UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ERYCA TATYANE MARTINHO DE AMORIM

PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS: UMA APLICAÇÃO NO RAMO DE PETRÓLEO

MOSSORÓ
2015

ERYCA TATYANE MARTINHO DE AMORIM

PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS: UMA APLICAÇÃO NO RAMO DE PETRÓLEO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Dario José Aloise – UERN.
Co-Orientador: Prof. Dr. Carlos Heitor Pereira Liberalino – UERN.

MOSSORÓ

2015

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

Amorim, Eryca Tatyane Martinho de
Problema de escalonamento de projetos com restrição de recursos: uma aplicação no ramo de petróleo. / Eryca Tatyane Martinho de Amorim. – Mossoró, RN, 2015.

67 f.
Orientador: Prof. Dr. Dario José Aloise

Dissertação (Mestrado em Ciência da Computação) Universidade do Estado do Rio Grande do Norte. Universidade Federal Rural do Semi-Árido.

1. Ciência da Computação. 2. Escalonamento de Projetos - Restrições de Recursos. 3. Flow shop. 4. Otimização por Colônia de Formigas. I. Aloise, Dario José. II.Universidade do Estado do Rio Grande do Norte. III. Título.

UERN/BC CDD 004

ERYCA TATYANE MARTINHO DE AMORIM

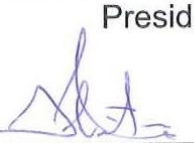
PROBLEMA DE ESCALONAMENTO DE PROJETOS COM RESTRIÇÃO DE RECURSOS: UMA APLICAÇÃO NO RAMO DE PETRÓLEO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação – associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação

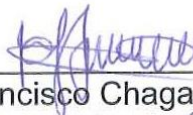
APROVADA EM: ____/____/_____

BANCA EXAMNINADORA

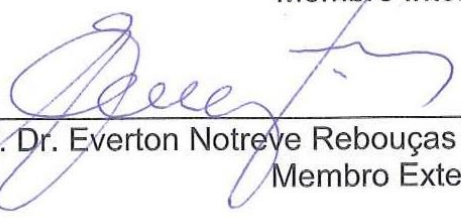
Prof. Dr. Dario José Aloise – UERN
Presidente



Prof. Dr. Carlos Heitor Pereira Liberalino – UERN
Co-Orientador



Prof. Dr. Francisco Chagas de Lima Junior – UERN
Membro Interno



Prof. Dr. Everton Notreve Rebouças Queiroz Fernandes – UnP
Membro Externo

Dedico aos meus pais, Joaquim Martinho da Silva e Isa Paiva de Amorim, que sempre me incentivaram a estudar, mostrando que a única herança que ninguém pode nos tomar são os conhecimentos adquiridos.

AGRADECIMENTOS

A Deus, primeiramente, pela minha vida.

Aos meus pais, Joaquim Martinho e Isa Paiva, por ter me ensinado a viver, me ensinando a ir de encontro aos meus objetivos através do meu esforço. Se conquistei mais essa vitória, devo a vocês.

Ao meu orientador, Prof. Dr. Dario José Aloise pelo incentivo e disponibilidade, pela palavra amiga e de apoio nos momentos difíceis, pela confiança depositada a mim. Por ter acreditado que eu conseguiria chegar ao final dessa batalha. Foi uma honra ter sido orientada pelo senhor.

Ao meu co-orientador, Prof. Dr. Carlos Heitor Pereira Liberalino pela orientação, presteza, disponibilidade, compreensão e tolerância, qualidades estas de suma importância para a construção de um trabalho dissertativo.

As Universidades Estadual do Rio Grande do Norte e Federal Rural do Semi-Árido, por me oferecer e proporcionar os recursos necessários para elaboração e execução deste trabalho.

Aos meus amigos, em especial, Márcio Furukava e João Phellipe, a Juliene, Thiago Henrique, Danielle, Caio, Giannini, Monallisa, Adair, Karol, Tatiane, Camila, Rayane, Olga, Carol, Raphaella, pelo apoio moral e de força nos momentos delicados e de queda e pelos momentos de descontração para “desviar o peso do meu pensamento”. Sem vocês não teria terminado esse desafio.

A CAPES pelo apoio financeiro.

A todos, que direta ou indiretamente, tornaram esse sonho possível.

Muito Obrigada!!

"Vai ter com a formiga, ó preguiçoso; olha para os seus caminhos, e sê sábio. Pois ela, não tendo chefe, nem guarda, nem dominador, prepara no verão o seu pão; na sega ajunta o seu mantimento.
O preguiçoso, até quando ficarás deitado? Quando te levantarás do teu sono?
Um pouco a dormir, um pouco a tosquenejar; um pouco a repousar de braços cruzados; assim sobrevirá a tua pobreza como o meliante, e a tua necessidade como um homem armado".

Provérbios 6: 6-8

RESUMO

Este trabalho propõe mostrar a aplicabilidade na prática do escalonamento de tarefas de lançamento de linhas de dutos de petróleo utilizando a metaheurística colônia de formigas para resolução do problema *flow shop*. Este problema é classificado na literatura como NP-difícil e consiste em determinar o sequenciamento de tarefas de um projeto, objetivando a minimização do *makespan* no projeto, o tempo total do processo, assim como gerenciar os recursos disponíveis de forma otimizada, tudo isso considerando as restrições de precedência entre as tarefas, fornecendo assim, uma ordem dos serviços a serem executados. Para tanto, foram desenvolvidas quatro parâmetros para comparação da eficiência da aplicação, sendo, duas programações, executada pelo Microsoft Excel, denominadas sequencial e paralela, a otimização pela colônia de formigas implementada em Java e os valores ótimos encontrados pelo CPLEX via MATLAB. Com relação aos resultados obtidos pelo CPLEX e ACO, temos um GAP de até 3% no valor encontrado para o *makespan*, sendo a ACO resolvida em tempo computacional inferior. Após realização dos testes e comparação dos resultados obtidos foi observado que para instâncias pequenas, a aplicação da colônia de formigas reduziu o valor do *makespan* em 50% em relação aos usados de maneira empírica e para instâncias maiores em 25%, sendo estes valores bastante significativos quando tratamos de locação de equipamentos.

Palavras-Chave: Escalonamento de Projetos com Restrições de Recursos. *Flow shop*. Otimização por Colônia de Formigas. NP-difícil. Linha de dutos de petróleo.

ABSTRACT

This work proposes to show the applicability in practice of scheduling tasks launch lines of oil pipelines using meta-heuristic ant colony to resolve the flow shop problem. This problem is classified in the literature as NP-hard and is to determine the sequence of a project tasks, aiming to minimize the makespan in the project, the total process time, as well as manage the available resources optimally, all this considering the precedence constraints between tasks, thus providing a sequence of services to be performed. To this end, four parameters were developed for comparison of application efficiency, and two schedules, executed by Microsoft Excel, called sequential and parallel, optimizing the ant colony implemented in Java and the optimal values found by CPLEX via MATLAB. After performing the tests and comparison of results it was observed that for small instances, the application of ant colony reduced the value of the makespan by 50% and larger instances by 25%, which are quite significant values when dealing with equipment rental.

Keywords: Scheduling Projects with Resource Constraints. Flow shop. Optimization by Ant Colony. NP-hard. Line oil pipeline.

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

ACO	*Ant Colony Optmization*
AG	Algoritmo Genético
ASP	Autorização de Serviço Parcial
CLF	Controlador de Lógica Fuzzy
CM	Construção e Montagem
FSP	Problema de Programação de *Flow Shop*
MI	Manutenção e Inspeção
OM	Ordem de Manutenção
PCP	Planejamento e Controle da Produção
PPP	Problemas de Programação de Projetos
PSP	Problema de Escalonamento de Projetos

SUMÁRIO

1 INTRODUÇÃO

O escalonamento das tarefas de um projeto de diversas naturezas, como por exemplo, de venda ou de prestação de serviços, se faz presente nas ações organizacionais de uma empresa. Essas tarefas devem ser executadas em um tempo determinado, levando em conta elementos como: mão-de-obra especializada, equipamentos adequados, dentre outras, objetivando a minimização dos custos ou a maximização das receitas.

Planejar as etapas de um projeto complexo não é uma atividade fácil, devido ao grande número de serviços e os detalhes que a permeiam. Saber quais tarefas deverão ser executadas, num tempo determinado, pela mão-de-obra e equipamentos disponíveis é um problema combinatório complexo, classificado como NP-difícil (CRAVO, 2009), e muitos são os fatores que poderão influenciar no planejamento e execução.

No ramo petrolífero existem muitas empresas que fornecem seus serviços para a perfuração de poços de petróleo, assim como a sua busca, recuperação, transporte, refino e distribuição. Mais especificamente: construção e manutenção de instalações dos poços, lançamento de dutos de petróleo, fornecimento de alimentação para funcionários, contratação de mão-de-obra, venda de matéria prima, dentre outros.

Nesse contexto, portanto, existe a necessidade de modelar um escalonamento as tarefas de um projeto a serem executadas por uma prestadora de serviços do segmento de construção e montagem de instalações, as quais são solicitadas por uma empresa petrolífera, aqui denominada “cliente”. Esses serviços são solicitados através de Autorizações de Serviços Parciais (ASP) baseando-se no efetivo de mão-de-obra e dos equipamentos disponíveis, no prazo estabelecido para execução dos serviços pelo cliente, minimizando, assim, os custos.

Este problema de escalonamento foi observado numa empresa terceirizada, prestadora de serviços da Petrobras na cidade de Mossoró/RN, que executa serviços no ramo do petróleo e gás. Mais especificamente, de Construção e Montagem (CM) e de Manutenção e Inspeção (MI), onde ambos descrevem: escopo do serviço, tempo de execução, valor pago pelo serviço ao prestador e localização do serviço.

A problemática considerada nesse trabalho gira em torno das tarefas parciais, as quais são solicitadas pelo cliente. Ocorre que muitas solicitações são enviadas antes mesmo das anteriores serem concluídas, o que gera um entrave na escolha de qual tarefa será executada inicialmente, tendo em vista a sua prioridade, custo de operação, disponibilidade de mão-de-obra e equipamentos para execução. Cabe então à empresa executora escalonar todas as tarefas, sem ultrapassar o período estipulado para execução.

O presente trabalho objetiva, especialmente, utilizar métodos de otimização que defina a sequência das tarefas em cada máquina na instalação de linhas de dutos de petróleo, com intuito de minimizar o *makespan*, ou seja, o tempo total do projeto. As tarefas serão inseridas e excluídas à medida que são solicitadas e concluídas, respectivamente.

O trabalho está dividido da seguinte forma: a seção 2 aborda a descrição do problema e apresenta uma revisão bibliográfica; a seção 3 trata da modelagem matemática do problema; na seção 4 são elencados os métodos exatos e as heurísticas a serem utilizadas no trabalho; na seção 5 são expostos os resultados computacionais encontrados e, por fim, na seção 6 são explanadas as considerações finais.

2 REVISÃO BIBLIOGRÁFICA

A seguir discorreremos brevemente sobre os Modelos de Programação de Projeto, onde abordaremos na seção 2.1 sobre os Problemas da Produção e suas características. Na seção 2.1.2 será explicado o Problemas de Programação de Projetos e seus pontos principais. A seção 2.1.3 aborda sobre os Problemas de Planejamento (*schedulling*) e suas definições. A seção 2.1.4 aborda sobre a Programação em ambientes *flow shop* e suas particularidades. Os métodos exatos e aproximativos serão abordados na seção 2.2, sendo a otimização por colônia de formigas abordada de maneira mais completa na seção 2.2.2.1, tendo em vista ser o nosso objeto de estudo.

2.1 MODELOS DE PROGRAMAÇÃO DE PROJETO

2.1.1 Programação da Produção

A produção de qualquer produto ou serviço pode ser vista como um processo, ou um conjunto de processos, que tem como objetivo transformar insumos em produtos ou serviços. A execução, controle e planejamento das atividades de produção são de responsabilidade do Planejamento e Controle da Produção (PCP), o qual deve garantir que sejam produzidos os produtos certos, na quantidade certa e no tempo certo (SLACK et al., 2002). O propósito do planejamento e controle da produção é garantir que ocorra de maneira eficaz e gere produtos e serviços da forma planejada, com a qualidade que os clientes esperam (SLACK, 2002).

Graves (1981) definiu a programação da produção como sendo a alocação, no tempo, dos recursos disponíveis de produção de tal forma que satisfaça, da melhor maneira possível, um conjunto de critérios. Para ele, o problema de programação da produção envolve um conjunto de tarefas a serem realizadas e os critérios, que podem envolver decisões entre o término mais cedo e/ou mais tarde de cada uma delas.

Para Baker (1974) a programação da produção é definida, de um modo geral, como alocação de recursos no passar do tempo.

Este problema de programação da produção é NP-árduo (GAREY et al, 1976; RINNOOY KAN, 1976), e portanto, a busca por uma solução ótima apresenta importância mais teórica do que prática, direcionando as pesquisas para o desenvolvimento de métodos heurísticos e metaheurísticos, haja vista esses métodos serem desenvolvidos inicialmente para problemas clássicos, aplicáveis na realidade, porém em pouca profundidade.

2.1.2 Problemas de Programação de Projetos

Programação de projetos é a aplicação de habilidades, técnicas e intuição adquiridas através do conhecimento e experiência para desenvolver modelos de programação eficaz. O modelo de programação integra e organiza logicamente componentes diversos do projeto, tais como atividades, recursos e relações lógicas, para aumentar a probabilidade da conclusão do projeto de sucesso dentro do período determinado (NEGREIRO, 2013).

O Problema de Programação de Projetos (PPP) consiste em encontrar um instante de início para todas as tarefas de forma a minimizar um ou mais objetivos, como, por exemplo, o instante do término do projeto (*makespan*) quando os recursos são limitados. Existe uma grande variedade de problemas de acordo com os elementos do projeto. Atividades podem ser executadas de diversos modos, como, por exemplo, um trabalhador em oito períodos de tempo ou quatro trabalhadores em dois períodos de tempo (ARENALES, 2007).

Existem categorias de recursos, tais como recursos renováveis, que estão disponíveis em cada período (máquinas, equipamentos, mão-de-obra), e recursos não-renováveis, que são limitados ao longo do horizonte de planejamento (capital disponível). Além disso, há uma diversidade de objetivos, tais como minimização do tempo total de um projeto (*makespan*) ou do custo atribuído ao projeto, como também tempo de atraso ou de adiantamento, maximização da qualidade ou do valor presente, quando existe um fluxo de caixa ao longo do projeto, com despesas

com atividades e receitas geradas ao término de partes do projeto (ARENALES, 2007).

