desvio relativo dos tempos computacionais – limite de 1.800 segundos.



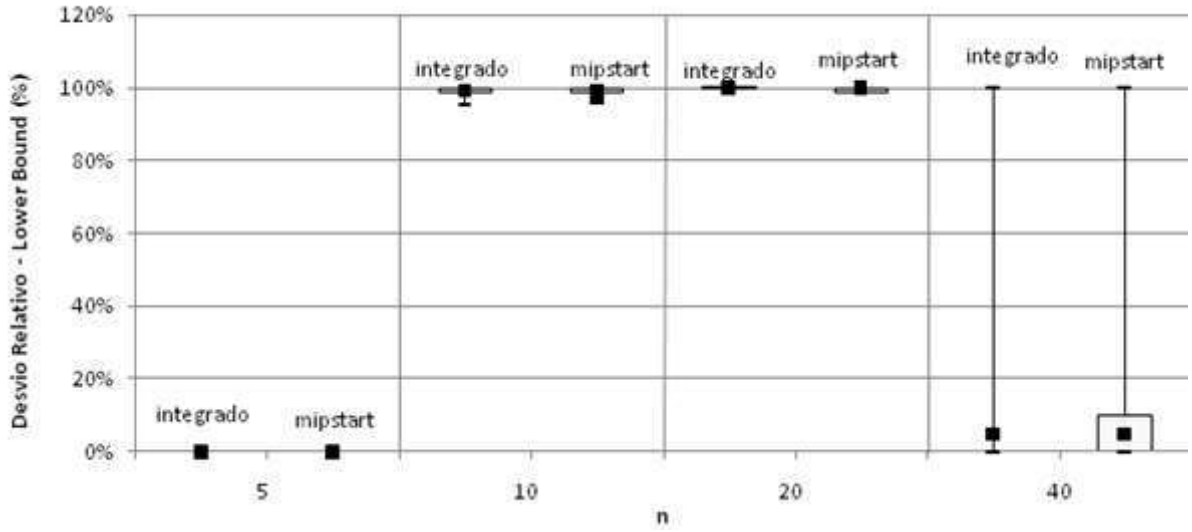
O desvio relativo médio do tempo, no limite de 1.800 segundos, está descrito no gráfico 5.4 e este representa a relação entre os dados de tempo obtidos por um respectivo modelo e as médias dos tempos dos três modelos considerando tarefas de quantidades iguais. Como verificado no tempo limite de 120 segundos, para *n* igual à 5 tarefas, o desvio é maior porque o intervalo considerado de tempos médio ficou entre 0 e 1 segundo. Assim, houve uma diferença média de aproximadamente 72% do modelo desacoplado em relação às médias dos tempos de execução de todos os modelos. Em valor absoluto, o modelo desacoplado obteve média de execução de 0,05 segundos. Para *n* igual à 40 os desvios de tempo são quase iguais pois todos atingiram o limite de 1.800 segundos. Com o comparativo das execuções de 120 e 1.800 segundos é possível afirmar que os modelos necessitaram de aproximadamente o mesmo tempo computacional para serem executados, em instâncias de mesmo número de tarefas, respondendo a quarta pergunta de guia da análise.

Gráfico 5.5 Desvio relativo (*GAP*) do *LB* – limite de execução 120 segundos.



O gráfico 5.5 demonstra o desvio relativo do limitante inferior (do inglês, *Lower Bound* - LB) comparando os modelos integrado e *mipstart*. Ao crescer os valores de *n*, número de tarefas, o desvio do LB é muito maior e próximo de 100% de desvio. O *lower bound* do método de busca do *branch and bound* pode ser definido como uma estimativa do menor valor possível que pode ser atingido por uma solução, assim, o LB então é sempre menor ou igual ao valor da melhor solução possível. O LB permite que algoritmo não explore certos nós, tendo a certeza de que não haverá perda de soluções melhores que as já encontradas até o momento.

Gráfico 5.6 Desvio relativo (*GAP*) do *LB* – limite de execução 1.800 segundos.



Pelo gráfico 5.6 é possível observar que mesmo com um tempo maior de execução o desempenho do LB ainda foi ruim para instâncias de *n* maiores.

Como a análise do *Lower Bound* é possível ser feita entre os problemas *mipstart* e o integrado, a resposta para a última pergunta de guia da análise dos resultados: "entre o *mipstart* e o integrado, qual conseguiu um melhor *Lower Bound*?" é a de que ambos obtiveram desempenhos praticamente iguais e próximos a zero em *n* iguais a 5, e próximos de 100% para *n* iguais a 10 ou 20.

6 CONSIDERAÇÕES FINAIS

Como já citado no trabalho, devido aos possíveis ganhos reais atrelado ao estudo de diferentes abordagens para os problemas de programação e sequenciamento de operações de produção e distribuição, foi visto que há interesse na busca de se obter soluções satisfatórias para determinado problema ou até mesmo obter uma solução ótima, sendo que esta não exija um tempo computacional de resolução muito elevado.

A razão para evolução dessas pesquisas tem principal origem no fato de que, ao longo dos anos, há o constante avanço da tecnologia, por exemplo, a automação de processos e melhoria no rastreamento de veículos e também os mercados se tornam cada vez mais competitivos, com isso, as empresas buscam utilizar seus recursos de forma eficiente para, principalmente, reduzir custos e fortalecer os níveis de serviço ao cliente (CHEN, 2004; DÍAZ-MADROÑERO *et al*., 2015).

Focando no aprimoramento de técnicas de resolução desta classe de problemas, o presente estudo explorou uma pesquisa no tema de sistemas integrados de produção-distribuição na qual o objetivo central foi analisar um comparativo entre resoluções de subproblemas produção e distribuição de um sistema desacoplado, um modelo integrado e um modelo integrado utilizando de solução inicial proveniente de uma abordagem desacoplada. Para cumprir esse objetivo, inicialmente uma análise da literatura forneceu bases para a especificação do problema e para os modelos matemáticos implementados.

Os modelos matemáticos foram executados em 240 instâncias e com dois tempos limites de execução e os resultados obtidos foram analisados e comparados para derivar conclusões acerca do objetivo do trabalho.

Uma das primeiras observações da resolução dos algoritmos foi de que, considerando um total de 240 instâncias, para o número de tarefas iguais à 5, 10 e 20, a maioria delas conseguiu gerar alguma solução inicial. O número de instâncias, dentre o total de 240, que não geraram resultados foram 65 no tempo de execução de 120 segundos sendo que 60 destas pertenciam à número de tarefas iguais à 40. Para o número de tarefas iguais à 40, na execução de 120 segundos, nenhuma execução obteve solução, mas ao aumentar o tempo de solução para 1.800 segundos, os resultados obtidos foram muito próximos ao anterior, somente 6 instâncias geraram resultados, sendo que 3 destas pertenciam ao problema desacoplado e o restante do modelo *mipstart*. Apesar do tempo comparado ser 15 vezes maior ao da primeira

execução, o número absoluto de tempo ainda é de 30 minutos e, com isso, não pode se afirmar sobre efetiva limitação do mesmo quanto a este parâmetro de *n* igual à 40.

Uma observação importante é em relação ao desempenho dos modelos desacoplado, integrado e *mipstart* que, em *n* igual a 5, obtiveram o melhor desempenho e atingiram a solução ótima para praticamente todas instâncias. Para *n* igual a 10 no tempo de 120 segundos, mesmo que muito próximo do resultado do *mipstart*, o modelo desacoplado atingiu o melhor resultado. Em *n* igual à 20 tarefas, e mesmo tempo de execução, o resultado do desvio relativo da F.O foi menor no *mipstart.* No tempo de execução de 1.800 segundos, o *mipstart* obteve o melhor desempenho em todas as demais instâncias, 10, 20 e 40 tarefas. O valor médio obtido de melhoria foi consideravelmente superior ao obtido na execução de 120 segundos, se comparado ao segundo menor desvio da F.O. Na execução de menor tempo a melhoria obtida foi de 1,93% e de 11,15% na execução de maior tempo.

