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

Giuliano Frascati

Programação da Produção em máquina única com setup dependente da sequência e terceirização permitida: uma abordagem de Otimização por Colônia de Formigas

São Carlos
2014

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

Giuliano Frascati

Programação da Produção em máquina única com setup dependente da sequência e terceirização permitida: uma abordagem de Otimização por Colônia de Formigas

Dissertação de Mestrado
Universidade Federal de São Carlos
Engenharia de Produção
Orientador: Roberto Tavares Neto

São Carlos
2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

F841pp

Frascati, Giuliano.
Programação da produção em máquina única com *setup* dependente da sequência e terceirização permitida : uma abordagem de otimização por colônia de formigas / Giuliano Frascati. -- São Carlos : UFSCar, 2014.
96 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Pesquisa operacional. 2. Programação da produção. 3. Terceirização. 4. Meta-heurística. 5. Máquina única. I. Título.

CDD: 658.4034 (20ª)

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO
UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
Rod. Washington Luís, Km. 235 - CEP. 13565-905 - São Carlos - SP - Brasil
Fone/Fax: (016) 3351-8236 / 3351-8237 / 3351-8238 (ramal: 232)
Email : ppgep@dep.ufscar.br

FOLHA DE APROVAÇÃO

Aluno(a): Giuliano Frascati

DISSERTAÇÃO DE MESTRADO DEFENDIDA E APROVADA EM 18/02/2014 PELA COMISSÃO JULGADORA:

Prof. Dr. Roberto Fernandes Tavares Neto
Orientador(a) PPGEP/UFSCar

Prof. Dr. Marcelo Seido Nagano
EESC/USP

Prof. Dr. Paulo Rogério Politano
DC/UFSCar

Prof. Dr. Mário Otávio Batalha
Coordenador do PPGEP

Agradecimentos

Agradeço primeiramente a oportunidade de trilhar um caminho repleto de aprendizado e perseverança na direção da conquista dos meus sonhos.

Agradeço ao meu orientador Professor Doutor Roberto Fernandes Tavares Neto pela orientação impecável, disponibilidade e parceria ao longo do nosso projeto.

Á minha mãe, Solange Pavão, pela fé e esperança sempre depositadas em mim.

Á minha namorada Letícia De Carli Novaes por todo o apoio, paciência e carinho.

Ao meu colega Márcio Rogério da Silva que guiou meus passos para a entrada no Programa de Pós-Graduação em Engenharia de Produção da UFSCar e à república Los Cowborjas e seus integrantes, por compartilhar comigo mais este objetivo alcançado.

Programação da Produção em máquina única com setup dependente da sequência e terceirização permitida: uma abordagem de Otimização por Colônia de Formigas

Resumo: Diversos problemas de *scheduling* são classificados na literatura como NP-Difíceis, o que significa que os custos computacionais das soluções desenvolvidas usando métodos exatos conhecidos são muito altos para esses problemas. No caso da possibilidade de terceirização de parte das tarefas existentes se torna vital inserir essas decisões nos problemas de *scheduling* visando à obtenção de resultados ótimos para os objetivos de desempenho. O presente trabalho trata de um caso como esse: um ambiente de máquina única onde os tempos de *setup* são dependentes da sequência de execução das operações e com a possibilidade de terceirização. O objetivo é determinar a sequência de operações executadas no ambiente de máquina única e o conjunto de operações a serem terceirizadas de forma que nenhuma das ordens de serviço seja entregue com atraso e o custo de terceirização seja mínimo. A aplicação de meta-heurísticas, como o ACO (*Ant Colony Optimization*) abre um novo horizonte para o desenvolvimento de soluções para problemas este, classificado como NP-Difícil, sobretudo quando aplicadas em conjunto com métodos de busca local para o refinamento das soluções. Os resultados demontram que o algoritmo híbrido incluindo ACO e busca local, obteve resultados significativos, atingindo a resposta ótima em 94,7% dos problemas.

Palavras-chave: *scheduling*, máquina única, terceirização, meta-heurísticas, algoritmo de colônia de formigas

Sequence-dependent-setup-time scheduling problem with outsourcing allowed: applying Ant Colony Optimization

Abstract: *Many scheduling problems found in the literature are classified as NP-Hard, which means that the computational costs of the solutions within known exact mathematical methods can be very time consuming. In the case of partial outsourcing it is essential to consider the outsourcing decisions inside the scheduling problem to achieve optimal results from outsourcing. This project discusses the following issue: a single machine environment where the setup times are sequence-dependent and there is an outsourcing option. The goal is to determinate the set of jobs that will be outsourced and the production sequence of the jobs that will be performed in-house, aiming to eliminate the total tardiness of all jobs, witch is a NP-Had problem. New approaches regarding meta-heuristics, like ACO (Ant Colony Optimization) show a new horizon for this kind of issues. The hybrid algorithm, including ACO and local search methods, reached the optimal values in 94,7% of the problems.*

Key Words: scheduling, single machine, outsourcing, meta-heuristics, ant colony optimization

Lista de Figuras

Lista de Quadros

Lista de Tabelas

Sumário

1. Introdução

Para manter-se competitiva, a função produção de uma organização deve atender ao mesmo tempo os objetivos de qualidade, custo e tempo. Tal organização deve manter seus processos de produção eficazes e eficientes, otimizando indicadores como custo e tempo, além de garantir padrões de qualidade requeridos pelos clientes.

Preocupando-se com os planos de produção futuros e o controle dos planos passados (dentre as demais funções da Administração da Produção) o Planejamento e Controle da Produção (PCP) tem como seu principal objetivo balancear o suprimento e a demanda das operações, garantindo que os objetivos de desempenho de qualidade, custo e tempo sejam atingidos.

Segundo Slack *et al.* (2009), no longo prazo as decisões são baseadas nos recursos necessários para atingir os objetivos estratégicos esperados. No médio prazo é necessário avaliar o atendimento da demanda global, parcialmente desagregada. Já no curto prazo são tomadas as decisões de sequenciamento e programação da produção, ou seja, é feita a alocação dos recursos disponíveis à demanda totalmente desagregada.

Baker (1974) descreve a atividade de programação de operações como um processo de tomada de decisão que ocorre diretamente após algumas decisões serem tratadas, como por exemplo: quais serão os produtos a serem produzidos; em qual escala eles serão produzidos e quais recursos estarão disponíveis para sua produção. As decisões de programação da produção assumem as respostas para estas perguntas e se tornarão importantes em situações onde a capacidade dos recursos é fixa e associada a comprometimentos de investimentos de longo prazo.

Diante de um cenário onde o horizonte de planejamento corresponde a horas ou poucos dias, a rapidez com a qual é necessário tomar decisões é de extrema importância. No curto prazo, portanto, deve ser tomada a decisão de qual tarefa será alocada a qual recurso a cada momento. No entanto, muitos dos problemas relacionados à programação de operações podem implicar em custos computacionais extremamente altos para sua solução através de algoritmos exatos.

A resolução de problemas de *scheduling* normalmente visa minimizar ou maximizar um ou mais objetivos da produção, como prazos de entrega, somatório de atrasos, quantidade de

produtos em estoque ou tempo de execução total da atividade (*makespan).* Obter a programação ótima para alocar as tarefas no sistema produtivo implica na otimização dos critérios de desempenho analisados.

Além disso, um grande conjunto de restrições é usualmente encontrado em ambientes produtivos em operação. Um exemplo clásssico é a impossibilidade de espera entre duas operações consecutivas (comum em indústrias químicas). Sempre atender os prazos de entrega, ou seja, não permitir a entrega de tarefas com atraso, também é um objetivo importante para diversas organizações. De acordo com Mishra *et al.* (2008) os sistemas de manufatura estão se tornando mais complexos, ampliando a complexidade do trabalho dos gestores, planejadores e programadores da produção.

O advento da globalização incentivou maiores níveis de terceirização e a redefinição das fronteiras corporativas. Estruturas muitos diferentes daquelas tradicionais da velha economia estão presentes na nova economia global, necessitando uma revisão nas prioridades e indicadores de desempenho (HAYES *et al* 2005).

Apesar da literatura que trata da programação de operações ser bastante vasta, a grande variedade de problemas encontrados em indústrias traz à tona aspectos que foram pouco estudados, por exemplo, a terceirização parcial de tarefas. A terceirização parcial pode ser vista em diversos cenários de mercado como uma opção para acompanhar a demanda em sistemas produtivos com capacidade constante ou também em um cenário onde o ritmo de crescimento da capacidade da função produção é inferior ao ritmo de crescimento da demanda.

A compreensão da terceirização parcial como uma alternativa para o acompanhamento da demanda sem contar com um aumento da capacidade produtiva já existente na organização exige a análise detalhada do impacto das possibilidades de terceirização parcial para os objetivos de desempenho da produção no curto prazo. Nesse tipo de problema, considera-se que existe um conjunto de *n* tarefas que devem ser programadas para processamento no ambiente fabril da empresa ou realizadas por meio de subcontratação.

A importância desse tipo de problema é notada por Qi (2011) que afirma que “não é possível atingir o potencial máximo de benefícios entre uma relação de terceirização entre

organizações a menos que haja um planejamento da produção e *scheduling* eficazes e eficientes abrangendo a complexidade da terceirização".

Da mesma forma como a literatura de *scheduling* pouco considera possibilidades de terceirização, a literatura sobre terceirização também desconsidera aspectos de *scheduling,* por exemplo, Lee *et al.* (2000) tratam a terceirização como um meio de reduzir o número de competências centrais a organização, como a subcontratação de serviços de TI (Tecnologia da Informação). Os trabalhos que tratam da terceirização de apenas parte de um conjunto de tarefas são relativamente recentes, por exemplo, Lee e Sung (2008a,b) propõem soluções para o problema de *scheduling* onde *n* tarefas devem ser sequenciadas para execução em um ambiente de máquina única ou enviadas para execução terceirizada.

Todo problema de *scheduling* possui um ambiente de manufatura, que determina o tipo de fluxo para as ordens de produção. Ambientes produtivos podem ser classificados como ambientes de máquina única, máquinas paralelas, *flowshop* ou *jobshop* de acordo com o fluxo que as ordens de produção devem percorrer para estarem completas (c.f. SIPPER; BULFIN, 1997, SLACK *et al.*, 2009 e FERNANDES; GODINHO FILHO, 2010).

Lee e Sung (2008b) consideram dois problemas de *scheduling* em ambiente de máquina única. O problema de minimização do atraso máximo e dos custos de terceirização e também o problema de minimização do atraso total e dos custos de terceirização, ambos descritos como problemas *NP-Hard* (também chamados de NP-Difícil).

No presente trabalho, a possibilidade de terceirização apresentada por Lee e Sung (2008a,b) é considerada em um problema de sequenciamento com tempos de *setup* dependentes da sequência em um ambiente de máquina única. O problema também leva em conta a restrição da eliminação de atrasos, tendo como objetivo minimizar os custos de terceirização e garantindo que não exista nenhuma tarefa entregue com atraso. Este problema, assim como problema tratado por Lee e Sung (2008b), é NP-Difícil.

Para as tarefas terceirizadas, é considerado apenas um custo de terceirização e assume-se que todas as tarefas podem ser terceirizadas e concluídas dentro de seus respectivos prazos. A possibilidade de terceirização tem como objetivo eliminar os atrasos existentes e garantir que a

carga de trabalho alocada para o processamento no ambiente de máquina única interno à organização não gere atrasos.

A terceirização, neste caso, garante a existência de uma solução factível para todos os problemas dessa categoria, uma vez que é feita a consideração de que quando aplicada a terceirizada sempre é possível cumprir o prazo de entrega da tarefa alocada a um terceiro. Esse problema, desconsiderando a possibilidade de terceirização, já foi abordado por vários autores (c.f. Gagné *et al.* (2002), Gupta e Smith (2006), Liao e Juan (2007), e Tasgetiren *et al.* (2009)), que apresentam soluções baseadas em algoritmos evolutivos e meta-heurísticas para solução do problema com tempos de *setup* dependentes da sequência em ambiente de máquina única, caracterizado na literatura como NP-Difícil.

Quanto aos métodos disponíveis para a solução de problemas combinatórios NP-Difíceis, Sipper e Bulfin (1997), ao considerar o problema de programação de operações em ambiente de máquina única com tempos de *setup* dependentes da sequência sugerem, entre outros, a aplicação de algoritmos como o *Branch-and-Bound* que sistematicamente enumera um conjunto de possíveis sequências enquanto busca por um valor ótimo para a função objetivo do problema. No entanto, esse tipo de estratégia é ineficiente para problemas industriais em escala normalmente encontrada em situações reais – em diversos casos, o tempo de processamento necessário para esse algoritmo encontrar a solução do problema de programação de operações pode superar algumas horas, mesmo em um computador de última geração.

Dentro deste contexto, este trabalho propõe e implementa um algoritmo baseado na meta-heurística Optimização por Colônia de Formigas (*Ant Colony Optimization* – ACO), para solucionar o problema de programação de tarefas com tempos de *setup* dependentes da sequência em um ambiente de máquina única e com terceirização permitida, com a restrição de não existirem atrasos para a entrega de nenhuma das tarefas.

Diferentemente de outros algoritmos exatos, como os disponíveis em pacotes comerciais como o CPLEX (IBM ILOG CPLEX Optimization Studio), que são capazes de comprovar a solução ótima para um problema de análise combinatória, uma meta-heurística poderá fornecer a resposta ótima para o problema, mas não comprová-la. Nesse sentido é essencial que a meta-heurística implementada seja robustamente validada para o problema proposto.

O algoritmo ACO desenvolvido será, avaliado a partir da comparação de resultados obtidos com a aplicação de um modelo de programação inteira mista (MIP – *Mixed Integer Programming*), desenvolvido neste trabalho para o problema apresentado. Os problemas de teste são solucionados através do software comercial GAMS 22.2 que utiliza o algoritmo CPLEX para resolver problemas de programação inteira mista. Em seguida as soluções obtidas pelo ACO serão comparadas de acordo com a qualidade das respostas obtidas para o custo de terceirização, que deve ser minimizado, e o tempo de execução dos algoritmos.

De acordo com a revisão bibliográfica apresentada na seção a seguir, a presente pesquisa é pioneira no contexto da aplicação da possibilidade de terceirização parcial como forma de viabilização do objetivo da eliminação total dos atrasos em um ambiente produtivo de máquina única com tempos de *setup* dependentes da sequência.

Para apresentar essa pesquisa, o presente documento é dividido da seguinte forma: a seguir é apresentada a teoria acerca dos aspectos importantes ao problema: programação de operações, terceirização e ACO; na seção 3 encontra-se descrita a metodologia de pesquisa aplicada ao trabalho; na seção 4 são detalhados as soluções propostas, MIP e ACO, para o problema de máquina única com tempos de *setup* dependentes da sequência e terceirização permitida; na seção 5 são descritos os experimentos computacionais que compuseram este trabalho; na seção 6 é feita a comparação entre os resultados de cada abordagem, mostrando o grande potencial para aplicação da abordagem ACO para a solução de problemas de programação de operações e,por fim; a seção 7 conclui a respeito das descobertas mais importantes desta pesquisa e indica oportunidades para novos trabalhos.

2. Planejamento e Controle da Produção

As atividades do planejamento e controle da produção são normalmente relacionadas como uma função integrada responsável por conectar o suprimento e a demanda. Os recursos da função produção possuem a capacidade de suprir o consumidor com suas demandas específicas. A função de planejamento da produção deve ser responsável por definir em que momentos esses recursos processarão as diferentes ordens de produção dos consumidores e a função de controle da produção deve garantir que função produção siga o planejado (SLACK *et al.*, 2009).

Segundo Sipper e Bulfin (1997) as organizações são comandadas por indivíduos que tomam decisões para manter a organização na direção de seus objetivos. Em relação aos horizontes de planejamento, convencionam-se três tipos de períodos a serem analisados: longo, médio e curto prazo. As decisões no longo prazo se relacionam com os objetivos estratégicos da função produção e no médio prazo a capacidade da operação é analisada de forma agregada. No curto prazo, onde os períodos de tempo analisados são normalmente dias, ou mesmo horas, as decisões operacionais deverão fornecer com detalhes o planejamento para atender a demanda totalmente desagregada.

No curto prazo, portanto, os gerentes de produção estarão preocupados com o equilíbrio entre a qualidade, a rapidez, a confiabilidade, a flexibilidade e os custos da função produção. Segundo Slack *et al.* (2009), é improvável que os gestores tenham tempo para realizar cálculos detalhados dos efeitos de suas decisões de planejamento e controle de curto prazo sobre todos esses objetivos.

Sipper e Bulfin (1997) afirmam que a redução dos níveis de estoque e da capacidade ociosa em um sistema produtivo torna a programação da produção uma atividade crítica e essencial dentro do PCP. Estes autores ainda argumentam que muitas organizações constroem manualmente a programação para suas respectivas funções de produção. No entanto, a necessidade de gerenciar pedidos urgentes ou falhas de equipamentos pode gerar mudanças inesperadas nos planos de produção traçados.

Para superar estes problemas podem ser utilizados sistemas computadorizados para programação da produção com capacidade finita, onde dados armazenados em bases de dados

alimentam o sistema computadorizado que fornece soluções para a programação das operações baseando-se em um modelo de *scheduling* apropriado (SIPPER; BULFIN, 1997).

Segundo Slack *et al.* (2009) um sistema de suporte a decisão é aquele que auxilia o processo decisório gerencial. A partir de informações do sistema produtivo fornecidas ao sistema de suporte a decisão, como os prazos de entrega e tempo de processamento das tarefas a serem programadas, deve ser possível a criação de cenários produtivos que otimizem os objetivos de desempenho desejados para o sistema produtivo.

Diversos autores como Baker (1974), Sipper e Bulfin (1997), Slack *et al.*, (2009) e Fernandes e Godinho Filho (2010) relacionam melhorias na função de programação da produção de organizações com um grande potencial para a redução de custos e inclusive tempo para produção de seus produtos.

Para alguns conjuntos de problemas de programação de operações existem regras ou procedimentos simples e capazes de otimizar determinados objetivos de desempenho. Como, por exemplo, para a minimização no estoque em processo em um ambiente de máquina única aplica-se a regra de ordenação SPT (*Shortest Processing Time* – Menor Tempo de Processamento) que ordena as tarefas pelo volume de processamento que cada uma implica ao sistema produtivo, sendo que nesse problema os prazos de entrega são ignorados (SIPPER e BULFIN, 1997).

Também no ambiente produtivo de máquina única, a minimização do atraso médio das tarefas é facilmente obtida a partir da aplicação da regra de ordenação EDD (*Earliest Due Date* – Menor Prazo de Entrega) que ordena as tarefas em relação aos seus prazos de entrega. E a minimização do número de tarefas com atraso pelo procedimento de Moore que remove, antes de uma tarefa sequenciada com atraso, a tarefa de maior tempo de processamento e a posiciona no final da sequência de tarefas a serem processadas (BUFFA e SARIN, 1987). Algoritmos construtivos um pouco mais complexos, mas ainda de fácil implementação são facilmente encontrados na literatura (c.f. SIPPER; BULFIN, 1997, SLACK *et al.*, 2009 e FERNANDES; GODINHO FILHO, 2010).

As decisões tomadas no curto prazo fornecem à função produção o roteiro de produção dos itens a serem processados imediatamente, ou em alguns dias. Qualquer tempo gasto com o processo de tomada de decisão a respeito da sequência em que as ordens devem ser

imediatamente processadas em um ambiente produtivo implicará no adiamento do momento em que essas ordens estarão realmente disponíveis para o processamento. Tal fato dificulta a utilização de métodos exatos para a obtenção de soluções para programação da produção, visto que, em problemas que considerem, por exemplo, a dependência dos tempos de *setup* em relação à ordem que as tarefas são executadas, mesmo para o ambiente de máquina única, obter a minimização dos atrasos pode representar horas de processamento em um computador de última geração, mesmo com a aplicação de algoritmos avançados como *Branch-and-Bound* ou modelos de programação matemática.

Buscando alcançar boas soluções para um problema combinatório complexo como o apresentado, este trabalho destina-se ao estudo do problema de programação da produção com tempos de *setup* dependentes da sequência e terceirização permitida em ambiente produtivo de máquina única, onde o objetivo é a eliminação dos atrasos através da terceirização das tarefas. Este estudo é composto de: (1) o desenvolvimento e implementação de um modelo de programação inteira mista para a resolução do problema proposto e (2) o desenvolvimento e implementação de uma rotina computacional para suporte a decisão, baseada na meta-heurística ACO.

O presente trabalho concentra-se na área de programação de operações devido ao grande potencial para a redução de custos e tempo que uma solução eficiente de *scheduling* pode representar e dada à complexidade das decisões envolvidas no processo de programar operações em determinados sistemas produtivos torna-se extremamente válido o esforço da aplicação de uma técnica recente, como o ACO, a um problema NP-Difícil de programação de operações.

As seções a seguir constroem o referencial teórico da pesquisa, que trata da aplicação da meta-heurística ACO em problemas de *scheduling*, sobretudo em problemas sobre o ambiente de máquina única, dos conceitos da terceirização parcial e das estruturas da meta-heurística de otimização por colônia de formigas.

2.1. Programação de Operações

A decisão da programação das operações ocorre apenas após a função de planejamento ter respondido a respeito do que produzir, o quanto produzir e quais recursos estarão disponíveis

para produção. Nessa etapa de tomada de decisões, torna-se relevante decidir em qual sequência as ordens liberadas para produção serão processadas no ambiente produtivo.

Os gerentes de produção utilizam os resultados de programação da produção em muitos aspectos da tomada de decisão. No nível mais amplo, está o planejamento da capacidade, onde a programação da produção revelará a necessidade adicional de capacidade e também o tipo de capacidade requisitada. Em alguns casos, um *scheduling* eficiente pode melhorar índices de utilização de recursos eliminando a necessidade de investimentos em capacidade (BUFFA; SARIN, 1987).

A teoria de programação de operações centra-se principalmente nos modelos matemáticos e algoritmos relacionados à função de *scheduling*. A perspectiva teórica da área é predominantemente quantitativa, sendo seu objetivo principal capturar a estrutura do problema em uma forma matemática concisa, de maneira a traduzir os objetivos da tomada de decisão em funções objetivo explicitas e as restrições da programação de operações em restrições matemáticas (BAKER, 1974).

A eficácia e a eficiência das atividades de programação de operações têm impacto direto nos custos e *lead time* de produção de uma organização. Segundo Chase *et al.* (2006) existem quatro funções que devem ser executadas dentro das atividades de *scheduling* e controle:

1) Alocar ordens, equipamentos e pessoal para os centros de trabalho ou localizações específicas. Essencialmente para o planejamento da capacidade no curto prazo;
2) Determinar a sequência de produção das ordens;
3) Produzir, ou despachar, as ordens;
4) Controlar o chão de fábrica, envolvendo:
    a) Controlar o progresso das ordens despachadas;
    b) Expedir ordens críticas ou atrasadas.

Ainda segundo esses autores, os objetivos comuns da atividade de *scheduling* são (1) cumprir os prazos de entrega; (2) minimizar o *lead time*; (3) minimizar os custos ou tempos de *setup*; (4) minimizar o estoque em processo e; (5) maximizar a utilização dos recursos.

Para organizações diferentes os objetivos a serem atingidos através da programação da produção eficiente serão provavelmente distintos. Diferentes ambientes produtivos e restrições variadas também modificarão a maneira como a produção deverá ser programada.

É comum encontrar na literatura quatro grupos principais de ambientes de manufatura: (i) ambientes de máquina única, onde todas as tarefas devem ser realizadas por uma única máquina; (ii) ambientes de máquinas paralelas, onde as tarefas devem ser realizadas em uma das máquinas de um conjunto de máquinas que podem realizar a mesma operação; (iii) *flowshop*, onde uma tarefa possui operações a serem realizadas em máquinas distintas, sendo que a sequência de máquinas utilizadas por todas as tarefas é a mesma; (iv) *jobshop*, onde uma tarefa possui operações a serem realizadas em máquinas distintas, sendo que a sequência de máquinas utilizadas por cada tarefa não necessariamente é a mesma (MOCCELLIN, 1999).

Nagano *et al.* (2004) caracterizam os problemas de programação de tarefas em sistemas de produção de acordo com as variáveis K e M_s definindo a complexidade de cada nível dos problemas. A Figura 1 mostra que para o problema de máquina única abordado nesse trabalho $K = 1$ e $M_1 = 1$, existindo apenas uma máquina e um estágio de produção.

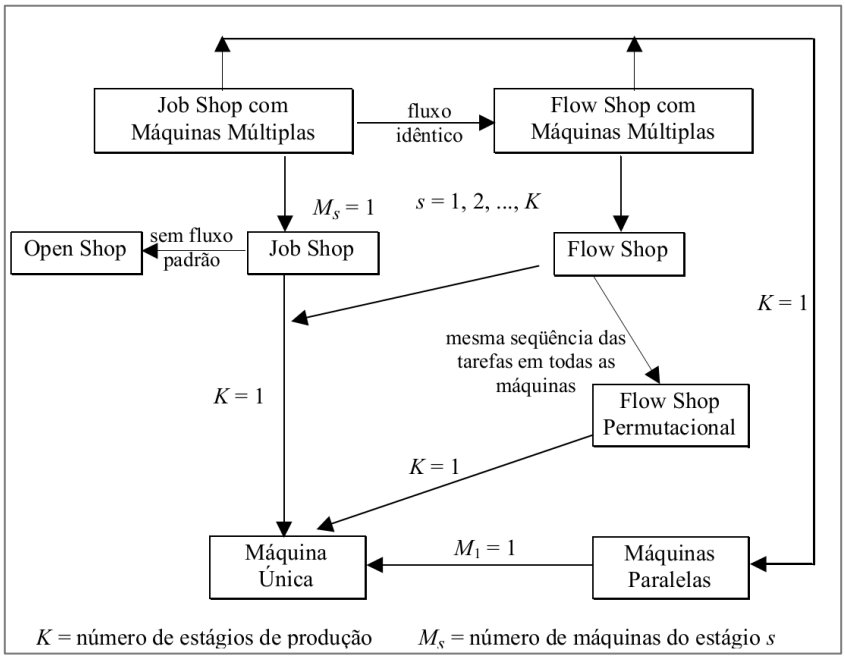


Figura 1. Relação entre as classes de problemas de programação de operações em máquinas
Fonte: Nagano *et al.* (2004, p. 115)

Allahverdi *et al.* (1998) apresentam uma revisão de literatura sobre o tema de *scheduling* classificando os problemas de acordo com as características dos tempos de *setup* das tarefas e com os ambientes de produção de máquina única, máquinas paralelas, *flowshop* e *jobshop*. Esse trabalho fornece uma revisão da pesquisa direcionada a problemas de *scheduling*, com e sem a formação de lotes de produção e também em relação à existência ou não da dependência dos tempos de *setup* em relação à ordem de processamento das tarefas. Nesta revisão 74 trabalhos apresentam problemas de máquina única, 28 apresentam problemas de máquinas paralelas, 40 focam em problemas em ambiente *flowshop* e 14 tratam de problemas em ambiente *jobshop*. Do total de 186 trabalhos revisados, 47,3% abordam problemas com a dependência da sequência das tarefas em relação aos tempos de setup (70,2% em máquina única, 53,6% em máquinas paralelas, 47,5% em *flowshop* e 14,3% em *jobshop*) os demais consideram problemas onde os tempos de setup são independentes da sequência das tarefas. No entanto, nenhum dos problemas apresentados explicita relações de terceirização entre organizações.

O problema de *scheduling* em ambiente de máquina única com terceirização permitida e tempos de *setup* dependentes da sequência é classificado como NP-Difícil, dado que o problema no mesmo ambiente sem a possibilidade de terceirização também é NP-Difícil. Esse conjunto de problemas combinatórios apresenta características que dificultam a aplicação de métodos tradicionais, como programação matemática, para sua solução devido ao imenso espaço amostral gerado com as soluções possíveis para o problema. Neste trabalho serão propostas duas alternativas de solução para o problema apresentado. A primeira abordagem, um modelo de programação inteira, tem a função de validar os resultados obtidos com a segunda abordagem, um algoritmo baseado na meta-heurística de otimização por colônia de formigas.

As duas abordagens de solução propostas para o problema possuem características distintas. O modelo MIP foi capaz de fornecer soluções ótimas do problema, quando existem recursos computacionais suficientes para tal. Mas, como é bem conhecido, o custo computacional cresce proibitivamente conforme o tamanho do problema aumenta, sendo esse o maior obstáculo para o uso de MIP em aplicações industriais reais (OMAR *et al.* 2007).

O ACO, apesar de não fornecer garantias de resultados ótimos para as respostas obtidas, é eficaz em obter boas soluções com baixíssimo custo computacional (c.f. TAVARES NETO,

2010). Segundo Tavares Neto (2012a) “como alternativa aos modelos MIP, o ACO, para o caso específico dos problemas de *scheduling*, tem provado seu valor em diversas ocasiões”.

O referencial que se segue abordará primeiramente as questões a respeito da importância da terceirização para as organizações e também dentro das decisões de *scheduling* e em seguida será apresentada a meta-heurística ACO e suas aplicações na área de programação de operações.

2.2. Terceirização

O trabalho de Wilhelm *et al*. (2013), onde é apresentado um modelo matemático para o planejamento de longo prazo de uma cadeia de suprimentos com possibilidade de subcontratação, instalação e fechamento de facilidades, aborda o tema da terceirização com características de metodologia similar ao presente trabalho. Apesar de estarem tratando de um problema diferente do escopo deste trabalho, esses autores aplicam a mesma estratégia proposta neste trabalho para a verificação e validação de seu modelo, a partir de um modelo MIP. Seus resultados mostram que de um total de 24 casos de teste, 58,3% das simulações demoram mais que uma hora de execução.