Existem algumas nomenclaturas bastante utilizadas em trabalhos técnicos, que são definidas segundo Baker (1982), tais como:

- Tarefa – que também pode ser designada por “ordem”, “produto”, “trabalho” (*job*), “corrida”, entre outros. Pode ser um produto ou um lote de produtos idênticos, que devem ser processados pelos recursos produtivos.
- Máquina – que também pode ser “posto”, “processador”, “equipamento” entre outros, é o recurso produtivo destinado a executar uma operação. Outro recurso produtivo é o recurso humano que pode ser designado por “homem”.
- Operação ou processamento – é o trabalho realizado pelo recurso produtivo sobre a tarefa.
- Tempo de processamento – é o tempo necessário para se concluir uma operação sobre uma tarefa. Neste trabalho, o tempo de processamento incluirá os tempos de preparação, transporte, colocação e retiradas dos recursos produtivos.
- Roteiro – neste trabalho, roteiro será a sequência ordenada de recursos produtivos pelos quais devem passar as tarefas. Cada tarefa terá um roteiro independente.
- Programa – é uma sequência viável, onde estão especificadas as datas de início e término de processamento de cada tarefa;
- Tempo ocioso – no presente trabalho, a referência a tempo ocioso estará associada ao intervalo de tempo em que uma máquina pode ficar parada quando, estando disponível e tendo uma tarefa para atender, ela espera para atender outra tarefa. Outros tempos em que as máquinas possam estar paradas, sem tarefa nenhuma na fila, não serão designados por tempo “ocioso”. Deste modo, inserção de tempo ocioso será fazer a máquina ficar parada mesmo que tendo uma tarefa pronta para ser atendida na fila.

Baker (1997) afirma que *schedule* (ou escalonamento) caracteriza um plano tangível, tal como de horários de ônibus ou o de aulas. Usualmente um

escalonamento informa quando certas coisas devem acontecer; ela mostra o plano de tempos de certas atividades e responde a questão: - quando alguma coisa terá lugar? A resposta para essa questão usualmente nos informa o horário (data), como por exemplo: uma aula de matemática se inicia às 13:00 hrs e finaliza às 17:00 hrs. Entretanto, outra resposta válida poderia ser em termos de sequência e não horário (data).

Tendo definido *schedule*, Baker (1997) definiu também *scheduling (*programação) como sendo, implicitamente, o processo de geração de *schedules* e que a estrutura dos problemas de programação na indústria é a seguinte: um conjunto de tarefas a serem executadas por um conjunto de recursos disponíveis. Deste modo, dado um conjunto de tarefas e um conjunto de recursos, o problema de programação seria o de determinar os tempos detalhadamente de execução de cada tarefa dentro da capacidade de recursos disponíveis.

A finalidade de um sistema de programação da produção é a utilização eficiente de recursos limitados na fabricação de produtos de modo a satisfazer a demanda dos clientes e gerar lucros para a empresa. As restrições incluem a disponibilidade de recursos e a satisfação dos clientes. A satisfação do cliente consiste em fornecer uma variedade de produtos a baixo custo e entregá-lo sem atraso e com qualidade (OLIVEIRA, 2002).

A classificação dos problemas de programação deve considerar fatores internos, que a influenciam por meio das requisições tecnológicas determinadas principalmente pelo fluxo padrão das tarefas nas máquinas, pelo número e tipos de máquinas disponíveis e pela precedência das tarefas (NAGANO, 1998).

O fluxo padrão consiste no direcionamento das tarefas com relação às máquinas, de tal forma que sempre utilize a mesma máquina para processar determinada tarefa numa data fixa. O número de tipos de máquinas e a quantidade de máquinas de cada tipo disponíveis para desempenhar as tarefas são importantes para a classificação da programação, pois materializam o tamanho da alocação dos recursos no tempo, permitindo a utilização específica dos métodos de otimização (SOUZA, 2009).

Para Nagano (1998) os principais problemas de programação encontrados na literatura são: *Job Shop*; *Open Job Shop*; *Batch Shop*; *Flow Shop*; *Batch/Flow Shop*; *Manufacturing Cells* (células de manufatura); *Assembly Shop* (estação de montagem); *Assembly Line* (linha de montagem); *Transfer Line* (linha de

transferência); *Flexible Transfer Line* (linha de transferência Flexível). Este trabalho considera o problema do tipo *flow shop*, o qual definiremos mais adiante.

2.1.3 Problema de Planejamento (*scheduling*)

Planejamento é um processo de otimização, no qual máquinas e recursos limitados são designados ao longo do tempo para atender diversas solicitações que são compostas de diversas atividades, denominadas tarefas. Esta designação é feita de tal maneira que sejam respeitadas integralmente as restrições de tempo de execução das atividades e o limite de capacidade do conjunto de máquinas e recursos usados por todas as atividades (PINEDO, 2005).

Em um modelo de fabricação, tem-se normalmente um recurso chamado "máquina" e uma tarefa que feita pela máquina, que é tipicamente chamada de *job*. Num processo de produção, o *job* pode estar ligado a uma única operação (task) ou várias operações que têm que ser realizadas em máquinas diferentes (PINEDO, 2005).

As principais decisões envolvidas nesse nível são: designação (sequenciamento) de tarefas (jobs), recursos e programação (planejamento) das tarefas em cada recurso, isto é, a sequência de processamento das tarefas e o instante de início e término do processamento de cada tarefa. Em geral, o problema de planejamento é caracterizado por três conjuntos: conjunto $T = \{T_1, T_2, \ldots, T_n\}$ de n tarefas, conjunto $M = \{M_1, M_2, \ldots, P_m\}$ de m máquinas e o conjunto $R = \{R_1, R_2, \ldots, R_s\}$ de s tipos de recursos (BLAZEWICZ *et al*., 2007).

Os principais tipos de problema de planejamento são: *Job Shop*: n tarefas e m máquinas, em que cada tarefa é processada nas m máquinas, de acordo com um roteiro e sem paradas das tarefas para execução de outras; *Open Shop*: Têm as mesmas características do *job shop*, com a permissão de parar tarefas para execução de outras e pode não haver relação de precedência entre as operações; *Flow Shop*: é um caso particular do *job shop*, em que as n tarefas têm o mesmo roteiro nas m máquinas (BLAZEWICZ *et al*., 2007; BRUCKER, 2007; ARENALES *et al*., 2007).

O Problema de Planejamento consiste em programar as atividades de forma que nenhuma infrinja suas relações de precedências e não extrapole as quantidades de recursos disponíveis. Cada atividade pode possuir um ou mais modos de execução, com diferentes combinações de duração e consumo de recursos (SANTOS, 2013).

Dois elementos são de fundamental importância para problemas de planejamento: as tarefas, que constituem cada uma das etapas do projeto a ser executado, e os recursos, que são os insumos necessários para que uma tarefa seja executada. As tarefas estão conectadas entre si através de relações de precedência que determinam a ordem em que as tarefas podem ou não ser executadas. Normalmente, estas relações são do tipo *finish-to-start*, ou seja, é preciso que uma tarefa predecessora seja completamente executada antes de uma tarefa sucessora começar a ser. Também é muito comum que uma tarefa possa ter mais de uma predecessora. Neste caso, todas as predecessoras precisam ter sido executadas antes da tarefa em questão começar. O objetivo mais comum é fazer com que todas as tarefas do projeto sejam executadas o mais rapidamente possível, respeitando as restrições de precedência e de utilização dos recursos (SILVA, 2012).

Os recursos que são necessários para a execução das tarefas são o outro elemento a ser administrado no PSP. Uma classificação bastante tradicional os divide em dois grupos: recursos renováveis e recursos não-renováveis. Os recursos renováveis são aqueles que, após ser utilizados na execução de uma tarefa do projeto, ficam novamente disponíveis para ser utilizados em outra tarefa ainda não executada. Alguns exemplos de recursos desta classe são as máquinas (escavadeiras, tratores, computadores) e os profissionais (engenheiros, programadores, assistentes). Estes recursos podem ser reutilizados ao final de uma etapa de projeto. Os recursos são classificados como não-renováveis se eles estiverem disponíveis uma única vez durante todo horizonte de tempo no qual deve ser tratado o problema. Uma vez que eles são utilizados (consumidos) não é mais possível contar com eles até o fim do problema. Exemplos mais comuns são combustíveis e dinheiro, entre outros (SILVA, 2012).

Para Silva (2012) tradicionalmente, os problemas de planejamento não supõem a geração e sim o consumo dos recursos que são dados de entrada com valores pré-definidos uma vez que ou estes têm caráter não-renovável ou têm sua taxa de renovação bem definida pelo problema, como visto em (VALLS, 2008) e

(NONOBE, 2002). Estes cenários, no entanto, não são capazes de modelar certas situações onde, a partir do término da execução de uma etapa do projeto, esta passa a gerar recursos adicionais.

2.1.4 Programação em *Flow Shop (Flow Shop Schedule)*

O problema de programação de *flow shop* (FSP) é um problema onde se mantêm a mesma ordem de programação das n tarefas em todas as m máquinas. Basicamente com o objetivo de determinar entre as $n!$ possíveis sequências aquela que minimize alguma função objetivo estabelecida, tais como tempo total da programação das tarefas (*makespan*), tempo ocioso das máquinas (*idletime*), tempo de fluxo das tarefas (*flowtime*), entre outros (GIGANTE, 2010).

Em um *flow shop*, segundo Baker (1997), cada tarefa tem sua própria sequência de processamento com fluxo linear unidirecional, ou seja, sem retorno no fluxo. O fluxo pode ser discreto, contínuo ou “semicontínuo”, cada tarefa só pode ser atendida por uma máquina de cada vez e cada máquina só pode atender uma tarefa por vez, não podendo a tarefa retornar para atendimento em máquinas anteriores. No caso mais simples, cada tarefa exige o mesmo conjunto de atividades, na mesma sequência e no mesmo conjunto de máquinas. Assim, no ambiente *flow shop*, n tarefas devem ser programadas para processamento em um conjunto de m máquinas distintas, tendo o mesmo fluxo de processamento, o fluxo unidirecional. Quando, em todas as máquinas, a ordem de processamento das tarefas é a mesma, tem-se o *flow shop permutacional*.

A figura 01 mostra o esquema genérico de um *flow shop*, conforme Baker (1997), onde as tarefas podem entrar por qualquer máquina, pular máquinas, mas sempre seguindo um fluxo unidirecional.

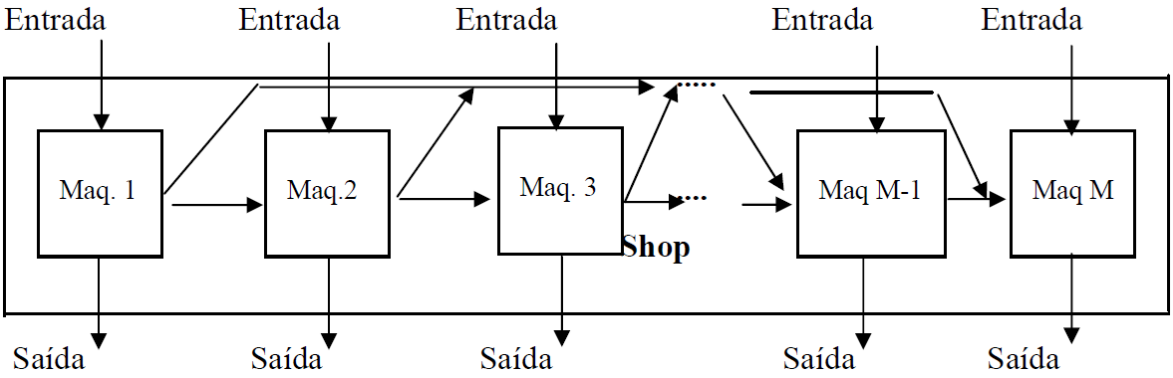


Figura 01 – Fluxo de processamento em um *flow shop* generalizado.

Fonte: Souza (2009)

O FSP é classificado como NP-difícil para a maioria dos problemas clássicos. Já são considerados NP-difícil os seguintes problemas: $F_2 \mid\mid \sum C_j$, um *flow shop* com duas máquinas (F_2) em série (||) com o objetivo de minimizar o somatório do tempo de término de todas as tarefas (C_j); $F_2 \mid\mid L_M$, um *flow shop* com duas máquinas (F_2) em série (||) com o objetivo de minimizar o máximo atraso (L_M); $F_3 \mid\mid C_M$, um *flow shop* com três máquinas (F_2) em série (||) com o objetivo de minimizar o *makespan* (C_M) (PINEDO, 2008).

De acordo com Gigante (2010) as hipóteses consideradas no problema de escalonamento de *flow shop* são:

a) cada máquina está disponível continuamente, sem interrupções;
b) cada máquina pode processar apenas uma tarefa de cada vez;
c) cada tarefa pode ser processada por uma máquina de cada vez;
d) os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos;
e) as tarefas têm a mesma data de liberação, a partir da qual, qualquer uma pode ser programada e executada;
f) os tempos de preparação das operações nas diversas máquinas são incluídos nos tempos de processamento e independem da sequência de operações em cada máquina;
g) uma vez iniciadas as operações nas diversas máquinas, elas não devem ser interrompidas.

Na figura 02 temos um exemplo representativo de sequenciamento de um problema de escalonamento *flow shop* para n = 3 e m = 5, onde se observa que a tarefa 1 é iniciada na máquina 1, depois na máquina 2, na máquina 3, na máquina 4 e, por último na máquina 5. Tão logo a máquina 1 esteja liberada da tarefa 1, será iniciada a tarefa 2, que será executada posteriormente, pela máquina 3, 4 e 5 e, assim, até que todas as tarefas sejam processadas por todas as máquinas sequencialmente. É observado também, que a execução da tarefa 2 só será iniciada na máquina 2 após a finalização na máquina 1.