Assim, as principais conclusões do trabalho são de que o modelo *mipstart*, com solução inicial de um problema desacoplado aplicada a um modelo integrado, obteve desempenho superior se comparado aos outros dois. Para instâncias de maiores e mesmo *n*, número de tarefas, no menor tempo de execução, 120 segundos, o modelo *mipstart* representou uma melhoria média de 11,29% nos resultados médios obtidos pela F.O em relação ao modelo desacoplado e de 1,93% em relação ao modelo integrado. Já no limite de tempo estabelecido de 30 minutos, também nas instâncias de maior *n*, o *mipstart* foi superior em aproximadamente 18,24% em relação ao problema desacoplado e 11,15% se comparado ao algoritmo integrado que não utiliza de uma solução inicial de modelo desacoplado como partida, o que é um resultado muito eficiente considerando-se que se trata de um modelo MIP e que o tempo total de execução foi relativamente baixo mesmo considerando o maior tempo limite de execução, de 1.800 segundos.

Através dos resultados também foi possível perceber que, embora o modelo desacoplado não necessariamente gere o resultado ótimo, nos problemas simples muitos dos resultados ótimos encontrados são próximos aos valores do modelo integrado. Mais que isso, ao aumentarmos o tamanho do problema, o modelo integrado se torna inviável, e a única abordagem capaz de gerar soluções, mesmo que sem garantia de otimalidade, é o modelo desacoplado.

Uma sugestão de pesquisa futura seria a utilização de outras formas de resolução e compará-las com os resultados obtidos com o MIP. Adicionalmente, entende-se que tal problema explorado na atual pesquisa tem um grande potencial para ser explorado através de

técnicas heurísticas e meta-heurísticas. Além disso, seria interessante analisar o desempenho do *Lower Bound* comparando-os com de outros possíveis métodos de resolução para o mesmo problema, e se este pode ou não ter afetado consideravelmente a eficácia das soluções obtidas. Em instâncias de n=5, o desvio relativo do LB ficou muito próximo de zero, mas para instâncias de 10, 20 e 40, os desvios ficaram muito próximo de 100%.

Outra possível vertente da pesquisa seria adaptá-la a um estudo de caso, verificar se as técnicas aplicadas fornecem resultados satisfatórios e analisar se há ganhos consideráveis de uma solução inicial desacoplada em um problema integrado de produção e distribuição para determinada situação.

Também como pesquisas futuras, sugere-se expandir as análises para outros tipos de fluxos produtivos que sejam mais complexos, como por exemplo, o *flow shop* e verificar se os resultados são diferentes dos obtidos com máquina única. Além disso, como visto no capítulo de revisão bibliográfica, é possível aplicar outras características ao modelo distribuição para que se aproxime mais da realidade das empresas, como por exemplo, múltiplos veículos com capacidades variadas.

REFERÊNCIAS BIBLIOGRÁFICAS

Al-DULAIMI B.F.; ALI H.A. **Enhanced Traveling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA).** Proceedings of World Academy of Science Engineering and Technology, v.38, p.296-302, 2008.

BAKER K. R. **Introduction to Sequencing and Scheduling**, Wiley, New York. K. R., 1974.

BAKER, K.R.; TRIETSCH, D.: **Principles of Sequencing and Scheduling.** John Wiley and Sons, New York, 2009.

BALLOU, R. H. **Logística Empresarial:** transportes, administração de materiais e distribuição física. São Paulo: Atlas, 1993.

BALLOU, R. H. **Gerenciamento da Cadeia de Suprimentos:** Logística Empresarial. Bookman editora, 2009.

BERTRAND J. W. M.; FRANSOO, J. C. **Operations management research methodologies using quantitative modeling**. International Journal of Operations & Production Management, v. 22, n.2, p. 241-264, 2002.

BODIN, L. D. **Twenty Years of Routing and Scheduling**. Operations Research, v. 38, p.571-579, 1990. Disponível em: <https://doi.org/10.1287/opre.38.4.571>.

BOWERSOX, J. D.; CLOSS J. D.; COOPER, B. M.; BOWERSOX, J. **Gestão logística da cadeia de suprimentos**. AMGH Editora, 2013.

BRAH, S. A.; LOO, L. L. **Heuristics for scheduling in a flow shop with multiple processors**. European Journal of Operational Research, v.113, n.1, p. 113–122, 1999. <https://doi.org/10.1016/s0377-2217(97)00423-2>.

BRYMAN, A. **Research methods and organization studies (contemporary social research)**. London: Routledge, v. 1, 1989.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M. L. **A Heuristic Algorithm for the n Job, m Machine Sequencing Problem**. Management Science, v.16, n.10, p.630–637, 1970. <https://doi:10.1287/mnsc.16.10.b630>.

CAUCHICK, P. A.; FLEURY A.; PEREIRA C. H.; NAKANO D. N.; LIMA E. P.; TURRIONI J. B.; LEE L; MORABITO R.; MARTINS R. A.; SOUSA R.; COSTA S. E. G.; PUREZA V. **Metodologia de pesquisa para engenharia de produção e gestão de operações.** Rio de Janeiro: Elsevier: ABEPRO, 2012.

CHANG, Y.C.; LEE, C. Y. **Machine scheduling with job delivery coordination**. European Journal of Operational Research, v. 158, p. 470–487, 2004.

CHEN, Z. L. Integrated production and distribution operations: taxonomy, models, and review, 2004. *In*: SIMCHI-LEVI, WU, S. D.; SHEN, Z.J. **Handbook of quantitative supply chain analysis:** modeling in the e-business era. Kluwer Academic Publishers, Boston, p. 711–745, 2004.

CHEN, Z. L. **Integrated production and outbound distribution scheduling**: review and extensions. Operations Research, v.58, p.30–148, 2010. <http://dx.doi.org/10.1287/opre.1080.0688>.

CHENG, B.; LEUNG, J.; LI, K. **Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method.** Computers & Industrial Engineering, v. 83, p. 217–225, 2015.

CHENG, B.; LEUNG, J; LI, K. **Integrated scheduling on a batch machine to minimize production, inventory and distribution costs**. European Journal of Operational Research, v.258, p. 104–112, 2017.

CLARKE, G.; WRIGHT, J. W. **Scheduling of Vehicles from a Central Depot to a Number of Delivery Points**. Operations Research, v.12, n. 4, p. 568-581, 1964.

COOK, S. A. **The complexity of theorem-proving procedures**. Symposium on Theory of Computing (STOC), ed. 3, 1971.

DAKIN, R. J. **A tree-search algorithm for mixed integer programming problems**. The Computer Journal, v.8, n.3, p.250–255, 1965.

DANTZIG G. B.; FULKERSON D. R.; JOHNSON S. M. **Solution of a Large-Scale Traveling-Salesman Problem**, Operations Research, v. 2, p. 393–410, 1954.

DANTZIG, G. B.; RAMSER, J. H. **The truck dispatching problem**. Management Science, v.6, v.1, p. 80–91, 1959.

DESROSIERS, J.; DUMAS, Y., SOLOMON, M.; SOUMIS, F. **Time constrained routing and scheduling**. Handbooks in operations research and management science, v.8, 1995.

DEVAPRIYA, P.; FERRELL, W.; GEISMAR, N**. Integrated production and distribution scheduling with a perishable product**. European Journal of Operational Research, v. 259, p.906–916, 2017.

DÍAZ-MADROÑERO, M.; PEIDRO, D.; MULA, J. **A review of tactical optimization models for integrated production and transport routing planning decisions**. Computers & Industrial Engineering, v.88, p.518–535, 2015.

DIJKSTRA, E. W. **A note on two problems in connexion with graphs**. Numerische Mathematik, 1959. In: CORMEN, et al. **Introduction to Algorithms**. MIT Press, v.3, 2010.

DUTRA, F. A. F.; ERDMANN, R. H. **Uma nova abordagem para o estudo do planejamento e controle da produção (PCP)**: a ótica da teoria da complexidade. Revista GEPROS [online], v.1, p.195, 2006.

EHM, J.; FREITAG M. **The Benefit of Integrating Production and Transport Scheduling**. Procedia CIRP. v.41, p. 585–590, 2016.

EHRGOTT, M. **A discussion of scalarization techniques for multiple objective integer programming**. Annals of Operations Research, Springer, v. 147, n. 1, p. 343–360, 2006.

ERENGUC, S.S.; SIMPSON, N.C.; VAKHARAI, A.J. **Integrated production/distribution planning in supply chains**: an invited review, European Journal of Operational Research 115, p. 219–236, 1999.

FERRIS, C. M.; LIU, Y. **Modelling demand response in organized wholesale energy markets**. Optimization Methods and Software. v.31, n.5, p.1064-1088, 2016.