Vários trabalhos encontrados na literatura abordam esse tema como uma forma de eliminar atividades não essenciais àquelas responsáveis por agregar valor aos consumidores. Dessa forma, funções de apoio, como serviços de Tecnologia da Informação, podem ser terceirizadas para que a organização foque seus recursos nas capacidades centrais da função produção.

Nesse sentido, Gewald e Dibbern (2009) afirmam que função de TI tem sido considerada uma das principais candidatas a terceirização de processos de negócios (BPO - *Business Process Outsourcing*), o que consiste em delegar todo um processo de negócios para a terceirização. Conduzindo um *survey* esses autores analisam os riscos e benefícios da terceirização dos serviços de TI em bancos alemães.

Ravindran (2012) argumenta que apesar do tema ter atingido certa maturidade, exceto por um pequeno conjunto de boas práticas, existem poucos modelos preditivos para tomada de decisão de terceirização. Esse autor propõe um modelo para o tratamento do posicionamento dos recursos na interface entre a organização e a subcontratada que oferece os serviços de TI. Tanto

Mahalik (2011) como Shi e Dong (2012) trabalham sobre o problema quantitativo da escolha da organização para subcontratação de serviços de TI.

Várias são as decisões que devem ser tomadas quando uma organização escolhe pela subcontratação. Por exemplo, Antelo e Bru (2010) tratam de como a organização pode se valer de informações disponíveis no futuro próximo e também do aprendizado que a terceirização pode oferecer a respeito da eficiência e custos dos processos produtivos das organizações escolhidas para subcontratação. Estes autores aplicam uma abordagem com um modelo de opções reais para terceirização, mas desconsideram os aspectos de programação das operações e se voltam para as análises competitivas relacionadas à escolha da subcontratação ou da reestruturação da capacidade.

Essa metodologia de modelagem matemática com um modelo de opções reais também é apresentada por Jian e Xu (2009). Esses autores concluem que quanto maior é a volatilidade dos preços dos produtos e da demanda maiores são as opções de lucros a partir da terceirização, da mesma forma melhores opções existem quando os custos de terceirização são voláteis, mas ao contrário, quando os custos de produção interna são incertos, menores são as possibilidades para o lucro com a terceirização. Maior incerteza da demanda do mercado, dos preços dos produtos e dos custos de terceirização favorecerá a terceirização, já incerteza com relação aos custos de produção interna prejudicará a terceirização da produção.

Apesar da literatura que trata da terceirização ser ampla, é escasso o número de trabalhos que abordam a possibilidade de terceirização dentro de problemas de *scheduling*. A maioria dos trabalhos encontrados considera a tomada de decisão de terceirização no nível estratégico da organização, ou seja, relacionada ao planejamento em um horizonte de longo prazo. Por exemplo, Kok (2000) e Humphreys *et al.*(2009) tratam das decisões de “fazer” ou “comprar” do ponto de vista da estratégia de operações, ambos aplicando modelos matemáticos como técnicas para resolução.

Moon (2010) destaca a necessidade de mais modelos de suporte a decisão para a terceirização. Esse autor discute a respeito dos riscos de mercado para a terceirização, apresentando um modelo tem como objetivo determinar o período ótimo para ao investimento em terceirização parcial da produção.

Alvares e Stenbacka (2007) levantam algumas questões a respeito da terceirização parcial, por exemplo, em respeito a como determinar quando uma organização deve optar pela subcontratação e o que deve ser realizado com recursos subcontratados e como a terceirização deve ser representada na função operação. A partir de uma metodologia de modelagem e simulação, esses autores investigam tais aspectos dentro de um ambiente de incertezas de mercado para determinar resultados ótimos em relação às questões propostas apresentando um modelo de opções reais para terceirização.

Nesse contexto, o presente trabalho se diferencia dessa literatura abordando a terceirização parcial como forma de melhorar o desempenho da organização em relação às decisões de planejamento de curto prazo. Em cenários onde a demanda é incerta a terceirização parcial pode exercer papel crucial em estabilizar o suprimento e a demanda.

A seguir, a literatura sobre a consideração da terceirização parcial dentro das decisões de programação de operações é apresentada como forma de atingir o objetivo de produção da garantia de que os consumidores receberão seus produtos dentro dos prazos de entrega estabelecidos, mesmo que não exista capacidade suficiente para o processamento desses produtos com os recursos da organização.

2.3. Programação de Operações com Terceirização Permitida

A partir do advento da globalização e do desenvolvimento da tecnologia da informação a terceirização ganhou força na indústria de manufatura. No entanto, a aplicação da terceirização para a melhoria da função de programação de operações demanda a decisão entre infindáveis possibilidades de *scheduling*.

É comum que problemas de programação de operações se diferenciem conforme o ambiente produtivo para outro entre diferentes organizações. A otimização das atividades de programação da produção é importante em todos os cenários produtivos, tornando-se necessário o conhecimento de modelos capazes de solucionar os casos onde é permitido terceirizar processos produtivos a outras organizações. Tavares Neto (2012a) afirma que o conceito de permitir a terceirização parcial em problemas de *scheduling* pode aumentar a complexidade do problema, com novas estruturas e aspectos a serem tratados.

Graham *et al.* (1979) classificam os problemas de *scheduling* conforme as variáveis α/β/ϒ, que definem o ambiente produtivo (α) as restrições do problema (β) e o objetivo (ϒ). De acordo com essa nomenclatura o modelo para o problema abordado nesta pesquisa é representado por $1/s_{ij}, T_j = 0/OC$ onde $\alpha = 1$ representa o ambiente de máquina única, as restrições são a dependência dos tempos de *setup* em relação à sequência das tarefas processadas nesse ambiente e a necessidade de cumprir todas as ordens dentro do prazo (atraso zero), permitindo que algumas delas sejam terceirizadas, mas minimizando *OC* que é o custo de terceirização.

Lee *et al.* (2002) tratam da programação de operações envolvendo uma cadeia de suprimentos com a possibilidade de terceirização. Esses autores desenvolvem em seu trabalho um algoritmo genético para a solução do problema de *scheduling* que integra as decisões de em qual máquina uma tarefa deve ser processada e em qual sequência as tarefas devem ser processadas, onde o objetivo é garantir os prazos de entrega através do uso de terceirização. Nesse trabalho o modelo matemático para a solução do problema de sequenciamento é representado por uma derivação do modelo para o problema do caixeiro viajante (TSP – *Traveling Salesman Problem*).

Havendo a possibilidade de atribuir uma tarefa para terceiros a primeira consideração a ser feita é a representação do transporte entre a organização e a subcontratada para realização das tarefas, observando o transporte pode envolver variações de tempo e capital. A segunda é a necessidade de se explicitar os custos de produção que, em ambiente terceirizado, podem não ser expressos apenas em função do tempo de processamento das tarefas (indicador usualmente encontrado em problemas de *scheduling*), mas sim de um custo pago a organização subcontratada para o processamento da tarefa.

Nesse sentido, observa-se, por exemplo, os trabalhos de Lee e Sung (2008a,b), que consideram, além do tempo de processamento p_j, um prazo de entrega d_j e um custo de terceirização o_j, as tarefas em um problema de terceirização devem conter um *leadtime* l_j que expresse o tempo necessário para que a subcontratada entregue essa tarefa. No modelo descrito neste trabalho a implicação do transporte fica agregada.

Estratégias distintas podem ser aplicadas para determinar o conjunto de tarefas a serem terceirizadas, denominado o conjunto O_π, e uma sequência S_π contendo as tarefas que serão realizadas usando a capacidade produtiva interna, ordenado de forma a minimizar a soma

ponderada entre o tempo total de finalização das tarefas e o custo total de terceirização. Por exemplo, Lee e Sung (2008b) aplicam um algoritmo de programação dinâmica (*Dynamic Programming* – DP) e Tavares Neto (2010) aplica um ACO para a solução de seus respectivos problemas. Mishra *et al*. (2008) aplicam o método heurístico *Simulated Annealing* (SA) para a solução do problema de sequenciamento representado por um modelo para a cadeia de suprimentos envolvendo a possibilidade de terceirização para atender demandas excedentes a capacidade interna da organização de forma a garantir que os prazos de entrega sejam atendidos.

Para o problema de *scheduling* em ambiente de máquina única com uma opção de terceirização, Qi (2008a) apresenta soluções baseadas em programação dinâmica para os objetivos da minimização dos custos de terceirização, minimização do *makespan* e minimização do atraso máximo. Este autor mostra que os três problemas apresentados são NP-Difíceis e conclue que futuras pesquisas devem abordar diferentes condições como mais opções para terceirização de outras funções objetivo.

Qi (2008b) também aplica um algoritmo de programação dinâmica para a solução do problema de *scheduling* em ambiente *flowshop* com duas máquinas e opção de terceirização para o primeiro estágio de produção. Qi (2009) compara a solução de programação dinâmica para o mesmo problema com uma solução heurística com complexidade computacional reduzida. O autor conclui que a heurística é eficiente, sobretudo para problemas com maior número de tarefas indicando a escalabilidade do algoritmo.

Chan *et al.* (2009) enfatizam que as estratégias de suporte a decisão para programação da produção em que as organizações se apoiam, como programação linear e algoritmos como o *Branch-and-Bound*, não são válidas para os cenários modernos com alta complexidade nos sistemas produtivos. Esses autores desenvolvem um algoritmo baseado em SA, o ESCSA (*Enhanced Swift Converging Simulated Annealing*) para o problema de sequenciamento de tarefas com uma opção de terceirização na cadeia de suprimentos e concluem que mais pesquisas devem ser aplicadas à solução de problemas mais complexos e em maior escala.

Qi (2011) apresenta um sistema produtivo em dois estágios com três possibilidades de relações. Uma com capacidade interna e subcontratada para ambos os estágios, outra com capacidade externa para o primeiro e interna para o segundo estágio e a terceira com capacidade interna e subcontratada para ambos os estágios, mas com diferentes subcontratadas para cada estágio. Ele mostra soluções baseadas em programação dinâmica para os três diferentes tipos de

relações entre organizações e suas subcontratadas com objetivo de minimizar os custos de terceirização e o *makespan* das tarefas processadas dentro da capacidade produtiva da organização. Esse trabalho demonstra o potencial da terceirização em relação à melhoria do *makespan* do sistema produtivo. Analisando diferentes tipos de relações de terceirização o autor conclui que há a necessidade de um modelo mais genérico que englobe múltiplas opções de terceirização em qualquer estágio de produção e com múltiplos estágios de produção. Observa-se nessa pesquisa uma maior preocupação com as características dos modelos propostos quanto ao desempenho dos algoritmos responsáveis pela solução do problema.

Choi e Chung (2011) mostram que o problema de *scheduling* em ambiente *flowshop* com duas máquinas e terceirização permitida é NP-Difícil, mas não apresentam uma estratégia para solucionar o problema. Tavares Neto e Godinho Filho (2011a) obtêm bons resultados para a solução do problema de *scheduling*, aplicando um algoritmo baseado na meta-heurística de colônia de formigas, com objetivo da minimização do atraso total em um ambiente produtivo de *flowshop* permutacional com terceirização permitida.

Lee e Choi (2011) provam que o problema combinatório da minimização do custo total de terceirização e do *makespan* para o problema com dois estágios de produção com uma opção de terceirização no segundo estágio é NP-Difícil e propõem uma heurística gulosa para sua solução. Os autores enfatizam que problemas similares de programação da produção, como o *flowshop* com mais de duas máquinas, também são NP-Difíceis e outros objetivos de desempenho devem ser tratados por pesquisas futuras.

Tavares Neto e Godinho Filho (2011b) propõem um algoritmo com duas meta-heurísticas combinadas, ambas baseadas na meta-heurística ACO, para a solução do problema de *scheduling* em um ambiente *flowshop* com terceirização permitida. Os experimentos computacionais realizados pelos autores encorajam a aplicação da meta-heurística de otimização por colônia de formigas em outros problemas de programação de operações com a possibilidade de terceirização.

Tavares Neto (2012a) discorre sobre as opções de solução para problemas recentes que incorporam a possibilidade de terceirização. Este autor trata de diferentes ambientes produtivos e também diferentes meta-heurísticas baseadas em colônias de formigas. Para a análise da

eficiência das meta-heurísticas propostas o autor apresenta um comparativo com a uma solução MIP que utiliza o algoritmo CPLEX.

Moghaddam *et al.* (2012) utilizam a terceirização como forma de eliminar quaisquer atrasos em um problema de *scheduling* em ambiente *flowshop* com duas máquinas e reentrada de tarefas. Esses autores propõem um algoritmo genético para a resolução do problema, uma vez que esse é NP-Difícil.

Tavares e Godinho Filho (2012b) desenvolvem um ACO para a minimização de dois objetivos, a soma ponderada dos custos de terceirização e do *makespan* das tarefas sequenciadas dentro da capacidade produtiva da organização. Eles concluem que o ACO á mais eficiente que a solução apresentada por Lee e Sung (2008a).

Mokhtari *et al.* (2012) apresentam um algoritmo evolutivo para o problema de *scheduling* que visa a minimização do custo de terceirização, que é permitida como forma de satisfazer os prazos de entrega, em um ambiente produtivo de *flowshop*. Os autores apresentam diferentes cenários com diferentes políticas e capacidades de terceirização e concluem que o algoritmo proposto é capaz de encontrar boas soluções com baixo tempo de processamento para o algoritmo.

Mokhtari e Abadi (2013) propõem uma abordagem com a aplicação de programação dinâmica para o problema de *scheduling* que envolve um estágio de produção interna em ambiente de máquinas paralelas não relacionadas e um estágio com subcontratados com sistemas produtivos de máquina única capazes de realizar todas as tarefas. A terceirização é vista, por esses autores, como uma forma de aumentar o desempenho da função de programação da produção, destacando que a terceirização é uma prática comum para os momentos em que a demanda excede a capacidade da operação produtiva.

Chung e Choi (2013) provam que o problema de *scheduling* com terceirização permitida em ambiente de *flowshop* com duas máquinas é NP-Difícil. Esses autores implementam uma heurística de aproximações para resolver o problema.

Em cenários onde a demanda requisitada é superior à capacidade produtiva interna da organização, a transferência de parte da produção para um terceiro, pode garantir o

acompanhamento da demanda sem a necessidade investimento em capacidade e estoque em processo. Um plano eficiente para aplicação da terceirização pode melhorar os *lead times* de produção, reduzir os custos totais da produção e dessa forma tornar a organização mais competitiva.

Quadro 1. Trabalhos revisados que consideram a terceirização em problemas de *scheduling*

Fonte	Estratégia de Solução	Ano da Publicação
Lee *et al.* (2002)	Algoritmo Genético	2002
Lee e Sung (2008a)	Programação Dinâmica	2008
Lee e Sung (2008b)	Programação Dinâmica	2008
Mishra *et al.* (2008)	*Simulated Annealing*	2008
Qi (2008a)	Programação Dinâmica	2008
Qi (2008b)	Programação Dinâmica	2008
Qi (2009)	Heurística	2009
Chan *et al.* (2009)	ESCSA	2009
Tavares Neto (2010)	ACO	2010
Qi (2011)	Programação Dinâmica	2011
Choi e Chung (2011)	---	2011
Tavares Neto e Godinho Filho (2011a)	ACO	2011
Lee e Choi (2011)	Heurística	2011
Tavares Neto e Godinho Filho (2011b)	ACO	2011
Moghaddam *et al.* (2012)	Algoritmo Genético	2012
Tavares Neto (2012a)	ACO e MIP	2012
Tavares e Godinho Filho (2012b)	ACO	2012
Mokhtari *et al.* (2012)	Algoritmo Evolutivo	2012
Mokhtari e Abadi (2013)	Programação Dinâmica	2013
Chung e Choi (2013)	Heurística	2013

O tema que trata da consideração da terceirização como uma opção que deve ser inserida nas decisões de curto prazo, neste caso as decisões de programação de operações, é recente como pode ser visto a partir da revisão de literatura apresentada. É possível destacar que uma das principais preocupações dos autores da área é a complexidade combinatória dos problemas de decisão de programação de operações e, portanto, a maioria dos trabalhos encontrados se concentra em propor novos métodos capazes de solucionar problemas combinatórios complexos com menor tempo de execução, e fornecer respostas com alta qualidade em relação aos valores ótimos para os problemas propostos. O Quadro 1 resume a revisão de literatura construída a respeito da consideração de opções de terceirização parcial em problemas de programação de operações.

Dentre as abordagens apresentadas o presente trabalho propõe a aplicação da meta-heurística de otimização por colônia de formigas por duas razões principais. A complexidade algorítmica do ACO é menor que soluções de programação dinâmica e a segunda razão é uma característica das meta-heurísticas, especificamente o baixo acoplamento que as estruturas do algoritmo apresentam entre si, potencializando a hibridização dessa estratégia de solução com outras estratégias como a busca local, que podem ser aplicadas para a melhoria do desempenho do algoritmo. A seção a seguir trata do tema da otimização por colônia de formigas com mais detalhes.

2.4. Otimização por Colônia de Formigas

Colorni *et al.* (1991) definiram o sistema de formigas (AS – *Ant System*) como "uma abordagem distribuída para a solução de problemas e otimização baseada no resultado das iterações de baixo nível entre simples agentes cooperativos que não possuem ciência do seu comportamento cooperativo". O ACO é uma meta-heurística proposta recentemente para solução de problemas combinatórios complexos. Sua fonte de inspiração é a trilha de feromônios e o comportamento de formigas reais que utilizam feromônios como um meio de comunicação (GAMBARDELLA e DORIGO, 1996).

Na revisão a respeito de meta-heurísticas feita por Mullen *et al.* (2009), o ACO é sugerido como uma boa alternativa para a obtenção de soluções precisas e em um tempo significativamente menor para problemas de otimização combinatória classificados como NP-Difíceis. Segundo esses autores, a transição entre as formigas naturais e artificiais envolve o uso de agentes computacionais simples que trabalham em cooperação se comunicando através de trilhas de feromônios artificiais.

Analogamente as formigas reais, onde os caminhos mais curtos, entre as fontes de alimento e o ninho, recebem progressivamente maiores taxas de deposição de feromônios devido à movimentação de um número cada vez maior de formigas uma vez que essa rota mais curta é descoberta. A Figura 2 ilustra como ocorre o acúmulo de feromônios no caminho mais curto encontrado.

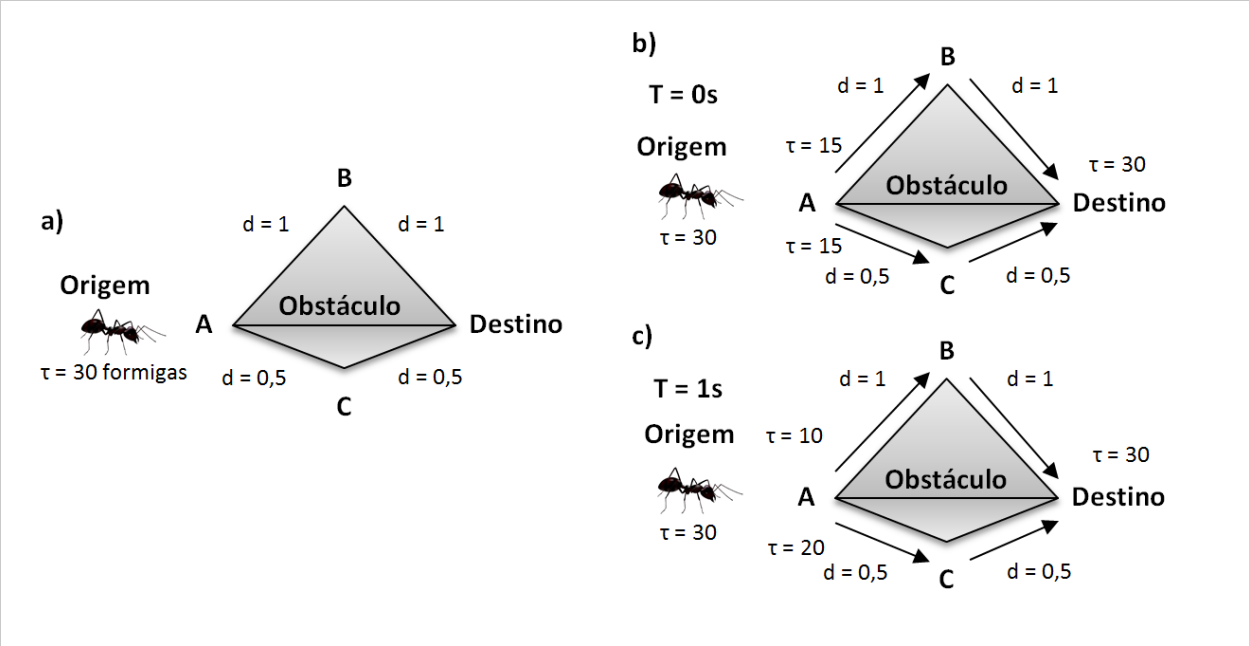


Figura 2. Um exemplo de formigas artificiais
(a) Grafo inicial com as distâncias. (b) No tempo *t = 0* não existem trilhas de feromônio nas arestas do grafo, portanto, as formigas escolhem os caminhos com probabilidades iguais. (c) No tempo *t = 1* a trilha é mais forte nos caminhos mais curtos, portanto, mais formigas escolhem esse caminho.
Fonte: Adaptado de Dorigo *et al.*(1996, p. 30, Fig. 2)

Em um ACO as formigas, ou agentes computacionais, são responsáveis por percorrer os nós de uma solução factível para o problema. As soluções, para o mesmo problema, apresentadas por diversas formigas são avaliadas e então as atualizações locais e globais nos níveis de feromônio são executadas (DORIGO *et al.*, 1996).

```
procedimento metaheurísticaACO
        AtividadesDeScheduling
                GerenciarTransiçãoDasFormigas ()
                GerenciarFeromônios ()
                RefinarSolução () {opcional}
        Fim das AtividadesDeScheduling
Fim da metaheurísticaACO
```

Figura 3. Pseudocódigo ACO
Fonte: Adaptado de Dorigo e Stützle (2002)

A meta-heurística baseada em colônia de formigas pode ser formalmente definida de acordo com as três funções básicas presentes na Figura 3. Deve existir um sistema que gerencia as regras de transição das formigas sobre a região factível do problema e uma estratégia de

deposição e evaporação de feromônios artificiais. Outras estratégias podem ser opcionalmente adicionadas para alcançar determinados objetivos, como a utilização de algoritmos de busca local para a melhoria das respostas obtidas pelas gerações de formigas.

Segundo Gambardella e Dorigo (1996) a regra de transição é definida conforme dois aspectos: as informações heurísticas do problema na forma de uma função denominada visibilidade (η); e uma matriz (ζ) que armazena as trilhas artificiais de feromônio. Esses autores também apresentam duas diferentes regras para a atualização dos níveis de feromônio artificial: i) uma regra de atualização local, realizada por cada agente da colônia; e ii) uma regra de atualização global a partir dos resultados anteriores dos agentes. Juntamente com uma regra de evaporação dos níveis de feromônio, as regras de atualização são responsáveis por atualizar informações sobre as soluções exploradas de um problema combinatório.

A função de visibilidade é construída a partir das informações disponíveis para a solução do problema, por exemplo, as distâncias de transporte o problema clássico do caixeiro viajante, utilizadas por Dorigo *et al.* (1996) e outros. Essas informações constituem o componente heurístico do algoritmo e devem ser ponderadas por fatores a serem determinados de maneira a beneficiar os objetivos de otimização do problema.

Dorigo e Stützle (2002) relacionam a importância desses fatores à exploração das soluções factíveis. A determinação dos fatores pode levar a obtenção de soluções locai sem estágios iniciais do algoritmo, em alguns casos estratégias que ampliam a exploração dos dados podem fornecem melhores soluções, sendo importante analisar o balanço de exploração das soluções factíveis.

A determinação desses fatores também pode ser automatizada. Por exemplo, López-Ibáñez *et al.* (2011, p. 1) desenvolveram o software IRACE com o propósito de configurar automaticamente os parâmetros de algoritmos de otimização conforme testes realizados em um conjunto de *benchmark*. Segundo os autores o IRACE está focado "particularmente na configuração automática para meta-heurísticas, ou seja, algoritmos de otimização de propósito geral como algoritmos evolutivos, ACO e outros algoritmos estocásticos".

Em alguns casos também pode ser necessário o refinamento das soluções obtidas a partir da exploração de soluções factíveis realizada pelos agentes do algoritmo. Nesses casos diferentes

estratégias podem ser implementadas de acordo com as características dos problemas combinatórios sendo tratados. Por exemplo, Gagné *et al.* (2002) aplicam uma etapa de busca local baseada no algoritmo 3-opt.

A meta-heurística ACO tem sido aplicada em diversas áreas do conhecimento que apresentam problemas de otimização combinatória complexos, por exemplo, Wang *et al.* (2011) aplicam o ACO para resolver um problema de roteamento dinâmico e alocação de espectro em redes ópticas; Pourhashemi e Ansarey (2012) demonstram a aplicabilidade do ACO para otimização dinâmica do controle de consumo de energia em um veículo híbrido; Tian e Chen (2012) utilizam a abordagem ACO para o mapeamento de profundidade com alta resolução para vídeos em três dimensões; Prabhakar e Singh (2013) aplicam ACO para otimização de redes de trafego de veículos.

Na seção 4.3 encontra-se detalhado o ACO proposto. A seguir é construído o referencial teórico a respeito da aplicação de ACO para a solução de problemas de *scheduling* em ambiente produtivo de máquina única.

2.5. ACO Aplicado a *Scheduling*

Quanto a aplicação da meta-heurística ACO para problemas de programação de operações em ambiente de máquina única, o primeiro trabalho encontrado é o de Bauer *et al.* (1999). Esse trabalho apresenta um algoritmo ACO para a minimização do atraso total em um ambiente de máquina única, onde cada tarefa j ($j = 1, ..., n$) está disponível no tempo zero e tem um tempo de processamento p_j. As variáveis d_j e C_j representam os prazos de entrega e os tempos de término e $T_j = max\{0, C_j - d_j\}$ é o atraso para cada tarefa. Para a potencialização dos resultados, um algoritmo de busca local foi aplicado juntamente com o ACO. Os autores utilizaram *benchmarks* já existentes na literatura para este problema como forma de validação do algoritmo proposto, e destacam que o ACO obteve bons resultados, sobretudo para grandes instâncias, aquelas com 100 tarefas.

Merkle e Middendorf (2000) apresentam uma solução baseada na meta-heurística ACO para o problema de *scheduling* com pesos e minimização do tempo total de atraso em ambiente de máquina única. Den Besten *el al*. (2000) trata do mesmo problema, mas implementa um algoritmo de busca local que vasculha a vizinhança das respostas geradas pelo ACO, melhorando

significativamente a qualidade das soluções geradas. Ambos os autores realizam simulações sobre os problemas de teste da biblioteca *ORLibrary* para o problema de máquina única com pesos.

Para o problema com tempos de *setup* dependentes da sequência, Gagné *et al.* (2002) apresentam uma comparação do algoritmo ACO com outras heurísticas, concluindo que o ACO é capaz de resolver tal problema para tamanhos reais encontrados em situações industriais. O ACO proposto por esses autores é resultado da combinação da meta-heurística de colônia de formigas com um algoritmo de busca local 3-opt.

Merkle e Middendorf (2003) comparam os resultados do ACO para o problema com pesos e o problema sem pesos em ambiente de máquina única. Esse autores comparam os resultados da meta-heurística desenvolvida com os apresentados por Congram *et al.* (2002) que resolvem o problema a partir de um algoritmo de busca local.

Holthaus e Rajendran (2005) apresentam um ACO, com a aplicação subsequente de procedimentos de busca local, para o problema de *scheduling* em ambiente de máquina única, objetivando a minimização do atraso total ponderado. Esses autores validam seu algoritmo fazendo uma comparação de *benchmark* com os dados de outros algoritmos para o mesmo problema disponíveis na literatura.

No mesmo ambiente, mas para o problema da minimização do atraso total com pesos, Liao e Juan (2007) também propõem um algoritmo de colônia de formigas com busca local e observam que ele supera 86% dos resultados encontrados para as instâncias do *benchmark* apresentado.

Kashan e Karimi (2008) apresentam um *framework* baseado na meta-heurística de colônia de formigas para o problema de *scheduling* com formação de lotes em ambiente de máquina única e famílias incompatíveis. Os autores concluem que o *framework* proposto, denominado ACF (*Ant Colony Framework – Framework* de Colônia de Formigas) supera os demais algoritmos apresentados para comparação, principalmente para instâncias com maior número de tarefas.

Preocupados com a exploração eficiente das soluções factíveis, Cheng *et al.* (2008) propõem uma adição de uma estratégia caótica ao algoritmo de colônia de formigas, denominado CACO (*Chaotic Ant Colony Optimization*), que amplifica a aleatoriedade de escolha do movimento dos agentes do algoritmo, para ampliar a exploração dos dados e evitar que o algoritmo retorne soluções ótimas locais. O ambiente nesse trabalho é de máquina única, mas para produção de lotes não idênticos, com tempos de *setup* dependentes da sequência na qual as tarefas são processadas. Os autores geraram os problemas de teste com diferentes distribuições para os tempos de processamento e os tempos de *setup* das tarefas e validaram o algoritmo proposto através da comparação com outros dois métodos de solução, *simulated annealing* e um algoritmo genético.

Anghinolfi *et al*. (2008) desenvolvem um algoritmo de colônia de formigas adaptativo para minimizar a necessidade de afinamento inicial dos fatores para o problema de *scheduling* com pesos e tempos de *setup* dependentes da sequência. Esses autores definem um estágio de busca local, que explora randomicamente a vizinhança da solução gerada a cada geração de formigas e validam o algoritmo proposto comparando-o com um *benchmark* disponível na literatura para o problema analisado.