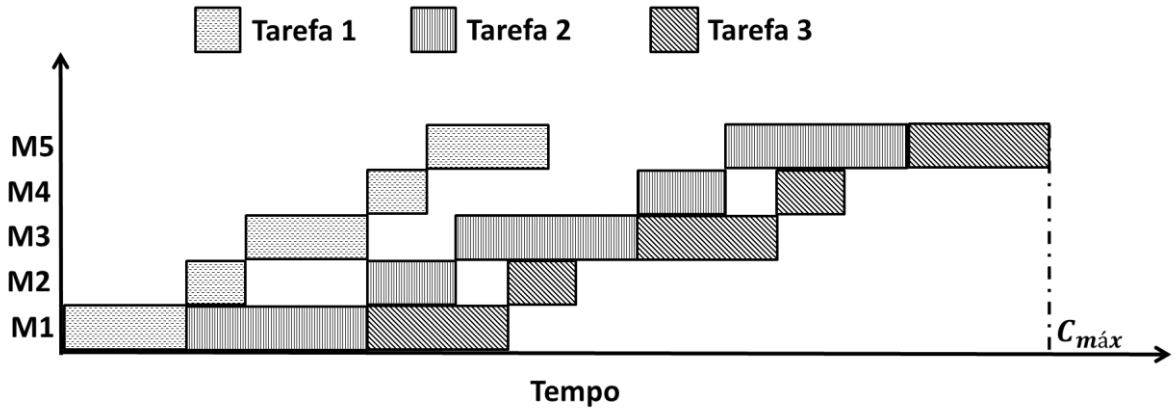


Figura 02 – Exemplo de um escalonamento *flow shop* para n = 3 e m = 5.

Fonte: Autoria Própria (2015).

Legenda:

M1, M2, M3, M4, M5 – Tarefa executada pelas máquinas 1, 2, 3, 4 e 5, respectivamente.

$C_{máx}$ – *Makespan* encontrado

Observe que um problema de escalonamento *flow shop* envolvendo apenas 10 tarefas apresenta 3.628.800 sequências possíveis de programação (PÉREZ, 2011; GIGANTE, 2010).

Muitos esforços têm sido feitos no sentido de resolver o problema de escalonamento *flow shop* envolvendo diferentes metodologias. Por exemplo, Widmer; Hertz (1989) propuseram um método heurístico para resolver problema de escalonamento *flow shop* com o objetivo de minimizar apenas o *makespan*. Este método é composto de duas fases: a primeira fase considera uma sequência inicial correspondendo a uma solução do problema do caixeiro viajante, e a segunda tenta melhorar essa solução usando técnicas de busca tabu. Já o trabalho de Ho (1995)

apresenta uma heurística para minimizar apenas o tempo de fluxo médio para o problema de escalonamento *flow shop* (PÉREZ, 2011).

2.2 MÉTODOS EXATOS E APROXIMATIVOS

2.2.1 Exatos

Desde que Johnson (1954) propôs uma solução ótima (exata) para o problema de n tarefas sendo processadas em 2 máquinas, vários outros métodos foram desenvolvidos para resolver o problema de sequenciamento em sistemas *flow shop* de n tarefas e m máquinas, minimizando o critério do *makespan*. Em todos eles, muitas restrições foram feitas, as quais, embora distantes de situações reais, simplificaram o problema (GIGANTE, 2011).

Palmer (1965) propôs um índice denominado *slope index*, a partir do qual se estabelece a sequência de processamento das tarefas nas máquinas. Tal índice é calculado de forma que as tarefas em que os tempos de processamento tendem a crescer na sequência das máquinas devem receber maior prioridade na programação, ou seja, devem ocupar as primeiras posições na ordem de execução (GIGANTE, 2011).

A seguir trataremos dos três tipos básicos de algoritmos exatos: os baseados em árvores de precedência, os baseados no conceito de alternativas de modo e atraso e os baseados em alternativas de modo e extensão.

1. Algoritmos baseados em árvores de precedência

Algoritmos baseados em árvores de precedência (*precedence tree*) consistem em construir ao longo de uma árvore de pesquisa diferentes escalonamentos, escolhendo nos ramos da árvore pares de modo-atividade que possam ser escalonados assim como instantes de tempos mais cedo em que esse escalonamento pode ser realizado. Um ramo termina quando a atividade escalonada

é a fictícia que representa o fim do projeto. O processo de *backtracking* realizado ao longo da árvore gera soluções alternativas que podem ser comparados pela função objetivo. Um caminho que leva da raiz da árvore até um nível mais baixo corresponde a uma sequência de escalonamento válida de atividades, isto é, respeita as relações de precedência (BRUCKER, 1999). Algoritmos baseados em árvore de precedência foram propostos por (PATTERSON, 1989) e melhorados posteriormente por (SPRECHER, 1994) e (SPRECHER, 1996).

2. Algoritmos baseados no conceito de alternativas de modo e atraso

Algoritmos baseados no conceito de alternativas de modo e atraso (*mode and delay alternatives*) baseiam-se na construção de uma árvore de pesquisa na qual cada nó está associado a um determinado instante de tempo. Em cada nó escalonam-se temporariamente atividades cujas antecessoras já tenham sido escalonadas. Isso é feito escolhendo entre vários pares de modo-atividade possíveis. Se alguma restrição de recurso tiver sido violada com esse escalonamento, calculam-se conjuntos de atividades dentre as que acabaram de ser escalonadas. Esses conjuntos representam as atividades que podem tornar válido o escalonamento parcial realizado no nó da árvore se todas forem atrasadas. Entre os vários conjuntos, escolhe-se um dos conjuntos mínimos, isto é, um conjunto tal que se for removida uma atividade qualquer o atraso das atividades restantes não será suficiente para garantir a validade do escalonamento parcial (LEAL, 2007); (BRUCKER, 1999).

3. Algoritmos baseados em alternativas de modo e extensão

Algoritmos baseados em alternativas de modo e extensão (*mode and extension alternatives*) são parecidos com o anterior. A diferença está no fato de não ser permitido efetuar escalonamentos parciais nos nós da árvore de pesquisa que viole restrições de recursos. O conceito de extensão referido no nome do algoritmo consiste em um conjunto de pares modo-atividade escalonáveis num determinado nó, e que estendem o escalonamento parcial do nó no nível superior sem que

nenhuma restrição ligada aos recursos seja violada (LEAL, 2007); (BRUCKER, 1999).

2.2.2 Aproximativos

Heurísticas ou métodos de aproximação, podem ser descritas como sendo regras de escolha que devem ser seguidas para a decisão de qual tarefa ser escalonada em um dado momento. Ao contrário dos métodos exatos, os métodos baseados em heurísticas não garantem que uma solução ótima será encontrada, mas sim uma solução boa (SOARES, 1994).

De acordo com os estudos de Colin (2007) heurísticas são procedimentos generalistas que são adaptados à cada problema que se deseja atuar. Sua aplicação, ainda para Colin (2007) não é decorrente da otimalidade da função (pois métodos heurísticos não garantem a otimização do problema em que são aplicadas), e sim por sua flexibilidade e menor custo computacional. O menor custo computacional torna o uso de heurísticas possível para a resolução de problemas cuja solução por meio de métodos exatos seja proibitiva devido ao tempo necessário para a produção da resposta (TAVARES NETO, 2010).

Já uma Metaheurística, conforme explana Carabetti (2010, p.22) tem a capacidade de gerar soluções suficientemente boas em um tempo razoável para problemas de elevada complexidade, as metaheurísticas são procedimentos heurísticos que guiam outras heurísticas, “experimentando” o espaço de soluções além do ótimo local.

A seção 2.2.2.1 aborda mais especificamente a metaheurística Otimização por Colônia de Formigas tendo em vista ter sido utilizada nesse trabalho como método de resolução.

2.2.2.1 Otimização por Colônia de Formigas (ACO)

A Metaheurística Otimização por Colônia de Formiga (ACO) foi criada por Colorni, Dorigo e Maniezzo (1991) é uma metaheurística de base populacional que utiliza um conjunto de agentes (formigas artificiais) para percorrer um grafo conectado, de modo a construir uma solução à um problema de otimização combinatória. As formigas movem-se vértice a vértice ao longo das arestas do grafo construído, explorando as informações fornecidas pelos valores de feromônio depositado e, desta forma constroem uma solução para o problema (CARABETTI, 2010).

De acordo com Tavares Neto; Godinho Filho (2013) o algoritmo ACO busca imitar o comportamento de formigas reais indo do ninho até a fonte de alimento. Para isso, agentes computacionais (“formigas”) são posicionados em um grafo e forçados a se movimentar pelos nós até que uma condição de parada seja satisfeita. As setas representam ilustradas na figura 03, representam os níveis de feromônio dos caminhos que a formiga pode percorrer, sendo o comprimento da seta a quantificação desse feromônio. O caminho feito pela formiga é uma solução do problema.

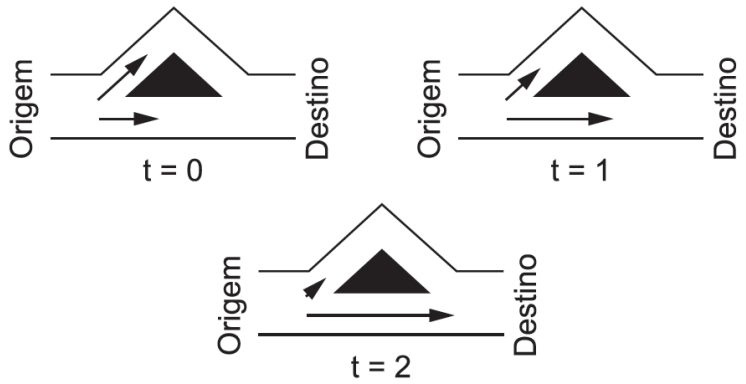


Figura 03 – Comportamento básico da formiga em um ACO.
Fonte: Tavares Neto; Godinho Filho (2013).

A figura 03 mostra como a heurística foi desenvolvida por Colorni, Dorigo e Maniezzo (1991), onde se pode observar que no instante t=0s a formiga escolhe, de maneira aleatória, por qual caminho seguir, dentre as soluções possíveis

disponíveis. À medida que as formigas vão chegando ao objetivo, a função de avaliação definida previamente analisa, a partir do percurso realizado, a qualidade da solução gerada, onde é depositado o feronômio, que será responsável pelo aprendizado das melhores soluções.

Para que o feronômio não extrapole os níveis depositados Colorni, Dorigo e Maniezzo (1991) programaram uma estratégia de evaporação do feromônio, normalmente modelada como um declive linear do nível atual de feromônio existente, conseguindo aqui escolher o melhor caminho com maior frequência que os demais (TAVARES NETO; GODINHO FILHO, 2013).

Logo, se as formigas escolherem o maior caminho, demorarão para retornar, tendo assim, uma maior evaporação do feromônio. O caminho mais longo tem maior nível de evaporação quando comparado ao caminho mais curto. Após algumas iterações, ou seja, a passagem das formigas pelos melhores caminhos se tem o caminho mais favorável (TAVARES NETO, 2010).

A compreensão a respeito de como formigas podem estabelecer caminhos mais curtos entre a fonte de alimento e seu ninho, visto que são animais cegos, é um problema estudado pelos etnologistas que possibilitou o desenvolvimento da classe de algoritmos de Colônias de Formigas. Foi verificado que, quando as formigas passam por sua trilha, depositam uma substância, chamada de feromônio, que é utilizada para a comunicação entre os indivíduos da colônia, como também serve de forma de auxílio à tomada de decisão para as outras formigas (CARABETTI, 2010).

Para Dorigo et al. (1999), os algoritmos baseados em colônias de formigas possuem, de uma forma geral, as seguintes características:

- apesar de cada formiga ser capaz de encontrar uma solução, soluções de boa qualidade só emergirão como resultado da iteração coletiva entre as formigas;
- cada formiga faz uso apenas de suas próprias informações, além das informações locais sobre o nó que visita;
- as formigas se comunicam apenas de forma indireta, através da leitura e escrita nas variáveis de rastro de feromônio;
- as formigas não são adaptáveis, pelo contrário, modificam a forma como o problema é representado e percebido para as outras formigas.

A figura 04 mostra como o algoritmo de otimização por colônia de formigas foi modelado, conforme proposto por Colorni, Dorigo e Maniezzo (1991) e citado nos estudos de Tavares Neto (2010) onde afirma que o conjunto de formiga constrói uma solução repetidas vezes, a qual é construída através de escolhas de caminhos seguindo a regra de transição, onde é aplicada a regra de atualização local de feromônios a cada movimento da formiga. Posteriormente todas as formigas formarem a solução, que é uma sequencia de movimentos de uma formiga, aplica-se a regra de atualização global de feromônios, onde pode-se complementar com o uso da busca local. Esse processo é executado até que a condição de parada seja satisfeita, podendo ser o número de iterações, ou o tempo computacional tiver sido extrapolado ou ainda a quantidade de vezes que a solução tiver sido repetida.

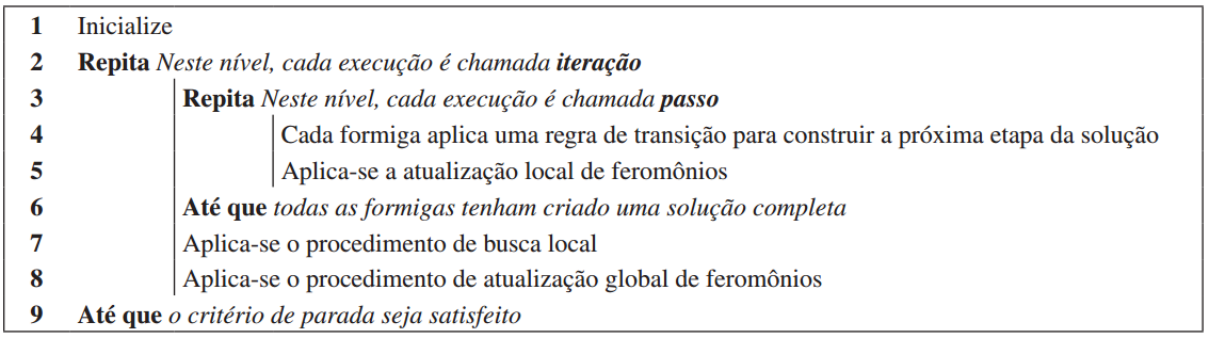


Figura 04 – Algoritmo da Metaheuristica ACO.

Fonte: Tavares Neto; Godinho Filho (2013).

Para Carabetti (2010) a metaheurística consiste de uma etapa de inicialização e de três componentes algorítmicos, ainda de acordo com a Figura 11, cuja ativação é regulada pelo laço enquanto/faça. Essa construção é repetida até que um critério de parada, geralmente definido pelo número máximo de iterações ou tempo máximo de processamento, seja satisfeito.