FISCHETTI, M.; SARTOR, G.; ZANETTE, A. **MIP-and-refine matheuristic for smart grid energy management**. International Transactions in Operational Research. v.2, 2013. <http://dx.doi.org/10.1111/itor.12034>

FLOOD, M. M. **The travelling salesman problem**. Operations Research Journal, v.4, p. 61-75, 1955.

FUCHIGAMI, H. Y.; MOCCELLIN, J. V. **Influência da relação entre os tempos de processamento e de *setup* em *flow shop* híbridos**. RevistaPODes, v.6, n.2, p.318-331, 2014**.**

FUMERO, F.; VERCELLIS, C. **Synchronized development of production, inventory and distribution schedules**. Transportation Science, v.33, n.3, p.330–340, 1999.

FUMERO, Y.; CORSANO, G.; MONTAGNA, J.M. **A mixed integer linear programming model for simultaneous design and scheduling of flowshop plants.** Applied Mathematical Modelling, v. 37, p. 1652-1664, 2013.

GAREY M. R.; JOHNSON D. S.; STOCKMEYER L. **Some Simplified NP-Complete Graph problems**, Theoretical Computer Science, v.1, n.3, 1976.

GEISMAR, H. N.; LAPORTE G.; LEI, L.; SRISKANDARAJAH C. **The integrated production and transportation scheduling problem for a product with a short lifespan**, Informs Journal on Computing, v.20, n.1, 2008.

GELAREH, S., GENDRON, B., HANAFI, S. et al. **The selective traveling salesman problem with draft limits**. Journal of Heuristics, v.26, p.339–352, 2020.

GLEIXNER, A. M.; STEFFY, D. E.; WOLTER, K. **Iterative Refinement for Linear Programming**. INFORMS Journal on Computing, v.28, ed.3, p.449–464, 2016. <http://dx.doi.org/10.1287/ijoc.2016.0692>.

GRAHAM, R. L. **Combinatorial Scheduling Theory**. Mathematics Today Twelve Informal Essays. Springer, NY, 1978.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**.São Paulo: Atlas, v.6, 2008.

GILLETT, B. E.; MILLER, L. R. **A Heuristic Algorithm for the Vehicle-Dispatch Problem**. Operations Research, v.22, n.2, p.340–349, 1974. <http://dx.doi.org/10.1287/opre.22.2.340>.

GOMES, C. F. S. **Gestão da Cadeia de Suprimentos integrada à Tecnologia da Informação.** Cengage Learning Editores, 2004.

GRAVES, S.C. **A review of production scheduling**. Operations Research, v.29, n.4, p.646-675, 1981.

GUPTA, J. N. D.; GULATI, N.; SHARMA, S.; SINGLA, P. **Optimal two stage flow shop Scheduling to minimize the rental Cost including Job Block Criteria, set up times and processing times associated with probabilities.** European Journal of Business and Management [online], v.3, n.3, p. 85-103, 2011.

HADDADZADE M.; RAZFAR M. R.; ZARANDI M. H. F. **Integration of process planning and job shop scheduling with stochastic processing time**. The International Journal of Advanced Manufacturing Technology, v. 71, n. 1–4, p. 241–252, 2014.

HAO, J.; CAO, L.; JIANG, D. **Integrated Production-Distribution Scheduling Problem with Multiple Independent Manufacturers**. Hindawi Publishing Corporation - Mathematical Problems in Engineering, v. 15, 2015.

HURTER, A.P.; VAN BUER, M.G. **The Newspaper Production/Distribution Problem.** Journal of Business Logistics, v.17, p. 85–107, 1996.

JOHNSON, S. M. **Optimal two- and three-stage production schedules with setup times included**. Naval Research Logistics Quarterly, v. 1, n.1, p. 61-68, 1954.

KARP, R. M. **Reducibility among Combinatorial Problems. Complexity of Computer Computations**, Plenum Press, New York, p. 85–103, 1972.

KHALILI, M.; TAVAKKOLI-MOGHADDAM, R. **A multi-objective electromagnetism algorithm for a bi-objective flowshop scheduling problem**. Journal of Manufacturing Systems, v.31, n.2, p.232–239, 2012. <http://dx.doi:10.1016/j.jmsy.2011.08.002>

KIM, S. H.; BAE, S. H. **Optimal solution to the vehicle routing problem by adopting a meta-heuristic algorithm**. Transportation Planning and Technology, v.39, n.6, p. 574–585, 2016. <http://doi:10.1080/03081060.2016.1187808>

LABADIE, N.; PRINS, C. **Vehicle Routing Nowadays:** Compact Review and Emerging Problems. Production Systems and Supply Chain Management in Emerging Countries: Best Practices, p. 141–166, 2012. <http://dx.doi.org/10.1007/978-3-642-26004-9_8>

LENSTRA, J. K.; RINNOOY, A. H. G. **Computational Complexity of Discrete Optimization Problems.** Annals of Discrete Mathematics, p. 121–140, 1979. <http://dx.doi.org/10.1016/s0167-5060(08)70821-5>

LI, K., ZHANG, X.; LEUNG, J. Y. T.; E YANG, S. L. **Parallel machine scheduling problems in green manufacturing industry**. Journal of Manufacturing Systems, v. 38, p. 98–106, 2016.

LUGO, P. L. M. **Modelos de otimização e métodos de solução para o planejamento da produção e distribuição na indústria de móveis**. Tese (Doutorado em Engenharia de

Produção) - Universidade Federal de São Carlos. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-graduação em Engenharia de Produção, 263 p., 2018.

MACCARTHY, B. L.; LIU, J. **Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling**. International Journal of Production Research, v.31, n.1, p.59–79, 1993.<http://dx.doi.org/10.1080/00207549308956713>

MANNE, A.S. **On the job-shop scheduling problem**. Operations Research, v. 8, p. 219–223, 1960.

MILLER C.E.; TUCKER A.W.; ZEMLIN R.A. **Integer programming formulation of traveling salesman problems**. Journal of the ACM, v.7, n. 4, p. 326–329, 1960.

MIN, H.; ZHOU, G. **Supply chain modeling: past, present and future**. Computers & Industrial Engineering, v.43, p. 231-249, 2002.

MOLE, R. H.; JAMESON, S. R. **A Sequential Route-building Algorithm Employing a Generalised Savings Criterion.** Journal of the Operational Research Society, v.27, n.2, p.503–511, 1976. <http://dx.doi.org/10.1057/jors.1976.95>

MOONS, S.; RAMAEKERS, K.; CARIS, A; ARDA Y. **Integrating production scheduling and vehicle routing decisions at the operational decision level:** A review and discussion. Computers & Industrial Engineering, v. 104, p. 224–245, 2017.

MORAIS, F. M.; MOCCELLIN, J. V. **Métodos heurísticos construtivos para redução do estoque em processo em ambientes de produção flow shop híbridos com tempos de *setup* dependentes da sequência**. Gest. Prod. [online], v.17, n.2, p.367-375, 2010. Disponível em: <http://dx.doi.org/10.1590/S0104-530X2010000200011>.

MORAIS, M. F.; MENEGARDE, J. K.; CANTIERE, P. C. **Regras de prioridade e critérios de desempenho adotados em problemas de programação da produção em ambientes flow shop**. IV Encontro de Produção Científica e Tecnológica, p. 1-9, 2009.

NADERI B.; M. ZANDIEHB M.; SHIRAZI M.A.H.A. **Modeling and scheduling a case of flexible flowshops**: Total weighted tardiness minimization. Computers & Industrial Engineering, v. 57, p. 1258-1267, 2009.

NAGANO, M. S.; MOCCELLIN, J. V. ***A high quality solution constructive heuristic for flow shop sequencing***, Journal of the Operational Research Society, v.53, n.12, p. 1374-1379, 2002.

NAGANO, M. S.; MOCCELLIN, J. V. **Flow shop híbrido com estágios gargalos.** In: XXXV SBPO, Natal. **Anais...** Natal: UFRN, 2003.

NAWAZ, M.; ENSCORE E. E. JR.; HAM I. **A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**. Omega, v.11, n.1, p. 91-95, 1983.

OSMAN, I.; POTTS, C. **Simulated annealing for permutation flow-shop scheduling**.

Revista OMEGA, The International Journal of Management Science, v.17, n.6, p.551–557, 1989.