Xu *et al.* (2008a) e Xu *et al.* (2008b) também propõem uma variação da meta-heurística ACO para a solução de um problema de *scheduling* em ambiente de máquina única. Em ambos os trabalhos as meta-heurísticas apresentadas são validadas a partir da comparação com *benchmarks* de outros algoritmos disponíveis o problema abordado.

M'Hallah e Alhajraf (2008) propõem o ACH (*Ant Colony Optimization Hybrid Heuristic*) que representa o resultado da aplicação de um ACO com uma heurística que combina métodos de busca local e algoritmos genéticos. O desempenho do ACH é analisado a partir da comparação dos seus resultados e dos resultados de um ACO padrão, também desenvolvido pelos autores, e ainda os resultados ótimos obtidos com a aplicação do algoritmo CPLEX, a partir do software GAMS. O padrão de comparação utilizado foi a razão entre a resposta média do ACH a partir de 10 execuções e a solução ótima obtida a partir do CPLEX.

Thiruvady *et al*. (2009) integram o algoritmo de colônia de formigas conhecido como MMAS (ver Stützle e Hoos (2000) para a definição do MMAS) com programação de restrições

(CP – *Constraint Programming*) para paralelizar a construção de soluções do ACO e também o método de busca *beam search*. Os autores testam o desempenho de diferentes combinações entre os algoritmos apresentados para um conjunto de problemas já apresentado na literatura, mas também gerando novos problemas de teste para avaliação dos algoritmos.

Para o problema da minimização do somatório do atraso ponderado, Cheng *et al*. (2009) propõem um algoritmo híbrido com a aplicação do ACO e de um conjunto de regras de eliminação para reduzir o espaço de busca. O algoritmo proposto obtêm respostas ótimas para 99,5% das instâncias para um dos *benchmarks* apresentados e 99% no outro.

Cheng *et al*. (2010) introduzem parâmetros de incerteza, em um modelo *fuzzy*, para o ambiente de máquina única com formação de lotes de tamanhos não idênticos. Os autores aplicam a abordagem ACO em conjunto com a implementação de método estocástico para a construção das respostas de forma a ampliar a exploração das soluções factíveis para evitar a convergência imatura para soluções ótimas locais. Para avaliar o desempenho do ACO desenvolvido os autores comparam as suas respostas com as de outros dois algoritmos, Algoritmo Genético (GA – *Genetic Algorithm*) e SA, para os problemas apresentados por Melouk *et al.* (2004).

Xu *et al*. (2012), para o problema de máquina única com tamanhos de lote variáveis, avaliam o desempenho do ACO solucionando os problemas de teste gerados a partir de um algoritmo genético, o CPLEX e também outras duas heurísticas mostrando que os resultados do ACO combinado com a seleção de uma lista de candidatos para a probabilidade de escolha das tarefas tem um desempenho mais robusto que os demais algoritmos.

Madureira *et al*. (2012) demonstram a eficiência e a eficácia da meta-heurística de colônia de formigas para as instâncias da *ORLibrary* para o problema de *scheduling* da minimização do atraso total com pesos, em ambiente de máquina única. Esses autores combinam o ACO com um algoritmo de busca local para obter melhores resultados

Outros trabalhos como Gupta e Smith (2005) e Tasgetiren *et al.* (2008) também abordam o ambiente de máquina única com tempos de *setup* dependentes da sequência, mas apresentam diferentes abordagens de solução. O primeiro avalia o desempenho do algoritmo GRASP (*Greedy*

Randomized Adaptive Search Procedure) e o segundo propõe um algoritmo evolutivo discreto para solucionar o problema de *scheduling*.

Quadro 2. Trabalhos com aplicações de ACO em problemas de scheduling em máquina única

Fonte	**Problema de *Scheduling***	**Implementação**
Bauer et al. (1999)	$1 \mid\mid \sum T_i$ *e* $1 \mid\mid \sum w_i.T_i$	ACO + Busca Local
den Besten et al. (2000)	$1 \mid\mid \sum w_i.T_i$	ACO + Busca Local
Merkle and Middendorf (2000)	$1 \mid\mid \sum w_i.T_i$	ACO
Gagné et al. (2002)	$1 / s_{ij} / \sum T_i$	ACO + Busca Local
Merkle and Middendorf (2003)	$1 \mid\mid \sum w_i.T_i$	ACO
Holthaus and Rajendran (2005)	$1 \mid\mid \sum w_i.T_i$	ACO + Busca Local
Liao and Juan (2007)	$1 / s_{ij} / \sum w_i.T_i$	ACO + Busca Local
Kashan and Karimi (2008)	$1/batch$, imcompativel, $s_{ij} / \sum w_i.T_i$	ACF
Cheng et al. (2008)	$1 / batch / C_{max}$	CACO
Anghinolfi et al. (2008)	$1 / s_{ij} / \sum w_i.T_i$	ACO Adaptativo + Busca Local
Xu et al. (2008a)	$1 / batch$, não identicos / C_{max}	BCACO (*Batch Sequence Chaotic ACO*)
Xu et al. (2008b)	$1 / batch$, não identicos / C_{max}	ACO
M'Hallah e Alhajraf (2008)	$1/d_j / \sum E_j + T_j$	ACH (ACO + Busca Local +Algoritmos Genéticos)
Cheng et al. (2009)	$1 \mid\mid \sum T_i$	ACO + Cortes
Thiruvady et al. (2009)	$1 / s_{ij}$ / M	ACO (MMAS) + CP + *Beam Search*
Cheng *et al*. (2010)	$1 / batch$, não identicos / C_{max}	ACO
Xu et al. (2012)	$1 / r_{j.} s_{j,}$ batch / C_{max}	ACO
Madureira *et al*. (2012)	$1 \mid\mid \sum w_i.T_i$	ACO + Busca Local

O Quadro 2 mostra a relação dos trabalhos revisados que contemplam aplicações da meta-heurística de otimização por colônia de formigas para problemas de programação de operações em ambientes de máquina única.

Dentre todos os autores estudados nenhum deles faz considerações sobre a possibilidade de terceirização parcial de tarefas. Trabalhos que relacionam o tema da terceirização com as decisões do planejamento e controle da produção no curto prazo, focando-se em problemas de *scheduling*, são recentes como demonstra a literatura de terceirização apresentada. A seção a seguir apresenta a metodologia adotada para a presente pesquisa e a descrição das etapas que a compõem.

3. Metodologia

Para se determinar a abordagem de pesquisa do trabalho proposto, é necessário que se entenda que este parte de um modelo de um problema a ser resolvido. Este modelo é composto por variáveis mensuráveis, como: tempo de processamento, prazos de entrega e outras, que são alteradas e cujo efeito sobre a resolução do problema é mensurado, neste caso, através de uma função objetivo.

Martins (2010) apresenta alguns dos métodos mais apropriados para a pesquisa em Engenharia de Produção utilizando uma abordagem quantitativa, como:

- Experimento;
- Quase-Experimento;
- Pesquisa de Avaliação (*surveys*);
- Modelagem/Simulação.

Dentro dos métodos apresentados por Martins (2010) observa-se, baseando-se em trabalhos posteriores na mesma área, que para o presente trabalho aplica-se como método de pesquisa a modelagem/simulação, pois é possível definir essa pesquisa como quantitativa. A pesquisa quantitativa é de grande importância para a gestão de operações. Esta abordagem é vantajosa quando:

- O problema é complexo e se faz necessária uma análise quantitativa para se chegar a uma conclusão;
- O problema possui elementos de importância ou urgência como, por exemplo, problemas que levam em consideração questões de segurança;
- O problema é novo, e como tal, não existe uma "base empírica" para se tomar decisões;
- O problema é facilmente automatizado, minimizando assim o tempo gasto para sua resolução.

Uma das alternativas para programar a produção em sistemas produtivos complexos e dinâmicos é o uso de modelagem e simulação. Como mostrado em Arenales *et al.* (2007, p. 4), a partir dos processos de modelagem, análise, inferência e julgamento, é possível que as regras sejam extraídas dos ambientes produtivos e modeladas com relações matemáticas, as quais

podem ser aplicadas técnicas matemáticas e tecnologia para validação e visualização das conclusões que determinado modelo sugere.

Morabito e Pureza (2010) descrevem o processo de modelagem matemática de acordo com a Figura 4.

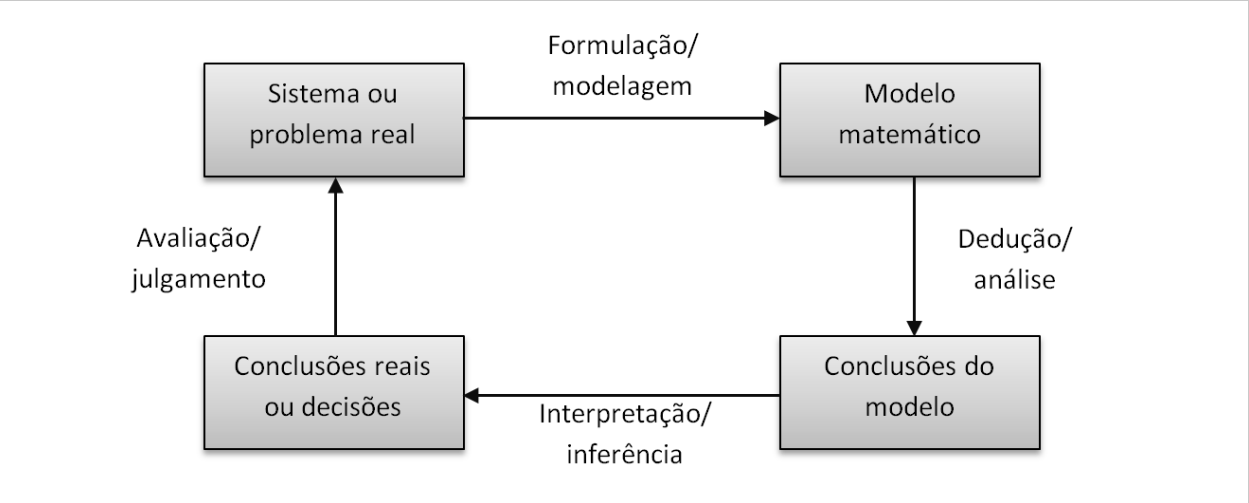


Figura 4. Processo de modelagem
Adaptado de: Morabito e Pureza (2010, cap. 8, p.180)

Yadav e Gupta (2008) fazem uma revisão das metodologias de pesquisa aplicadas ao estudo da terceirização, mostrando que diferentes metodologias estão sendo aplicadas em diferentes regiões do mundo e, levantando a necessidade de pesquisas que explorem melhor o fenômeno, com estudos que abordem diferentes ângulos como as relações de governança entre os elos da cadeia de suprimentos, os próprios relacionamentos da cadeia e a mensuração do sucesso da terceirização.

O método de modelagem e simulação possui duas diferentes abordagens: empírica e axiomática (BERTRAND; FRANSOO, 2002). Sendo a abordagem axiomática mais adequada a tratar-se da resolução de um problema proposto a partir de um modelo não empírico onde os problemas de teste serão gerados artificialmente. Esses autores subdividem a abordagem axiomática em normativa ou descritiva, sendo na normativa o objetivo central a obtenção de um modelo para o problema proposto e na descritiva a obtenção de uma solução para o modelo, como o trabalho parte da definição do modelo e o foco é a busca de uma forma mais eficiente para obtenção da solução desse modelo essa pesquisa é considerada descritiva.

Os problemas de teste serão submetidos à solução através de programação inteira mista para tal a modelagem do problema proposto será realizada na linguagem de modelagem GAMS/CPLEX 22.2. A partir dos dados das soluções geradas pelo GAMS serão analisadas duas variáveis, o custo de terceirização obtido e o tempo de processamento do algoritmo para obter essa resposta.

Em seguida os mesmos problemas de teste serão solucionados por um algoritmo ACO, que será capaz de fornecer as respostas para os problemas de teste em um tempo computacional muitas vezes menor que o algoritmo exato. E sendo o objetivo do problema tratado nesta pesquisa a minimização do custo de terceirização, espera-se que o ACO seja capaz de encontrar o mesmo custo de terceirização encontrado pelo GAMS, ou um custo vezes menor, para todos os problemas de teste.

Seguindo este objetivo de pesquisa, o método que será empregado no presente trabalho é o de modelagem/simulação axiomática descritiva, pois busca obter um meio mais rápido para solucionar o modelo apresentado. A literatura mostra aplicações da metodologia de modelagem e simulação para a solução de problemas buscando valores ótimos para o aperfeiçoamento de sistemas produtivos que permitem a subcontratação de recursos de outras organizações, porém grande parte das publicações foca-se nos aspectos da estratégia de operações, deixando de lado as questões de programaçãode operações.

A seção a seguir apresenta uma proposta baseada na meta-heurística ACO para solucionar o problema de *scheduling* com tempos de *setup* dependentes da sequência e terceirização permitida, em ambiente de máquina única, utilizando a terceirização como forma de garantir a restrição da inexistência de atrasos. São apresentadas as soluções desenvolvidas para o problema de programação de operações em ambiente de máquina única com tempos de *setup* dependentes da sequência e terceirização permitida. A seção 5 detalha como foram conduzidos os experimentos computacionais desta pesquisa e na seção 6 os resultados da aplicação das soluções propostas aos problemas de teste gerados são comparados para a avaliação do algoritmo ACO desenvolvido neste trabalho.

4. Resolvendo o problema $1/s_{ij},T_j=0/OC$

Com objetivo de definir matematicamente o problema, um modelo de programação inteira mista é inicialmente apresentado. Um conjunto de problemas de teste foi gerado e solucionado a partir do modelo. As soluções dos problemas foram armazenadas e, em seguida, o algoritmo ACO proposto, descrito na seção 4.3, é aplicado para o mesmo grupo de problemas e os dados obtidos através de ambos os algoritmos são comparados.

4.1. Definição Formal do Problema

A solução do problema da minimização do custo total de terceirização para as tarefas com pesos para a terceirização é proposta neste trabalho. O modelo proposto tem como objetivo minimizar o custo total de terceirização para as tarefas a serem processadas e ao mesmo tempo garantir que nenhuma das tarefas seja entregue com atraso. Os atrasos, representados pela variável T_j, devem ser sempre iguais a zero. É considerado que sempre é possível terceirizar uma tarefa e garantir que ela seja entregue dentro do prazo sob a pena de um custo de terceirização, definido por o_j para cada tarefa. Dessa maneira quando os prazos de entrega (d_j) não puderem ser cumpridos, será necessário a recorrer à terceirização para eliminação do atraso.

Quadro 3. Variáveis para o problema

T_j	Atraso para uma determinada tarefa j, para $j = 1,...,n$
C_j	Tempo de término de uma determinada tarefa j, caso ela seja processada em ambiente de máquina única, para $j = 1,...,n$
d_j	Prazo de entrega da tarefa j, para $j = 1,...,n$
p_j	Custo de processamento da tarefa j dentro da capacidade da organização, para $j = 1,...,n$
o_j	Custo de terceirização da tarefa j, para $j = 1,...,n$
s_{ij}	Custo de *setup* para execução da tarefa j se precedida imediatamente pela tarefa i, para $i = 1,...,n$; $j = 1,...,n$
y_j	Variável binária de decisão que define se uma tarefa j será ou não terceirizada, $y_j \in \{0,1\}$, para $j = 1,...,n$
x_{ij}	Variável binária de decisão que define a tarefa j é imediatamente precedida pela tarefa i, para $i = 1,...,n$; $j = 1,...,n$
OC	Custo total de terceirização de acordo com as tarefas j terceirizadas, para $j = 1,...,n$

As variáveis que compõem o modelo estão descritas no Quadro 3. O objetivo do problema é minimizar o custo de terceirização garantindo que nenhuma tarefa seja entregue com atraso. A função objetivo é descrita na Equação 4.1.

$$\min OC = \sum_{j} y_j . o_j \tag{4.1}$$

4.2. Modelo de Programação Inteira Mista

O modelo proposto foi implementado e validado a partir da linguagem de modelagem GAMS 22.2. As restrições do problema para sua solução utilizando-se programação inteira mista são as seguintes, onde:

• As Equações 4.2 definem o atraso para cada tarefa;

• As Equações 4.3 definem o período de término da tarefa;

• As Equações 4.4 restringem que o atraso seja igual zero;

• As Equações 4.5 garantem que todas as tarefas sejam sequenciadas ou terceirizadas;

• As Equações 4.6 evitam que as tarefas sejam sequenciadas mais de uma vez.

$$T_j \geq C_j - d_j, \forall \, \mathrm{j} \in N^* \tag{4.2}$$

$$C_j = \sum_{j} x_{ij}\left(p_j + s_{ij}\right) , \forall \, i \in N^* \tag{4.3}$$

$$T_j = 0 , \forall \, \mathrm{j} \in N^* \tag{4.4}$$

$$\sum_{j} x_{ij} + y_j = 1 , \forall \, i \in N^* \tag{4.5}$$

$$\sum_{j} x_{ij} \leq 1 , \forall \, i \in N^* \tag{4.6}$$

Além disso, é necessário eliminar a possibilidade da formação de ciclos fechados disjuntos, assim as Equações 4.7 são responsáveis por garantir o fluxo da sequência.

$$\sum_{i} x_{ki} = \sum_{j} x_{jk} , \forall \, \mathrm{k} \in N^* \tag{4.7}$$

Para eliminar a possibilidade de subcaminhos é preciso garantir que existirá apenas uma sequência para as tarefas sequenciadas em ambiente de máquina única e que esta não contenha

somente as tarefas terceirizadas. A Equação 4.8, Equação 4.9, Equação 4.10 e Equação 4.11 são responsáveis por preservar essa restrição.

$$u_0 = 1 \tag{4.8}$$

$$u_i \geq 2, \forall\, i > 1 \tag{4.9}$$

$$u_i \leq i\ + \sum_j y_j\ , \forall\, i > 1 \tag{4.10}$$

$$u_i\ -\ u_j + 1 \leq (i-1)(1 - x_{ij}), \forall\, i \in N^* \tag{4.11}$$

Tanto a matriz x_{ij} como o vetor y_j representam as variáveis binárias de decisão do problema, responsáveis respectivamente, por armazenar a sequência em que as tarefas devem ser executadas em ambiente de máquina única e quais tarefas serão terceirizadas.

$$x_{ij}, y_j \in \{0,1\}\ , \forall\, i, \mathrm{j} \in N^* \tag{4.12}$$

Todas as demais variáveis devem ser maiores ou iguais a zero.

$$p_j, d_j, o_j, s_{ij} \geq 0\ , \forall\, i, \mathrm{j} \in N^* \tag{4.13}$$

4.3. Algoritmo de Colônia de Formigas

Neste trabalho é considerado que a solução de um problema de máquina única e terceirização permitida é caracterizado por duas informações em sua solução: i) o conjunto ordenado S_π com a sequência das tarefas a serem processadas em ambiente de máquina única; e ii) o conjunto O_π com a lista das tarefas que devem ser executadas usando terceiros. Essa decisão é tomada a partir da ponderação de diversos fatores relacionados com as características do problema, que determinam a visibilidade η para cada decisão e também a partir da matriz de feromônios artificiais ζ que armazena a informação dos agentes computacionais durante a execução da meta-heurística.

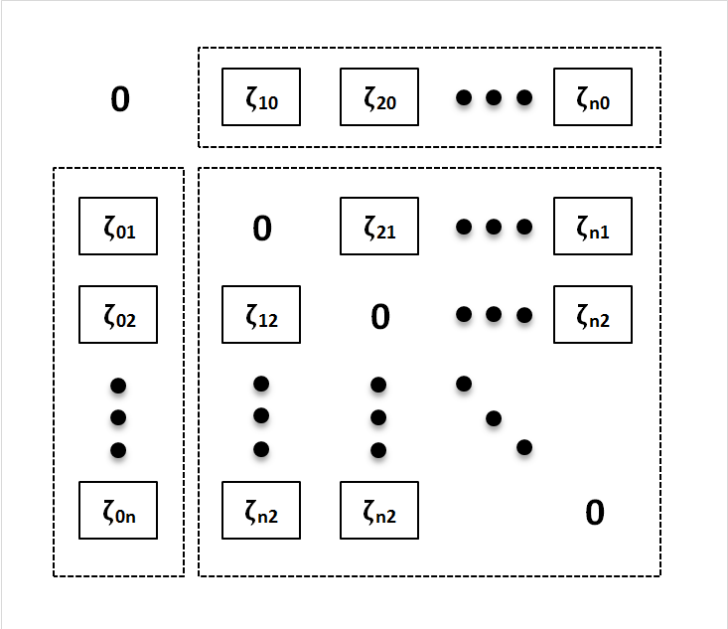


Figura 5. Ilustração da matriz de feromônios

Na matriz apresentada na Figura 5 as posições onde $i = j$ são definidas como zero por padrão. As posições restantes onde $j = 0$ representam a informação sobre a escolha da tarefa inicial e as posições onde $i = 0$ representam a informação para a escolha da terceirização das tarefas. As demais posições cujo formato é uma matriz de ordem n representam a escolha da sequência em que as tarefas serão programadas em ambiente de máquina única.

A função de visibilidade η determina a preferência heurística dos agentes computacionais por sequenciar ou terceirizar uma tarefa j imediatamente após ter sequenciado ou terceirizado uma tarefa i. No entanto, as informações heurísticas do problema, ou seja, os tempos de processamento, tempos de *setup*, prazos de entrega e custos de terceirização, influenciam de maneira diferente a escolha entre sequenciar em ambiente de máquina única ou terceirizar determinada tarefa, portanto, a função de visibilidade é definida separadamente para cada opção. A Equação 4.13 descreve as funções de visibilidade.

$$\eta_{ij} = \begin{cases} {\eta_{ij}}^1, para\ o\ caso\ das\ tarefas\ sequenciadas\ em\ máquina\ única \\ {\eta_{ij}}^2, para\ o\ caso\ das\ tarefas\ terceirizadas \\ 0, para\ o\ caso\ das\ tarefas\ já\ terceirizadas\ ou\ sequenciadas \end{cases} \quad (4.13)$$

Uma lista com as tarefas já sequenciadas e terceirizadas determina valor zero para a visibilidade da tarefa, impedindo que esse nó do grafo seja visitado novamente. A diferenciação entre as duas funções de visibilidade ocorre apenas pelos fatores que ponderam cada uma das

informações em cada função. A Equação 4.14 e Equação 4.15 mostram a regra de visibilidade proposta.

$$\eta_{ij}{}^{1} = {}^{1}\!/_{p_j{}^{\alpha 1}} . {}^{1}\!/_{d_j{}^{\alpha 2}} . {}^{1}\!/_{o_j{}^{\alpha 3}} . {}^{1}\!/_{s_{ij}{}^{\alpha 4}} \ , \forall\, i \leq\ n, \mathrm{j} \leq n \tag{4.14}$$

$$\eta_{ij}{}^{2} = {}^{1}\!/_{p_j{}^{\mu 1}} . {}^{1}\!/_{d_j{}^{\mu 2}} . {}^{1}\!/_{o_j{}^{\mu 3}} . {}^{1}\!/_{s_{ij}{}^{\mu 4}} \ , \forall\, i \leq\ n, \mathrm{j} \leq n \tag{4.15}$$

Os fatores α, na Equação 4.14, devem ser determinados de forma a favorecer apenas as tarefas onde a execução em máquina única colabora para a garantia das restrições existentes. Já os fatores μ, presentes na Equação 4.15, são determinados de forma que apenas as tarefas em que a terceirização beneficia a minimização do custo de terceirização total.

Seguindo a literatura (c.f. Dorigo *et al.*, 1996; Dorigo e Stützle, 2002), a probabilidade de escolha da próxima tarefa, determinada como "regra de transição" na literatura analisada, leva em conta a matriz de feromônios artificiais (ζ) e a funçãode visibilidade (η), juntamente com um fator β responsável pela ponderação da informação heurística determinada pela função de visibilidade. Porém, dadas as características do problema tratado nesta pesquisa, especificamente devido a opção de terceirização, foram utilizadas duas funções de probabilidade, uma responsável pela escolha do sequenciamento das tarefas em ambiente de máquina única, denominada $P_{ij}{}^{1}$, e outra responsável pela escolha da terceirização das tarefas, denominada $P_{ij}{}^{2}$. A Equação 4.16 e a Equação 4.17 mostram como a regra de transição é definida para o ACO proposto. É importante observar que $\sum_j P_{ij}{}^{1} + P_{ij}{}^{2} = 1$ para todos os casos.

$$P_{ij}{}^{1} = \zeta_{ij} . \eta_{ij}{}^{1\beta 1} \Big/ \sum_j \zeta_{ij} . \eta_{ij}{}^{1\beta 1} + \sum_j \zeta_{0j} . \eta_{ij}{}^{2\beta 1} , \forall\, i \in N^* \tag{4.16}$$

$$P_{ij}{}^{2} = \zeta_{0j} . \eta_{ij}{}^{2\beta 2} \Big/ \sum_j \zeta_{ij} . \eta_{ij}{}^{1\beta 2} + \sum_j \zeta_{0j} . \eta_{ij}{}^{2\beta 2} , \forall\, i \in N^* \tag{4.17}$$

A cada iteração do agente computacional, uma variável pseudo-aleatória entre zero e um é criada e seu valor determina se a iteração determinará o sequenciamento ou a terceirização de

uma tarefa. A Figura 6 ilustra a regra de transição implementada para o agente computacional. O método Mover() determina a cada iteração, com base nas equações descritas acima, qual a opção escolhida pelo agente computacional. As opções dadas à formiga são, dentre um conjunto de *n* tarefas, escolher uma tarefa para ser sequenciada em ambiente de máquina única ou uma tarefa para ser terceirizada. É importante observar que o total de tarefas presentes nos conjuntos S_π e O_π será igual ao total de *n* tarefas, ou seja, $n_s + n_o = n$.

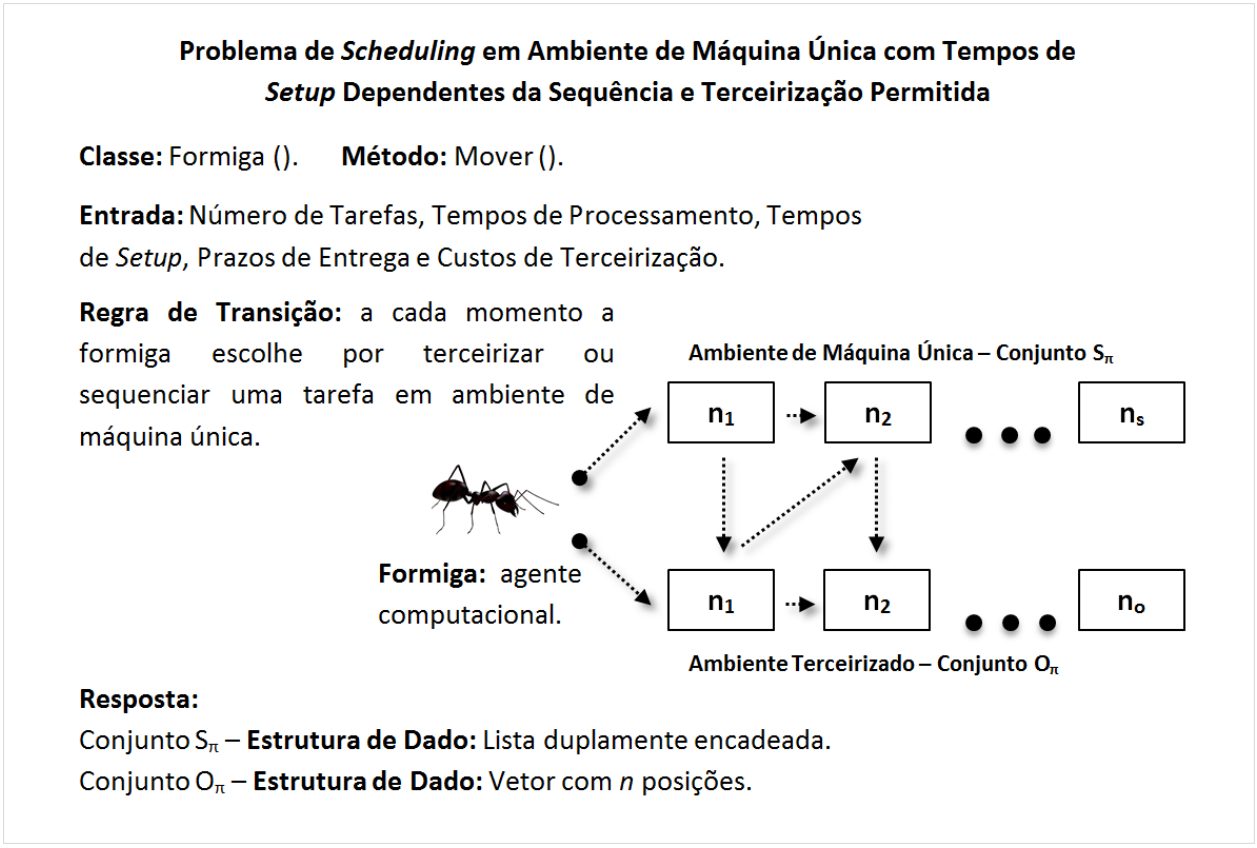


Figura 6. Ilustração do funcionamento da regra de transição do agente computacional

No método Mover() a formiga é responsável por escolher, com base na probabilidade definida na Equação 4.16 e na Equação 4.17, entre sequenciar uma determinada tarefa em ambiente de máquina única ou terceirizar uma tarefa. Caso a escolha seja sequenciar a tarefa internamente é feita uma verificação para garantir a restrição da Equação 4.4. Se a tarefa não possuir uma folga suficiente entre o seu prazo de entrega e os tempos de processamento e *setup* necessários para ser completada a visibilidade será zero, garantindo que ela será terceirizada.