Primeiro inicializa com um conjunto de formigas para a construção de soluções do problema em questão, soluções estas compostas pelo sequenciamento das tarefas em cada máquina. Logo após, cada formiga vai depositando o feromônio ao longo do caminho. A minimização do *makespan* dá-se com o somatório do custo que cada formiga leva para percorrer o caminho (FONSECA, 2010).

A tabela 01 mostra as principais aplicações da metaheurística ACO, segundo Blum (2005a).

Tabela 01 – Principais aplicações da metaheurística ACO.

Problema	**Autores**
Problema do Caixeiro Viajante	Dorigo, Maniezzo, e Colorni (Dorigo et al., 1991), (Dorigo, 1992), (Dorigo et al., 1996) Dorigo e Gambardella (Dorigo e Gambardella, 1997b), (Dorigo e Gambardella, 1997a) Stützle e Hoos (Stützle e Hoos, 1997), (Stützle e Hoos, 2000).
Problema Quadrático de Alocação	Maniezzo (Maniezzo, 1999) Maniezzo e Colorni (Maniezzo e Colorni, 1999) Stützle e Hoos (Stützle e Hoos, 2000)
Problemas de Agendamentos (*scheduling*)	Stützle (Stützle, 1998) den Besten, Stützle e Dorigo (den Besten et al., 2000) Gagné, Price e Gravel (Gagné et al., 2002) Merkle, Middendorf e Shmeck (Merkle et al., 2002) Blum (respectivamente, Blum e Sampels) (Blum, 2004), (Blum, 2005b)
Problema de Roteamento de Veículos	Gambardella, Taillard e Agazzi (Gambardella et al., 1999) Reimann, Doerner e Hartl (Reimann e Hartl, 2004)
Problema de Horário Escolar	Socha, Sampels e Manfrin (Socha et al., 2003)
Set Packing	Gandibleux, Delorme e T'Kindt
Coloração de Grafos	Costa e Hertz (Costa e Hertz, 1997)
Problema de Menor Supersequencia	Michel e Middendorf (Michel e Middendorf, 1998)
Problema de Ordenação Sequencial	Gambardella e Dorigo (Gambardella e Dorigo, 2000)
Problemas de Satisfação de Restrições	Solnon (Solnon, 2002)
Problema de Clique Máximo	Bui e Rizzo Jr (Bui e Rizzo, 2004)
Problema de Caminhos Disjuntos	Blesa e Blum (Blesa e Blum, 2004)
Problemas Multi-Objetivos	Guntsch e Middendorf (Guntsch e Middendorf, 2003) Lopéz-Ibánez, Paquete e Stutzle (López-Ibáñez et al., 2004) Doerner, Gutjahr, Hartl, Strauss e Stummer (Doerner et al., 2004)

Problemas Dinâmicos (estocásticos)	Guntsch e Middendorf (Guntsch e Middendorf, 2001) Bianchi, Gambardella e Dorigo (Bianchi et al., 2002)

No próximo capítulo são apresentados os resultados computacionais encontrados com a implementação dos quatro parâmetros de escalonamento, objeto de estudo deste trabalho.

3 DESCRIÇÃO DO PROBLEMA

O problema abordado na presente dissertação almeja otimizar o escalonamento das tarefas de um projeto. Particularmente trata da atividade de lançamento de linhas de dutos de petróleo, onde se leva em consideração os seguintes aspectos: o tempo de execução das tarefas, o custo e a ociosidade envolvida na utilização dos equipamentos.

A motivação para a resolução deste problema se deu devido à ociosidade de equipes e equipamentos observados numa empresa prestadora de serviço do ramo de petróleo, onde o planejamento não era executado de modo otimizado, verificando-se por diversas vezes, equipamentos disponíveis no pátio por falta de planejamento. Vale mencionar também que os equipamentos eram alocados para executar as tarefas em campo e, que muitas vezes aguardavam a conclusão de outras tarefas.

A figura 05 demonstra graficamente como são inicializados a solicitação dos serviços. Inicialmente, o cliente solicita ao prestador, através de uma ordem de serviço a ser executado nomeada como Autorização de Serviços Parciais (ASP) ou na Ordem de Manutenção (OM).

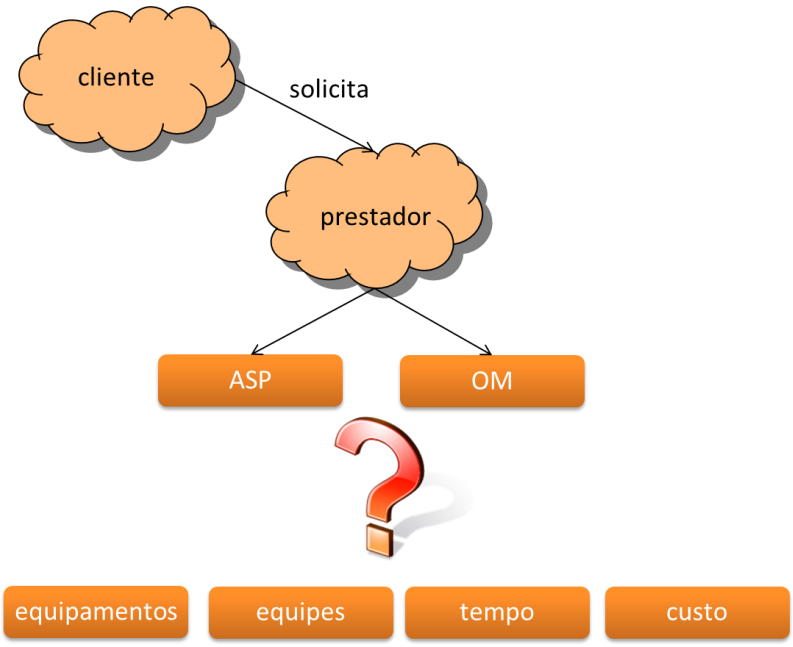


Figura 05 – Modelo de solicitação de um serviço.

Fonte: Autoria própria (2015).

O prestador em seguida, mediante seu planejamento, escalona a sequência dos serviços a serem executados baseando-se na disponibilidade dos seus equipamentos e equipes, com o menor tempo de execução e custo envolvido, buscando a excelência nas suas atividades, com a lucratividade das obras e satisfação do cliente.

A figura 06 mostra o maquinário necessário para a execução do serviço de lançamento de linhas (dutos) de produção de petróleo (ou oleodutos), ou seja, retroescavadeiras, caminhões munck e caçambas.

Figura 06 – Tipos de maquinários para escalonamento. (A) Retroescavadeira, (B) Caminhão Munck, (C) Caçamba.

Fonte: Novas (2015).

Na tarefa de lançamento de uma linha de dutos de petróleo temos as seguintes operações (em parênteses tem-se o maquinário necessário):

A. Abertura de vala (retroescavadeira);

B. Desfile de tubos e descarga de areia nova (caminhão munck e caçamba);

C. Enchimento de ½ vala com areia nova para assentamento dos dutos (retroescavadeira);

D. Lançamento de linha na vala e descarga de areia nova ao lado da calha (caminhão munck e caçamba, como ilustrado na figura 07);

E. Enchimento de mais ½ vala para cobertura da linha (retroescavadeira).

Figura 07 – Lançamento de dutos em valas, após preenchimento com colchão de areia.

Fonte: NOVAS (2015).

A tarefa de desfile de tubos é a disposição dos tubos na superfície para posterior lançamento na vala. O conjunto de operações que compõem a execução da tarefa de lançamento de uma linha de produção, as quais estão associadas equipamentos, deve ser realizada de forma sequencial. Dessa maneira, os equipamentos podem ficar indisponíveis para a execução das tarefas em outros serviços concorrentes, gerando assim, uma ociosidade, pois o equipamento permanece à disposição de uma tarefa enquanto outras estão sendo executadas. Entretanto, isso não ocorre para serviços concorrentes, pois para cada tarefa de um serviço deve ser disponibilizado um equipamento. Logo, torna-se necessária a paralelização na utilização desses equipamentos para que os mesmos fiquem menos ociosos ou sejam utilizados da melhor forma.

A figura 08 mostra o grafo das tarefas em sequência, onde os vértices representam os estados e as arestas as tarefas, como também suas precedências.

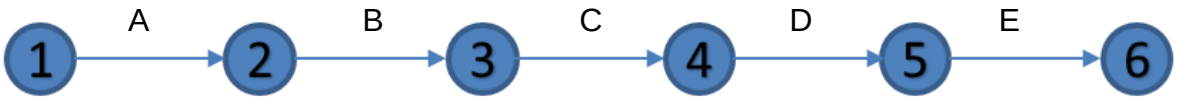


Figura 08 – Grafo representativo das tarefas executadas de maneira sequencial.

Fonte: Autoria Própria (2015).

Detalhadamente, um lançamento de linha ou tarefa consiste na seguinte sequência de passos:

A. Inicialmente em dispor de uma retroescavadeira para a abertura da vala, que é uma escavação onde se faz um colchão de areia para assentar a tubulação ou linha, denominada atividade A.

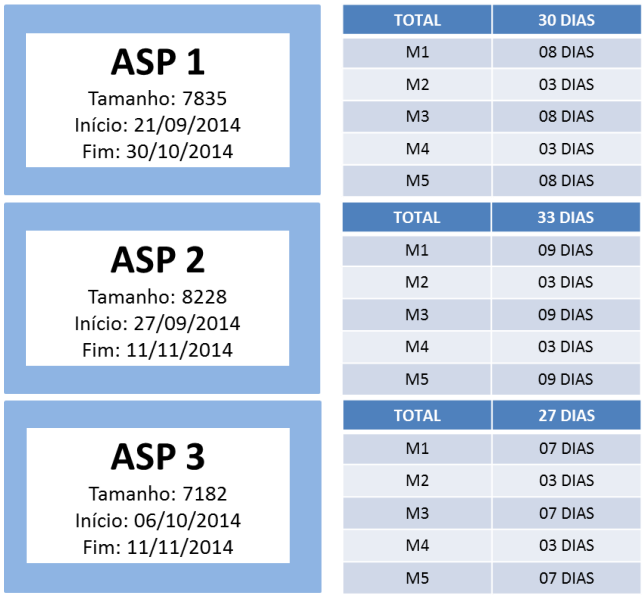
B. Para a atividade B, o caminhão munck deverá estar no local e dispor os tubos ao lado da vala, como também a caçamba que deverá descarregar a areia nova (sem pedras) para preenchimento da vala.

C. A atividade C utiliza uma retroescavadeira para preencher ½ vala, chamado de colchão de areia, para acomodar os dutos.

D. A atividade D com o uso de um caminhão munck transfere os dutos para o colchão formado na vala e a caçamba descarrega areia nova ao longo da vala.

E. A atividade E, com o uso de uma retroescavadeira, preenche a vala com a areia nova já descarregada.

A figura 09 mostra um exemplo com três tarefas denominadas ASP (Autorização de Serviço Parcial) 1, 2 e 3, os quais contêm seu tamanho (da linha de dutos) e suas datas de início e término de execução, onde a partir do tamanho em metros, se calcula os dias necessários para realização das tarefas.



TOTAL	30 DIAS
M1	08 DIAS
M2	03 DIAS
M3	08 DIAS
M4	03 DIAS
M5	08 DIAS

TOTAL	33 DIAS
M1	09 DIAS
M2	03 DIAS
M3	09 DIAS
M4	03 DIAS
M5	09 DIAS

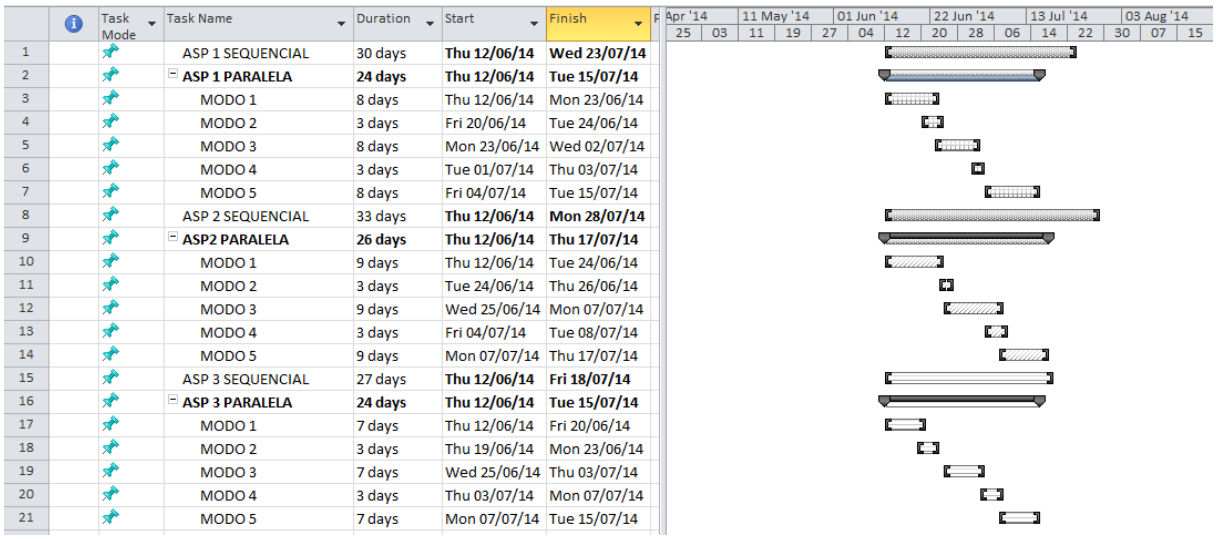
TOTAL	27 DIAS
M1	07 DIAS
M2	03 DIAS
M3	07 DIAS
M4	03 DIAS
M5	07 DIAS

Figura 09 – Descrição do escopo das ASP.

Fonte: Autoria Própria (2015).

Nas tabelas dessa figura, encontram-se dispostos os dados da descrição de cada uma das ASP´s e do tempo de execução das máquinas correspondentes, sendo M1, M2, M3, M4 e M5 os equipamentos (também chamadas “máquinas”).

A figura 10 mostra o planejamento das tarefas e seu diagrama de Gantt, onde a otimização se mostra na paralelização do uso dos equipamentos e das atividades. Neste diagrama, temos três ASP´s programadas em sequencial e em paralelo. Na primeira, ilustrada em quadriculado, temos uma tarefa com duração de 30 dias quando planejadas sequencialmente contra 21 dias de modo paralelo. Já na segunda tarefa, ilustrada em linhas diagonais, temos 33 dias programados em sequencial contra 25 dias em paralelo e, finalmente, a terceira tarefa, ilustrada em linhas paralelas, temos 27 dias em sequencial contra 20 dias em paralelo.