PETTIE, S.; RAMACHANDRAN, V.**A shortest path algorithm for real-weighted undirected graphs**. SIAM J. Comput, v.34, n.6, 2005.

PINEDO, M. **Scheduling:** Theory, Algorithms, and Systems. Prentice Hall, v.3, NJ, EUA, 2008.

POLACEK M.; DOERNER K. F.; HARTL R. F.; KIECHLE G.; MARC REIMANN M. **Scheduling periodic customer visits for a traveling salesperson**. European Journal of Operational Research, v. 179, n. 3, 2007.

PUGAZHENDHI, S.; THIAGARAJAN, S.; RAJENDRAN, C.; ANANTHARAMAN, N. **Performance enhancement by using non-permutation schedules in flowline-based manufacturing systems**. Computers & Industrial Engineering, v,44, n.1, p. 133–157, 2003. <http://dx.doi.org/10.1016/s0360-8352(02)00189-4>.

RAFIEI, H.; SAFAEIA.; RABBANIB, M. **Integrated production-distribution planning problem in a competition-based four-echelon supply chain**. Computers & Industrial Engineering, v. 119, p.85–99, 2018.

RAKKE, J.; CHRISTIANSEN, M.; FAGERHOLT, K.; LAPORTE, G. **The Traveling Salesman Problem with Draft Limits**. Computers & Operations Research, v.39, p.2161-2167, 2012. <http://dx.doi.org/10.1016/j.cor.2011.10.025>.

REIMANN, M., TAVARES NETO, R., & BOGENDORFER, E. **Joint optimization of production planning and vehicle routing problems:** A review of existing strategies. Pesquisa Operacional, v.34, p.189–214, 2014.<http://dx.doi.org/10.1590/01017438.2014.034.02.0189>.

RONCONI D.; BIRGIN, E. **Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness**, Just-in Time Systems. s, Springer Series on Optimization and Its Applications, v. 60, 2011.

RUIZ R.; SIVRIKAYA F; URLINGS S. T. **Modeling realistic hybrid flexible flowshop scheduling problems**. Computers & Operations Research, v. 35, p.1151–1175, 2008.

SADJADI S. J.; ARYANEZHAD M. B.; ZIAEE M. **The general flowshop scheduling problem**: Mathematical Models. Journal of Applied Sciences, v. 8, n.17, p. 3032-3037, 2008.

SAWIK, T. **Integrated supply, production and distribution scheduling under disruption risks**. Revista Omega, v.62, p.131–144, 2016. <http://dx.doi.org/10.1016/j.omega.2015.09.005>.

SCHOLZ-REITER, B.; MAKUSCHEWITZ, T.; NOVAES, A. G. N.; FRAZZON, E. M.; LIMA JUNIOR, O. F. **An approach for the sustainable integration of production and transportation scheduling. International Journal of Logistics Systems and Management**, v. 10, p. 158–179, 2011.

SIMCHI-LEVI.; CHEN X.; BRAMEL, J. **The Logic of Logistics:** Theory, Algorithms, and Applications for Logistics Management, Springer, v.3, 2014.

SLACK, N. et al. **Administração da Produção- Edição compacta**. São Paulo: Atlas, 1999.

SMITH, W. E. **Various optimizers for single-stage production**. Naval Research Logistics Quarterly, v.3, p.59–66, 1956.

GERHARDT, E. T; SILVEIRA, T. D. **Métodos de Pesquisa,** Porto Alegre: Editora da UFRGS, v.1, 2009.

TAVARES NETO, R. F; NAGANO, M. S. **An Iterated Greedy approach to integrate production by parallel machines and distribution by a single capacitated vehicle.** Swarm and Evolutionary Computation, 2019.

THOMAS, D. J.; GRIFFIN, P. M. **Coordinated supply chain management**. European Journal of Operational Research, v. 94, p. 1–15, 1996.

TUBINO, D. F. **Planejamento e Controle da Produção:** Teoria à prática. São Paulo: Atlas, 2007, 190 p.

ULRICH, C. A. **Integrated machine scheduling and vehicle routing with time windows**, European Journal of Operational Research, v. 227, p.152-165, 2013.

VAN BUER, M.G.; WOODRUFF, D.L.; OLSONR.T. **Solving the Medium Newspaper Production/Distribution Problem**. European Journal of Operations Research, v. 115, p. 237-253, 1999.

WAGNER, H. M. **An integer linear programming model for machine scheduling**. Naval Research Logistics Quarterly, v.6, n.2, p.131-140, 1959. <http://dx.doi.org/10.1002/nav.3800060205>

WANG JB. **Single-machine scheduling with general learning functions**. Computers & Mathematics with Applications, v. 56, n.8, p. 1941-1947, 2008.

WANG D.Y.; GRUNDER O.; MOUDNI A. EL. **Integrated scheduling of production and distribution operations**: A review. International Journal of Industrial and Systems Engineering, v. 19, n.1, p. 94-122, 2015.

WANG J. **Mathematical Models for Optimization Problem in Flowshop Scheduling**: Rev. Téc. Ing. Univ. Zulia, v. 39, n.6, p. 257-263, 2016. <http://dx.doi.org/10.21311/001.39.6.31>.

WILSON, J.M. **Alternative formulations of a flow-shop scheduling problem**. Journal of the Operational Research Society, v. 40, p.395–399, 1989.