Após a construção das respostas a partir da regra de transição que foi implementada no método Mover(), dois outros métodos foram construídos para refinar as soluções obtidas. As

respostas construídas a partir do agente computacional são então submetidas a um algoritmo de busca local que avalia se a solução pode ser melhorada buscando os resultados em sua vizinhança. A vizinhança das soluções criadas consiste basicamente de soluções que possuam tarefas, anteriormente terceirizadas, não mais terceirizadas, ou então soluções que possuam tarefas sequenciadas com menor custo de terceirização que as tarefas terceirizadas na solução anterior, desde que os prazos de entrega possam ser cumpridos após a troca das tarefas entre os dois conjuntos, dadas as restrições do problema.

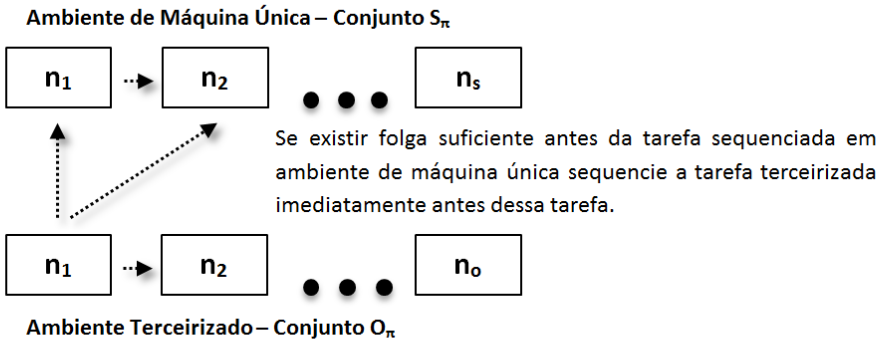
Explorar a vizinhança das soluções criadas pode ser definido como duas diferentes rotinas. Uma primeira rotina deve realizar a tentativa de inserção de tarefas que estão no conjunto O_π em possíveis folgas de tempo que existam no conjunto S_π, desde que nenhuma tarefa do conjunto S_π contenha atrasos após a inserção. A outra rotina deve realizar a tentativa da troca entre tarefas dos conjuntos, contanto que a tarefa anteriormente sequenciada em ambiente de máquina única possua um custo de terceirização menor que uma tarefa terceirizada, desde que nenhuma tarefa do conjunto S_π contenha atrasos após a troca.

O algoritmo de busca local foi implementado sem recursões, ou seja, o conjunto O_π é sempre percorrido uma única vez para a verificação de tarefas que possam ser inseridas ou trocadas. Este algoritmo possui duas funções, uma que recebe as soluções criadas e tenta inserir tarefas do conjunto O_π no conjunto S_π e outra que tenta realizar trocas entre as tarefas dos conjuntos. A busca local é aplicada apenas a melhor solução obtida a partir de uma geração de formigas.

Problema de *Scheduling* em Ambiente de Máquina Única com Tempos de *Setup* Dependentes da Sequência e Terceirização Permitida

Classe: Formiga () **Método:** ListaInsere ()

Se existirem tarefas terceirizadas, tente inserir cada uma das tarefas em todas as posições possíveis da sequência de tarefas programadas para o ambiente de máquina única.



Busca Local: Explora a vizinhança das soluções para descobrir se existem tarefas terceirizadas que poderiam ser sequenciadas sem gerar atrasos.

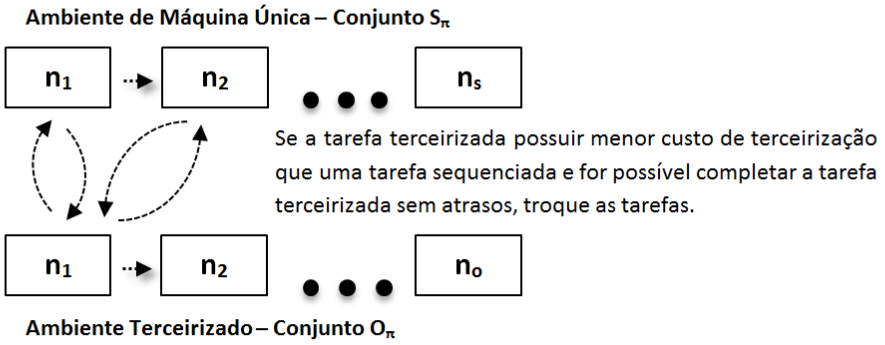
Figura 7. Método de inserção da busca local

A Figura 7 e a Figura 8 descrevem o funcionamento dos métodos de busca local implementados para o problema proposto.

Problema de *Scheduling* em Ambiente de Máquina Única com Tempos de *Setup* Dependentes da Sequência e Terceirização Permitida

Classe: Formiga () **Método:** ListaTroca ()

Se existirem tarefas terceirizadas, tente trocar cada uma das tarefas por uma tarefa sequenciada em ambiente de máquina única que possua menor custo de terceirização que a tarefa atualmente terceirizada, desde que seja possível completar a tarefa sem atraso em ambiente de máquina única na posição da sequência onde a troca ocorrer.



Busca Local: Explora a vizinhança das soluções para descobrir se existem tarefas terceirizadas com custo de terceirização maior que uma tarefa previamente sequenciada.

Figura 8. Método de Troca da Busca Local

Para o ACO proposto foram utilizadas seis formigas a cada geração e a atualização dos níveis de feromônios é feita de forma elitista, excluindo as atualizações locais e privilegiando apenas a melhor resposta obtida a cada geração. As seis soluções são avaliadas e apenas a melhor terá os valores da matriz de feromônios artificiais, que representam a sequência determinada para as tarefas em ambiente de máquina única e as tarefas terceirizadas, bonificados.

Definindo como ACO o algoritmo baseado na meta-heurística de colônia de formigas sem a utilização das rotinas de refinamento, e como ACOBL o algoritmo com a aplicação dos métodos de busca local descritos, a Figura 9 descreve o funcionamento do algoritmo implementado.

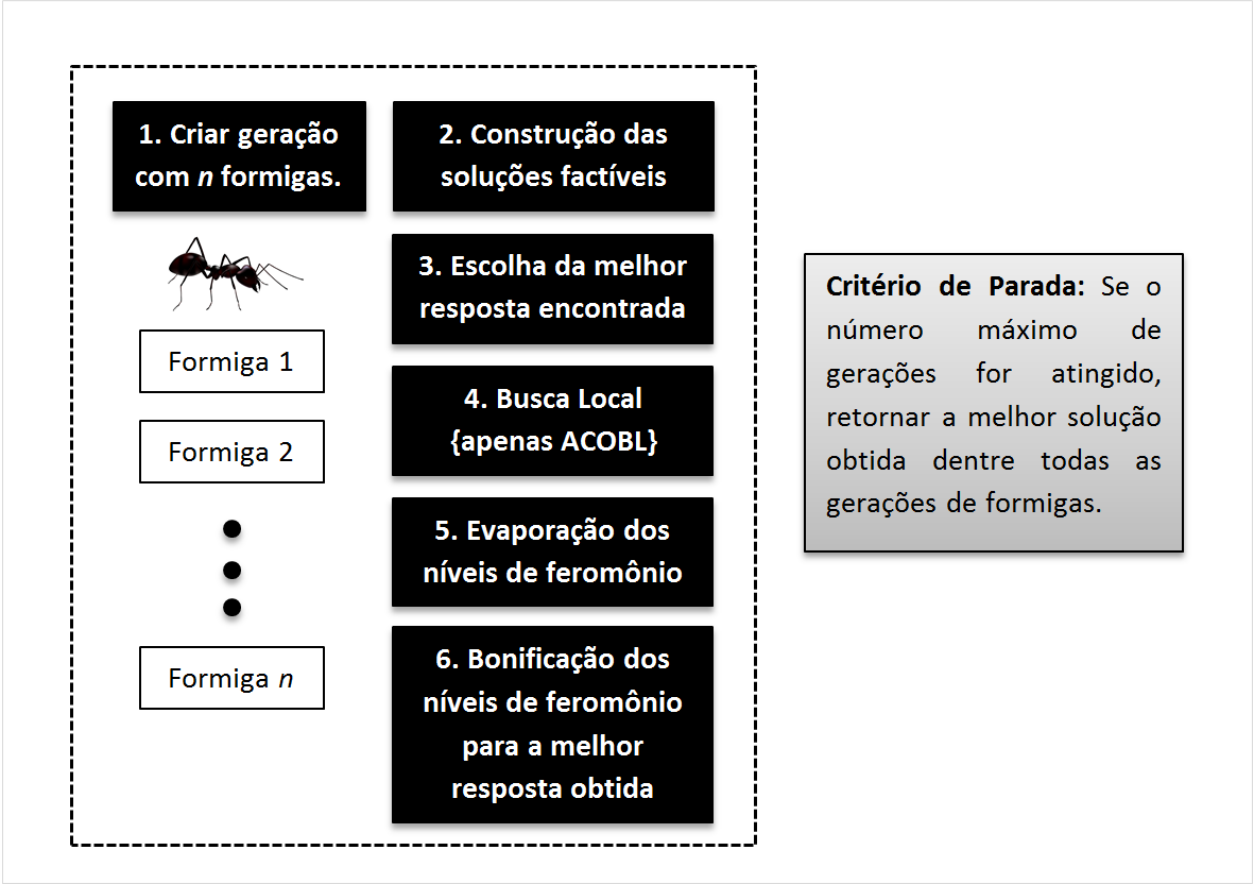


Figura 9. Passos do ACO implementado

Para que os agentes computacionais sejam capazes de encontrar melhores respostas em suas execuções os parâmetros que determinam a função de visibilidade (η) e também o fator β devem ser definidos para tal. A definição destes parâmetros também representa um problema combinatório, onde determinados os conjuntos de valores para cada um dos parâmetros, o problema consiste em determinar quais valores de parâmetros são capazes de direcionar os agentes computacionais para a resposta ótima do problema de programação de operações dentro do limite de tempo de execução estipulado para a meta-heurística.

A determinação destes parâmetros para uma meta-heurística é denominada como afinação, ou *tuning*, de acordo com a literatura revisada. No presente trabalho a afinação foi realizada a partir do pacote IRACE[1] para o software R. Ambos os softwares disponíveis gratuitamente para o sistema operacional Linux. Esse algoritmo é responsável por analisar

[1] **Nota:** O pacote IRACE está disponível a partir do endereço http://iridia.ulb.ac.be/irace/.

estatisticamente a relação entre os parâmetros para a execução da meta-heurística, determinando qual o melhor conjunto para a solução dos problemas de teste fornecidos.

Uma das entradas do software IRACE representa os conjuntos de valores dentro dos quais o software realizará experimentos para a obtenção dos melhores valores para os parâmetros da meta-heurística. O Quadro 4 apresenta os conjuntos definidos, de acordo com a sixtaxe do IRACE, para o *tuning* do ACO proposto.

Quadro 4. Domínios numéricos dos parâmetros da meta-heurística

Parâmetros do ACO	**Conjunto Numérico**
Máximo de Feromônio (MAXF)	c (5, 10, 15)
Mínimo de Feromônio (MINF)	c (20, 30, 40)
Taxa de Bonificação (TB)	c (1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0)
Taxa de Evaporação (TE)	c (0.85, 0.86, 0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99)
μ_1	r (0.001, 5.0)
μ_2	r (0.001, 5.0)
μ_3	r (0.001, 5.0)
μ_4	r (0.001, 5.0)
α_1	r (0.001, 5.0)
α_2	r (0.001, 5.0)
α_3	r (0.001, 5.0)
α_4	r (0.001, 5.0)
β_1	c (0.85, 0.86, 0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.0)
β_2	c (0.85, 0.86, 0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.0)

Os experimentos que seguem têm como objetivo avaliar o desempenho do ACO desenvolvido neste trabalho em relação à qualidade da resposta e também ao tempo necessário para obtê-la. Serão analisados duas composições diferentes das estruturas implementadas, a primeira, o ACO consiste apenas da implementação da regra de transição e da estratégia de gerenciamento dos níveis de feromônios, e o ACOBL consiste da adição dos métodos de busca local ao primeiro algoritmo.

5. Experimentos Computacionais

Os experimentos computacionais realizados foram divididos em dois grupos: o grupo de Experimentos Preliminares e o grupo de Experimentos Finais. Ambos os experimentos tem como objetivo avaliar o desempenho do algoritmo desenvolvido para a solução do problema proposto, no entanto, a realização do primeiro experimento levantou algumas questões de pesquisa que deram origem a um novo experimento.

Em relação à limitação de tempo para a execução do ACO, as características da meta-heurística, observadas na literatura, mostram que o algoritmo deve obter melhores resultados que o modelo MIP, dada a restrição de tempo imposta ao algoritmo CPLEX, sobretudo para os problemas de teste com maior número de tarefas. Em todos os experimentos apresentados nessa pesquisa o número de gerações de forminhas utilizadas para o ACO e ACOBL foi 1000, baseando-se na literatura, por exemplo, Dorigo *et al.* (1996) definem este valor como 5000 para todos os experimentos realizados. Estes autores destacam que os valores utilizados para este parâmetros devem ser definidos pelo usuário de acordo com a disponibilidade de recursos para a obtenção da solução.

Também é bem conhecido que os parâmetros da meta-heurística influenciam significativamente o comportamento do algoritmo ACO, tanto em relação à sua capacidade de convergência quanto à sua capacidade de obter uma resposta com valor mais próximo ao ótimo. Portanto, para o funcionamento da meta-heurística, é necessário que estes parâmetros que definem o comportamento do algoritmo sejam determinados de maneira que o ACO seja capaz de encontrar as respostas ótimas para os problemas.

Sobre esse tema, Birattari (2009) apresenta a definição formal do problema da configuração de uma meta-heurística. O software IRACE, apresentado por López-Ibáñez *et al.* (2011), que implementa o teste estatístico não paramétrico de Friedman, recebe como entradas o software de uma meta-heurística, um ou mais problemas de teste e os conjuntos numéricos que define o domínio para os valores dos parâmetros da meta-heurística. A partir da realização dos experimentos, que o usuário pode configurar para a operação o IRACE, o software fornece como saída os cinco melhores conjuntos com os valores para os fatores encontrados nos experimentos executados.

Em ambos os grupos de experimentos, existe a necessidade de se determinar os parâmetros presentes na função de visibilidade, na regra de transição da meta-heurística e também nos procedimentos de gerenciamento da matriz de feromônios. Neste trabalho a determinação desses parâmetros foi realizada de forma automática a partir do software IRACE.

Todos os experimentos computacionais foram realizados em uma máquina com processador Intel(R) Core(TM) i7-3770 CPU @ 3.80GHz com 12Gb de memória RAM. Os experimentos com o software GAMS 22.2 utilizaram o sistema operacional Windows 7 Service Pack 1 (64 bits) e os experimentos com o software IRACE e com os algoritmos baseados na meta-heurística ACO implementados utilizaram o sistema operacional Linux Mint 15 (64 bits) instalados na mesma máquina. A seguir, é descrita a forma como foram gerados os problemas de teste para os experimentos e as características destes experimentos, incluindo aqueles realizados com o IRACE para a determinação dos parâmetros da meta-heurística.

5.1. Geração dos Problemas de Teste

Para validação do algoritmo uma amostra de problemas foi gerada para os experimentos. A geração dos dados aleatórios foi feita a partir de um gerador de números pseudo-aleatórios implementado em C/C++ no decorrer da pesquisa. Os arquivos com os problemas gerados possuem problemas de teste consecutivos de acordo com o tamanho de cada problema.

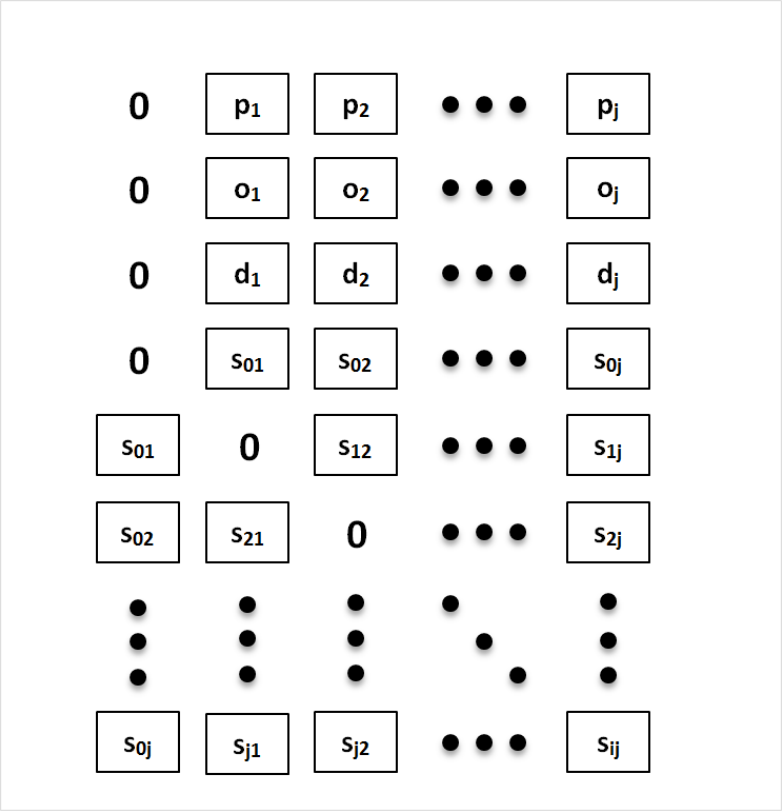


Figura 10. Ilustração do formato dos dados aleatórios gerados

Com relação à interface de execução do sistema desenvolvido neste trabalho, um arquivo com extenção de texto (txt) deve conter os problemas de teste um após o outro para determinado tamanho do problema, o ACO implementado recede como entrada este arquivo além de diversos parâmetros como o número de gerações, os valores para os parâmetros determinados a partir do IRACE e o tamanho do problema a ser solucionado. A Figura 10 ilustra o formato de um problema de teste como uma matriz de dados.

5.2. Características do Experimento Preliminar

Como não existe *benchmark* disponível para o problema proposto nesta pesquisa, foi necessário criar um conjunto de problemas de teste para a avaliação dos algoritmos. Os dados do primeiro experimento basearam-se nos dados apresentados pela biblioteca *ORLibrary* (BEASLEY, acessado em Julho de 2013), para o problema em ambiente de máquina única com tempos de *setup* dependentes da sequência.

O principal objetivo do experimento preliminar é avaliar a escalabilidade dos dois algoritmos. Para atingir esse objetivo serão gerados problemas com 20, 40, 60, 80 e 100 tarefas.

Nestes problemas de teste, os tempos de processamento e os custos de terceirização das tarefas obedecem a uma distribuição uniforme [1,100] enquanto os tempos de *setup* obedecem à distribuição uniforme [1,10]. O experimento preliminar lida apenas com uma configuração entre os tempos de processamento e os tempos de *setup*.

Já os prazos de entrega são determinados de acordo os valores das variáveis TF (*Tardiness Factor*) e RDD (*Relative Due Date*) e correspondem à distribuição uniforme $\left[P\left(1 - TF - \frac{RDD}{2}\right), P\left(1 - TF + \frac{RDD}{2}\right)\right]$, onde TF um fator de atraso, RDD o alcance relativo das datas de entrega e P representando a soma dos tempos de processamento das tarefas, $P = \sum_{j=1}^{n} p_j$. A Tabela 1 descreve as possibilidades de distribuição utilizadas para os prazos de entrega nos dados da biblioteca *ORLibrary*.

Tabela 1. Valores de RDD e TF e as distribuições dos prazos de entrega para a *ORLibrary*

TF	RDD	Conjunto para Distribuição Uniforme dos Prazos de Entrega
1,0	1,0	[(-0,5 x P),(0,5 x P)]*
1,0	0,8	[(-0,4 x P),(0,4 x P)]*
1,0	0,6	[(-0,3 x P),(0,3 x P)]*
1,0	0,4	[(-0,2 x P),(0,2 x P)]*
1,0	0,2	[(-0,1 x P),(0,1 x P)]*
0,8	1,0	[(-0,3 x P),(0,7 x P)]*
0,8	0,8	[(-0,2 x P),(0,6 x P)]*
0,8	0,6	[(-0,1 x P),(0,5 x P)]*
0,8	0,4	[(0 x P),(0,4 x P)]*
0,8	0,2	[(0,1 x P),(0,3 x P)]
0,6	1,0	[(-0,1 x P),(0,9 x P)]*
0,6	0,8	[(0 x P),(0,8 x P)]*
0,6	0,6	[(0,1 x P),(0,7 x P)]
0,6	0,4	[(0,2 x P),(0,6 x P)]
0,6	0,2	[(0,3 x P),(0,5 x P)]
0,4	1,0	[(0,1 x P),(1,1 x P)]
0,4	0,8	[(0,2 x P),(1,0 x P)]
0,4	0,6	[(0,3 x P),(0,9 x P)]
0,4	0,4	[(0,4 x P),(0,8 x P)]
0,4	0,2	[(0,5 x P),(0,7 x P)]
0,2	1,0	[(0,3 x P),(1,3 x P)]
0,2	0,8	[(0,4 x P),(1,2 x P)]
0,2	0,6	[(0,5 x P),(1,1 x P)]
0,2	0,4	[(0,6 x P),(1,0 x P)]
0,2	0,2	[(0,7 x P),(0,9 x P)]

*** Os conjuntos assinalados foram ignorados para a geração dos problemas de teste.**

Para gerar os problemas de teste do experimento preliminar foi utilizada uma estratégia similar à aplicada na biblioteca *ORLibrary*, mas foram ignorados os resultados ($TF + RDD/2 \geq 1$) para eliminar os intervalos valores negativos e garantir que os prazos de entrega possuam um valor mínimo de 10% da variável *P*, totalizando 14 níveis de problemas de teste para os quais foram gerados três problemas cada.

Por não existir base de dados comparativa os problemas gerados foram solucionados primeiramente a partir do software GAMS, que invoca o solver comercial CPLEX. No entanto, sabe-se que esse algoritmo apresenta dificuldades em solucionar grandes problemas de teste com os recursos computacionais (tempo de processamento e memória) disponíveis. No caso, foi necessário impor uma restrição de tempo de execução para as tentativas de resolução usando programação matemática: todos os problemas receberam um tempo máximo para execução do algoritmo de 9000 segundos, considerado um parâmetro razoável para a obtenção de um plano de produção para a programação das operações que é considerado no horizonte de curto prazo das organizações.

O experimento preliminar analisou a escalabilidade do algoritmo gerando problemas de teste com 20, 40, 60, 80 e 100 tarefas. Dessa maneira o experimento foi composto de 42 problemas de teste de cada tamanho, sendo três problemas para cada nível, que foram submetidos à solução através do CPLEX e do ACO proposto neste trabalho.

Os dados dessas execuções foram armazenados e comparados com os dados da execução do algoritmo ACO proposto para os mesmo problemas de teste. O tempo necessário para a execução do ACO não é proibitivo, mas a qualidade da resposta gerada pela meta-heurística é potencialmente maior quanto maior é o tempo de execução disponível para o algoritmo.

Foi escolhido arbitrariamente apenas um problema entre cada tamanho dos problemas de teste para o *tuning* do ACO para o Experimento Preliminar com objetivo de identificar se o tamanho do problema influencia a definição destes parâmetros. Para tal os experimentos realizados a partir do software IRACE estão descritos na Tabela 2.

Tabela 2. *Tuning* do experimento preliminar com software IRACE

		Configurações do IRACE	
Experimento de *Tuning*		**Máximo de Experimentos**	**Dígitos dos Valores Reais**
1	Problema de Teste nº 1 de 20 Tarefas	1000	3
2	Problema de Teste nº 1 de 40 Tarefas	1000	3
3	Problema de Teste nº 1 de 60 Tarefas	1000	3
4	Problema de Teste nº 1 de 80 Tarefas	1000	3
5	Problema de Teste nº 1 de 100 Tarefas	1000	3

A partir das respostas destes cinco experimentos foram executadas simulações com o ACO desenvolvido para analisar os parâmetros obtidos a partir do software IRACE.

Os resultados deste experimento demonstraram que o CPLEX não foi capaz de comprovar a resposta ótima para a maioria dos problemas apresentados no tempo de 9000 segundos. Já o ACO se iguala, ou supera o CPLEX em relação à qualidade da resposta em 100% dos casos para o ACOBL e 96,6% dos casos para a implementação básica do ACO, quando considerada a melhor resposta encontrada dentre 30 execuções dos algoritmos. A seção 6.1 descreve o resultado do experimento preliminar com mais detalhes.

A realização deste experimento levantou uma questão a respeito da influência da relação entre os dados de entrada do problema de *scheduling* (tempos de processamento, tempos de *setup*, prazos de entrega e custos de terceirização) e as características de ambos os algoritmos. Para obter as respostas destas questões de pesquisa, um novo experimento foi realizado e sua proposta encontra-se descrita a seguir.

5.3. Características do Experimento Final

O experimento preliminar revelou que, mesmo para os problemas de teste com 20 tarefas, o algoritmo CPLEX não foi capaz de apresentar o resultado ótimo para na grande maioria dos casos. Testes com o modelo MIP implementado revelaram que para problemas com 11 tarefas algoritmo CPLEX poderia atingir algumas horas de processamento para verificar a resposta ótima do problema, já problemas com 12 tarefas não foram solucionados em um prazo de cinco horas de processamento. Portanto, este experimento considerou apenas problemas de teste com 10 tarefas. Nota-se, no entanto, a necessidade de testar o desempenho do algoritmo de acordo com

esta nova proposta de experimento também para problemas com maior número de tarefas, por exemplo, 50 tarefas e 100 tarefas. Porém de acordo com o escopo e prazo desta pesquisa não foi possível concluir tal experimento.

Este experimento analisa a relação entre os dados de entrada para os problemas e o impacto das características desses dados na configuração dos espaços de busca derivados de cada problema. Buscou-se entender se diferentes formas de se definir tempos de *setup* e tempos de processamento das tarefas afetaram o desempenho do algoritmo, ou se os parâmetros devem ser ajustados periodicamente de acordo com as oscilações de um sistema produtivo.

A relação entre os tempos de processamento e *setup* foi analisada gerando-se problemas de teste a partir das distribuições uniforme descritas na Tabela 3.

Tabela 3. Distribuições uniformes para os tempos de processamento e *setup*

Conjunto para Distribuição Uniforme dos Tempos de Processamentos	**Conjunto para Distribuição Uniforme dos Tempos de *Setup***
[1,100]	[1,10]
[1,100]	[1,25]
[1,100]	[1,50]
[1,100]	[1,100]
[1,50]	[1,10]
[1,50]	[1,25]
[1,50]	[1,50]
[1,50]	[1,100]
[1,25]	[1,10]
[1,25]	[1,25]
[1,25]	[1,50]
[1,10]	[1,10]
[1,10]	[1,25]

Os dados dos custos de terceirização foram gerados para o experimento preliminar de acordo com uma distribuição uniforme [1,100], permitindo a diferenciação entre tarefas com maior propensão a serem terceirizadas, aquelas com baixo custo de terceirização, e tarefas com menores chances de serem terceirizadas, aquelas com alto custo de terceirização. Para analisar cenários com diferentes relações para o custo de terceirização, foram utilizadas quatro distribuições uniformes com os seguintes limites: [50,50], [35,65], [20,80] e [5,95]. Cada um desses cenários pondera de maneira diferente a relação de terceirização, quando o custo de terceirização é igual entre todas as tarefas o problema se reduz a minimização do número de

tarefas terceirizadas, no entanto, aumentando a distância entre os parâmetros da distribuição pondera-se o custo de terceirização na escolha das tarefas terceirizadas.

Os prazos de entrega foram gerados de acordo com uma simplificação da estratégia apresentada pela biblioteca *ORLibrary*. Para o limite inferior das distribuições foi utilizado o valor de (*0,2*P*) onde $P = \sum_j p_j + s_{j-1,j}$. Dessa maneira, garante-se que não existirão tarefas a serem programadas cujos prazos de entrega não podem ser cumpridos no ambiente de máquina única. Já o limite superior para as distribuições varia em quatro diferentes níveis: (*0,5*P*), (*0,7*P*), (*0,9*P*) e (*1,1*P*), gerando cenários com diferentes necessidades de terceirização. A Tabela 4 apresenta os experimentos realizados com o software IRACE.

Tabela 4. *Tuning* do experimento final com software IRACE

		Configurações do IRACE	
Experimento do *Tuning*		*Máximo de Experimentos*	*Dígitos Reais*
1	Problema de Teste nº 1	1000	3
2	Problema de Teste nº 1	10000	3
3	Problema de Teste nº 40	1000	3
4	Problema de Teste nº 40	10000	3
5	Problemas de Teste nº 2, 6, 18, 20, 31, 37, 43 e 45	1000	3
6	Problemas de Teste nº 2, 6, 18, 20, 31, 37, 43 e 45	10000	3
7	Problemas de Teste nº 55, 63, 69, 74, 74, 91, 93 e 100	1000	3
8	Problemas de Teste nº 55, 63, 69, 74, 74, 91, 93 e 100	10000	3
9	Problemas de Teste nº 109, 111, 121, 127, 132, 142, 150 e 153	1000	3
10	Problemas de Teste nº 109, 111, 121, 127, 132, 142, 150 e 153	10000	3
11	Problemas de Teste nº 157, 168, 174, 180, 188, 191, 199 e 205	1000	3
12	Problemas de Teste nº 157, 168, 174, 180, 188, 191, 199 e 205	10000	3

Com 13 diferentes configurações para os tempos de processamento e *setup* e quatro níveis para os prazos de entrega e quatro possibilidades para os custos de terceirização foram construídos 208 tipos diferentes de problemas de teste. Para o *tuning* do ACO desse experimento, foram escolhidos dois problemas de teste para experimentos com apenas um problema de teste disponível ao IRACE e também conjuntos de oito problemas dentre cada um dos diferentes níveis de problema quanto aos prazos de entrega, ou seja, foram escolhidos oito problemas entre os problemas de teste de um a 52, oito problemas entre 53 e 104, oito problemas entre 105 e 156 e oito problemas de teste entre os problemas de número 157 a 208. Além disso, foram realizados

experimentos com numero máximo de 1000 experimentos e também 10000 experimentos de acordo com as configurações disponíveis para o IRACE.