	Task Mode	Task Name	Duration	Start	Finish
1		ASP 1 SEQUENCIAL	30 days	**Thu 12/06/14**	**Wed 23/07/14**
2		**ASP 1 PARALELA**	**24 days**	**Thu 12/06/14**	**Tue 15/07/14**
3		MODO 1	8 days	Thu 12/06/14	Mon 23/06/14
4		MODO 2	3 days	Fri 20/06/14	Tue 24/06/14
5		MODO 3	8 days	Mon 23/06/14	Wed 02/07/14
6		MODO 4	3 days	Tue 01/07/14	Thu 03/07/14
7		MODO 5	8 days	Fri 04/07/14	Tue 15/07/14
8		ASP 2 SEQUENCIAL	33 days	**Thu 12/06/14**	**Mon 28/07/14**
9		**ASP2 PARALELA**	**26 days**	**Thu 12/06/14**	**Thu 17/07/14**
10		MODO 1	9 days	Thu 12/06/14	Tue 24/06/14
11		MODO 2	3 days	Tue 24/06/14	Thu 26/06/14
12		MODO 3	9 days	Wed 25/06/14	Mon 07/07/14
13		MODO 4	3 days	Fri 04/07/14	Tue 08/07/14
14		MODO 5	9 days	Mon 07/07/14	Thu 17/07/14
15		ASP 3 SEQUENCIAL	27 days	**Thu 12/06/14**	**Fri 18/07/14**
16		**ASP 3 PARALELA**	**24 days**	**Thu 12/06/14**	**Tue 15/07/14**
17		MODO 1	7 days	Thu 12/06/14	Fri 20/06/14
18		MODO 2	3 days	Thu 19/06/14	Mon 23/06/14
19		MODO 3	7 days	Wed 25/06/14	Thu 03/07/14
20		MODO 4	3 days	Thu 03/07/14	Mon 07/07/14
21		MODO 5	7 days	Mon 07/07/14	Tue 15/07/14

Figura 10 – Programação das tarefas e diagrama de Gantt.

Fonte: Autoria Própria (2015).

Ainda de acordo com a figura 10, podemos afirmar que o planejamento de forma paralela nos deu um ganho de 09 dias na primeira ASP, 08 dias para a segunda e 07 dias para a terceira. Porém, como veremos adiante, as tarefas concorrem com relação aos equipamentos, o qual é mostrado na figura 11 um exemplo de escalonamento de tarefas x equipamentos. As ASP´s foram dispostas em sequencial (A) e em paralelo (B) para melhor percepção dos ganhos. Porém, buscou-se escalonar as tarefas minimizando a ociosidade dos equipamentos. Com isso, mostra-se que com o planejamento adequado e paralelizado, o projeto será otimizado.

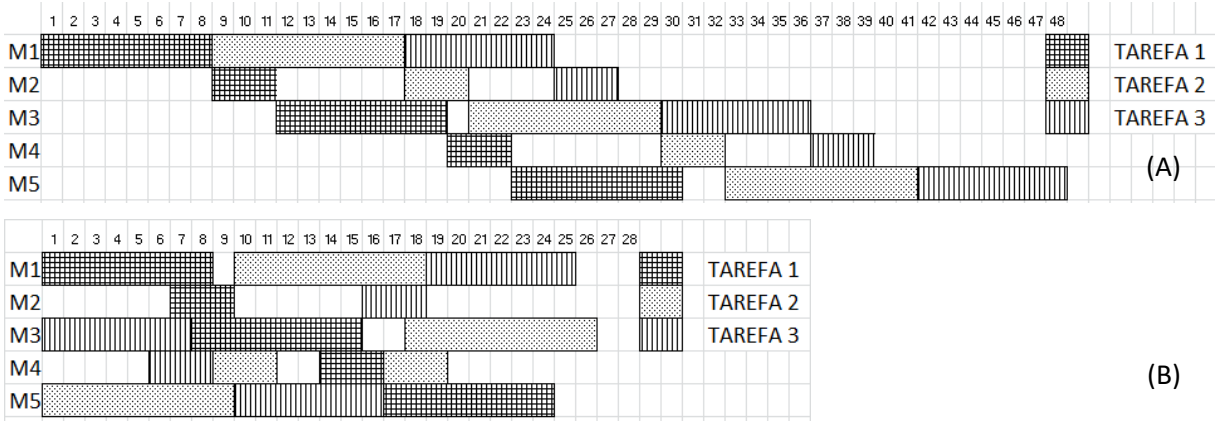


Figura 11 – (A) Uso dos equipamentos de forma serial.
(B) Uso dos equipamentos de forma paralela.
Fonte: Autoria Própria (2015).

Ainda de acordo com a figura 11, observamos que houve uma redução de 15 dias no *makespan* das locações de equipamentos. Observa-se também que em modo sequencial temos 51 dias ociosos de retroescavadeira contra 06 dias de modo paralelo. Já para caçamba / munck temos 70 dias ociosos no modo sequencial contra 34 de modo paralelo, que poderão ser utilizados para execução de tarefas menores ou que tenham um prazo menos comprometido.

3.1 MODELAGEM MATEMÁTICA

Este capítulo apresenta e discute a modelagem matemática, tomando por base a modelagem apresentada por Rubén, Funda, Thijs (2006), encontrado no trabalho de Souza (2009), que aborda a programação de tarefas em *flow shop*.

Utilizaremos, portanto, as mesmas variáveis de decisão para definir a sequência das máquinas, como também os tempos de término das tarefas em cada uma dessas máquinas. No trabalho de Souza (2009) foram incluídas restrições para evitar ciclos de tarefas numa dada máquina. A função objetivo visa minimizar o *makespan* para conclusão das tarefas, sendo essa a diferença em relação ao trabalho de Souza (2009).

Abaixo segue as hipóteses utilizadas na programação:

1. Cada máquina estará disponível continuamente sem interrupções;
2. Cada máquina pode processar apenas uma tarefa de cada vez;

3. Cada tarefa pode ser processada por uma máquina de cada vez;

4. Os tempos de processamento das tarefas nas diversas máquinas são determinados e fixos;

5. As tarefas terão a mesma data de liberação, a partir da qual, qualquer uma pode ser programada e executada;

6. Para cada tarefa i haverá um intervalo de tempo $[d_i, f_i]$, chamado de janela de tempo em que a tarefa não sofrerá nenhuma punição se sua data de término estiver dentro deste intervalo, onde d_i é o tempo inicial para a tarefa i e f_i é o tempo final para tarefa i;

7. Os tempos de preparação das operações nas diversas máquinas são incluídos nos tempos de processamento e independem da sequência de operação em cada máquina;

8. As operações nas diversas máquinas, uma vez iniciadas, não devem ser interrompidas; e, finalmente,

9. Por ser um *flow shop*, o fluxo de atendimento será unidirecional, deste modo, as tarefas não poderão retornar para máquinas anteriores no roteiro.

A notação utilizada para a formulação é a seguinte:

T: é um número suficientemente grande (horizonte de planejamento).

Índices:

i/j: índice para as tarefas;

m: índice para as máquinas;

Conjuntos de tarefas e máquinas:

G_m: conjunto de tarefas que visitam a máquina m;

CM_i: conjunto das máquinas visitadas pela tarefa i;

M: conjunto de máquinas;

N: conjunto de tarefas há uma tarefa fictícia de índice zero;

Parâmetros:

$P_{i,m}$: tempo de processamento da tarefa i na máquina m;

d_i: instante inicial (limite inferior) da janela de tempo para a conclusão da tarefa i;

f_i: instante final (limite superior) da janela de tempo para a conclusão da tarefa i;

n_m: número de tarefas que passam pela máquina m, (para cada máquina há uma tarefa fictícia, de índice zero, com tempo de processamento nulo);

lm_i: última máquina pela qual passa a tarefa i;

$ma_{m,i}$: máquina antecessora da máquina m no fluxo da tarefa i;

Variáveis:

C_{mi}: instante de término ou conclusão da tarefa i na máquina m;

$fl_{m,i,j}$: fluxo da tarefa i para a tarefa j na máquina m (variável introduzida para evitar que haja um ciclo de tarefas numa dada máquina);

$X_{m,i,j}$: variável binária que assume valor 1 quando a tarefa i precede a tarefa j na máquina m e 0 caso contrário;

A formulação matemática do problema é:

$$Min \sum_{i \in N} C_{mi} \quad (1)$$

Sujeito a:

$$\sum_{\substack{i \neq j \\ i \in \{G_m, 0\}}} X_{m,i,j} = 1, j \in N, m \in CM_i \quad (2)$$

$$\sum_{\substack{i \neq j \\ i \in G_m}} X_{m,j,i} \leq 1, j \in N, m \in CM_i \quad (3)$$

$$\sum_{\substack{h \neq j, h \neq i \\ h \in \{G_m, 0\}}} X_{m,h,i} \geq X_{m,i,j} \; ; \; i,j \in N; m \in CM_i \cap CM_j \quad (4)$$

$$X_{m,i,j} + X_{m,j,i} \leq 1 \; ; i \in N; j = i+1, \dots, m; m \in CM_i \cap CM_j \quad (5)$$

$$\sum_{i \in \{G_m\}} X_{m,0,i} \leq 1 \quad ; m \in M \quad (6)$$

$$c_{m,i} \geq C_{m,j} + p_{i,m} - (1 - X_{m,j,i})T \quad ; m \in M; j \in \{G_m, 0\}, j \neq i \quad (7)$$

$$c_{m,i} \geq C_{ma_{m,i},i} + p_{i,m} \quad ; \; m \in M; i \in G_m \quad (8)$$

$$\sum_{j \in G_m} fl_{m,0,i} \geq n_m \quad ; m \in M \quad (9)$$

$$\sum_{\substack{j\in\{G_m,0\}\\ j\neq 1}} fl_{m,j,i} - \sum_{\substack{j\in G_M\\ j\neq 1}} fl_{m,i,j} = 1 \quad ;m \in M; i \in G_m \tag{10}$$

$$fl_{m,i,j} \leq n_m X_{m,i,j} \quad ;m \in M; i \in G_m; j \in G_m; i \neq j \tag{11}$$

$$x_{m,i,j} = 0 \;\; ou \; x_{m,i,j} = 1 \quad ;m \in M; i,j \in N, i \neq j \tag{12}$$

$$fl_{m,i,j} \geq 0 \quad ;m \in M; i,j \in N, i \neq j \tag{13}$$

A equação (1) expressa a função objetivo do problema. O objetivo é definir a sequência das tarefas em cada máquina com intuito de minimizar o *makespan*. A restrição (2) expressa a garantia do antecessor da tarefa j na máquina m, sendo esta uma tarefa real ou fictícia; a restrição (3) expressa a garantia de que há apenas uma tarefa que sucede a tarefa j na máquina m. A restrição (4) expressa a garantia de que, a máquina m passar da tarefa i para a tarefa j, seja preciso ter processado antes a tarefa i, onde a antecessora foi uma tarefa real h ou a tarefa fictícia zero.

A restrição (5) expressa a garantia de que, se na máquina m, j sucede i, então i não pode suceder j na máquina m. Esta restrição evita ciclos de tarefas com duas únicas tarefas, porém não evita ciclos maiores. A restrição (6) impõe que haverá, no máximo, uma tarefa inicial para cada máquina; a restrição (7) impõe que, se a tarefa i for processada na máquina m após a tarefa j, ela não poderá ser iniciada antes do término da tarefa j. A restrição (8) garante que o processamento da tarefa i na máquina m não poderá começar antes que termine o processamento da tarefa i na máquina anterior a m dentro do seu roteiro. ela é desnecessária em problemas de máquina única.

As restrições (9), (10) e (11) evitam a formação de ciclos de tarefas numa dada máquina. Isto é, evitam soluções como, por exemplo, $x_{m,0,j} = 1, x_{m,j,k} = 1, x_{m,k,i} = 1 \; e \; x_{m,i,j} = 1$ mostrado na figura 12.

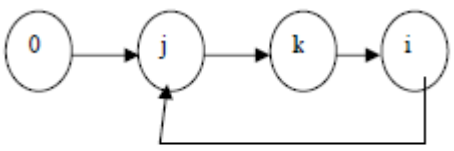


Figura 12 – Exemplo de ciclo, onde 0 é a tarefa fictícia.

Fonte: Souza (2009)

3.2 OTIMIZAÇÃO POR COLÔNIA DE FORMIGAS / *CPLEX*

Como cenário para aplicação do problema foram definidas instâncias com o tamanho da linha a ser lançada, data inicial e final para conclusão da tarefa e tempo necessário para execução da tarefa por cada uma das cinco máquinas executoras. Por tratar-se de um *flow shop* as tarefas inicializam sempre na máquina 1 (M1) e, posteriormente, executadas pelas máquinas 2 (M2), 3 (M3), 4 (M4) e 5 (M5), sequencialmente.

A estratégia utilizada pela colônia de formigas foi fornecer inicialmente todas as possíveis soluções de escalonamento das tarefas, onde à medida que as interações ocorriam, ou seja, que as formigas depositavam o feromônio, calculava-se o custo de cada tarefa. Posteriormente, encontra-se a minimização do *makespan* calculando o somatório dos custos de cada formiga. A figura 13 representa um modelo de sequenciamento de tarefas do tipo *flow shop* em cinco máquinas, onde cada tarefa é representada por uma cor. Esta representação mostra como essas tarefas serão executadas ao longo do tempo em um ambiente *flow shop* na prática.

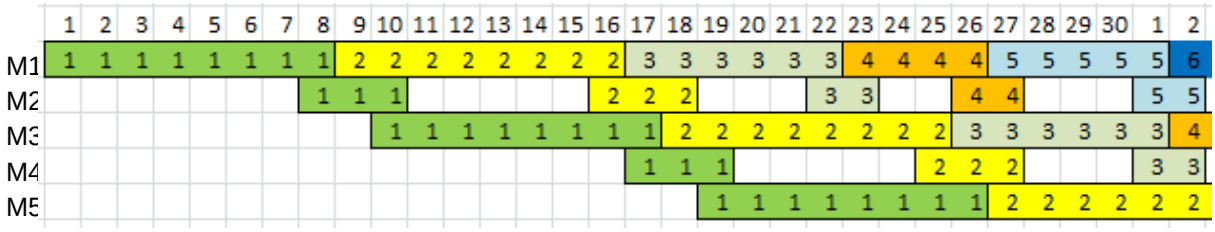


Figura 13 – Representação da programação em gráfico de Gantt.