APÊNDICE A – Algoritmo desacoplado de produção + VRP no software *Python*

APÊNDICE A

```
import numpy as np
import cplex
from problem import Problem
import sys
import re
from modeloVRP import criaModeloVRP

class Job: #Para usar o sistema de ordenação do Python
def __init__(self, id, pi):
        self.id = id      #Id do job
self.pi = pi    #Tempo de processamento

def __lt__(self, other):
        return self.pi<other.pi
    def __le__(self, other):
        return self.pi<= other.pi
    def __eq__(self, other):
        return self.pi == other.pi
    def __ne__(self, other):
        return self.pi != other.pi
    def __gt__(self, other):
        return self.pi>other.pi
    def __ge__(self, other):
        return self.pi>= other.pi
    def __str__(self):
        return "{0}: {1}".format(self.id, self.pi)

def criaSolucaoDesacoplado(arquivoEntrada, logFile, solFile, timelimit):
    arquivoInstancia_semDir = re.split("/|\\\\", arquivoEntrada)[-1]

    problema = Problem()
```

```
    problema.read(arquivoEntrada)

    #Resolve o problema do scheduling: SPT para minimizar tempo de fluxo
jobs = [Job(i, problema.rho[i]) for i in range(1, problema.n)]
jobs.sort()
    jobs = [Job(0, problema.rho[0])] + jobs

#Calculo de quando as tarefas estarao prontas para serem enviadas
temposLiberacao = []
libAtual = 0.0
    for j in jobs:
libAtual += j.pi
        temposLiberacao.append([j.id, libAtual])

    #Cria modelo VRP
    modelvrp = criaModeloVRP(problema, problema.n)

    #incluindo a restrição dos tempos de liberacao no modelo
for t in temposLiberacao:
modelvrp.variables.set_lower_bounds("C_{0}".format(t[0]), t[1])

fobj = open(logFile, "w")
modelvrp.set_log_stream(fobj)
modelvrp.set_warning_stream(fobj)
modelvrp.set_error_stream(fobj)
modelvrp.set_results_stream(fobj)
modelvrp.parameters.workmem.set(1000)
modelvrp.parameters.mip.limits.treememory.set(10000)
modelvrp.parameters.mip.strategy.file.set(2)
modelvrp.parameters.timelimit.set(timelimit)
initTime = modelvrp.get_time()
#arquivo VRP que é utilizado no desacoplado
def criaModeloVRP(problema, n):
```

```
        model = cplex.Cplex()

        #tamanhos = Lido do arquivo (problema.delta[i][j])

        #BIGM
        BIGM_R = float(np.sum(problema.delta)+ np.sum(problema.rho))


    var = ["w_{0}_{1}_{2}".format(i,j,r) for i in range(n) for j in range(n) for r in
range(n)]
    model.variables.add(names = var, types = ["B"]*len(var))

        var = ["A_{0}".format(i) for i in range(n)]
    model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var),
ub=[problema.kappa]*len(var))

    var = ["R_{0}".format(r) for r in range(n)]
    model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))

        var = ["D_{0}".format(i) for i in range(n)]
    model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))

        var = ["z"]
    model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0])

        var = ["u_{0}".format(i) for i in range(n)]
    model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))

        var = ["C_{0}".format(i) for i in range(n)]
    model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))


        #constraints related to w variables
```

```
        for r in range(1, n): #(4.8)
var = ["w_0_0_{0}".format(r-1), "w_0_0_{0}".format(r)]
coef = [1, -1]
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [0], senses="L")

        for i in range(1,n): #(4.9)
            for j in range(1,n):
                var = ["w_{0}_{1}_{2}".format(i,j,r) for r in range(n)]
coef = [1]*len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="L")

        for r in range(n): #(4.10)
            var = ["w_0_{0}_{1}".format(j,r) for j in range(n)]
coef = [1]*len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

        for r in range(n): #(4.11)
            var = ["w_{0}_0_{1}".format(j,r) for j in range(n)]
coef = [1]*len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

        for j in range(1,n): #(4.12)
            for r in range(n):
var = ["w_{0}_{1}_{2}".format(j,j,r)]
coef = [1]
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [0], senses="E")
```

```
        for i in range(1,n):  #(4.13)
            var = ["w_{0}_{1}_{2}".format(i,j,r) for j in range(n) for r in range(n)]
coef = [1] * len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

        for i in range(1,n): #(4.14)
            for r in range(n):
var = ["w_0_0_{0}".format(r)]
                coef = [1]
var2 = ["w_{0}_{1}_{2}".format(i,j,r) for j in range(1,n)]
                coef2 = [1] * len(var2)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var+var2, val = coef+coef2)], rhs = [1], senses="L")

        for i in range(1,n): #(4.15)
            for r in range(n ):
              var = ["w_{0}_{1}_{2}".format(i,h,r) for h in range(n) if h != i]
val = [1]*len(var)
              var1 = ["w_{0}_{1}_{2}".format(h,i,r) for h in range(n) if h != i]
              val1 = [-1]*len(var1)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var+var1, val = val+val1)], rhs = [0], senses="E")

#constraints related to vehicle capacity (A)

        BIGM = sum(problema.sigma)

        for j in range(1,problema.n):  #(4.16)
            for i in range(problema.n):
              if j != i:
                for r in range(problema.n):
var = ["A_{0}".format(j), "A_{0}".format(i),"w_{0}_{1}_{2}".format(i,j,r)]
```

```
        coef = [1, -1,-BIGM]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [problema.sigma[j] -
1*BIGM], senses="G")



              for r in range(n):  #(4.17)
                 for i in range(1,n):
                    for j in range(n):
        if i != j:
                           var =
["R_{0}".format(r),"C_{0}".format(i),"w_{0}_{1}_{2}".format(i,j,r)]
                           coef = [1,-1,-1*BIGM_R]
                           model.linear_constraints.add(
                              lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [-1*BIGM_R],
senses="G")



        for i in range(1,n):  #(4.18)
                    for r in range(n):
        var = ["D_{0}".format(i),"R_{0}".format(r),"w_0_{0}_{1}".format(i,r)]
                       coef = [1,-1,-BIGM_R]
                       model.linear_constraints.add(
                          lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs =
[problema.delta[0][i] - BIGM_R], senses="G")

        for j in range(1, n):  #(4.19)
                    for i in range(n):
                       if j != i:
                        for r in range(n):
        var = ["D_{0}".format(j), "D_{0}".format(i),"w_{0}_{1}_{2}".format(i,j,r)]
                           coef = [1,-1,-BIGM_R]
                           model.linear_constraints.add(
```

```
                    lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs =
[problema.delta[i][j] - BIGM_R], senses="G")

        for r in range(1,n):  #(4.20)
                for i in range(1,n):
        var = ["R_{0}".format(r), "D_{0}".format(i),"w_{0}_0_{1}".format(i,r-1)]
                    coef = [1,-1,- BIGM_R]
                    model.linear_constraints.add(
                        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs =
[problema.delta[i][0] - BIGM_R], senses="G")


            #equacões para eliminar subrotas

            for i in range(0):  #(4.21)
                var = ["u_{0}".format(i)]
                coef = [1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

            for i in range(1,n):  #(4.22)
        var = ["u_{0}".format(i)]
                coef = [1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [2], senses="G")

            for i in range(1,n):  #(4.23)
        var = ["u_{0}".format(i)]
                coef = [1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [n], senses="L")

            for j in range(1,n):  #(4.24)
```

```
                for i in range(1,n):
                        if i != j:
                                var = ["u_{0}".format(i), "u_{0}".format(j),
"w_{0}_{1}_0".format(i,j)]
        coef = [1,-1,n-1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [n-2], senses="L")

                var = ["z"]
        val = [1]
                var1 = ["D_{0}".format(i) for i in range(n) ]
                val1 = [-1  fori in range(n) ]

        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var+var1, val = val+val1)], rhs = [0], senses="E")

        model.objective.set_linear(zip(["z"], [1]))
        model.objective.set_sense(model.objective.sense.minimize)

#retorno ao arquivo desacoplado e solve
        modelvrp.solve()
        endTime = modelvrp.get_time()
        modelvrp.solution.write(solFile)
        fobj.close()

            return jobs, modelvrp,
[arquivoInstancia_semDir,modelvrp.solution.get_objective_value(),
modelvrp.solution.MIP.get_best_objective(), modelvrp.solution.get_status(),
modelvrp.solution.get_status_string(), endTime-initTime]

        if __name__ == "__main__":

        arquivoInstancia = sys.argv[1]
```

```
        arquivoInstancia_semDir = re.split("/|\\\\", arquivoInstancia)[-1]
            logFile = arquivoInstancia_semDir+".node.log"
            solFile = arquivoInstancia_semDir+".sol"

            jobs, modelvrp, results = criaSolucaoDesacoplado(arquivoEntrada, logFile, solFile,
timelimit)

        print(arquivoInstancia_semDir,modelvrp.solution.get_objective_value(),
modelvrp.solution.MIP.get_best_objective(), modelvrp.solution.get_status(),
modelvrp.solution.get_status_string(), endTime-initTime, sep=";")
```

APÊNDICE B – Algoritmo integrado de produçãoe distribuição (máquina única e um veículo)