Para os experimentos de afinação foram utilizados apenas um exemplar de cada nível diferente para os problemas de teste, ou seja, 208 problemas de teste. Posteriormente um experimento mais extensivo foi conduzindo envolvendo 30 problemas de teste de cada nível, totalizando 6240 problemas de teste do problema analisado no Experimento Final. A seção 6.2 mostra com detalhes os resultados deste experimento.

6. Análise dos Resultados

A presente pesquisa forneceu resultados a respeito da qualidade de resposta dos algoritmos implementados, ACO e ACOBL, em relação a um modelo de programação inteira mista, também desenvolvido neste projeto na linguagem GAMS que utiliza o algoritmo CPLEX. As seções a seguir descrevem como estes algoritmos se comportaram nos experimentos realizados.

6.1. Análise dos Resultados do Experimento Preliminar

Dada a restrição de tempo máximo de execução de 9000 segundos, o pacote comercial utilizado (GAMS/CPLEX) foi capaz obter resultado ótimo em apenas dois problemas e com exceção de um único problema de teste (problema nº 17 com 40 tarefas) foi capaz de fornecer uma resposta factível em todos os demais problemas de teste gerados. Os dois problemas para as quais o CPLEX foi capaz de provar a solução ótima tem seus tempos de execução apresentados na Tabela 5.

Tabela 5. Respostas ótimas do modelo de programação inteira mista

	Problema de Teste nº 5	**Problema de Teste nº 21**
Número de Tarefas	20	20
Resposta Ótima (OC)	0	0
Tempo de Execução (s)	7	1985

Para as simulações com o ACO, primeiramente foram conduzidos cinco experimentos para a afinação da meta-heurística com problemas de 20, 40, 60, 80 e 100 tarefas. A partir dos experimentos com o software R e o pacote IRACE foram coletadas apenas a primeira resposta (melhor resposta) obtida pelo IRACE em cada experimento. A Tabela 6 e a Tabela 7 mostram os resultados obtidos para os 14 parâmetros do ACO.

Tabela 6. i) Resultados dos experimentos de *tuning* do ACO para o Experimento Preliminar

Tuning	**MAXF**	**MINF**	**TB**	**TE**	**β_1**	**β_2**
p1|20t	10	20	2,6	0,93	0,98	1
p1|40t	5	20	2,9	0,94	0,94	0,9
p1|60t	15	30	1,3	0,96	0,86	1
p1|80t	10	20	1,1	0,86	0,96	0,88
p1|100t	5	30	2	0,98	0,96	0,98
Sem Tuning	10	20	2,5	0,9	1	1

Tabela 7. ii) Resultados dos experimentos de *tuning* do ACO para o experimento preliminar

Tuning	α_1	α_2	α_3	α_4	μ_1	μ_2	μ_3	μ_4
p1|20t	0,863	1,891	2,001	3,293	1,704	2,869	2,620	3,574
p1|40t	4,33	2,74	1,73	3,3	4,78	2,98	3	3,94
p1|60t	4,53	1,83	2,36	4,14	3,33	4,41	2,04	1,26
p1|80t	1,39	2,98	1,21	3,6	0,42	0,48	2,2	4,14
p1|100t	2,94	4,42	1,22	4,7	1,51	3,89	4,22	2,67
Sem Tuning	1	1	1	1	1	1	1	1

Além disso, foram realizados outros cinco experimentos com o mesmo objetivo para o ACOBL. A Tabela 8 e a Tabela 9 mostram os resultados obtidos para a afinação da meta-heurística combinada com o procedimento de busca local desenvolvido.

Tabela 8. i) Resultados dos experimentos de tuning do ACOBL no experimento preliminar

Tuning	MAXF	MINF	TB	TE	β_1	β_2
p1|20t	10	30	1,7	0,9	0,99	0,92
p1|40t	15	30	1,8	0,86	1	0,95
p1|60t	15	20	1,7	0,9	0,89	0,95
p1|80t	15	30	2,1	0,99	0,9	0,87
p1|100t	5	40	3,5	0,9	0,96	0,89
Sem Tuning	10	20	2,5	0,9	1	1

Tabela 9. ii) Resultados dos experimentos de *tuning* do ACOBL no experimento preliminar

Tuning	α_1	α_2	α_3	α_4	μ_1	μ_2	μ_3	μ_4
p1|20t	2,54	1,56	0,12	3,03	4,06	4,68	2,91	2,09
p1|40t	0,47	2,58	3,99	4,76	2,55	0,23	4,9	2,14
p1|60t	1,23	4,88	4,04	1,21	0,33	1,31	0,84	1,6
p1|80t	2,66	3,66	2,24	3,32	0,04	4,42	4,35	2,67
p1|100t	2,25	4,33	1,56	4,83	1,23	0,3	3,62	1,85
Sem Tuning	1	1	1	1	1	1	1	1

Neste ponto todos os problemas de teste foram executados 30 vezes cada um utilizando cada uma das respostas de afinação obtidas, tanto com o ACO como com o ACOBL, e a melhor resposta dentre estas simulações foi coletada como resposta, de cada problema de teste, para a avaliação do comportamento dos parâmetros fornecidos pelo IRACE. A Figura 11 e a Figura 12 mostram os gráficos criados com os valores das somatórias dos custos de terceirização para os 42 problemas de cada tamanho diferente, incluindo os resultados obtidos através do GAMS e também utilizando a configuração “sem tuning” para os parâmetros, cujos valores foram determinados a partir de outros trabalhos aqui revisados (c.f. MERKLE; MIDDENDORF, 2003 e TAVARES NETO, 2010).

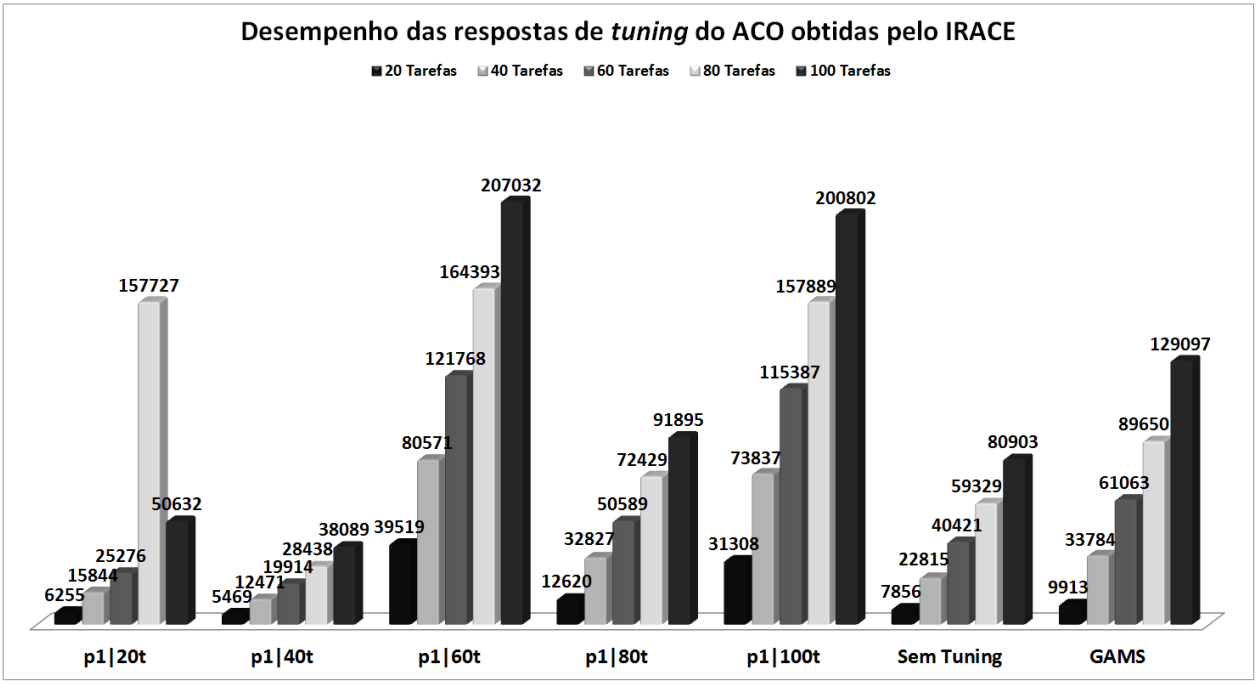


Figura 11. Desempenho das respostas de *tuning* do ACO para o experimento preliminar

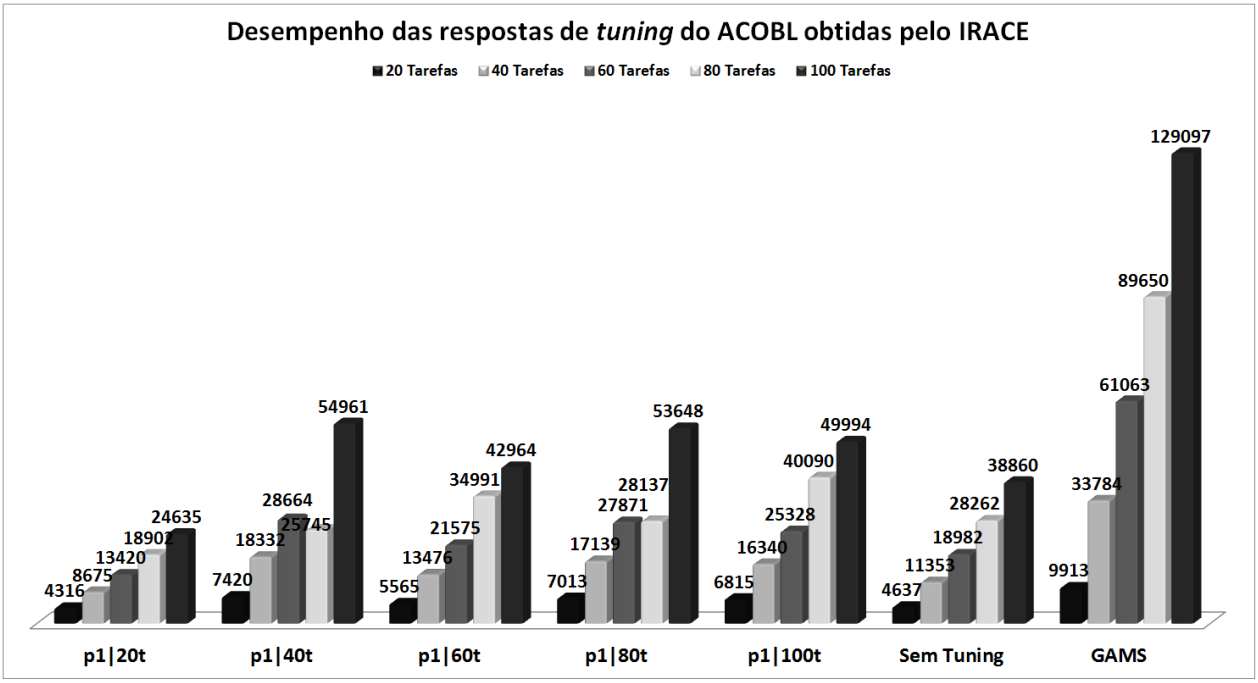


Figura 12. Desempenho das respostas de *tuning* do ACOBL para o experimento preliminar

Foi observado que, para o ACO, o experimento de *tuning* realizado com o problema nº 1 de 40 tarefas resultou na melhor afinação para todos os demais tamanhos de problemas. Da

mesma forma, para o ACOBL, o experimento de *tuning* com o problema nº 1 de 20 tarefas originou a melhor resposta para os parâmetros em todos os tamanhos de problema.

Nota-se também que diferentes respostas para estes parâmetros podem ocasionar comportamentos distintos para os algoritmos. No caso do ACO a resposta obtida da afinação utilizando o problema nº 1 de 20 tarefas apresenta um resultado ruim para os problemas de 80 tarefas, mas para os problemas de 20 tarefas suas respostas são apenas 14,4% maiores que as obtidas com a melhor afinação (p1|40t). Para o ACOBL as respostas p1|40t e p1|80t apresentam resultados melhores para problemas de 80 tarefas do que para problemas de 60 tarefas.

No caso da meta-heurísca baseada em otimização por colônia de formigas (ACO) foi observado que algumas respostas obtidas através do IRACE acarretaram um comportamento não convergente para o ACO. Já para o ACOBL as respostas divergiram entre as diferentes afinações dos parâmetros porêm todas as respostas de *tuning* testadas superaram os resultados do GAMS/CPLEX como pode ser observado na Figura 12. Também é evidente o ganho em qualidade de resposta que a aplicação da busca local proporcionou ao algoritmo inicial. No entanto, não é possível afirmar que exista uma única resposta para estes 14 parâmetros que possa operar de maneira ótima sobre todas as configurações de problemas propostas para este experimento.

A Figura 13**Erro! Fonte de referência não encontrada.** e a Figura 14 apresentam a comparação da qualidade de resposta obtida a partir do ACO e do ACOBL quando comparados ao modelo MIP implementado e solucionado pelo CPLEX. Observa-se que para em apenas 7 casos o ACO obteve respostas piores que as obtidas através do CPLEX enquanto o ACOBL supera ou iguala os resultados para todos os problemas de teste do experimento. Nota-se ainda que o ACO se iguala ao resultado do CPLEX em apenas três problemas de teste e o ACOBL em quatro problemas. O ACO foi capaz de encontrar custo de terceirização igual à zero em seis (2,8%) casos e o ACOBL encontrou custo de terceirização nulo em 27 (12,8%) casos de teste, nos quais o CPLEX retornou um custo de terceirização positivo.

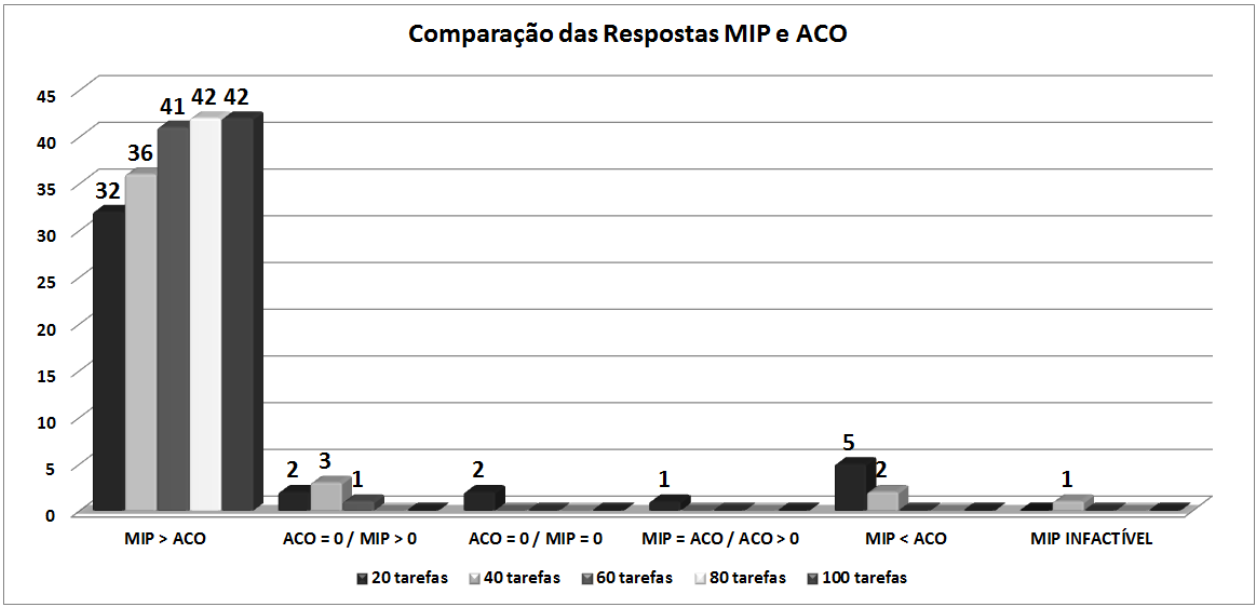


Figura 13. Comparação das respostas MIP e ACO

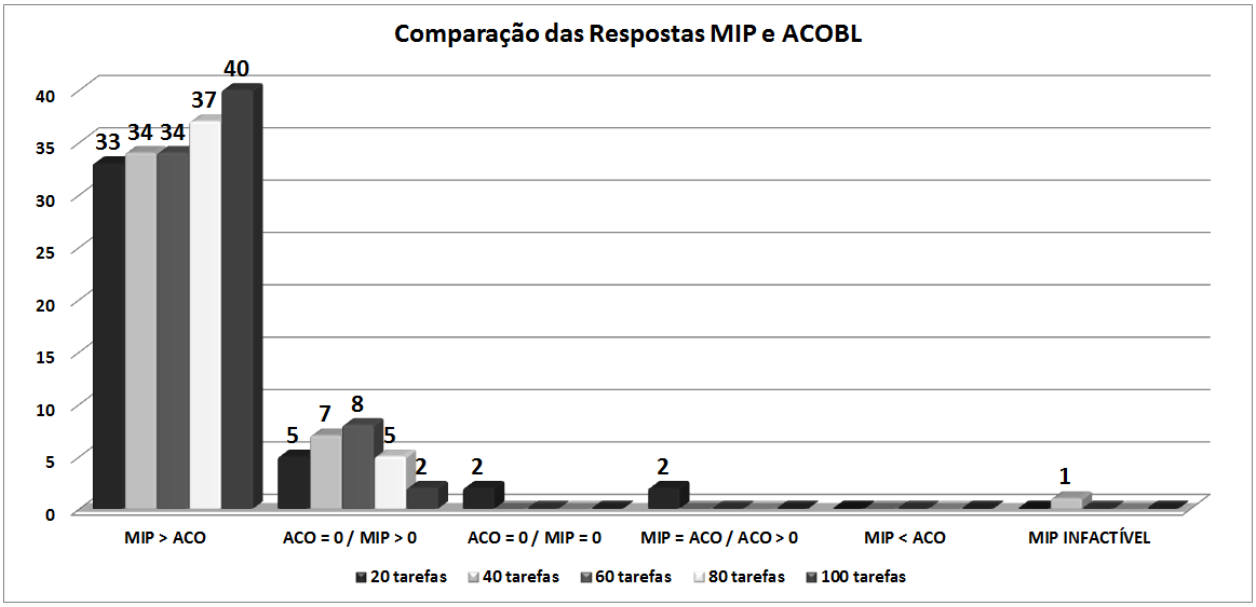


Figura 14. Comparação das Respostas MIP e ACOBL

As primeiras cinco colunas de cada um dos gráficos anteriores referem-se apenas as respostas cuja razão entre o valor obtido a partir do CPLEX e as respostas dos algoritmos implementados foi maior que um, ou seja, contêm os resultados onde as respostas do ACO e do ACOBL superaram a resposta obtida pelo CPLEX, excluindo as respostas ótimas obtidas tanto pelo ACO como pelo ACOBL. Em 91,9% dos casos de teste o ACO obteve respostas com custo de terceirização inferior ao obtivo pelo CPLEX, mas diferente de zero, e o ACOBL encontrou respostas menores que as do CPLEX em 84,8% dos problemas de teste, excluindo-se também as

respostas onde o ACOBL obteve resultado nulo para o custo de terceirização, portanto, obtendo o resultado ótimo para o problema. A Figura 15, a Figura 16, a Figura 17, a Figura 18, a Figura 19 e a Figura 20 comparam a razão entre as respostas obtidas a partir do modelo MIP e do ACO e ACOBL, respectivamente. Note que, para o ACO, em 10 casos para os problemas de 20 tarefas, seis casos para os problemas de 40 tarefas e um caso com 60 tarefas não se enquadram nesta categorização e recebem o valor zero no gráfico a seguir. E para o ACOBL em 178 problemas de teste os resultados do algoritmo implementado foi melhor que o resultado do CPLEX, mas ambos positivos, os demais resultados recevem valor zero no gráfico da Figura 18. Os dados dos 42 problemas de teste de cada tamanho foram divididos em três gráficos com 14 problemas de teste cada apenas para facilitar a visualização dos resultados.