Fonte: Autoria própria (2015).

Para ilustrar o funcionamento da optimização por colônia de formigas, teremos quatro ASP´s como exemplo. É necessário calcular inicialmente a quantidade de possíveis soluções viáveis para o problema, onde temos 4! = 24 sequências possíveis, ou seja: 1-2-3-4; 1-3-2-4; 1-4-2-3; 1-4-3-2; 1-2-4-3; 1-3-4-2; 2-1-3-4; 2-1-4-3; 2-3-1-4; 2-3-4-1; 2-4-1-3; 2-4-3-1; 3-1-2-4; 3-2-1-4; 3-4-1-2; 3-4-2-1; 3-1-4-2; 3-2-4-1; 4-1-2-3; 4-2-1-3; 4-3-1-2; 4-3-2-1; 4-2-3-1; 4-1-3-2.

Na tabela 02 temos os tempos (dias) de execução de cada tarefa (ASP) por máquina, onde para ASP 1 temos 8 dias de execução na máquina 1, 3 dias na

máquina 2, 8 dias na máquina 3, 3 dias na máquina 4 e 8 dias na máquina 5, totalizando 30 dias totais de trabalho e, assim, sucessivamente para as demais ASP´s.

Tabela 02 – Tempos de execução da tarefa por máquina.

ASP	M1	M2	M3	M4	M5	TOTAL
1	8	3	8	3	8	30
2	8	3	8	3	8	32
3	3	1	3	1	3	11
4	7	3	7	3	7	28

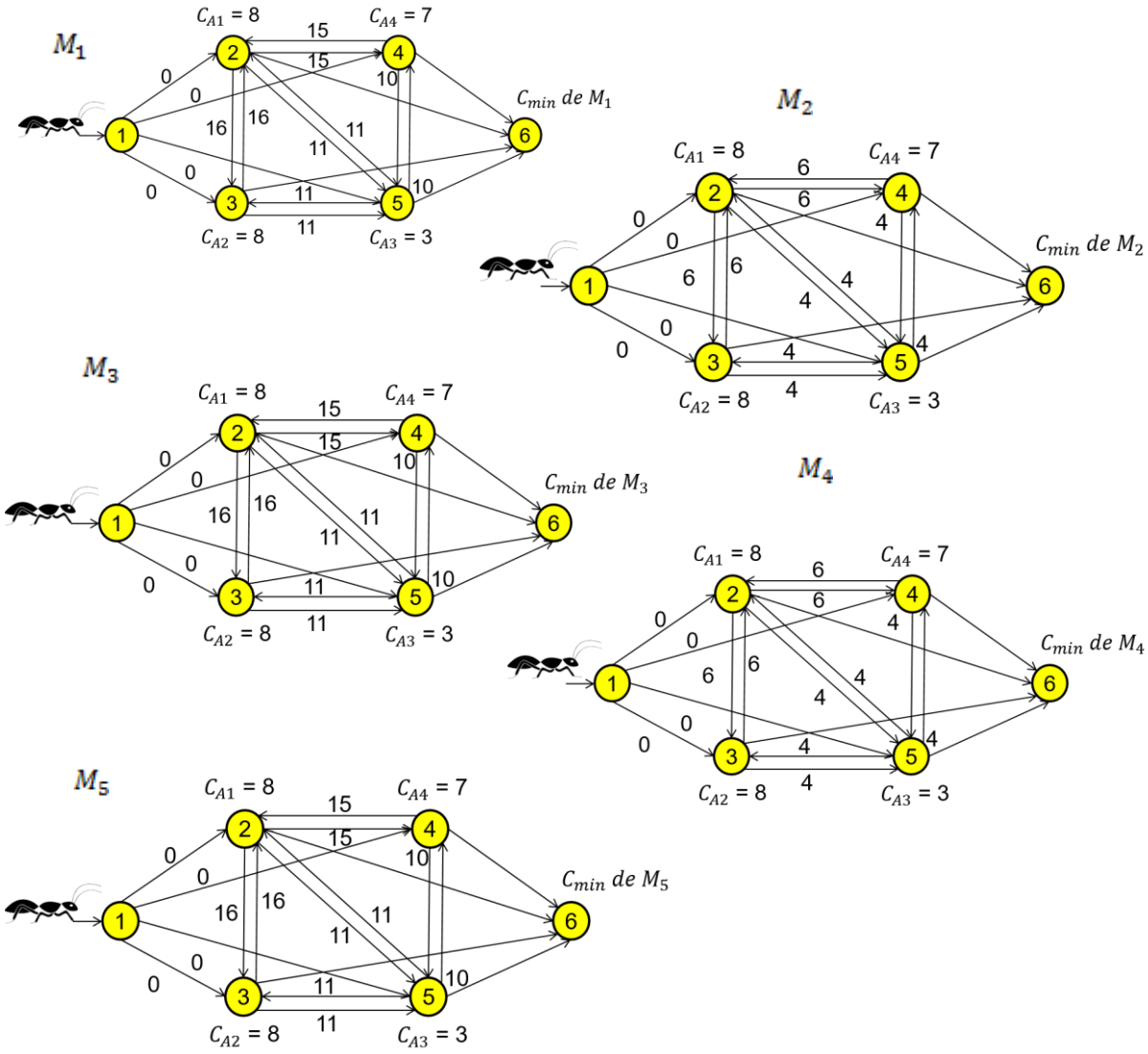
O algoritmo é inicializado a partir do conjunto de sequências possíveis e da matriz de custo dos tempos de execução das ASP´s (tabela 05), onde dispõe o quantitativo de dias necessários para executar, calculadas a partir dos tempos de execução da tarefa por máquina, conforme figura 14.

ASP	M1				M2				M3				M4				M5			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	0	16	11	15	0	6	4	6	0	16	11	15	0	6	4	6	0	16	11	15
2	16	0	11	15	6	0	4	6	16	0	11	15	6	0	4	6	16	0	11	15
3	11	11	0	10	4	4	0	4	11	11	0	10	4	4	0	4	11	11	0	10
4	15	15	10	0	6	6	4	0	15	15	10	0	6	6	4	0	15	15	10	0

Figura 14 – Matriz de custo para quatro ASP´s.

Fonte: Autoria própria (2015).

Cada formiga da colônia vai percorrendo uma sequência do conjunto de soluções escolhida aleatoriamente, onde efetua o depósito de feromônio no retorno à colônia para, assim, influenciar as próximas formigas que foram percorrer o caminho na escolha da solução. O valor do *makespan* é calculado à medida que a formiga escolhe seu percurso no grafo, ilustrado na figura 15, onde C_{A1}, C_{A2}, C_{A3}, C_{A4} que é o custo de execução da ASP 1 na máquina 1, 2, 3, 4 e 5, respectivamente.



$$C_{min}(makespan) = C_{min}\ de\ M_1 + C_{min}\ de\ M_2 + C_{min}\ de\ M_3 + C_{min}\ de\ M_4 + C_{min}\ de\ M_5$$

Figura 15 – Grafo de custo para execução das ASP´s.
Fonte: Autoria própria (2015).

O *flow shop* é realizado pela formiga a cada iteração, onde realiza uma escolha para cada máquina do *flow shop*, respeitando a disponibilidade da mesma, sendo retornado o tempo ao final do laço, sendo este, o tempo necessário para execução de todas as tarefas (*makespan*).

A figura 16 mostra os valores de *makespan* encontrados após o retorno da formiga à colônia, onde temos duas sequências apontadas como de menor *makespan*, no caso, 49 dias para execução das quatro ASP´s nas sequências 2-4-3-1 e 4-1-2-3.

1-2-3-4 – MAKESPAN = 54
1-3-2-4 – MAKESPAN = 53
1-4-2-3 – MAKESPAN = 70
1-4-3-2 – MAKESPAN = 50
1-2-4-3 – MAKESPAN = 70
1-3-4-2 – MAKESPAN = 54
2-1-3-4 – MAKESPAN = 52
2-1-4-3 – MAKESPAN = 69
2-3-1-4 – MAKESPAN = 53
2-3-4-1 – MAKESPAN = 50
2-4-1-3 – MAKESPAN = 68
2-4-3-1 – MAKESPAN = 49
3-1-2-4 – MAKESPAN = 72
3-2-1-4 – MAKESPAN = 73
3-4-1-2 – MAKESPAN = 69
3-4-2-1 – MAKESPAN = 70
3-1-4-2 – MAKESPAN = 68
3-2-4-1 – MAKESPAN = 70
4-1-2-3 – MAKESPAN = 49
4-2-1-3 – MAKESPAN = 72
4-3-1-2 – MAKESPAN = 52
4-3-2-1 – MAKESPAN = 54
4-2-3-1 – MAKESPAN = 53
4-1-3-2 – MAKESPAN = 52

Figura 16 – Menor valor de *makespan* encontrado.

Fonte: Autoria própria (2015).

Como várias formigas farão o mesmo percurso com a melhor solução encontrada, terão assim, depositado um maior número de feromônio ao longo do caminho. Aplica-se aqui o procedimento de atualização global de feromônio. A melhor solução encontrada é apresentada como resolução do problema, com o sequencial de dias em cada máquina, conforme ilustra figura 17.

	M1	M2	M3	M4	M5
ASP1	1.0 - 8.0	8.0 - 11.0	11.0 - 18.0	18.0 - 21.0	21.0 - 28.0
ASP2	8.0 - 16.0	16.0 - 19.0	19.0 - 27.0	27.0 - 30.0	30.0 - 38.0
ASP3	16.0 - 19.0	19.0 - 20.0	27.0 - 30.0	30.0 - 31.0	38.0 - 41.0
ASP4	19.0 - 27.0	27.0 - 30.0	30.0 - 38.0	38.0 - 41.0	41.0 - 49.0

Figura 17 – Representação da solução viável encontrada pelo ACO.

Fonte: Autoria própria (2015).

No próximo capítulo trataremos do referencial teórico para melhor embasamento sobre o problema estudado.

4 RESULTADOS COMPUTACIONAIS

Neste capítulo serão apresentados os resultados do experimento computacional realizado para solucionar o problema de sequenciamento de tarefas em ambiente *flow shop*, que visa comparar o planejamento manual, a metaheurística de Otimização por Colônia de Formigas e resultado obtido via CPLEX. Para isso, realizou-se o experimento em uma *workstation* Intel® Core™ i5-3337U CPU 1.80 GHz, 6 GB de memória RAM e Sistema Operacional Windows 8.1 de 64 Bits.

4.1 DETERMINAÇÃO DAS AMOSTRAS DO EXPERIMENTO

Para o experimento computacional foram utilizadas 300 ASP´s, retiradas do planejamento manual da empresa observada, sendo estas divididas em 10 classes, 20, 30, 50, 80, 100, 200, 220, 250, 280 e 300, ambas variando em 5 máquinas. O conjunto de instâncias encontram-se no apêndice 2 da versão online, disponível em: <http://www2.ufersa.edu.br/portal/cursos/posgraduacao/>.

4.2 IMPLEMENTAÇÃO

Utilizou-se a ferramenta Matlab versão 7.11.0.584 (R2010b) que tem uma linguagem de programação associada para implementar um processo. Também foram usadas as rotinas TOMLAB/CPLEX versão 8.1, que são um conjunto de programas para Matlab para a resolução de problemas de programação linear e quadrática. Para o desenvolvimento em Java foi usada a ferramenta IDE Eclipse Java EE IDE for Web Developers, versão Mars Release (4.5.0), build id 20150621-1200.

4.3 SOLUÇÃO E VALIDAÇÃO DO MODELO

O algoritmo com a metaheurística da Otimização por Colônia de Formigas foi desenvolvido utilizando a linguagem de programação Java. Para validação do modelo foi desenvolvido e adaptado dos estudos de Souza (2009), um modelo matemático para solucionar o problema e comparar com os resultados obtidos na utilização da Otimização por Colônia de Formigas. Este modelo foi resolvido utilizando as rotinas do TOMLAB/CPLEX versão 8.1. Para a programação manual foi utilizada planilha do Microsoft Excel 2010.

4.4 SOLUÇÃO DA PROPOSTA

As soluções obtidas serão descritas logo abaixo, sendo: a programação manual utilizada no planejamento das tarefas, os resultados obtidos utilizando a Otimização por Colônia de Formigas e, finalmente, as soluções de validação do modelo geradas pelo CPLEX.

No experimento computacional utilizaram-se 300 instâncias reais, coletadas em uma empresa prestadora de serviço do ramo de petróleo onde, conforme apresentado na tabela 03, dispunha de nome, tamanho da ASP, data de início e fim, dias totais para execução da tarefa, como também os dias necessários para execução em cada máquina. Como padrão para comparação entre os métodos a serem analisados, foram separados conjuntos de 20 ASP´s, 30 ASP´s, 50 ASP´s, 80 ASP´s, 100 ASP´s, 200 ASP´s, 220 ASP´s, 250 ASP´s, 280 ASP´s e 300 ASP´s.

Cabe ressaltar aqui que para se chegar à contagem dos dias utilizados em cada máquina, foi usada regra de 1000 km/dia para as retroescavadeiras e 2500 km/dia para a caçamba/munck.

Tabela 03 – Exemplificação das instâncias utilizadas.

NOME	TAMANHO	DIA INÍCIO	DIA FINAL	DIAS TOTAIS	DIAS P/ MÁQUINA				
					M1	M2	M3	M4	M5
ASP1	7835	1	45	45	8	3	8	3	8
ASP2	9182	1	52	52	8	3	8	3	8
ASP3	4246	1	30	30	3	1	3	1	3
ASP4	7817	1	51	51	7	3	7	3	7
ASP5	3957	1	49	49	6	3	6	3	6
ASP6	8326	1	42	42	5	2	5	2	5
ASP7	9000	1	57	57	9	4	9	4	9
ASP8	3996	1	39	39	6	3	6	3	6
ASP9	1492	1	62	62	10	4	10	4	10
ASP10	6750	1	43	43	5	2	5	2	5

A programação manual foi planejada usando-se o conhecimento diário de um técnico de planejamento da obra, onde com o auxílio de uma planilha do Microsoft Excel, contabilizou as horas de máquina para cada tarefa, num período de observação de 06 (seis) meses. Ocorre que usualmente as tarefas são executadas de modo sequencial, sendo aplicada uma melhoria quando o trabalho das máquinas dá-se de forma paralelizada. Para isso, contabilizou-se o somatório de horas/máquina para execução de 20 ASP´s, 30 ASP´s, 50 ASP´s, 80 ASP´s, 100 ASP´s, 200 ASP´s, 220 ASP´s, 250 ASP´s, 280 ASP´s e 300 ASP´s, com intuito de analisar os valores encontrados juntamente com as soluções computacionais.