APÊNDICE B

```
import cplex
import numpy as np
from problem import Problem

def criaModeloIntegrado(problema):
    model = cplex.Cplex()

    for i in range(problema.n):
        var = ["tempoProc_{0}".format(i)]
model.variables.add(names = var, types = ["B"]*len(var))

#Criação de variáveis

    #xij Binário

    var = ["x_{0}_{1}".format(i,j) for i in range(problema.n) for j in
range(problema.n)]
    model.variables.add(names = var, types = ["B"]*len(var))

    #Ci Continua, positiva

    var = ["C_{0}".format(i) for i in range(problema.n)]
model.variables.add(names = var, types = ["C"]*len(var))

#z Continua, positiva

    #Criação de restricoes
    #4.2

    for p in range(1,problema.n): #sum_i x_ip = 1 para todo i
        var = ["x_0_{0}".format(p)]
```

```
                coef = [1]*len(var)
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [0], senses="E",
names=["_Eq1_{0}".format(p)])

            #4.3
            for i in range(problema.n):
                var = ["x_{0}_{1}".format(i, p) for p in range(problema.n)]
        coef = [1]*len(var)
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E",
names=["_Eq2_{0}".format(i)])

            #4.4
            for p in range(problema.n):
                var = ["x_{0}_{1}".format(i, p) for i in range(problema.n)]
        coef = [1]*len(var)
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="L",
names=["_Eq3_{0}".format(p)])

            #4.5
            for p in range(problema.n-1):
                var = ["x_{0}_{1}".format(j, p+1) for j in range(problema.n)]
        coef = [1]*len(var)
                var2 = ["x_{0}_{1}".format(i, p) for i in range(problema.n)]
                coef2 = [-1]*len(var)
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var+var2, val = coef+coef2)], rhs = [0], senses="L")

            #4.6
            for i in range(problema.n):
                var = ["C_{0}".format(i), "x_{0}_0".format(i)]
```

```
        coef = [1, -1*BIGMS]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [problema.rho[i] - 1*BIGMS],
senses="G")

                #4.7
                BIGMS = sum(problema.rho)
                for i in range(problema.n):
                    for j in range(1, problema.n):
                      if i != j:
                          for p in range(problema.n-1):
                                var = ["C_{0}".format(j), "C_{0}".format(i),
"x_{0}_{1}".format(i,p), "x_{0}_{1}".format(j,p+1)]
        coef = [1, -1, -1*BIGMS, -1*BIGMS]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [problema.rho[j] - 2*BIGMS],
senses="G")


        def criaModeloVRP(problema, n):
                model = cplex.Cplex()

        #tamanhos = Lido do arquivo (problema.delta[i][j])

                #BIGM
                BIGM_R = float(np.sum(problema.delta)+ np.sum(problema.rho))



        var = ["w_{0}_{1}_{2}".format(i,j,r) for i in range(n) for j in range(n) for r in
range(n)]
        model.variables.add(names = var, types = ["B"]*len(var))

                var = ["A_{0}".format(i) for i in range(n)]
```

```
        model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var),
ub=[problema.kappa]*len(var))

        var = ["R_{0}".format(r) for r in range(n)]
        model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))

            var = ["D_{0}".format(i) for i in range(n)]
        model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))

            var = ["z"]
        model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0])

            var = ["u_{0}".format(i) for i in range(n)]
        model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))

            var = ["C_{0}".format(i) for i in range(n)]
        model.variables.add(names = var, types = ["C"]*len(var), lb=[0.0]*len(var))



            #constraints related to w variables

            for r in range(1, n): #(4.8)
        var = ["w_0_0_{0}".format(r-1), "w_0_0_{0}".format(r)]
        coef = [1, -1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [0], senses="L")

            for i in range(1,n): #(4.9)
                for j in range(1,n):
                    var = ["w_{0}_{1}_{2}".format(i,j,r) for r in range(n)]
        coef = [1]*len(var)
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="L")
```

```
        for r in range(n): #(4.10)
            var = ["w_0_{0}_{1}".format(j,r) for j in range(n)]
coef = [1]*len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

        for r in range(n): #(4.11)
            var = ["w_{0}_0_{1}".format(j,r) for j in range(n)]
coef = [1]*len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

        for j in range(1,n): #(4.12)
            for r in range(n):
var = ["w_{0}_{1}_{2}".format(j,j,r)]
coef = [1]
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [0], senses="E")

        for i in range(1,n):  #(4.13)
            var = ["w_{0}_{1}_{2}".format(i,j,r) for j in range(n) for r in range(n)]
coef = [1] * len(var)
model.linear_constraints.add(
lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

        for i in range(1,n): #(4.14)
            for r in range(n):
var = ["w_0_0_{0}".format(r)]
                coef = [1]
var2 = ["w_{0}_{1}_{2}".format(i,j,r) for j in range(1,n)]
                coef2 = [1] * len(var2)
model.linear_constraints.add(
```

```
        lin_expr=[cplex.SparsePair(ind=var+var2, val = coef+coef2)], rhs = [1], senses="L")

            for i in range(1,n): #(4.15)
                  for r in range(n ):
                    var = ["w_{0}_{1}_{2}".format(i,h,r) for h in range(n) if h != i]
        val = [1]*len(var)
                    var1 = ["w_{0}_{1}_{2}".format(h,i,r) for h in range(n) if h != i]
                    val1 = [-1]*len(var1)
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var+var1, val = val+val1)], rhs = [0], senses="E")

        #constraints related to vehicle capacity (A)

              BIGM = sum(problema.sigma)

              for j in range(1,problema.n):  #(4.16)
                  for i in range(problema.n):
                    if j != i:
                      for r in range(problema.n):
        var = ["A_{0}".format(j), "A_{0}".format(i),"w_{0}_{1}_{2}".format(i,j,r)]
        coef = [1, -1,-BIGM]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [problema.sigma[j] -
1*BIGM], senses="G")

               # Integrated constraints
              for r in range(n):  #(4.17)
                  for i in range(1,n):
                    for j in range(n):
        if i != j:
                          var =
["R_{0}".format(r),"C_{0}".format(i),"w_{0}_{1}_{2}".format(i,j,r)]
                          coef = [1,-1,-1*BIGM_R]
```

```
                    model.linear_constraints.add(
                        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [-1*BIGM_R],
senses="G")

        for i in range(1,n):  #(4.18)
                for r in range(n):
        var = ["D_{0}".format(i),"R_{0}".format(r),"w_0_{0}_{1}".format(i,r)]
                    coef = [1,-1,-BIGM_R]
                    model.linear_constraints.add(
                        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs =
[problema.delta[0][i] - BIGM_R], senses="G")

        for j in range(1, n):  #(4.19)
                for i in range(n):
                    if j != i:
                     for r in range(n):
        var = ["D_{0}".format(j), "D_{0}".format(i),"w_{0}_{1}_{2}".format(i,j,r)]
                        coef = [1,-1,-BIGM_R]
                        model.linear_constraints.add(
                            lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs =
[problema.delta[i][j] - BIGM_R], senses="G")

        for r in range(1,n):  #(4.20)
                for i in range(1,n):
        var = ["R_{0}".format(r), "D_{0}".format(i),"w_{0}_0_{1}".format(i,r-1)]
        coef = [1,-1,-200000]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [problema.delta[i][0] -
200000], senses="G")

            #equacões para eliminar subrotas

        for i in range(0):  #(4.21)
```

```
                var = ["u_{0}".format(i)]
        coef = [1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [1], senses="E")

            for i in range(1,n):  #(4.22)
                var = ["u_{0}".format(i)]
        coef = [1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [2], senses="G")

            for i in range(1,n):  #(4.23)
        var = ["u_{0}".format(i)]
                coef = [1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [n], senses="L")

            for j in range(1,n):  #(4.24)
                for i in range(1,n):
                        if i != j:
                            var = ["u_{0}".format(i), "u_{0}".format(j),
"w_{0}_{1}_0".format(i,j)]
        coef = [1,-1,n-1]
        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var, val = coef)], rhs = [n-2], senses="L")

            var = ["z"]
        val = [1]
            var1 = ["D_{0}".format(i) for i in range(n) ]
            val1 = [-1  fori in range(n) ]

        model.linear_constraints.add(
        lin_expr=[cplex.SparsePair(ind=var+var1, val = val+val1)], rhs = [0], senses="E")
```

```
model.objective.set_linear(zip(["z"], [1]))
model.objective.set_sense(model.objective.sense.minimize)

    return model
```

APÊNDICE C – *Mipstart*

APÊNDICE C

```
import sys
import os
import re
import cplex
import desacoplado
import modeloIntegrado
from problem import Problem

def insereSolucoesnoModelo(modeloOrigem, modeloDestino):
    #Criar vetor com todos os nomes das variáveis do modelo origem
    variaveis = [modeloOrigem.variables.get_names(i) for i in
range(modeloOrigem.variables.get_num())]
    _ind = modeloDestino.variables.get_names(variaveis)
_val = modeloOrigem.solution.get_values(variaveis)
modeloDestino.MIP_starts.add(cplex.SparsePair(ind=_ind, val=_val),
modeloDestino.MIP_starts.effort_level.check_feasibility)

def getIntegradoVazio(arquivoInstancia):
    problema = Problem()
    problema.read(arquivoInstancia)
    model = modeloIntegradoTavares.criaModeloIntegrado(problema)

    return model

def getIntegradoComValoresIniciais(arquivoInstancia, jobs, vrp):
model = getIntegradoVazio(arquivoInstancia)
    for i in range(len(jobs)):
        var = "x_{0}_{1}".format(jobs[i].id, i)
model.variables.set_lower_bounds(var, 1)

    for i in range(len(jobs)):
```

```
            for j in range(len(jobs)):
                for r in range(len(jobs)):
        var = "w_{0}_{1}_{2}".format(i,j,r)
        if vrp.solution.get_values(var) > 0.5:
        model.variables.set_lower_bounds(var, 1)

        model.set_log_stream(None)
        model.set_warning_stream(None)
        model.set_error_stream(None)
        model.set_results_stream(None)
        model.solve()
            return model

        def executaModeloIntegrado(arquivoInstancia_semDir, logFile, solFile, model,
timelimit):
        fobj = open(logFile, "w")
        model.set_log_stream(fobj)
        model.set_warning_stream(fobj)
        model.set_error_stream(fobj)
        model.set_results_stream(fobj)
        model.parameters.workmem.set(1000)
        model.parameters.mip.limits.treememory.set(10000)
        model.parameters.mip.strategy.file.set(2)
        model.parameters.timelimit.set(timelimit)
        initTime = model.get_time()
        model.solve()
        endTime = model.get_time()
        model.solution.write(solFile)

            return [arquivoInstancia_semDir,model.solution.get_objective_value(),
model.solution.MIP.get_best_objective(), model.solution.get_status(),
model.solution.get_status_string(), endTime-initTime]
```

```
if __name__ == "__main__":
arquivoInstancia = sys.argv[1]
arquivoInstancia_semDir = re.split("/|\\\\", arquivoInstancia)[-1]
    logFile = arquivoInstancia_semDir+".{0}.node.log"
    solFile = arquivoInstancia_semDir+".{0}.sol"

    timelimit = int(sys.argv[2])

    #Primeiro desacoplado
    jobs, modelvrp, results =
desacopladoTavares.criaSolucaoDesacoplado(arquivoInstancia,
logFile.format("desacoplado"), solFile.format("desacoplado"), timelimit)

print("desacoplado;"+";".join([str(r) for r in results]))

#iniciando o mipstart:
    modelSandbox = getIntegradoComValoresIniciais(arquivoInstancia, jobs,
modelvrp)

    #Criando um modelo integrado vazio:
    modelToRun = getIntegradoVazio(arquivoInstancia)

    #Inserir as variáveis do mipstart
insereSolucoesnoModelo(modelSandbox, modelToRun)

    results = executaModeloIntegrado(arquivoInstancia_semDir,
logFile.format("mipstart"), solFile.format("mipstart"), modelToRun, timelimit)
    print("mipstart;"+";".join([str(r) for r in results]))

#Resolvendo sem o MIP Start
    modelPuro = getIntegradoVazio(arquivoInstancia)
```

```
        results = executaModeloIntegrado(arquivoInstancia_semDir,
logFile.format("integrado"), solFile.format("integrado"), modelPuro, timelimit)
        print("integrado;"+";".join([str(r) for r in results]))
```