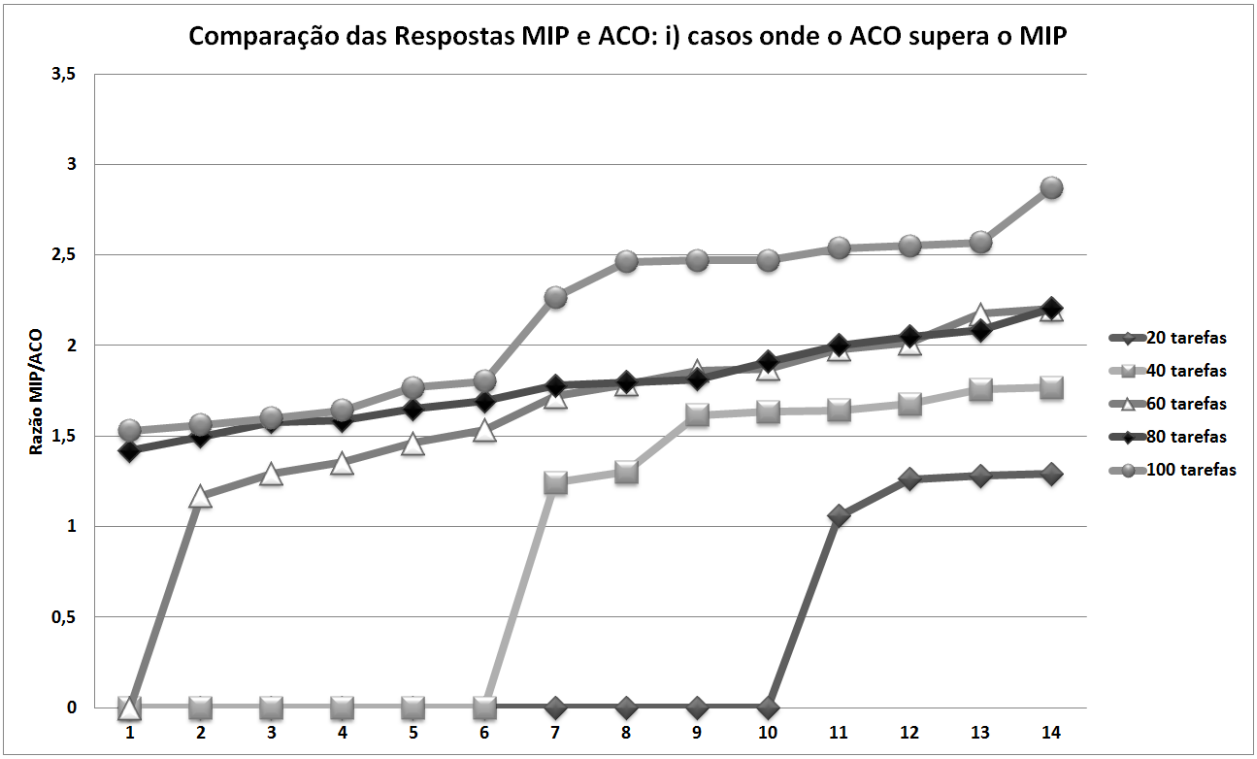


Figura 15. i) Comparação das respostas MIP e ACO

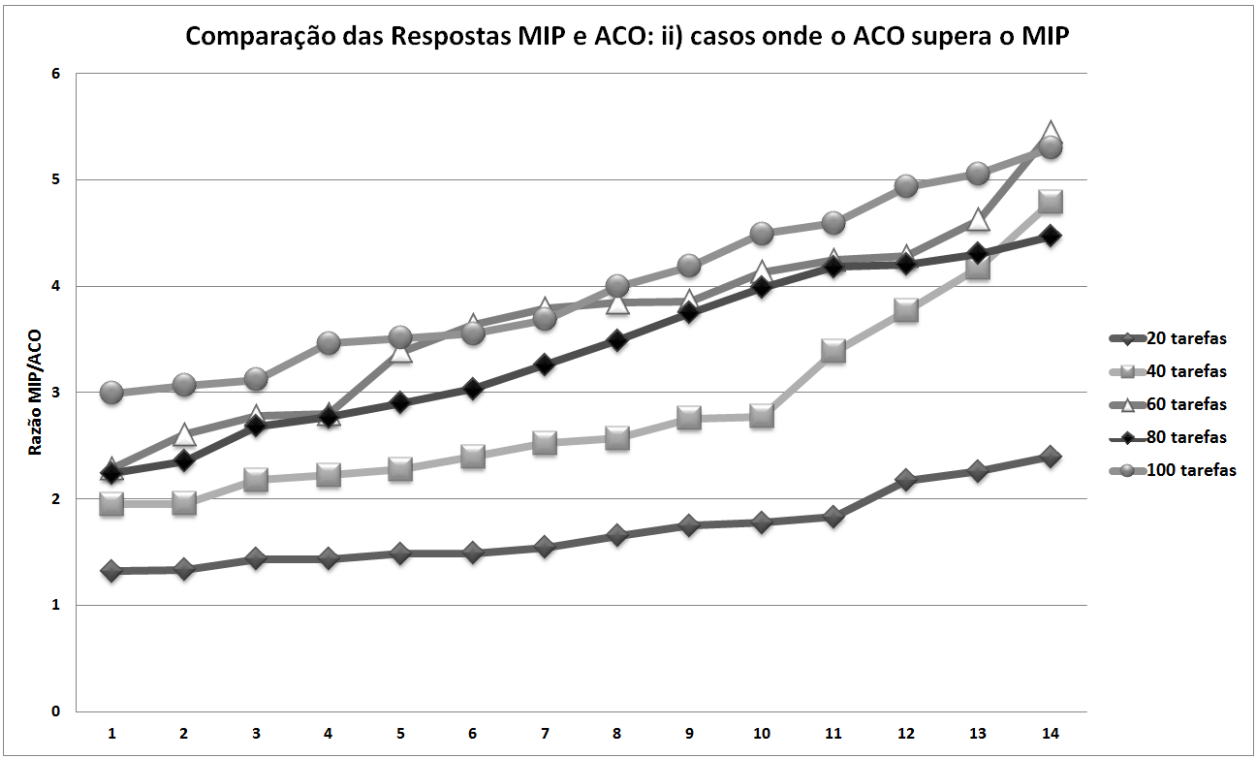


Figura 16. ii) Comparação das respostas MIP e ACO

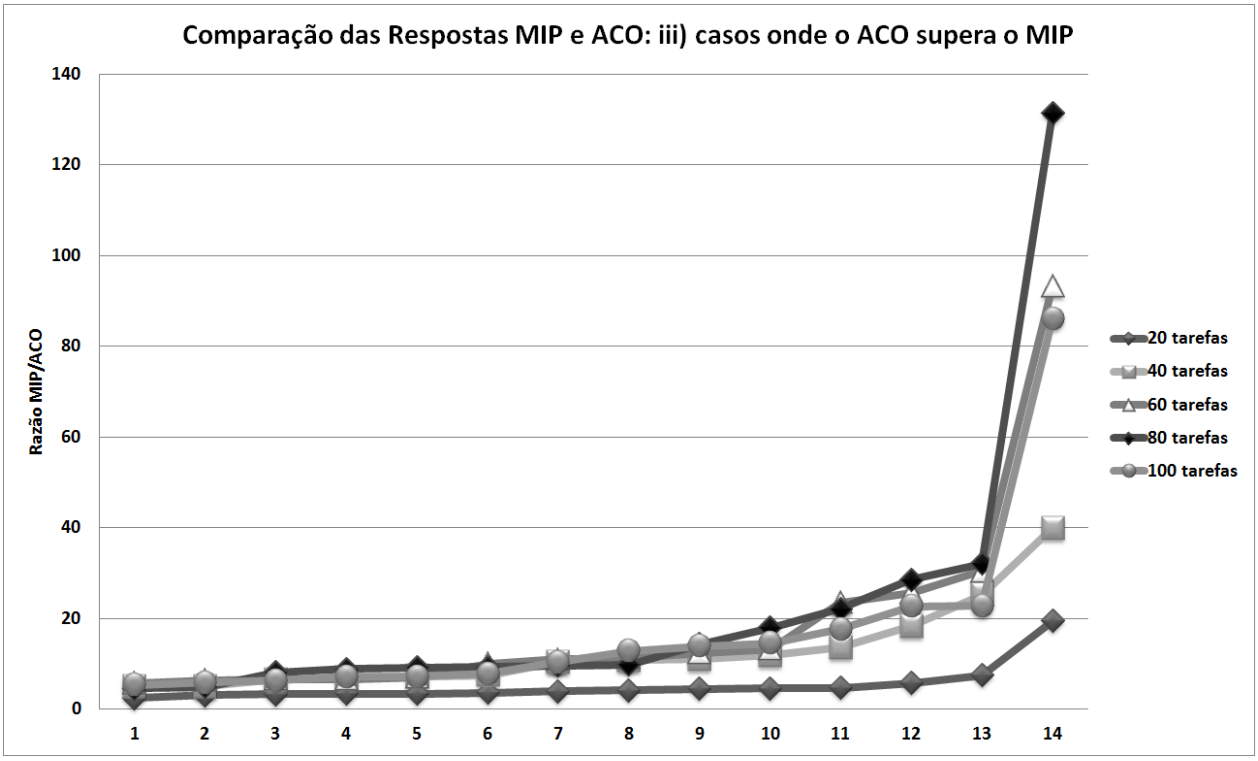


Figura 17. iii) Comparação das respostas MIP e ACO

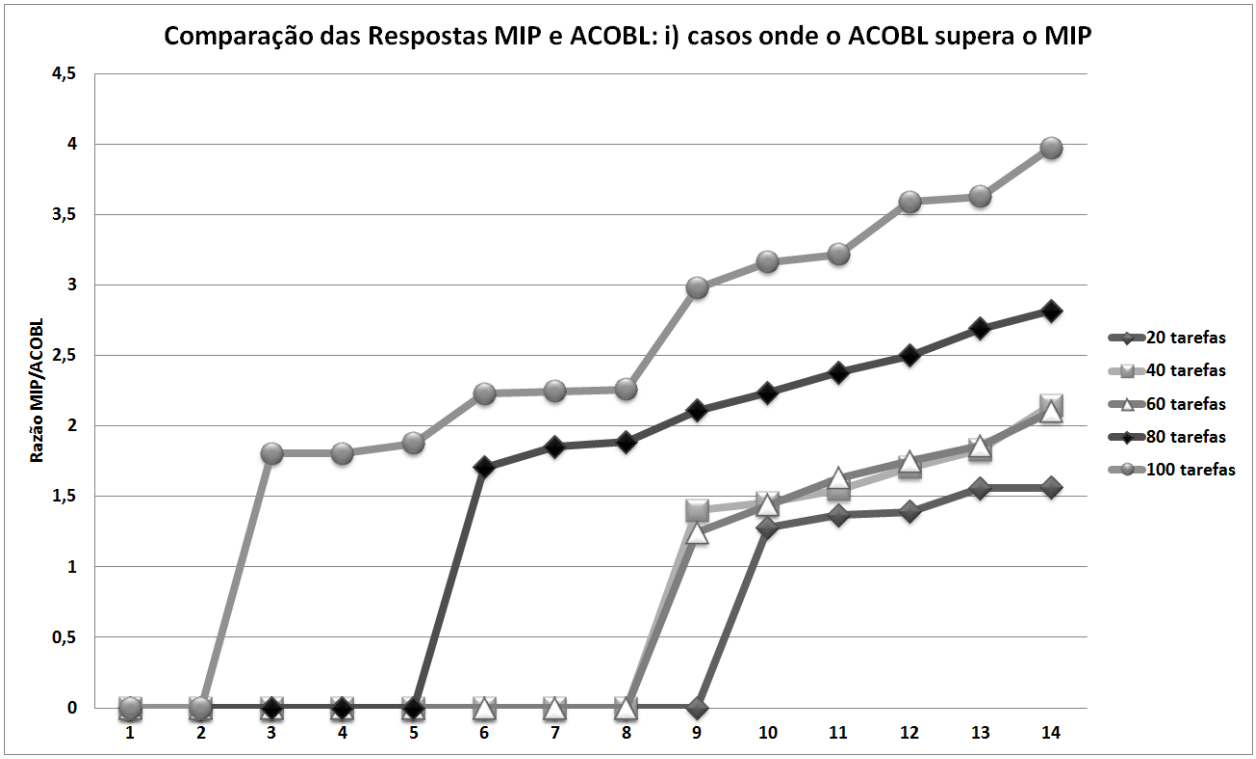


Figura 18. i) Comparação das respostas MIP e ACOBL

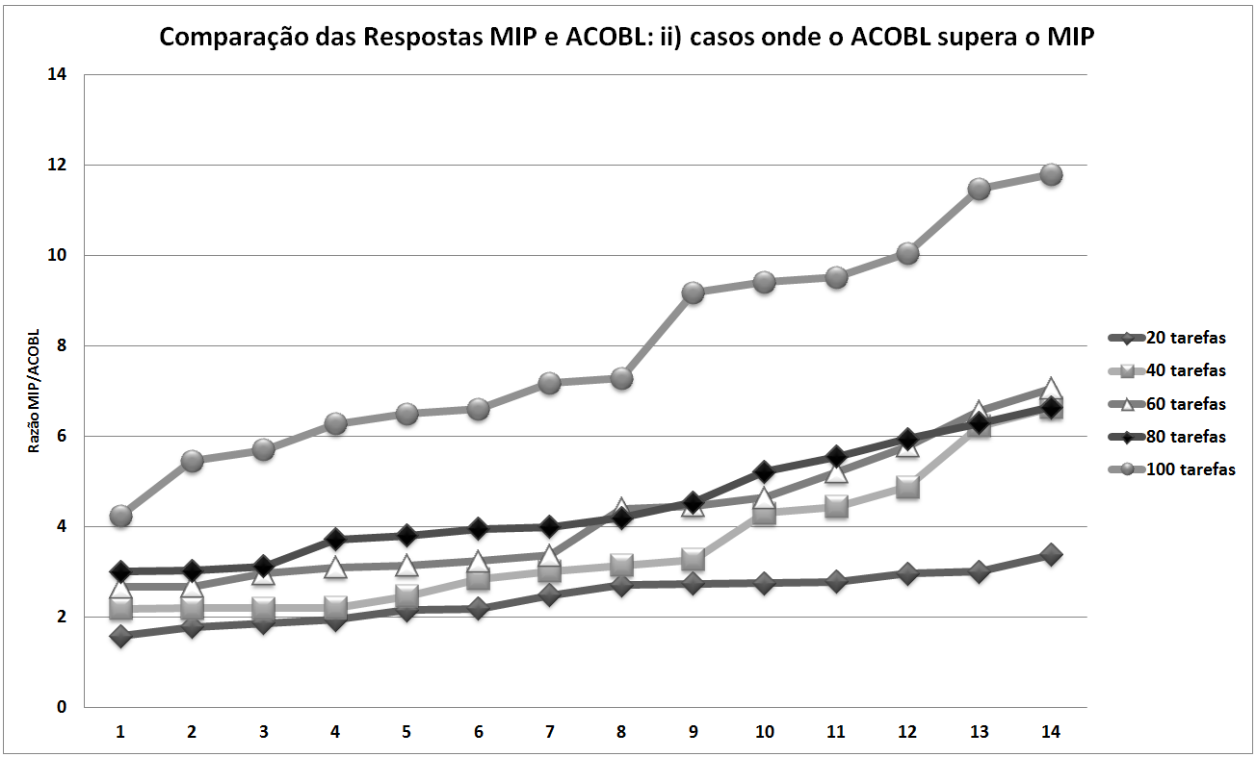


Figura 19. ii) Comparação das respostas MIP e ACOBL

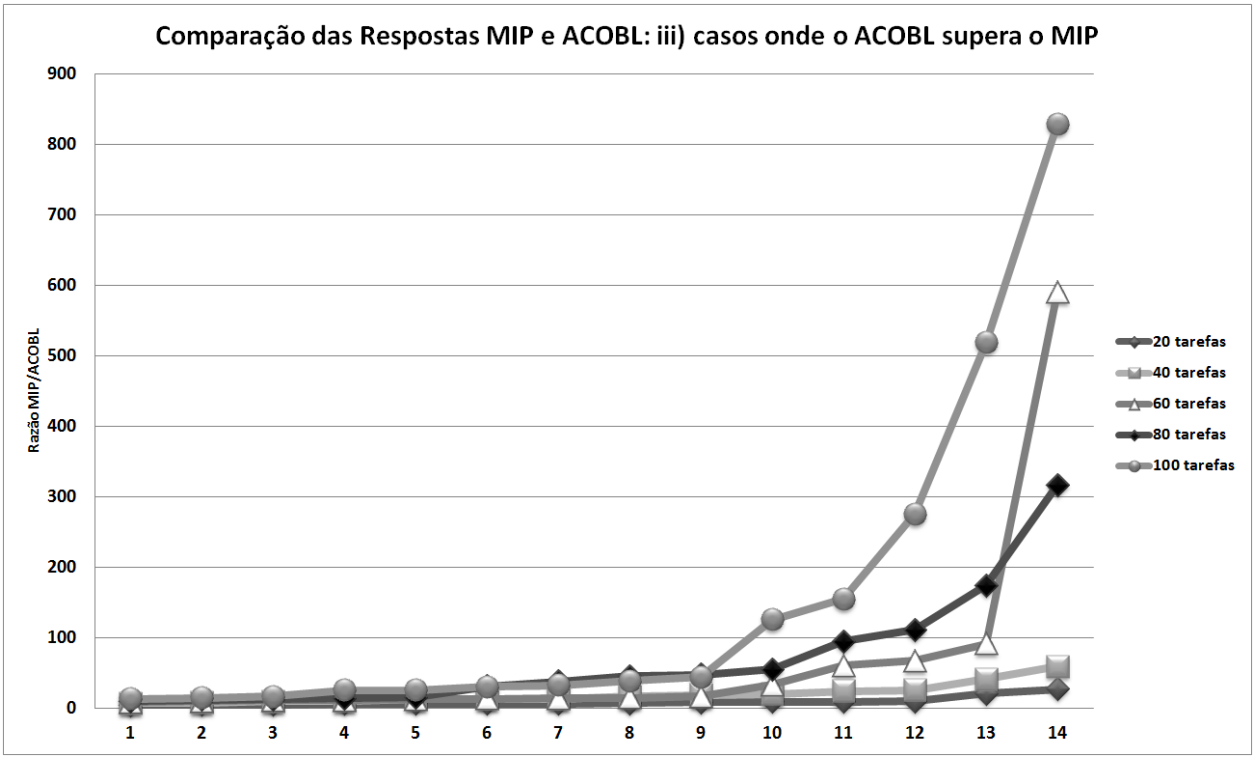


Figura 20. iii) Comparação das respostas MIP e ACOBL

Observa-se nos gráficos anteriores que o ACOBL fornece resultados razoavelmente superiores aos obtidos pelo ACO. A razão entre as respostas para o ACO é, em 35 casos de teste, maior que 10, em 27 casos, maior que cinco e em 82 dos problemas de teste o ACO obteve uma resposta menor que a metade da resposta encontrada pelo CPLEX. As respostas do ACOBL, por sua vez, resultaram em uma razão maior que 10 em 55 casos de teste, apenas maior que cinco em 36 problemas de teste e em 62 problemas o resultado da razão entre as respostas foi maior que dois. Nota-se ainda que para alguns casos a razão entre as respostas obtidas foi bastante alta, o ACOBL foi capaz de encontrar respostas mais de 100 vezes menores que o CPLEX, dado que o experimento realizado com o método exato possuiu uma restrição máxima para seu tempo de execução.

O algoritmo ACO obteve respostas inferiores às obtidas pelo CPLEX em apenas sete problemas de teste. Já o ACOBL superou os resultados do modelo de programação inteira mista em 100% dos problemas de teste fornecidos para o Experimento Preliminar. Dos 42 problemas de teste com 40 tarefas o software GAMS não foi capaz de fornecer uma solução factível em apenas um problema.

Como não foi possível obter as respostas ótimas para estes problemas foi utilizado como limitante inferior para a comparação da resposta a solução do problema simplificado $1/T_j = 0/OC$ que é facilmente solucionado a partir de um modelo de programação inteira mista, também utilizando a linguagem GAMS e o algoritmo CPLEX. A Figura 21 mostra a comparação entre as 30 execuções do algoritmo ACO, em formato *boxplot*, e também os resultados do GAMS para o problema $1/s_{ij}, T_j = 0/OC$ com seus respectivos limitantes inferiores.

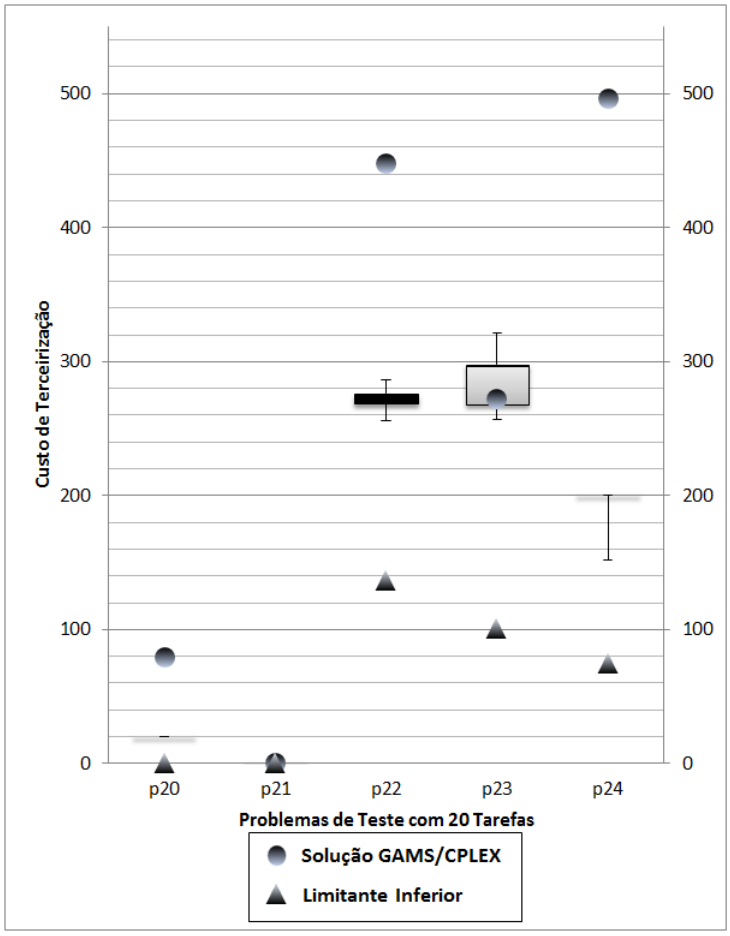


Figura 21. Simulações do ACO e resultados do GAMS para os problemas com 20 tarefas

Tabela 10. Resultados ACO/GAMS para os problemas de teste com 20 tarefas

	p20	p21	p22	p23	p24
MIN	20	0	256	257	152
Q1	20	0	269	267,75	200
MED	20	0	269	297	200
Q3	20	0	276,5	298	200
MAX	20	0	286	321	200
GAMS	79	0	448	272	496
LIMITE	0	0	137	101	75

As respostas do ACO para o problema nº 20 foram todas iguas a 20, onde o GAMS obteve solução igual a 79 e o limitante obtido, eliminando os tempos de setup do problema, foi zero. A Tabela 10 mostra com detalhes os resultados do gráfico da Figura 21. A Figura 22 e a Figura 23 apresentam alguns dos resultados para os problemas com 60 e 100 tarefas e a Tabela 11 e a Tabela 12 detalham os resultados para cada tamanho diferente do problema.

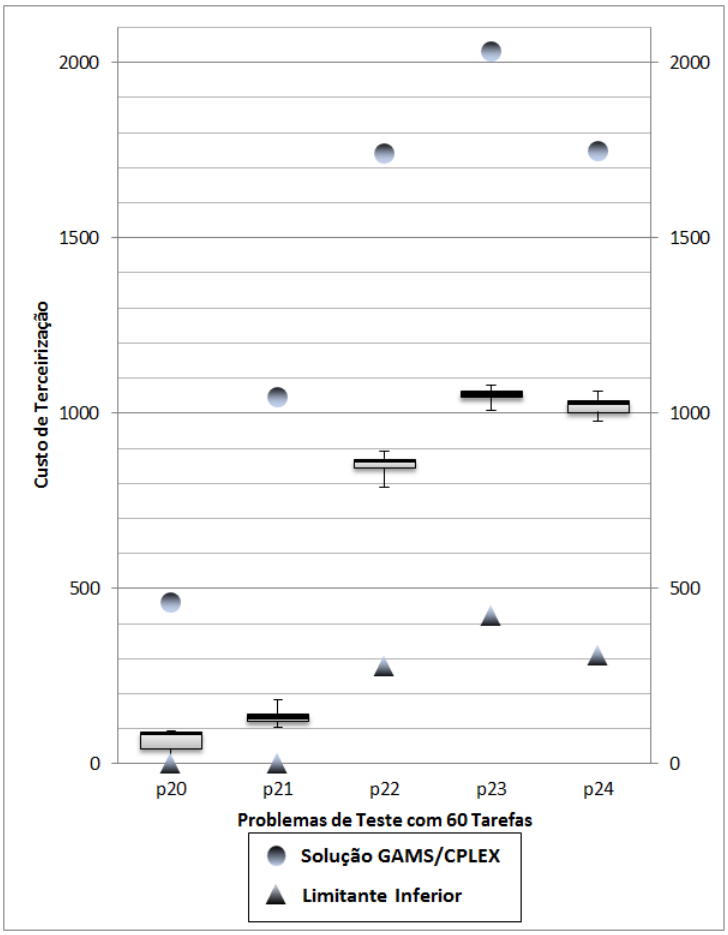


Figura 22. Simulações do ACO e resultados do GAMS para os problemas com 60 tarefas

Tabela 11. Resultados ACO/GAMS para os problemas de teste com 60 tarefas

	p20	p21	p22	p23	p24
MIN	0	104	790	1008	978
Q1	43,75	122,5	844	1046,5	1003,75
MED	84	129	861	1051,5	1026,5
Q3	92,25	143	870,5	1063,25	1037,5
MAX	93	184	892	1080	1063
GAMS	460	1046	1740	2032	1747
LIMITE	0	0	278	421	311

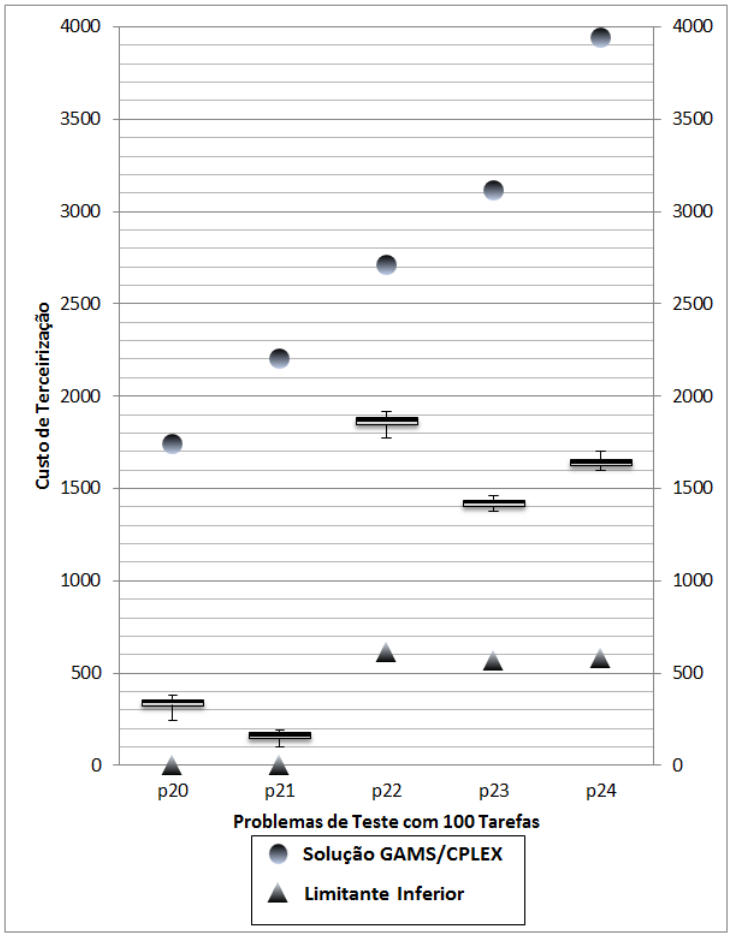


Figura 23. Simulações do ACO e resultados do GAMS para os problemas com 100 tarefas

Tabela 12. Resultados ACO/GAMS para os problemas de teste com 100 tarefas

	p20	p21	p22	p23	p24
MIN	245	97	1776	1374	1596
Q1	325,25	148	1846,25	1403,75	1625,75
MED	343,5	165	1865,5	1422	1641
Q3	358,25	183,25	1885	1438,75	1656,25
MAX	379	194	1918	1458	1703
GAMS	1740	2201	2714	3116	3942
LIMITE	0	0	613	571	582

Os mesmos problemas de teste foram solucionados utilizando a versão híbrida do ACO com um algoritmo de busca local. A Figura 24, a Figura 25 e a Figura 26 apresentam os resultados de alguns destes problemas. A Tabela 13, a Tabela 14 e a Tabela 15 detalham os dados para os problemas de teste apresentados nos gráficos das figuras anteriores.

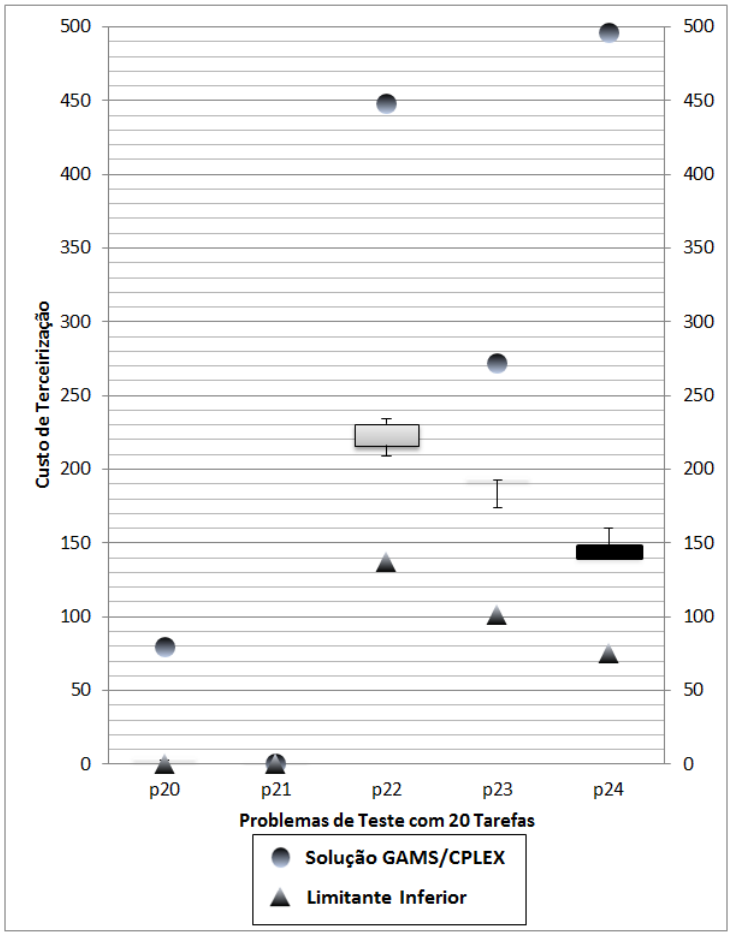


Figura 24. Simulações do ACOBL e resultados do GAMS para os problemas com 20 tarefas

Tabela 13. Resultados ACOBL/GAMS para os problemas de teste com 20 tarefas

	p20	p21	p22	p23	p24
MIN	3	0	209	174	140
Q1	3	0	216	193	140
MED	3	0	231	193	140
Q3	3	0	231	193	148
MAX	3	0	234	193	160
GAMS	79	0	448	272	496
LIMITE	0	0	137	101	75

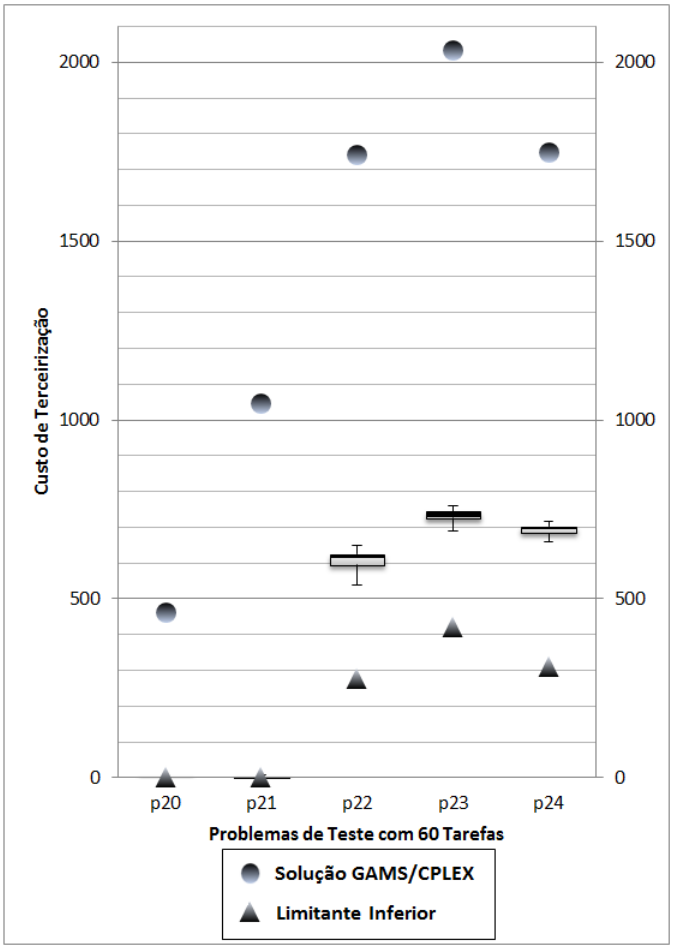


Figura 25. Simulações do ACOBL e resultados do GAMS para os problemas com 60 tarefas

Tabela 14. Resultados ACOBL/GAMS para os problemas de teste com 60 tarefas

	p20	p21	p22	p23	p24
MIN	0	0	539	689	659
Q1	0	0	595,25	725,25	684,5
MED	0	1	618,5	732	699,5
Q3	0	2,75	625,5	744	703
MAX	0	9	649	761	716
GAMS	460	1046	1740	2032	1747
LIMITE	0	0	278	421	311

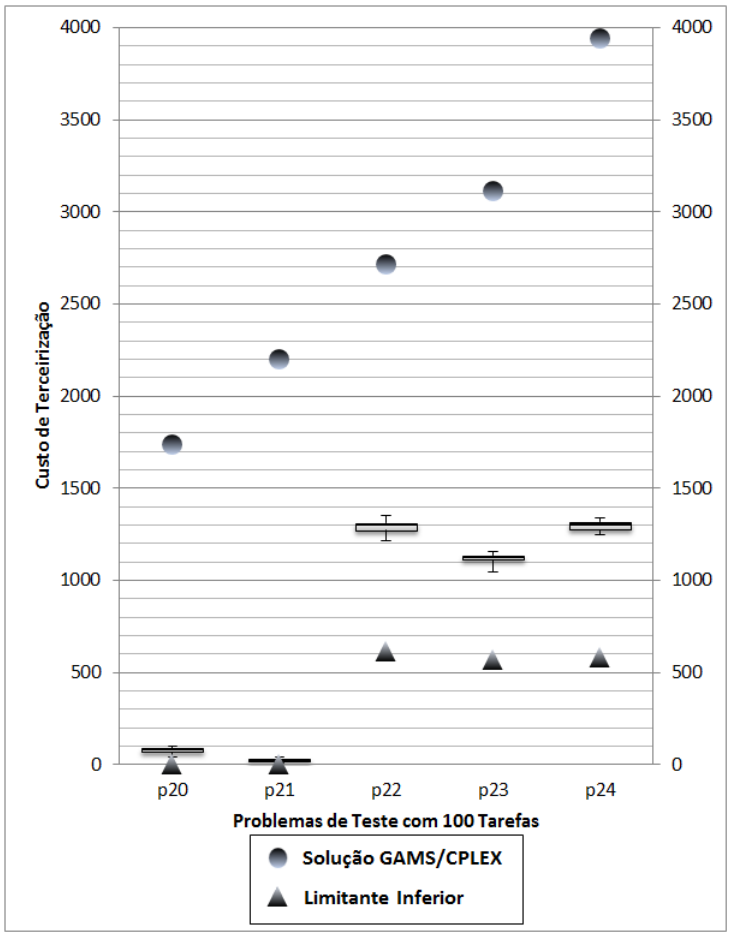


Figura 26. Simulações do ACOBL e resultados do GAMS para os problemas com 100 tarefas

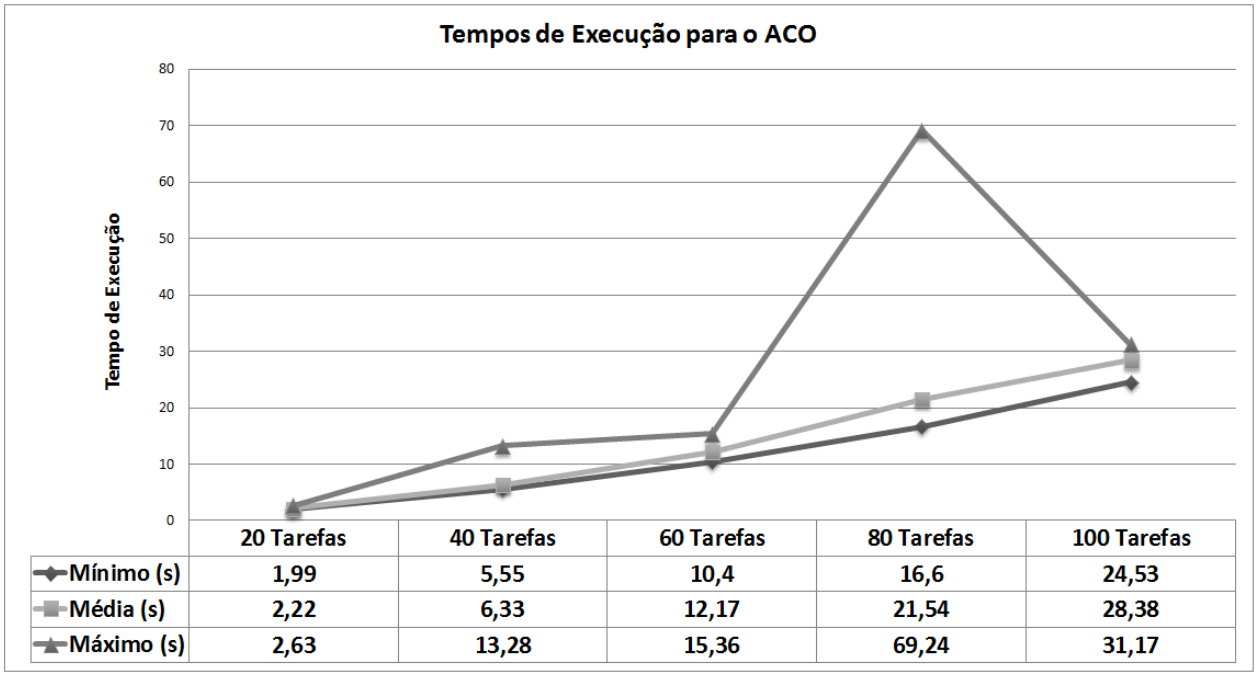
Tabela 15. Resultados ACOBL/GAMS para os problemas de teste com 100 tarefas

	p20	p21	p22	p23	p24
MIN	40	8	1218	1047	1247
Q1	69,5	19	1272,75	1113,25	1275,75
MED	81	26	1303,5	1128	1301,5
Q3	90,5	32	1311,5	1134	1316,75
MAX	103	41	1353	1156	1340
GAMS	1740	2201	2714	3116	3942
LIMITE	0	0	613	571	582

Em todos os gráficos apresentados é possível constatar que o ACO e o ACOBL se comportam da maneira prevista, ou seja, a distribuição dos dados dentre as 30 execuções de cada um dos algoritmos é pequena e similar em todos os tamanhos de problemas.