Observa-se no gráfico 01 que planejando 20 ASP´s são necessárias 12 retroescavadeiras para execução de tarefas em sequencial, contra 10 de forma paralela, tendo assim, uma economia de duas retroescavadeiras ao mês. Já para as caçambas/muncks seriam necessárias quatro para execução de forma sequencial, contra duas de forma paralela, tendo economia de duas caçambas/muncks.

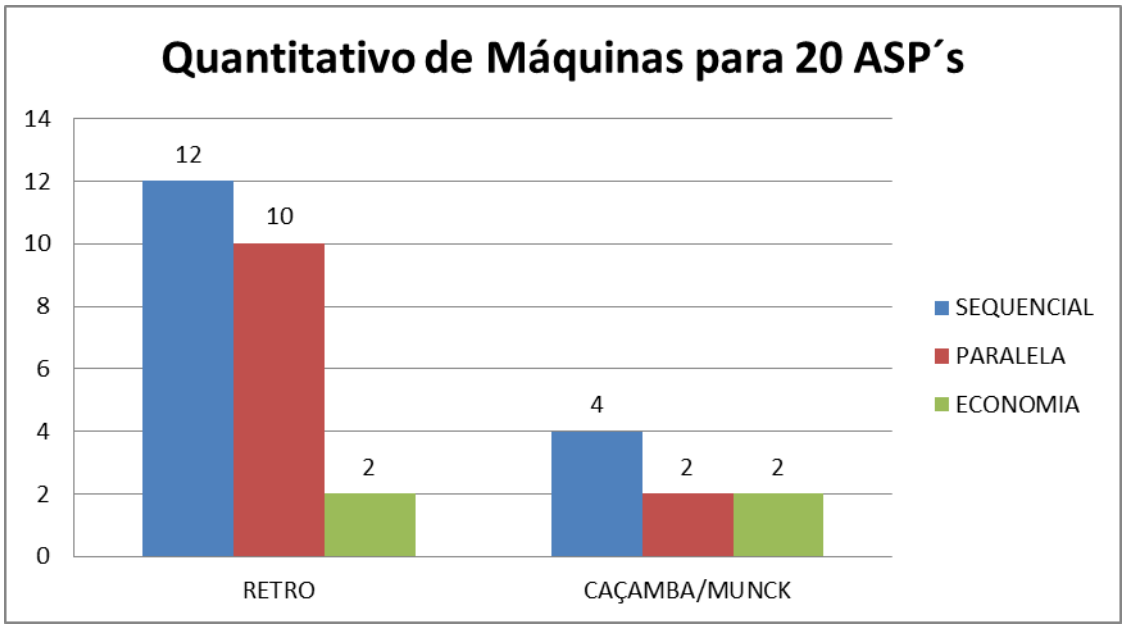


Gráfico 01 – Representação de quantitativo de máquinas utilizadas para 20 ASP´s.

O gráfico 02 mostra a economia de R$ 24.000,00 para retroescavadeiras e R$ 12.000,00 para caçamba/munck, totalizando R$ 36.000,00 em apenas 20 ASP´s.

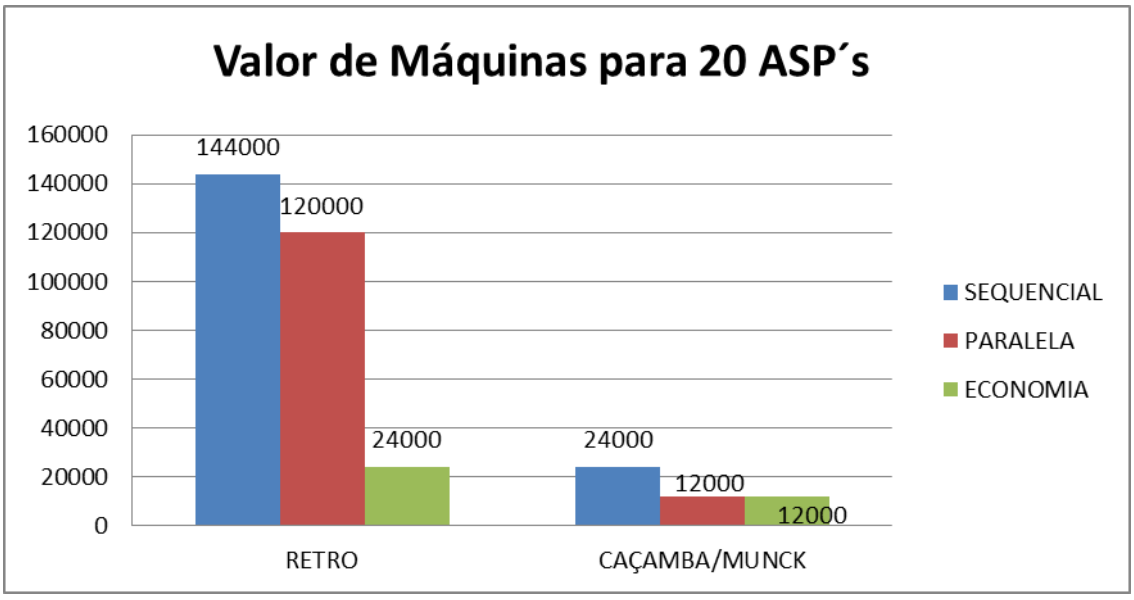


Gráfico 02 – Economia no pagamento de máquinas para execução de 20 APS´s.

De forma análoga, a tabela 04 mostra os ganhos referentes ao quantitativo e valores pagos nas programações com 20 ASP´s, 30 ASP´s, 50 ASP´s, 80 ASP´s, 100 ASP´s, 200 ASP´s, 220 ASP´s, 250 ASP´s, 280 ASP´s e 300 ASP´s.

Tabela 03 – Quantitativo dos ganhos nos padrões de 20, 30, 50, 80, 100, 200, 220, 250, 280 e 300.

ASP´S	RETROESCAVADEIRA				
	QUANT	SEQUENCIAL	QUANT	PARALELA	
20	12	R$ 144.000,00	10	R$ 120.000,00	
30	16	R$ 192.000,00	15	R$ 180.000,00	
50	24	R$ 288.000,00	21	R$ 252.000,00	
80	36	R$ 432.000,00	30	R$ 60.000,00	
100	60	R$ 720.000,00	52	R$ 624.000,00	
200	117	R$ 1.404.000,00	103	R$ 1.236.000,00	
220	126	R$ 1.512.000,00	112	R$ 1.344.000,00	
250	141	R$ 1.692.000,00	123	R$ 1.476.000,00	
280	150	R$ 1.800.000,00	132	R$ 1.584.000,00	
300	174	R$ 2.088.000,00	154	R$ 1.848.000,00	
ASP´S	**CAÇAMBA/MUNCK**				**GANHO COM OTIMIZAÇÃO**
	QUANT	**SEQUENCIAL**	**QUANT**	**PARALELA**	
20	4	R$ 24.000,00	2	R$ 12.000,00	R$ 36.000,00
30	8	R$ 48.000,00	3	R$ 18.000,00	R$ 42.000,00
50	7	R$ 42.000,00	4	R$ 24.000,00	R$ 54.000,00
80	10	R$ 60.000,00	6	R$ 36.000,00	R$ 96.000,00
100	16	R$ 96.000,00	10	R$ 60.000,00	R$ 132.000,00
200	32	R$ 192.000,00	18	R$ 108.000,00	R$ 252.000,00
220	36	R$ 216.000,00	20	R$ 120.000,00	R$ 264.000,00
250	40	R$ 240.000,00	22	R$ 132.000,00	R$ 324.000,00
280	42	R$ 252.000,00	23	R$ 138.000,00	R$ 330.000,00
300	48	R$ 288.000,00	28	R$ 168.000,00	R$ 360.000,00

Ainda conforme observamos na tabela 04, temos um ganho significativo quando paralelizamos o uso das máquinas, tendo em vista que com apenas 80 ASP´s pode-se economizar R$ 96.000,00 com a paralelização do uso dos equipamentos, pois na prática, esses equipamentos encontram-se à disposição da tarefa, podendo aguardar a conclusão de um serviço, executando outra tarefa.

O planejamento é efetuado através de um funcionário planejador de projetos, baseado em sua experiência de campo, onde se faz um acúmulo de horas/máquina e divide-se por 30 dias, tendo-se a quantidade de máquinas a serem utilizadas. Obviamente é feita apenas uma estimativa e a locação das máquinas podem ser aumentadas conforme necessidade, retirando assim, a precisão do planejamento.

Para saber qual seria a sequência de execução das tarefas é calculada a folga entre os dias disponíveis para execução no escopo da ASP e os dias necessários para sua realização pelas máquinas, conforme disposto na tabela 05.

Logo após, classifica-se da menor para a maior folga, tendo assim, o sequenciamento das tarefas.

Tabela 05 – Exemplo de ordenação para instância com 280 ASP´s a partir da sobra.

SEQ	TAM	INÍCIO	FIM	TOTAL	M1	M2	M3	M4	M5	Tempo Máquina	Sobra
ASP60	6841	1	8	7	1,32	0,55	1,32	0,55	1,32	3,0715	3,93
ASP69	17393	1	10	9	1,49	0,62	1,49	0,62	1,49	3,719333	5,28
ASP91	9832	1	68	67	16,57	6,90	16,57	6,90	16,57	61,50683	5,49
ASP72	1434	1	12	11	1,40	0,58	1,40	0,58	1,40	3,374333	7,63
ASP59	8318	1	24	23	4,00	1,67	4,00	1,67	4,00	13,318	9,68
ASP92	5627	1	25	24	3,79	1,58	3,79	1,58	3,79	12,53983	11,46
ASP97	18261	1	60	59	12,85	5,35	12,85	5,35	12,85	47,243	11,76
ASP58	13287	1	45	44	8,91	3,71	8,91	3,71	8,91	32,15883	11,84
ASP73	9815	1	35	34	6,30	2,63	6,30	2,63	6,30	22,15	11,85
ASP68	9235	1	36	35	6,50	2,71	6,50	2,71	6,50	22,91667	12,08

Como tratamos de locação, podemos solicitar máquinas a qualquer momento, sendo locações mensais ou diárias, o que torna o planejamento mais incerto, haja vista que no desenrolar de um projeto de obra, embora bem planejado, muitas peculiaridades são desvirtuadas, acarretando assim, ônus para o projeto.

Para avaliar a eficiência do método, analisou-se os tempos de respostas computacionais entre a Otimização por Colônia de Formigas e o resultado encontrado via CPLEX. Os resultados obtidos no experimento são encontrados na tabela 06.

A tabela 06 mostra a viabilidade na aplicação da metaheurística de Otimização por Colônia de Formigas quando comparada com os valores obtidos via CPLEX.

O modelo proposto conseguiu resolver de maneira exata e aproximada a minimização do *makespan* para as 10 instâncias, sendo observado um Gap em torno de 3%, mostrando uma equiparação dos valores otimizados encontrado pelos métodos exato e aproximado. O Gap foi obtido conforme equação (1).

$$Gap\ (\%) = \left| \frac{makespan\ CPLEX - makespan\ ACO}{makespan\ CPLEX} \right| \quad (1)$$

Tabela 06 – Tempo de resposta computacional das instâncias I1-I10 executadas pelo CPLEX e pela Colônia de Formigas.

Instâncias		CPLEX		ACO		GAP (%)
Nº	ASP	Makespan	Tempo de Execução	Makespan	Tempo de Execução	
I1	20	192	0,750s	197	0,730s	3%
I2	30	245	1,203s	283	0,866s	3%
I3	50	413	4,204s	427	1,539s	3%
I4	80	771	38,60s	782	4,011s	2%
I5	100	1558	76,956s	1627	6,799s	3%
I6	200	3140	1712,496s	3251	40,926s	2%
I7	220	3680	588,584s	3833	39,987s	2%
I8	250	4069	1411,821s	4139	61,998s	1%
I9	280	4165	4641,842s	4187	84,127s	1%
I10	300	4600	8549,712s	4713	107,721s	2%

Analisando os tempos computacionais obtidos, é possível afirmar que o CPLEX retornou resultados escala exponencial, haja vista para instâncias pequenas obteve-se resposta em até 40s. Para instâncias maiores, por exemplo, a I10, temos um tempo de resposta de 8549s, sendo caracterizado assim, um comportamento exponencial. Para a metaheurística ACO, ainda com relação ao tempo computacional, as instâncias foram resolvidas em pouco mais de 4s e para as instâncias maiores, ou seja, de I5 a I10 encontramos um tempo inferior a 110s, caracterizando assim, um comportamento linear.

Para compararmos o valor do *makespan* encontrado pelo experimento, em termos práticos, digamos que o planejamento da empresa disponha apenas de um conjunto de máquinas (três retroescavadeiras e duas caçambas/muncks), para fins de comparação, haja vista ser um sequenciamento de *flow shop* onde não poderíamos utilizar dois ou mais conjuntos. A tabela 07 mostra quantos dias seriam necessários para executar as instâncias em cada método de planejamento com apenas um conjunto de máquinas.

Na tabela 07 temos quatro métodos de avaliação do *makespan* da empresa. O primeiro, caracterizado por sequencial, dispõe as tarefas com quantitativos de dias por equipamentos de maneira sequencial, onde cada máquina inicia sua atividade após finalização da anterior e cada tarefa inicia após a finalização da anterior, sendo este o pior caso. O segundo método, denominado por paralelo é aquele onde as etapas 2, 3, 4 e 5 são iniciadas um dia antes da

finalização do seu antecessor, otimizando assim, quatro dias em cada tarefa, já que temos quatro máquinas inicializando um dia antes.

Tabela 07 – Comparativo de dias de trabalho executados por um conjunto de máquinas.