APÊNDICE D – Resultados obtidos na execução dos modelos: desacoplado, integrado e *mipstart*

APÊNDICE D

Nº instância	Instância	Execução de 120 segundos			Execução de 1.800 segundos		
		desacoplado	integrado	mipstart	desacoplado	integrado	mipstart
1	**rand_5_10_1_0**	506	506	506	506	506	506
2	**rand_5_10_1_1**	186	186	186	186	186	186
3	**rand_5_10_1_10**	825	825	825	825	825	825
4	**rand_5_10_1_11**	306	304	304	306	304	304
5	**rand_5_10_1_12**	440	440	440	440	440	440
6	**rand_5_10_1_13**	245	245	245	245	245	245
7	**rand_5_10_1_14**	163	163	163	163	163	163
8	**rand_5_10_1_15**	360	360	360	360	360	360
9	**rand_5_10_1_16**	474	474	474	474	474	474
10	**rand_5_10_1_17**	571	571	571	571	571	571
11	**rand_5_10_1_18**	225	225	225	225	225	225
12	**rand_5_10_1_19**	418	418	418	418	418	418
13	**rand_5_10_1_2**	412	412	412	412	412	412
14	**rand_5_10_1_3**	412	412	412	412	412	412
15	**rand_5_10_1_4**	340	340	340	340	340	340
16	**rand_5_10_1_5**	239	239	239	239	239	239
17	**rand_5_10_1_6**	348	348	348	348	348	348
18	**rand_5_10_1_7**	505	505	505	505	505	505
19	**rand_5_10_1_8**	348	348	348	348	348	348
20	**rand_5_10_1_9**	382	382	382	382	382	382
21	**rand_5_20_1_0**	434	434	434	434	434	434
22	**rand_5_20_1_1**	719	719	719	719	719	719
23	**rand_5_20_1_10**	407	407	407	407	407	407
24	**rand_5_20_1_11**	278	278	278	278	278	278
25	**rand_5_20_1_12**	512	512	512	512	512	512
26	**rand_5_20_1_13**	514	514	514	514	514	514
27	**rand_5_20_1_14**	362	362	362	362	362	362
28	**rand_5_20_1_15**	358	358	358	358	358	358
29	**rand_5_20_1_16**	385	385	385	385	385	385
30	**rand_5_20_1_17**	299	299	299	299	299	299
31	**rand_5_20_1_18**	702	702	702	702	702	702
32	**rand_5_20_1_19**	329	329	329	329	329	329
33	**rand_5_20_1_2**	198	198	198	198	198	198
34	**rand_5_20_1_3**	533	533	533	533	533	533
35	**rand_5_20_1_4**	304	304	304	304	304	304
36	**rand_5_20_1_5**	487	487	487	487	487	487
37	**rand_5_20_1_6**	514	514	514	514	514	514
38	**rand_5_20_1_7**	416	416	416	416	416	416
39	**rand_5_20_1_8**	253	253	253	253	253	253
40	**rand_5_20_1_9**	536	536	536	536	536	536
43	**rand_5_30_1_0**	353	353	353	353	353	353
44	**rand_5_30_1_1**	454	454	454	454	454	454
45	**rand_5_30_1_10**	405	405	405	405	405	405

Nº instância	Instância	Execução de 120 segundos			Execução de 1.800 segundos		
		desacoplado	integrado	mipstart	desacoplado	integrado	mipstart
46	**rand_5_30_1_11**	429	413	413	429	413	413
47	**rand_5_30_1_12**	525	525	525	525	525	525
48	**rand_5_30_1_13**	441	441	441	441	441	441
49	**rand_5_30_1_14**	309	309	309	309	309	309
50	**rand_5_30_1_15**	431	431	431	431	431	431
51	**rand_5_30_1_16**	594	594	594	594	594	594
52	**rand_5_30_1_17**	605	605	605	605	605	605
53	**rand_5_30_1_18**	479	479	479	479	479	479
54	**rand_5_30_1_19**	316	289	289	316	289	289
55	**rand_5_30_1_2**	449	449	449	449	449	449
56	**rand_5_30_1_3**	341	339	339	341	339	339
57	**rand_5_30_1_4**	333	333	333	333	333	333
58	**rand_5_30_1_5**	447	447	447	447	447	447
59	**rand_5_30_1_6**	543	543	543	543	543	543
60	**rand_5_30_1_7**	398	398	398	398	398	398
61	**rand_5_30_1_8**	541	541	541	541	541	541
62	**rand_5_30_1_9**	411	411	411	411	411	411
63	**rand_10_10_1_0**	1601	1679	1601	1733	1820	1733
64	**rand_10_10_1_1**	2503	2596	2503	2596	2780	2666
65	**rand_10_10_1_10**	1728	1772	1728	1821	1929	1821
66	**rand_10_10_1_11**	1337	1396	1337	1354	1496	1354
67	**rand_10_10_1_12**	1380	1398	1380	1421	1459	1483
68	**rand_10_10_1_13**	1094	1117	1094	1157	1288	1157
69	**rand_10_10_1_14**	1398	1431	1398	1485	1510	1485
70	**rand_10_10_1_15**	2359	2407	2359	2462	2775	2539
71	**rand_10_10_1_16**	930	941	930	969	974	975
72	**rand_10_10_1_17**	1614	1805	1614	1614	2093	1614
73	**rand_10_10_1_18**	1301	1340	1301	1340	1374	1340
74	**rand_10_10_1_19**	1285	1299	1285	1285	1407	1285
75	**rand_10_10_1_2**	983	1035	983	983	1134	1041
76	**rand_10_10_1_3**	1337	1431	1337	1337	1538	1337
77	**rand_10_10_1_4**	2970	2973	2970	3051	3074	3051
76	**rand_10_10_1_5**	687	687	687	687	741	687
78	**rand_10_10_1_6**	2393	2477	2393	2453	2610	2535
79	**rand_10_10_1_7**	1530	1624	1530	1572	1750	1572
80	**rand_10_10_1_8**	1448	1555	1448	1448	1627	1627
81	**rand_10_10_1_9**	1180	1250	1180	1209	1337	1209
82	**rand_10_20_1_0**	2328	2450	2328	2422	2671	2422
83	**rand_10_20_1_1**	1526	1570	1514	1608	1674	1608
84	**rand_10_20_1_10**	2192	2288	2192	2360	2414	2360
85	**rand_10_20_1_11**	2612	2719	2612	2787	2748	2793
86	**rand_10_20_1_12**	1883	1933	1883	1999	1974	1999
87	**rand_10_20_1_13**	1584	1639	1584	1692	1805	1692
88	**rand_10_20_1_14**	811	852	805	821	877	852
89	**rand_10_20_1_15**	1301	1318	1299	1301	1430	1301
90	**rand_10_20_1_16**	2603	2633	2594	2676	2995	2693