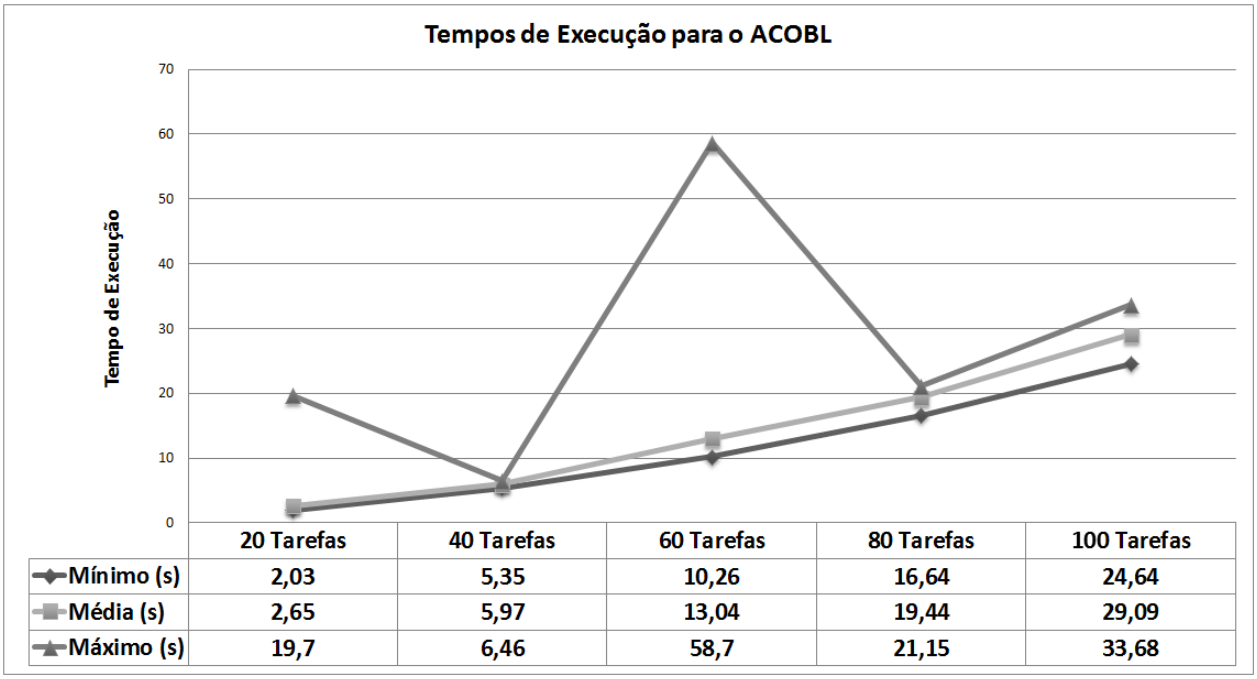
Com relação aos tempos de execução para o algoritmo CPLEX, com exceção dos problemas nº 5 e nº 21 onde as respostas ótimas foram encontradas, todos os problemas de teste atingiram o tempo limite de 9000 segundos. Para o ACO e o ACOBL foram realizadas 30

simulações para cada um dos 42 problemas de teste, a Figura 27 e a Figura 28 apresentam os valores mínimo, médio e máximo, obtidos através da análise dos tempos das 1260 simulações com cada tamanho diferente dos problemas de teste. Foram verificados apenas os tempos das melhores respostas de *tuning* obtidas.



	20 Tarefas	40 Tarefas	60 Tarefas	80 Tarefas	100 Tarefas
Mínimo (s)	1,99	5,55	10,4	16,6	24,53
Média (s)	2,22	6,33	12,17	21,54	28,38
Máximo (s)	2,63	13,28	15,36	69,24	31,17

Figura 27. Tempos de execução do ACO para o experimento preliminar



	20 Tarefas	40 Tarefas	60 Tarefas	80 Tarefas	100 Tarefas
Mínimo (s)	2,03	5,35	10,26	16,64	24,64
Média (s)	2,65	5,97	13,04	19,44	29,09
Máximo (s)	19,7	6,46	58,7	21,15	33,68

Figura 28. Tempos de execução do ACOBL para o experimento preliminar

Dentre os tempos de execução para os algoritmos ACO e ACOBL o valor máximo foi observado em um dos problemas de 60 tarefas para o ACO, sendo este tempo de 69,24 segundos. As médias dos tempos de execução crescem linearmente em relação ao tamanho dos problemas de teste e não excedem os 30 segundos mesmo para os problemas com 100 tarefas. A seguir são apresentados os resultados do experimento final.

6.2. Análise dos Resultados do Experimento Final

O experimento final foi composto de passos análogos ao experimento preliminar, modificando-se apenas a composição dos dados dos problemas de teste e os critérios para a execução do algoritmo CPLEX. Neste experimento os problemas de teste foram limitados ao tamanho de 10 tarefas e o CPLEX não recebeu nenhuma restrição de tempo, podendo sempre atingir a resposta ótima para os problemas.

Tanto para o ACO como para o ACOBL foram realizados os experimentos de *tuning* descritos na seção 5.3. Os resultados obtidos para o ACO estão descritos na Tabela 16 e na Tabela 17. Os resultados para a afinação do ACOBL são apresentados na Tabela 18 e na Tabela 19.

Tabela 16. i) Resultados dos experimentos de *tuning* do ACO para o experimento final

Tuning	MAXF	MINF	TB	TE	β_1	β_2
p1|m1	5	30	2,9	0,89	0,86	0,89
p1|m2	5	30	2,3	0,95	0,97	0,9
p40|m1	15	30	2,1	0,94	0,86	0,85
p40|m2	15	20	4	0,99	0,89	0,86
p1-52|m1	15	30	2,8	0,94	0,97	0,97
p1-52|m2	15	40	2,8	0,94	0,88	0,96
p53-104|m1	10	20	1,7	0,9	0,93	0,93
p53-104|m2	10	30	2,5	0,89	1	0,87
p105-156|m1	15	20	2,3	0,86	0,87	0,97
p105-156|m2	5	30	3,8	0,9	0,97	0,96
p157-208|m1	5	40	1,5	0,98	0,87	0,95
p157-208|m2	15	20	2,4	0,9	0,88	1
Sem Tuning	10	20	2,5	0,9	1	1

Tabela 17. ii) Resultados dos experimentos de *tuning* do ACO para o experimento final

Tuning	α_1	α_2	α_3	α_4	μ_1	μ_2	μ_3	μ_4
p1|m1	4,61	1,11	0,2	0,14	4,53	1,88	1,18	4,59
p1|m2	2,36	3,09	4,18	1,57	1,38	2,56	3,35	4,16
p40|m1	3,06	2,12	4,23	1,59	0,57	4,55	1,85	3,81
p40|m2	1,25	2,64	2,09	2,58	3,02	4,06	2,94	0,18
p1-52|m1	3,78	2,4	1,06	4,07	1,32	4,58	2,29	3,64
p1-52|m2	4,82	4,87	1,55	4,01	0,56	0,89	3,24	4,56
p53-104|m1	3,58	1,52	2,95	2,35	1,3	1,5	2,89	1,36
p53-104|m2	3,92	1,4	2,62	1,27	4,68	3,71	1,44	0,35
p105-156|m1	1,83	2,44	2,61	2,34	4,58	2,18	0,64	2,84
p105-156|m2	3,31	3,42	1,88	1,99	3,02	0,82	4,86	1,66
p157-208|m1	0,88	3,7	4,02	1,8	1,84	4,48	3,4	3,24
p157-208|m2	1,2	4,54	2,86	2,22	0,82	1,11	1,99	1,46
Sem Tuning	1	1	1	1	1	1	1	1

Tabela 18. i) Resultados dos experimentos de *tuning* do ACOBL para o experimento final

Tuning	MAXF	MINF	TB	TE	β_1	β_2
p1|m1	5	30	3,1	0,91	0,98	0,87
p1|m2	5	30	2	0,97	0,87	0,92
p40|m1	10	30	3,8	0,93	0,87	0,94
p40|m2	5	20	2,7	0,94	0,92	0,85
p1-52|m1	10	30	1,5	0,97	1	0,98
p1-52|m2	15	40	2,9	0,91	0,92	0,9
p53-104|m1	10	30	3,1	0,9	0,88	0,98
p53-104|m2	15	20	2,2	0,94	0,96	0,88
p105-156|m1	10	20	1,6	0,95	0,94	0,89
p105-156|m2	15	20	2	0,94	0,93	0,85
p157-208|m1	15	30	2,3	0,87	0,92	0,91
p157-208|m2	15	30	2,9	0,89	0,98	0,98
Sem Tuning	10	20	2,5	0,9	1	1

A partir destes resultados cada problema de teste foi solucionado 30 vezes com cada configuração diferente do algoritmo, sendo coletada a soma dos custos de terceirização entre os 208 problemas de teste que compuseram este experimento. A Figura 29 e a Figura 30 apresentam as resultados dos experimentos de *tuning* para o ACO e o ACOBL respectivamente.

Tabela 19. ii) Resultados dos experimentos de *tuning* do ACOBL para o experimento final

Tuning	α_1	α_2	α_3	α_4	μ_1	μ_2	μ_3	μ_4
p1|m1	0,91	3,1	4,19	0,31	1,91	4,21	0,87	3,95
p1|m2	4,71	1,91	2,8	3,12	2,16	0,97	2,61	0,97
p40|m1	3,48	3,05	4,76	0,4	0,43	2,62	3,13	0,09
p40|m2	4,16	1,75	4,73	1,99	1,16	4,19	4,2	1,46
p1-52|m1	3,39	3,32	4,78	1,18	3,5	2,92	3,2	4,49
p1-52|m2	4,32	3,94	4,67	1,8	3,44	2,59	4,24	2,23
p53-104|m1	2,19	3,16	1,06	0,58	3,43	0,49	2,19	0,84
p53-104|m2	1,24	2,53	2,25	3,98	1,08	4,5	1,38	2,55
p105-156|m1	4,07	2,75	2,42	2,16	0,89	0,92	4,47	4,75
p105-156|m2	0,1	3,41	2,81	1,84	3,68	4,87	1,11	3,95
p157-208|m1	2,28	2,79	0,81	3,67	0,04	2,03	0,38	0,32
p157-208|m2	0,72	2,48	3,14	0,01	4,9	2,25	3,34	1,58
Sem Tuning	1	1	1	1	1	1	1	1

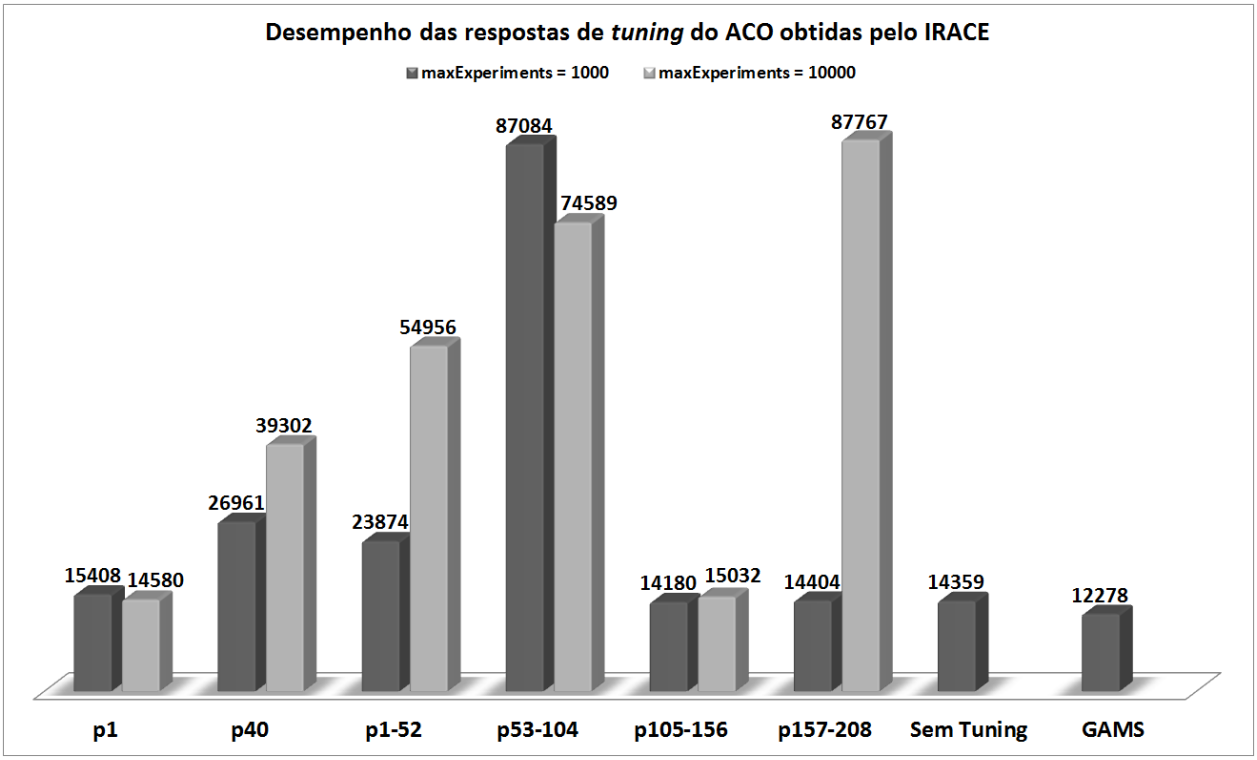


Figura 29. Desempenho das respostas de *tuning* do ACO para o experimento final

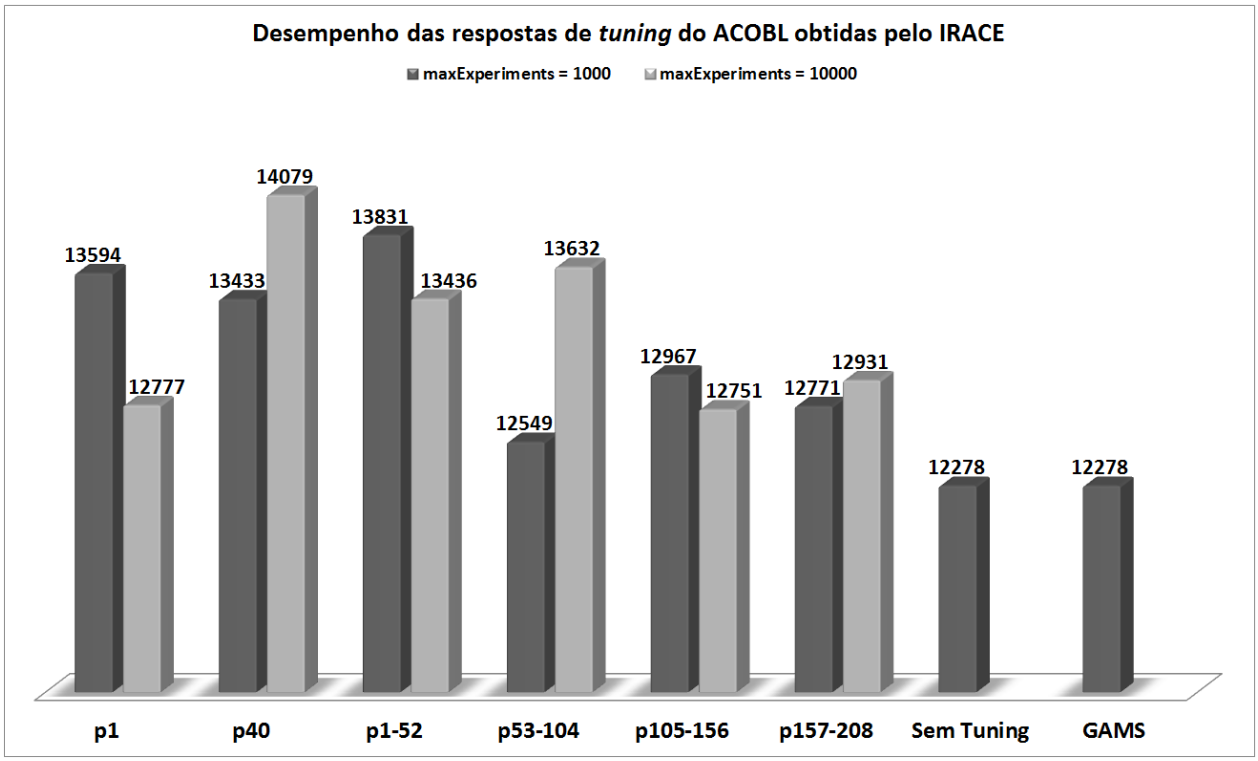


Figura 30. Desempenho das respostas de *tuning* do ACOBL para o experimento final

Destaca-se que o ACOBL foi capaz de encontrar as respostas ótimas para todos os problemas utilizando uma configuração que não pondera os dados da regra de transição e da função de probabilidade para a escolha das tarefas. Os problemas testados são compostos de poucas tarefas e dessa forma a população de agentes computacionais, criada de maneira uniforme, é capaz de amostrar um grande número de soluções alcançando as respostas ótimas em todos os casos, quando são realizadas 30 execuções para cada problema de teste.

Utilizando um número máximo de experimentos maior (configurado no IRACE) apesar de ampliar a exploração das soluções para o problema combinatório da afinação da meta-heurística não foi capaz de gerar respostas melhorer em todos os experimentos. No caso do ACO, a pior resposta para a afinação dos parâmetros foi obtida para o último quadrante de problemas (do problema nº 157 ao problema nº 208) no teste com número máximo de experimentos igual a 10000.

Com objetivo de analisar estatisticamente o desempenho dos algoritmos sobre um grande número de problemas de teste foi realizado um último experimento. Segundo Kennedy e Eberhart (2001, p. 434) pequenas amostras irão variar em torno da média da população observada, no entanto, para amostras que contêm 30 ou mais observações, a amostragem será distribuída de

maneira normal. Desta maneira para a realização do último experimento desta pesquisa foram gerados 30 diferentes problemas de teste para os 208 níveis de problemas existentes no experimento final. A Figura 31 e a Figura 32 mostram a comparação entre as respostas do ACO e do ACOBL, respectivamente, para o experimento final, onde a execução do CPLEX não foi sujeita a restrições e, portanto, foi possível avaliar a capacidade dos algoritmos em encontrar as respostas ótimas para os problemas de teste definidos neste experimento.

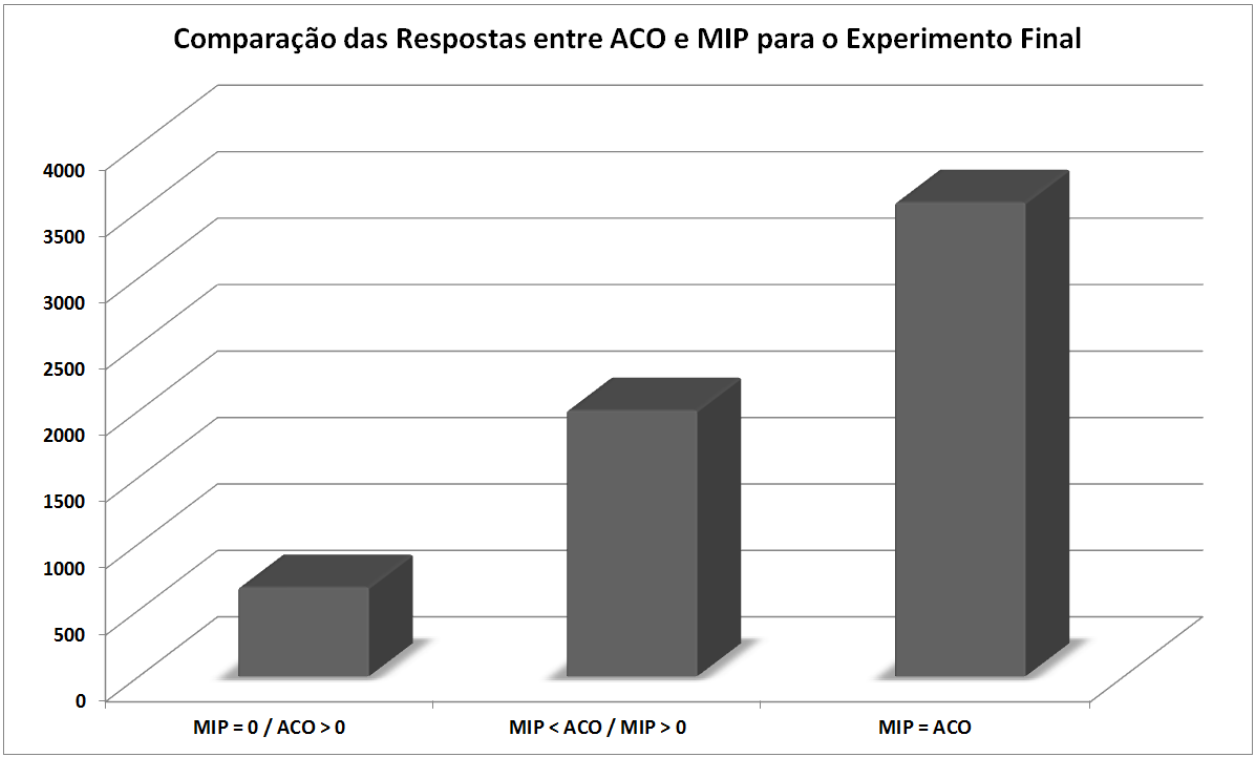


Figura 31. Comparação das respostas entre ACO e MIP para o experimento final

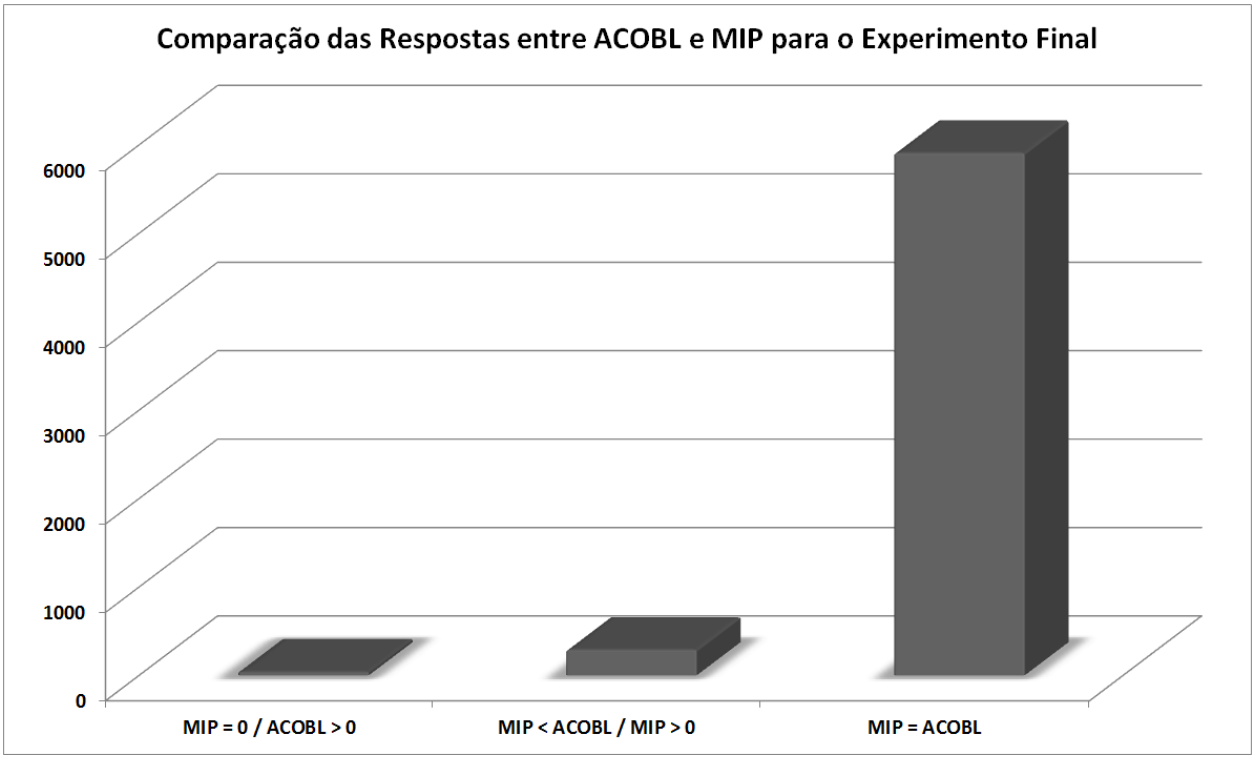


Figura 32. Comparação das respostas entre ACOBL e MIP para o experimento final

Em apenas 40 problemas de teste, dentre os 6240 problemas que compuseram este experimento, o CPLEX obteve resultado nulo para o custo de terceirização e o ACOBL não foi capaz de atingir o mesmo, para o ACO, no entanto, este resultado ocorreu em 671 casos de teste. O ACO não foi capaz de atingir respostas ótimas em outros 2002 problemas de teste, enquanto que para o ACOBL este resultado acontece em apenas 291 problemas. Os gráficos da Figura 33 e a Figura 34 mostram a comparação em a razão das respostas dos algoritmos implementados, apenas para os casos onde, ACO e ACOBL, não atingiram as respostas ótimas para os problemas, e as respostas obtidas através do método exato, que retornou as respostas ótimas para todos os problemas de teste, mas a um custo computacional muitas vezes maior que o necessário para a execução dos algoritmos implementados.