Nº instância	Instância	Execução de 120 segundos			Execução de 1.800 segundos		
		desacoplado	integrado	mipstart	desacoplado	integrado	mipstart
91	**rand_10_20_1_17**	2325	2451	2325	2328	2464	2443
92	**rand_10_20_1_18**	2282	2401	2282	2421	2517	2332
93	**rand_10_20_1_19**	1004	1029	994	1013	1045	1013
94	**rand_10_20_1_2**	1489	1595	1489	1595	1660	1611
95	**rand_10_20_1_3**	2009	2064	2009	2009	2411	2009
96	**rand_10_20_1_4**	1755	1800	1755	1797	1975	1797
97	**rand_10_20_1_5**	1984	2099	1984	1984	2102	1984
98	**rand_10_20_1_6**	1661	1774	1661	1713	1934	1661
99	**rand_10_20_1_7**	1871	2042	1871	1951	2070	1951
100	**rand_10_20_1_8**	1598	1583	1583	1598	1606	1583
101	**rand_10_20_1_9**	2236	2243	2236	2309	2496	2386
102	**rand_10_30_1_0**	1920	1920	1920	1986	2071	1986
103	**rand_10_30_1_1**	1490	1568	1490	1522	1756	1522
104	**rand_10_30_1_10**	1858	1910	1858	1858	2094	1965
105	**rand_10_30_1_11**	2336	2444	2336	2336	2545	2525
106	**rand_10_30_1_12**	1918	2008	1918	2142	2208	2147
107	**rand_10_30_1_13**	1760	1760	1760	1760	1906	1828
108	**rand_10_30_1_14**	952	923	952	952	961	991
109	**rand_10_30_1_15**	1994	1994	1994	1994	2140	2061
110	**rand_10_30_1_16**	1182	1182	1166	1212	1220	1212
111	**rand_10_30_1_17**	3189	3321	3189	3246	3421	3246
112	**rand_10_30_1_18**	2038	2046	2038	2105	2206	2105
113	**rand_10_30_1_19**	1618	1644	1616	1618	2088	1821
114	**rand_10_30_1_2**	1951	2014	1951	2014	2339	2014
115	**rand_10_30_1_3**	1576	1631	1576	1674	1697	1674
116	**rand_10_30_1_4**	1909	1995	1903	1920	2062	2032
117	**rand_10_30_1_5**	1317	1370	1313	1412	1482	1412
118	**rand_10_30_1_6**	1151	1190	1151	1173	1321	1177
119	**rand_10_30_1_7**	1090	1122	1090	1090	1295	1103
120	**rand_10_30_1_8**	1828	1828	1828	1890	2026	1925
121	**rand_10_30_1_9**	2199	2199	2199	2304	2413	2350
122	**rand_20_10_1_0**	7514	9021	7227	12720	14277	12720
123	**rand_20_10_1_1**	8752	10486	8367	11093	14436	10603
124	**rand_20_10_1_10**	6204	6447	5944	7444	8527	7712
125	**rand_20_10_1_11**	9846	10394	8680	11431	12987	11822
126	**rand_20_10_1_12**	10416	10030	10167	15460	12929	15724
127	**rand_20_10_1_13**	8821	11959	8479	14127	13385	12688
128	**rand_20_10_1_14**	8258	8150	7230	7509	9558	9387
129	**rand_20_10_1_15**	8680	9138	7029	16908	12141	11132
130	**rand_20_10_1_16**	8781	10827	8168	13406	12489	13052
131	**rand_20_10_1_17**	8742	10500	7703	12405	12194	10397
132	**rand_20_10_1_18**	10655	9407	9675	10900	11486	12323
133	**rand_20_10_1_19**	9227	9845	8205	14970	13542	12422
134	**rand_20_10_1_2**	13051	9117	11294	18610	12664	15985
135	**rand_20_10_1_3**	8966	6601	8966	10921	12443	11668
136	**rand_20_10_1_4**	5913	7675	5275	9122	8314	7819

Nº instância	Instância	Execução de 120 segundos			Execução de 1.800 segundos		
		desacoplado	integrado	mipstart	desacoplado	integrado	mipstart
137	**rand_20_10_1_5**	8227	8962	7196	14414	11224	11862
138	**rand_20_10_1_6**	14595	12770	11714	15734	15615	15734
139	**rand_20_10_1_7**	7033	7921	5970	10432	10267	9996
140	**rand_20_10_1_8**	8765	9110	8047	12294	11413	11186
141	**rand_20_10_1_9**	8949	8515	8949	16275	12246	15247
142	**rand_20_20_1_0**	6801	8805	6671	10882	10826	9871
143	**rand_20_20_1_1**	7958	9403	7811	11233	10758	10892
144	**rand_20_20_1_10**	8619	10156	7956	18764	11517	16505
145	**rand_20_20_1_11**	11562	10218	9570	12240	12846	12240
146	**rand_20_20_1_12**	10776	11485	10461	12564	13334	13648
147	**rand_20_20_1_13**	10261	0	8301	13574	10480	12630
148	**rand_20_20_1_14**	13998	13901	11413	24239	14194	14406
149	**rand_20_20_1_15**	9399	11039	8699	14091	12648	14091
150	**rand_20_20_1_16**	11342	11340	10526	16425	15531	15604
151	**rand_20_20_1_17**	11575	12176	9603	14034	15751	15185
152	**rand_20_20_1_18**	6697	10217	6238	14399	10257	10904
153	**rand_20_20_1_19**	12810	12175	11067	14845	16698	12817
154	**rand_20_20_1_2**	13481	11162	10199	15201	14812	15665
155	**rand_20_20_1_3**	11254	11780	10480	14629	14415	14967
156	**rand_20_20_1_4**	7294	7474	5.968	10502	10461	9034
157	**rand_20_20_1_5**	11415	12885	11031	16348	14008	13622
158	**rand_20_20_1_6**	10903	10032	10340	14667	13079	10895
159	**rand_20_20_1_7**	7193	8375	6392	11704	10279	10423
160	**rand_20_20_1_8**	9702	0	8429	14061	11340	11796
161	**rand_20_20_1_9**	9502	9774	9407	12834	12411	12625
162	**rand_20_30_1_0**	10575	12485	9171	20107	14050	17440
163	**rand_20_30_1_1**	12319	11392	9927	16917	13503	13100
164	**rand_20_30_1_10**	10651	13492	10277	13778	16097	15271
165	**rand_20_30_1_11**	6811	7785	5916	12841	11009	10879
166	**rand_20_30_1_12**	11454	11203	10566	19850	13284	14731
167	**rand_20_30_1_13**	9579	10180	8041	10212	12794	12068
168	**rand_20_30_1_14**	8690	9695	8055	16583	11567	12454
169	**rand_20_30_1_15**	10732	10815	10426	11806	12178	11487
170	**rand_20_30_1_16**	12291	11101	12291	19398	20510	18031
171	**rand_20_30_1_17**	9242	8988	7847	9304	13639	11840
172	**rand_20_30_1_18**	10176	10640	10176	16464	13811	16381
173	**rand_20_30_1_19**	7520	8949	7344	8891	10589	10613
174	**rand_20_30_1_2**	8643	9129	8068	10965	13336	12722
175	**rand_20_30_1_3**	10651	9192	9338	15007	15871	13211
176	**rand_20_30_1_4**	11228	11567	8722	19982	14940	13842
177	**rand_20_30_1_5**	9158	8414	7136	10680	14968	9677
178	**rand_20_30_1_6**	10682	10309	9986	19548	13681	12193
179	**rand_20_30_1_7**	8768	7200	8346	16198	13233	11985
180	**rand_20_30_1_8**	10237	9767	10237	14284	16247	13301
181	**rand_20_30_1_9**	12144	10589	10771	21540	14631	13203
182	**rand_40_10_1_4**	49342	0	47450	0	0	0

Nº instância	Instância	Execução de 120 segundos			Execução de 1.800 segundos		
		desacoplado	integrado	mipstart	desacoplado	integrado	mipstart
183	**rand_40_10_1_9**	74843	0	70202	0	0	0
184	**rand_40_20_1_1**	94444	0	77380	0	0	0