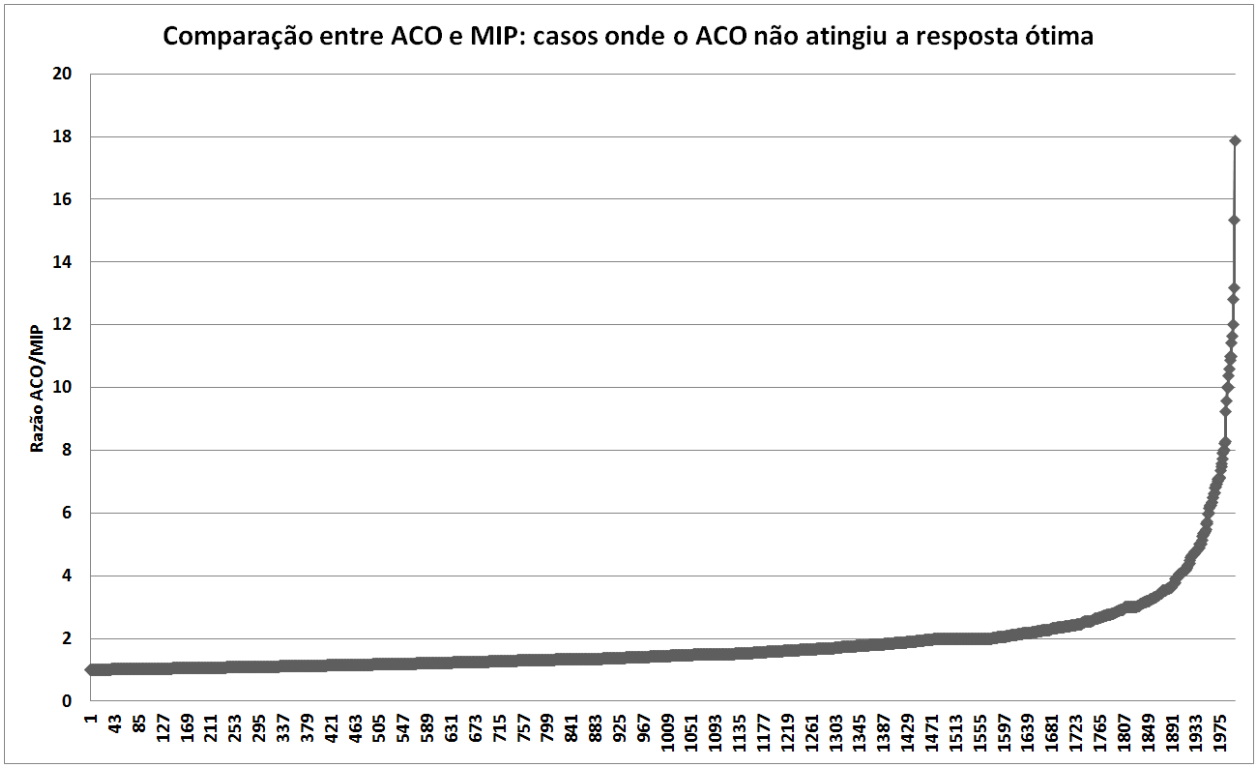


Figura 33. Potencial do ACO para obtenção de soluções ótimas

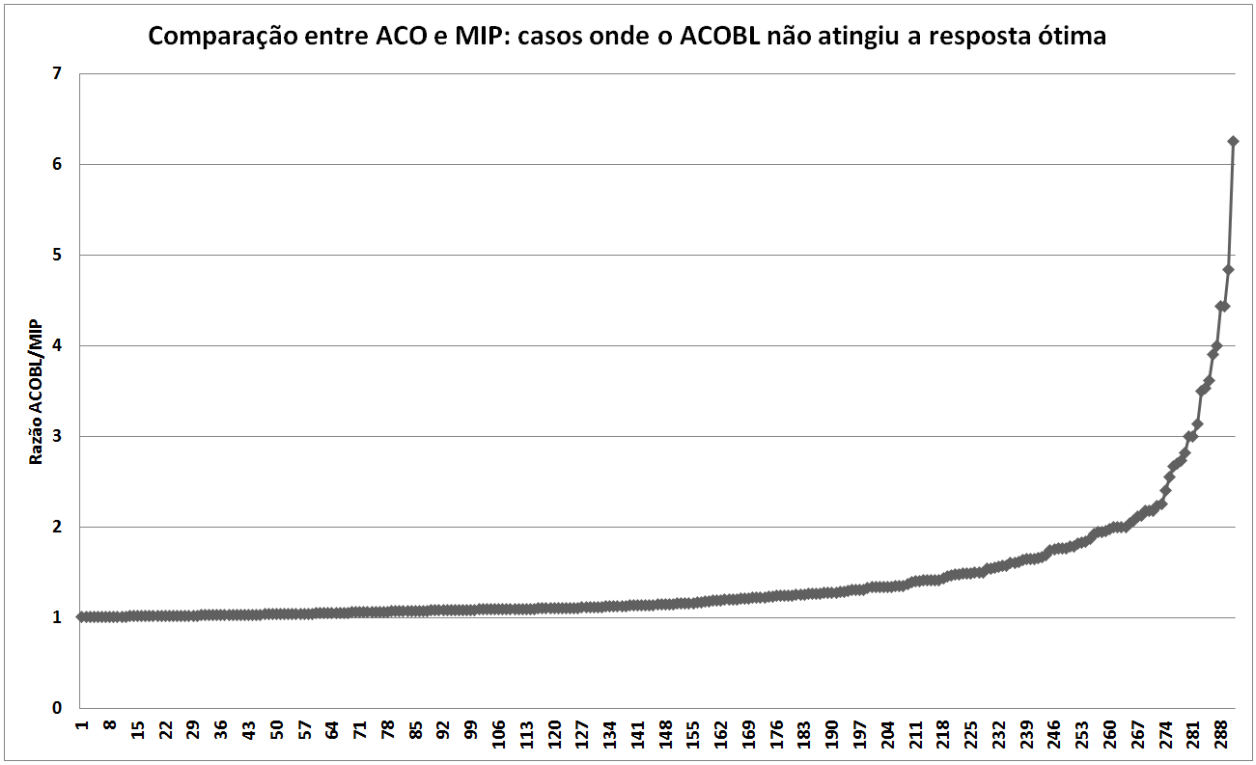


Figura 34. Potencial do ACOBL para obtenção de soluções ótimas

Além de 10,75% dos problemas de teste onde o ACO obteve um custo de terceirização e o CPLEX atingiu resultado nulo, o ACO não alcançou respostas ótimas em outros 32,08% dos

casos, dentre estes, mas em apenas 4,05% destes casos de teste a razão entre a resposta do método exato e do ACO foi maior que dois, mas no máximo igual a três, em 1,81% dos casos este resultado foi maior que três, mas inferior a cinco, em 0,75% dos casos a razão das respostas esteve entre cinco e 10, e para 0,19% dos problemas de teste a razão foi maior que 10.

O ACOBL em apenas 0,64% dos problemas não foi capaz de encontrar o resultado nulo para o custo de terceirização onde o método exato encontrou valor zero para objetivo do problema proposto. Para 1,92% dos problemas o ACOBL obteve respostas no máximo 10% maiores que o CPLEX, em 0,71% dos casos as respostas foram no máximo 20% maiores, em 0,5% dos problemas as respostas foram no máximo 20% maiores que as obtidas pelo CPLEX, em outros 0,51% dos casos o ACOBL obteve respostas no máximo 50% maiores, em 0,58% as respostas foram no máximo o dobro das encontradas pelo CPLEX e em 0,43% (27 casos) dos problemas de teste a razão entre os resultados foi maior que dois. Dentre todos os 6240 problemas de teste o ACOBL não alcançou a resposta ótima em apenas 331 (5,3%) casos.

A Figura 35 mostra como o algoritmo CPLEX se comportou em relação ao tempo de computação necessário para a obtenção das respostas ótimas para estes problemas de teste.

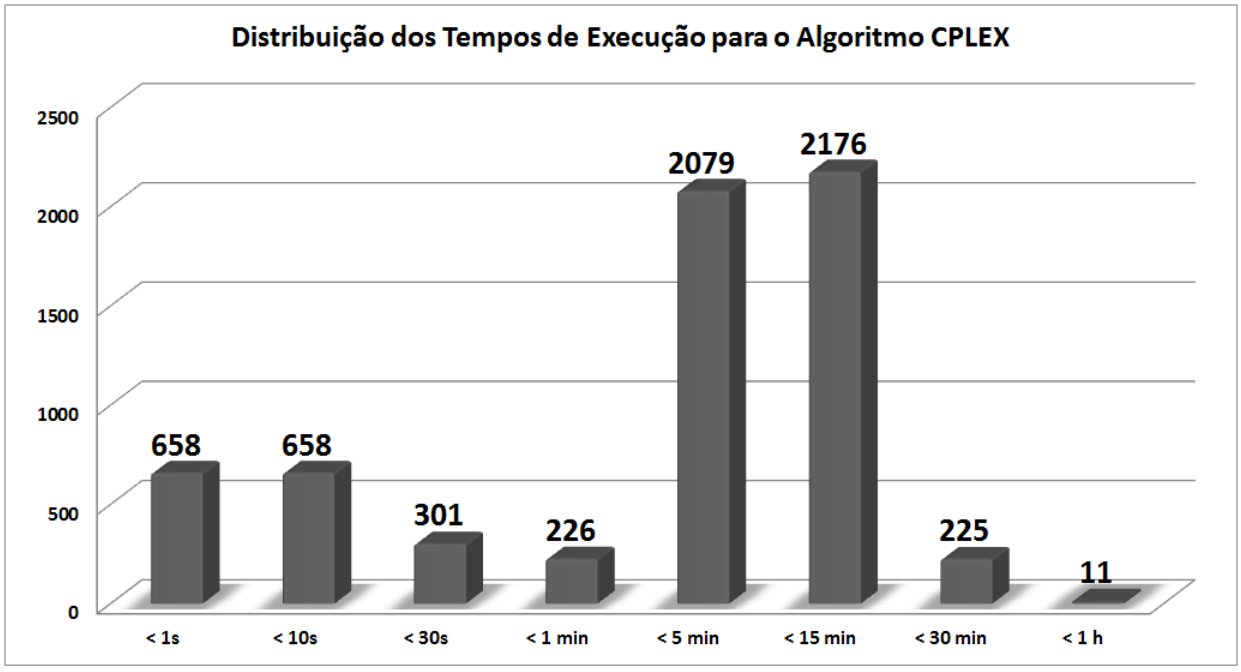


Figura 35. Tempos de execução do GAMS para o experimento final

Observa-se que grande parte dos problemas foi solucionada entre um e 15 minutos, no entanto, mesmo para problemas com apenas 10 tarefas, houve 11 problemas que necessitaram de

mais de 30 minutos para que o CPLEX alcança-se a resposta ótima. Já o ACO resultou em uma média de tempo de execução de 0,88 segundos entre os 6240 problemas, e o ACOBL apresentou média de 0,66 segundos, enquanto o CPLEX atingiu o tempo máximo de processamento de 3580 segundos e médio de 284 segundos.

N última etapa do experimento, dentre os 6240 problemas de teste disponíveis o ACO encontrou as respostas ótimas em 57,2% dos problemas e o ACOBL em 94,7% dos problemas. A seguir são feitas as conclusões desta pesquisa.

7. Conclusão

O problema do sequenciamento de operações em situações produtivas que permitam terceirização de ordens de serviço é algo pouco explorado na literatura científica. Assim, o presente trabalho teve como objetivo o desenvolvimento de bases teóricas para futuras aplicações técnicas em sistemas industriais reais. Para tal, desenvolveu-se estratégias em duas frentes: na primeira frente, foi elaborado e implementado um modelo de programação inteira mista; na segunda, criou-se um algoritmo baseado na meta-heuríticas ACO e uma rotina computacional de busca local para o refinamento das respostas. Uma vez tendo as duas frentes concluídas, buscou-se realizar testes para determinar as características e limitações do uso de cada uma das técnicas.

A implementação do algoritmo ACO apresentado nesse trabalho foi baseada na literatura que trata da meta-heurística *Ant Colony Optimization*, principalmente quanto às suas duas estruturas básicas: a regra de transição dos agentes e as regras atualização/evaporação da matriz de feromônios. Assim, os agentes são capazes de realizar movimentos básicos sobre o grafo que contém os dados disponíveis para o problema e acumulam as informações coletadas a respeito das soluções factíveis em uma matriz de feromônios artificiais.

Estas duas estruturas foram capazes de encontrar resultados superiores aos do modelo de programação inteira mista, em 96,6% das instâncias em um teste preliminar. A construção de uma estrutura adicional de busca local aumentou este percentual, de superioridade do algoritmo em relação aos dados obtidos da aplicação do MIP, para 100% nos problemas de teste definidos no experimento preliminar. A condução de um segundo experimento revelou que os algoritmos implementados têm enorme potencial para encontrar as repostas ótimas em problemas com pequeno número de tarefas. Com uma única tentativa, o ACO foi capaz de encontrar as respostas ótimas em 57,2% dos 6240 problemas de teste disponíveis e o ACOBL em 94,7% dos problemas.

Em ambos os casos percebeu-se que, os recursos computacionais necessários para a execução dos algoritmos, ACO e ACOBL, são muito pequenos quando comparados com os recursos necessários para a execução dos modelos matemáticos propostos, sendo que as duas estratégias sempre conseguiram obter soluções viáveis para os problemas de teste disponíveis.

Os experimentos de *tuning*, que utilizaram o IRACE, revelaram-se essenciais para atingir o melhor desempenho do algoritmo. Estes experimentos mostraram que diferentes conjuntos de

parâmetros ocasionarão comportamentos distintos para os algoritmos e, portanto, a afinação da meta-heurística representa um ponto crucial para a aplicação deste tipo de algoritmo em sistemas produtivos reais. Entendeu-se que a questão do *tuning* da meta-heurística representa um problema complexo e espera-se que pesquisas futuras busquem relacionar o comportamento do algoritmo com parâmetros ótimos utilizados para a configuração de meta-heurísticas.

Os resultados apresentados são encorajadores e mostram que a hibridização da meta-heurística de colônia de formigas com procedimentos de busca local fornece respostas com alta qualidade e com baixíssimos tempos de execução, pesquenos o suficiente para proporcionar a aplicação de uma ferramenta de suporte à decisão como esta mesmo em cenário onde existem apenas alguns minutos para a tomada de decisão sobre a programação das próximas ordens de produção.

Percebe-se que o presente trabalho permite sua continuidade através de quatro principais frentes de trabalho: a primeira, diz respeito à realização de estudos de campo para a aplicação das técnicas de otimização desenvolvidas nesta pesquisa em problemas de *scheduling* em sistemas industriais reais; a segunda relaciona-se com estudo e desenvolvimento de técnicas alternativas para a afinação dos parâmetros da meta-heurísticas; a terceira, quanto ao estudo e desenvolvimento de estratégias para o aprimoramento dos algoritmos propostos, como estratégias de busca local, e a quarta diz respeito à ampliação do escopo do problema estudado, integrando-se as decisões dos problemas de transporte (*Vehicle Routing Problem* – VRP) ao problema da programação das operações aqui estudado.

8. Referências Bibliográficas

ALLAHVERDI, A.; GUPTA, J.N.D., ALDOWAISAN, T. *A review of scheduling research involving setup considerations*, *International Journal of Management Science*, v. 27, p. 219-239, 1999.

ALVAREZ, L.H.R.; STENBACKA, R. *Partial outsourcing: A real options perspective.International Journal of Industrial Organization*, v. 25, p. 91-102, 2007.

ANGHINOLFI, D., BOCCALATTE, A., PAOLUCCI, M., VECCHIOLA, C. *Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In: Simulated Evolution and Learning. Springer, Berlin/Heidelberg*, pp. 411–420, 2008.

ANTELO, M.; BRU, L. *Outsoursing or restructuring: The dynamic choice.International Journal of Production Economics*, v. 123, p. 1-7, 2010.

ARENALES, M., ARMENTANO, V., MORABITO, R., YANASSE, H. Pesquisa Operacional, Elselvier, p. 163-289, 2007.

BAKER, K. R. *Introdution to Sequencing and Scheduling*. Wiley & Sons, 1974.

BAUER, A; BULLNHEIMER, B; HARTL, R. F.; STRAUSS, C. *Applying Ant Colony Optimization to solve the Single Machine Total Tardiness Problem. Technical Report. Vienna University of Economics and Business Administration*, 1999.

BEASLEY, J. E. *Or-Library: Weighted tardiness*. Disponível em <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html>, acessado em Julho de 2013.

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

BIRATTARI, M. *Tuning Metaheuristics: A Machine Learning Perspective, Studies in Computational Intelligence. Springer-Verlag, Berlin / Heidelberg*, v 197, 2009.

BUFFA, E. S.; SARIN, R. K. *Modern production/operations management*. Wiley, 1987.

CHAN, F. T. S.; KUMAR, V.; TIWARI, M. K. *The relevance of outsourcing and leagile strategies in performance optimization of an integrated process planning and scheduling model. International Journal of Production Research*, V. 47, N. 1, p. 119–142, 2009.

CHASE, R. B.; JACOBS, F. R.; AQUILANO, N. J. *Operations Management for Competitive Advantage*, ed. 11, McGraw-Hill Irwin, 2006.

CHENG, B.-Y., CHEN, H.-P., SHAO, H., XU, R., HUANG, G.Q. *A chaotic ant colony optimization method for scheduling a single batch-processing machine with non-identical job sizes. In: IEEE Congress on Evolutionary Computation*, p. 40–43, 2008.

CHENG, T.C.E., LAZAREV, A.A., GAFAROV, E.R. *A hybrid algorithm for the single-machine total tardiness problem. Comput. Oper.*, Res. 36 (2), p. 308–315, 2009.

CHENG, B.;LI, K.; CHEN, B.*Scheduling a single batch-processing machine with non-identical job sizes in fuzzy environment using an improved ant colony optimization, Journal of Manufacturing Systems*, v. 29, p. 29–34, 2010.

CHOI, B-C.; CHUNG, J. *Two-machine flow shop scheduling problem with an outsourcing option. European Journal of Operational Research*, v. 213 p. 66-72, 2011.

CHUNG, D-Y.; CHOI, B-C. *Outsourcing and scheduling for two-machine ordered flow shop scheduling problems. European Journal of Operational Research*, v. 226, p. 46-52, 2013.

COLORNI, A., DORIGO, M., MANIEZZO, V. *Distribuited optimization by ant colonies. In: European Conference of Artificial Life*. [S.l.: s.n.], 1991.

CONGRAM, R.K.; POTTS, C.N.; VAN DE VELDE, S.L. *An iterated dynasearch algorithm for the single-machine total weighted tar-diness scheduling problem.* INFORMS Journal on Computing, v. 14, p. 52–57, 2002.

DEN BESTEN, M., STÜTZLE, T., DORIGO, M., 2000. *Ant colony optimization for the total weighted tardiness problem. In: PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature. Springer-Verlag, London*, UK, p. 611–620, 2000.

DORIGO, M; MANIEZZO, V.; COLORNI, A. *The ant system: Optimization by a colony or cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-PartB*, v. 26, p. 29-41,1996.

DORIGO, M; STUTZLE, T. *The ant colony optimization metaheuristic: Algorithms, applications, and advances. Handbook of Metaheuristics, Kluwer Academic Publishers*, p. 251-285, 2002.

FERNANDES, C.F.F, GODINHO FILHO, M. Planejamento e Controle da Produção – Dos fundamentos ao Essencial, São Paulo: Atlas, 2002.

GAGNÉ, C., PRICE, W.L., GRAVEL, M. *Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-depen-dent setup times. J. Oper. Res. Soc.*, v. 53, p. 895–906, 2002.

GAMBARDELLA, L. M.; DORIGO, M. *Solving Symmetric and Asymmetric TSPs by Ant Colonies. IEEE Press*, p. 622-627, 1996.

GEWALD, H.; DIBBERN, J. *Risks and benefits of business process outsourcing: A study of transaction services in the German banking industry*. Information & Management, v. 46, p. 249–257, 2009.

GRAHAM, R.L., LAWLER, E.L., LENSTRA, J.K., RINNOOY, KAN A.H.G. *Optimization and approximation in deterministic machine scheduling: a survey.Annals of Discrete Mathematics 5*, 287–326, 1979.

GUPTA, S. R; SMITH, J. S. *Algorithms for single machine total tardiness scheduling with sequence dependent setups. European Journal of Operational Research*, v. 175, p. 722-739, 2006.

HAYES, R. H.; PISANO, G. P.; UPTON, D. M.; WHEELWRIGHT, S. D. *Operations, Strategy, and Technology: Pursuing the Competitive Edge*. Wiley, 2005.

HOLTHAUS, O.; RAJENDRAN, C. *A fast ant-colony algorithm for single-machine scheduling to minimize the sum of weighted tardiness of jobs*. *J. Oper. Res. Soc.*, v. 56, p. 947–953, 2005.

HUMPHREYS, P.; MCLVOR, R.; HUANG, G. *An expert system for evaluating the make or buy decision.Computers & Industrial Engineering*, v. 42, p. 567-585, 2002.

JIAN, H.; XU, M. *The value of production outsourcing: A real options perspective. In:* 2006 *International Conference on Management Science and Engineering*, [S.1.: s.n.], 2006.

KASHAN, A.H., KARIMI, B. *Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework. J. Oper. Res. Soc.*, v. 59, p. 1269–1280, 2008.

KENNEDY, J. F.; KENNEDY, J.; EBERHART, R. C. *Swarm Intelligence, Morgan Kaufmann*, p. 434, 2001.

KOK, T.G. *Capacity allocation an outsourcing in a process industry*. *International Journal of Production Economics*, v. 68, p. 229-239, 2000.

LEE, I. S.; SUNG, C. S. *Minimizing due date related measures for a single machine scheduling problem with outsourcing allowed.European Journal of Operational Research,* 2008a, p. 931-952, v. 186.

LEE, I. S.; SUNG, C. S. *Single machine scheduling with outsourcing allowed.Internacional Journal of Production Economics*, v. 111, p. 623-634, 2008b.

LEE, J.; HUYNH, M. Q.; CHI-WAI, K. R.; PI, S. *The evolution of outsourcing research: What is the next issue? In:33rd* Hawaii International Conference in System Sciences, [S.l.: s.n.], 2000.

LEE, K. CHOI, B-C. Two-stage production scheduling with an outsourcing option. European Journal of Operational Research, v. 213, p. 489–497, 2011.

LEE, Y. H.; JEONG, C. S.; MOON, C. *Advanced planning and scheduling with outsourcing in manufacturing supply chain*. Computers & Industrial Engineering, v. 43, p. 351-374, 2002.

LIAO, C.-J., JUAN, H.-C. *An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. Comput. Oper. Res.*, v. 34 (7), p. 1899–1909, 2007.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; STÜTZLE, T.; BIRATTARI, M. *The irace package, Iterated Race for Automatic Algorithm Configuration. Technical Report* TR/IRIDIA/2011-004, IRIDIA, *Université libre de Bruxelles, Belgium*, 2011.

M'HALLAH, R.; ALHAJRAF, A. *Ant Colony Optimization for the Single Machine Total Earliness Tardiness Scheduling Problem, Springer-Verlag Berlin Heidelberg*, p. 397–407, 2008.

MADUREIRA, A.; FALCÃO, D.; PEREIRA, I. *Ant Colony System based approach to Single Machine Scheduling Problems: Weighted tardiness scheduling problem*. *Nature and Biologically Inspired Computing* (NaBIC) , p-86-91, 2012.

MAHALIK, D. K. *Selection of Outsourcing Agency through AHP and Grey Relational Analysis: A Case Analysis*. Springer-Verlag Berlin Heidelberg 2011.

MARTINS, R. A. *Metodologia de pesquisa em engenharia de produção e gestão de operações*. In: MIGUEL, P. A. C. (Ed.). [S.1.]: Cap. Abordagens Quantitativa e Qualitativa, p. 45-62, Elsevier, 2010.

MELOUK S.; DAMODARAN P.; CHANG. P-Y. Minimizing makespan for single batch-processing machine with non-identical job sizes using simulated annealing. International Journal of Production Economics, v. 87(2), p. 141 – 148, 2004.

MERKLE, D.; MIDDENDORF, M. *An ant algorithm with a new pheromone evaluation rule for total tardiness problems. In: EvoWorkshops*., p. 287– 296, 2000.
MERKLE, D.; MIDDENDORF, M. *Ant colony optimization with global pheromone evaluation for scheduling a single machine. Appl. Intell.* 1, p. 105–111, 2003.

MISHRA, N.; CHOUDHARY, A. K.; TIWARI, M. K. *Modeling the planning and scheduling across the outsourcing supply chain: a Chaos-based fast Tabu–SA approach*. International Journal of Production Research, v. 46, nº 13, p. 3683–3715, 2008.

MOCCELLIN, J.V., Introdução à programação de operações em máquinas. Escola de Engenharia de São Carlos. USP, julho, 1999.

MOGHADDAM, A.; YALAOUI, F.; AMODEO, L. *An Efficient Meta-heuristic Based on Self-control Dominance Concept for a Bi-objective Re-entrant Scheduling Problem with Outsourcing. Springer-Verlag Berlin Heidelberg, Y. Hamadi and M. Schoenauer (Eds.)* , p. 467-471, 2012.

MOKHTARI, H.; ABADI, I. N. K. *Scheduling with an outsourcing option on both manufacturer and subcontractors. Computers & Operations Research*, v. 40, p. 1234-1242, 2013.

MOKHTARI, H.; ABADI, I. N. K.; AMIN-NASERI, M. R. *Production scheduling with outsourcing scenarios: a mixed integer programming and efficient solution procedure. International Journal of Production Research*, v. 50, n. 19, p. 5372-5395, 2012.

MOON, Y. *Efforts and efficiency in partial outsourcing and investment timing strategy under market uncertainty*. Computers & Industrial Engineering, v. 59, p. 24–33 2010.

MORABITO, R.; PUREZA, V. "Modelagem e simulação", em: Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações, Paulo A. C. Miguel (ed), Editora Campus/Elsevier, ISBN 978-85-352-3523-4, Rio de Janeiro, p. 165-192, 2010.

MULLEN, R. J.; MONEKOSSO, D.; BARMAN, P.; REMAGNINO, P. *A review of ant algorithms, Expert Systems with Applications*, v. 36, p. 9608-9617, 2009.

NAGANO, M. S. ; MOCCELLIN, J. V. ; LORENA, L. A. N. . Programação da Produção Flow Shop Permutacional com Minimização do Tempo Médio de Fluxo, 2004, SÃO JOÃO DEL-REI - MG. ANAIS DO XXXVI SBPO, v. 36, p. 114-123, 2004.

OMAR, M. K.; TEO, S. C.; SUPPIAH, Y. *Scheduling with Setup Considerations: An MIP Approach, Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner*,ISBN 978-3-902613-02-8, *Itech Education and Publishing, Vienna, Austria*, p.436, 2007.

POURHASHEMI, A. P.; ANSAREY, M. S. M. M. *Ant Colony Optimization Applied to Optimal Energy Management of Fuel Cell Hybrid Electric Vehicle, IV International Congress on Ultra Modern Telecommunications and Control Systems*, 2012.

PRABHAKAR, M.; SINGH, J. N.; MAHADEVAN G. *Defensive Mechanism for VANET in Game Theoretic Approach Using Heuristic Based Ant Colony Optimization, International Conference on Computer Communication and Informatics*, Coimbatore, India, 2012.

QI, X. *Coordinated Logistics Scheduling for In-House Production and Outsourcing.* IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, V. 5, N. 1, 2008a.

QI, X. Two-Stage Supply Chain Scheduling with an Option of Outsourcing in Stage One, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, V. 5, N. 1, 2008b.

QI, X. TWO-STAGE PRODUCTION SCHEDULING WITH AN OPTION OF OUTSOURCING FROM A REMOTE SUPPLIER, Systems Engineering Society of China & Springer-Verlag, 2009.

QI, X. *Outsourcing and production scheduling for a two-stage flow shop. International Journal Production Economics*, v. 129, p. 43-50, 2011.

RAVINDRAN, K. *Asset Transfer in IT Outsourcing: Divesting Commodities or Inviting Investment?* Springer-Verlag Berlin Heidelberg, 2012.

SHI, L.; DONG, H-B. *Application of Decision-Making Model Based on Structure Entropy in IT-Outsourcing Supplier Selection*. Springer-Verlag Berlin Heidelberg, 2012.

SIPPER, D. e BULFIN JR.; R.L. Production : Planning, Control and Integration, New York: Mc Graw Hill, 1997.

SLACK, N.; CHAMBERS, S.; JOHNSON, R. Administração da produção. 3.ed. São Paulo: Atlas, 2009.

STÜTZLE, T.; HOOS, H. H. MAX–MIN Ant System. Future Generation Computer Systems, v. 16, p. 889-914, 2000.

TASGETIREN, M. F.; PAN, Q-K., LIANG, Y-C. *A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times. Computers & Operations Research*, v. 36, p. 1900-1915, 2009.

TAVARES NETO, Roberto F. Proposta de solução de problemas de *scheduling* considerando possibilidade de terceirização usando a técnica de otimização por colônia de formigas. 148p. *Tese (Doutorado em Engenharia de Produção).* Departamento de Engenharia de Produção da *Universidade Federal de São Carlos, São Carlos, 2010.*

TAVARES NETO, R. F.; GODINHO FILHO, M. *A software model to prototype ant colony optimization algorithms. Expert Systems with Applications*, v. 38, p. 249-259, 2011a.

TAVARES NETO, R.; GODINHO FILHO, M. *An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. Computers & Operations Research* v. 38, p. 1286 –1293, 2011b.

TAVARES NETO, R. F. . *Moving to Outsourcing in Production Scheduling: Quantitative Approaches*. In: Fabiano E. Molinelli, Leonardo S. Paccagnella. (Org.). *Economics of Regulation and Outsourcing*. 1ed.: *Nova Publishers*, 2012a.

TAVARES NETO, R.F.; GODINHO FILHO, M. Otimização por colônia de formigas para o problema de sequenciamento de tarefas em uma única máquina com terceirização permitida. Revista Gestão & Produção. Artigo aceito para publicação, 2012b.

THIRUVADY, D.; BLUM, C.; MEYER, B.; ERNST, A. *Hybridizing beam-ACO with constraint programming for single machine job scheduling. In: Lecture Notes in Computer Science. Springer Link*, p. 30–44, 2009.

TIAN, J.; CHEN, L. *Depth image up-sampling using ant colony optimization, 21st International Conference on Pattern Recognition*, Tsukuba, Japão, 2012.

WANG, Y.; ZHANG, J.; ZHAO, Y. *ACO-based routing and spectrum allocation in flexible bandwidth networks. Photon Netw Commun*, v. 25, p. 135-143, 2013.

WILHELM, W.; HAN, X.; LEE, C. Computational comparison of two formulations for dynamic supply chain reconfiguration with capacity expansion and contraction. Computers & Operations Research, v. 40, p. 2340-2356, 2013.

XU, R.; CHEN, H-P.; ZHU, J-H.; SHAO, H. *A Hybrid Ant Colony Optimization to Minimize the Total Completion Time on a Single Batch Processing Machine with Non-identical Job Sizes, Fourth International Conference on Natural Computation*, p. 439-443, 2008a.

XU, R.; CHEN, H-P.; HUANG, G. Q.; SHAO, H. CHENG, B-Y. LIU, B-W. *Minimizing the total completion time on a single batch processing machine with non-identical job sizesusing ant*

colony optimization, Industrial Electronics and Applications. ICIEA 2008. 3rd IEEE Conference, p. 2211-2215, 2008b.

XU, R.; CHEN, H.; LI, X. *Makespan minimization on single batch-processing machine via ant colony optimization. Computers & Operations Research*, v. 39, p. 582 –593, 2012.

YADAV, V.; GUPTA, R.K. *A pragmatic and methodological review of research in outsourcing.Information Resources Management Journal*, v. 21, n. 1, p. 27-43, 2008.

ZHI-LONG, C., CHUNG-LUN, L. Scheduling with subcontracting options; *IIE Transactions*, v. 40, no. 12, p. 1171-1184, 2008.