INSTÂNCIAS	SEQUENCIAL (dias)	PARALELO (dias)	CPLEX (dias)	ACO (dias)	GAP CPLEX / SEQUENCIAL	GAP CPLEX / PARALELO	GAP ACO / SEQUENCIAL	GAP ACO / PARALELO
20 ASP	417,77	337,77	192	197	54%	43%	54%	42%
30 ASP	603,42	484,95	245	283	59%	49%	59%	42%
50 ASP	913,4	713,4	413	427	55%	42%	55%	40%
80 ASP	1323,81	1003,81	771	782	42%	23%	42%	22%
100 ASP	2205,29	1805,29	1558	1627	29%	14%	29%	10%
200 ASP	4410,59	3610,59	3140	3251	29%	13%	29%	10%
220 ASP	4824,88	3944,88	3680	3833	24%	7%	24%	3%
250 ASP	5323,98	4323,99	4069	4139	24%	6%	24%	4%
280 ASP	5734,4	4614,4	4165	4187	27%	10%	27%	9%
300 ASP	6615,88	5415,88	4600	4713	30%	15%	30%	13%

As equações (2) e (3) foram utilizadas no cálculo das melhorias encontradas na utilização dos métodos exato e aproximado, respectivamente. Para calcular a melhoria do CPLEX em relação ao planejamento sequencial utilizamos a equação (2) e a equação (3) para calcular a melhoria do ACO em relação ao planejamento sequencial.

$$Melhoria\frac{CPLEX}{Sequencial}(\%) = 1 - \left(\frac{Sequencial\ (dias)}{CPLEX\ (dias)}\right) \quad (2)$$

$$Melhoria\ ACO/Sequencial\ (\%) = 1 - \left(\frac{Sequencial\ (dias)}{ACO\ (dias)}\right) \quad (3)$$

Ainda de acordo com a tabela 07, temos o terceiro método que demonstra os valores ótimos obtidos pela implementação do *flow shop* no CPLEX e o, quarto e último método, os valores obtidos via implementação da metaheurística de otimização por colônia de formigas. As colunas melhoria cplex/sequencial e melhoria aco/sequencial mostram a relação entre os métodos e qual o ganho encontrado com a sua implantação.

A melhoria encontrada com a aplicação da metaheurística para instâncias pequenas, ou seja, de 20-80 ASP´s, tem uma média de 50% no ganho temporal, sendo este valor bastante significativo para as empresas que contratam a locação dos equipamentos, tendo suas horas/máquina locadas reduzidas pela metade. Já para instâncias entre 100-300 ASP´s temos uma melhoria de 25% de ganho, o que também é bastante representativo, pois reduz ¼ do tempo de horas/máquina locadas.

Com os resultados obtidos, pode-se afirmar que o modelo proposto pode ser utilizado na prática como ferramenta de apoio ao planejamento de ASP´s por empresas prestadoras de serviços de petróleo, haja vista ter buscado maior eficiência operacional para os equipamentos locados, minimizando assim, custos de projeto.

5 CONSIDERAÇÕES FINAIS

Os problemas de escalonamento de situações reais são resolvíveis na prática por pessoas qualificadas e experientes, mas que dependendo da sua competência, podem influenciar diretamente no planejamento final. Ressalta-se que qualquer situação cotidiana é passível de otimização, uma vez que possui algumas variáveis de decisão.

Para o desenvolvimento deste trabalho, foi observado numa empresa prestadora de serviço de petróleo, que alguns equipamentos locados, como retroescavadeiras, se encontravam inutilizados em campo e que serviços de outras locações do campo estavam sendo penalizados por não terem recurso disponível para execução dos serviços da mesma.

O objetivo principal desse trabalho foi minimizar o *makespan* de um projeto real que dispunha de tarefas de lançamento de linha de produção de petróleo, denominada de ASP, as quais mencionavam em seu escopo o tamanho da linha a ser lançada e o período a ser realizada a tarefa. De posse desses valores, foram calculados os tempos necessários para execução de cada tarefa por cada máquina, tendo em vista que, por tratar-se de equipamentos de locação, a empresa deve ter o real conhecimento de como estes são utilizados, para assim, dirimir custos desnecessários quando utilizados.

Para obtenção de resultados foi proposto à utilização através dos métodos: o CPLEX, um método exato e a Otimização por Colônia de Formigas, um aproximado. Para isso, utilizaram-se 10 instâncias contendo 20, 30, 50, 80, 100, 200, 220, 250, 280 e 300 ASP´s cada uma, onde ambas as metodologias retornaram valores para o *makespan*.

Para validar a utilização dos métodos de otimização exato e aproximado, comparou-se com os resultados adquiridos por um planejamento efetuado por um planejador de projetos com auxílio do Microsoft Excel, onde foi encontrado redução nos dias de locação por máquina na proporção de 50% para instâncias pequenas, ou seja, de 20-80 ASP´s e de 25% para as demais instâncias, sendo de 100-300 ASP´s.

Com relação à comparação entre os tempos computacionais obtidos, o método CPLEX embora retorne um *makespan* ótimo, leva um tempo de resposta

bastante oneroso quando comparado ao encontrado pela metaheurística de Otimização por Colônia de Formigas, o que inviabiliza a sua aplicação em problemas reais. Sendo assim, a metaheurística de Otimização por Colônia de Formigas se apresenta como uma metodologia viável para solução de problemas práticos como o escalonamento de tarefas de um lançamento de linhas de dutos para escoamento da produção de petróleo.

5.1 PUBLICAÇÕES

Foram publicados dois artigos em congressos:

- XLVI Simpósio Brasileiro de Pesquisa Operacional (SBPO), evento ocorrido de 16 a 19 de setembro de 2014, em Salvador - BA;
- VI Escola Potiguar de Computação e suas Aplicações (Época 2014), evento ocorrido de 26 a 28 de novembro em Santa Cruz – RN.

Foi submetido um artigo em periódico:

- RITA – Revista de Informática Teórica e Aplicada, do Instituto de Informática da Universidade Federal do Rio Grande do Sul – Qualis B4 – Ciência da Computação.

5.2 TRABALHOS FUTUROS

Apesar da proposta de utilização da metaheurística de Otimização por Colônia de Formigas aplicada ao problema prático de escalonamento de tarefas de lançamento de linhas de duto de produção de petróleo ter se mostrado eficiente em termos de tempos computacionais e resultados esperados, propõe-se como continuidade do estudo e/ou trabalhos futuros, aplicar outros métodos de otimização como o PSO, Busca Tabu, entre outros, como também metaheurísticas híbridas.

REFERÊNCIAS

AHUJA, H. N. **Construction performance control by networks**. New York, 1976.

AKKAN, C. A lagrangian heuristic for the discrete time-cost tradeoff problem for activity-on-arc project networks. **Technical report**, Koç University, Istambul, 1998.

AKKAN, C. Iterated local search algorithms for the discrete time-cost tradeoff problem. **Technical report**, Koç University, Istambul, 1999.

ARENALES, Marcos, ARMENTANO, Vinicius, MORABITO, Reinaldo, YANASSE, Horácio. **Pesquisa operacional**. Rio de Janeiro, 2007.

BANDELLONI, M.; TUCCI, M.; RINALDI, R. Optimal resource leveling using nonserial dynamic programming. **European Journal of Operational Research**, 78:162–177, 1994.

BAROUM, S. M.; PATTERSON, J. H. An Exact Solution Procedure for Maximizing the Net Present Value, chapter Handbook on Recent Advances in Project Scheduling, pages 107–134. **Kluwer Academic Publishers**, 1999.

BARTUSCH, M.; MÖHRING, R. H.; RADERMACHER, F. J. Scheduling project networks with resource constraints and time windows. **Operations Research**, 16:201–240, 1988.

BLA`ZEWICZ, J., ECKER, K. H., PESCH, E., SCHMIDT, G., WEGLARZ, J. Scheduling computer and manufacturing processes. In **Springer Verlag**, 1996.

BLAZEWICZ, J. et al. **Handbook on planejament**: from theory to aplications. Heidelberg: Springer Verlag, 2007.

BLUM, C. Ant colony optimization: Introduction and recent trends. **Physics of Life Reviews**, v. 2, p. 353–373, 2005a.

BRINKMANN, K.; NEUMANN, K. Heuristic procedure for resource constrained project scheduling with minimal and maximal time lags: The resource-leveling and minimum project duration problems. **Journal of Decision Systems**, 5:129–155, 1996.

BRUCKER, P. **Planejamento algorithms**. 5. Ed. Springer: Verlag Berlin Heidelberg: 2007.

BRUCKER, P.; DREXL, A.; MÖHRING, R.; NEUMANN, K.; PESCH, E. Resource-constrained project scheduling: notation, classification, models, and methods. Technical Report 112, **European Journal of Operational Research**, 1999.

BURGESS, A. R.; KILLEBREW, J. B. Variation in activity level on a cyclical arrow diagram. **Journal of Industrial Engineering**, 13:76–83, 1962.

CARABETTI, Eduardo Goecking. **Metaheurística colônia de formigas aplicada ao problema de roteamento de veículos em coleta e entrega e janela de tempo**. Dissertação (Mestrado). 2010. 70p. Cetro Federal de Educação Tecnológica de Minas Gerais. Belo Horizonte, 2010.

CHANG, C. K.; CHRISTENSEN, M. Genetic algorithms for project management. **Annals of Software Engineering**, 11:107–139, 2001.

CHANG, C. K.; JIANG, H. Time-line based model for software project scheduling with genetic algorithms. **Information and Software Technology**, 2008.

CHAVES, A. A. **Heurísticas híbridas com busca através de agrupamentos para o problema do caixeiro viajante com coleta de prêmios**. (Dissertação de mestrado) Pós-Graduação em Computação Aplicada, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, 2005.

CHRISTOFIDES, N.; ALVAREZ-VALDÉS, R.; TAMARIT, J. M. Project scheduling with resource constraints: a branch and bound approach. **European Journal of Operational Research**, 29:262–273, 1987.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. Distribuited optimization by and colonies. **European Conference of Artificial Life**. [s.l.: s.n.], 1991, p.134-142.

CRAVO, Gildásio Lecchi. **Escalonamento de projetos com restrições de recursos e múltiplos modos de processamento**: soluções heurísticas e uma aplicação à programação de manutenção industrial. (Tese), Universidade Federal do Espírito Santo, 2009.

DE REYCK, B. **Scheduling projects with generalized precedence relations: Exact and heuristics procedures**. PhD thesis, Katholieke Universiteit Leuven, 1988.

DE REYCK, B.; HERROELEN, W. An optimal procedure for the resource constrained project scheduling problem with discounted cash flows and generalized precedence relations. **Computers & Operations Research**, 25:1–17, 1998.

DE REYCK, B.; HERROELEN, W. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. **European Journal of Operational Research**, 119:538–556, 1999.

DEMEULEMEESTER, E. L.; HERROELEN, W. S. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. **Management Science**, 38:1803–1818, 1992.

DORIGO, Marco; DI CARO, Gianni e GAMBARDELLA, Luca M. Ant algorithms for discrete optimization. **Artificial Life**, v. 5, n. 2, 1999, p. 137–172.

EASA, S. M. Resource leveling in construction by optimization. **Journal of Construction Engineering and Management**, 115:302–316, 1989.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, 6:109–113, 1995.

FONSECA, Marcos Abraão de Souza. **Uma abordagem ACO para a programação reativa da produção**. Dissertação (Mestrado), 2010. 93p. Universidade Federal de São Carlos, São Carlos, 2010.

GAREY, M. R.; JOHNSON, D. S.; SETHI, R. The Complexity of Flowshop and Jobshop Scheduling, **Mathematics of Operations Research**, vol. 1, n. 2, 1976, pp.117-129.

GIGANTE, Rodrigo Luiz. **Heurística construtiva para a programação de operações *flow shop* permutacional**. Dissertação (Mestrado). 90 p. 2010. Escola de Engenharia de São Carlos da Universidade de São Paulo. São Paulo, 2010.

GLOVER, F. **Tabu search**: part i. ORSA Journal on Computing, 1(3):190–206, 1989.

GRAVES, S. G. A Review of production scheduling. **Operations Research**, Linthicum, v.29, n.4, 1981, p.646-675, July.

HARRIS, R. B. Packing method for resource leveling (pack). **Journal of Construction Engineering and Management**, 116:39–43, 1990.

HEILMANN, R. A branch-and-bound procedure for the multi-mode resource constrained project scheduling problem with minimum and maximum time lags. **European Journal of Operational Research**, 144:348–365, 2003.

HERROELEN, W. S.; VAN DOMMELEN, P., DEMEULEMEESTER, E. L. Project network models with discounted cash flows: a guided tour through recent developments. **European Journal of Operational Research**, 100:97–121, 1997.

HO, J. C. Flowshop sequencing with mean flowtime objective, **European Journal of Operation Research**, n. 81, 1995, p. 571-578.

ICMELI, O.; ERENGUC, S. S. A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. **Management Science**, 42:1395–1408, 1996.

ICMELI, O.; ERENGUC, S. S. A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. **Computers & Operations Research**, 8:841–853, 1994.

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

KENNEDY, J.; EBERHART, R. Particle Swarm Optimization. **Proceedings of IEEE International Conference on Neural Networks**, Perth, Australia, pp. 1942-1948, 1995.

KIM, K. W.; GEN, M. Hybrid genetic algorithm with fuzzy logic for resource constrained project scheduling. **Applied Soft Computing**, 2003.

LEAL, A. J. S. **Algoritmos de investigação operacional para um problema de sequenciamento de projetos**. (Mestrado), Universidade de Minho, Portugal, 2007.

LEVY, F. K.,; THOMPSON, G. L.; WIEST, J. D. Multiship, multishop, workload-smoothing program. **Naval Research Logistics Quarterly**, pages 37–44, 1962.

LONG, L. D.; OHSATO, A. Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. **International Journal of Project Management**, 2007.

LORENZONI, L. L. **Problema de escalonamento com restrição de recursos e múltiplos modos de processamento**: novos métodos de resolução e uma aplicação no contexto portuário. (Doutorado), Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, 2003.

MENDES, J. J. M. **Sistema de apoio a decisão para o planeamento de sistemas de produção tipo projeto**. (PHD), Universidade do Porto, 2003.

MOODIE, C. L.; MANDEVILLE, D. E. Project resource balancing by assembly line balancing techniques. **The Journal of Industrial Engineering**, 17:377–383, 1966.

NEGREIRO, Marcos; BARBOSA, Willame Tiberio. O problema de alocação de recursos e seleção de múltiplos projetos de ti. **Revista de Gestão e Projetos**, 4(2): 27–49, mai./ago. 2013.

NEUMANN, K.; ZIMMERMANN, J. Exact and heuristic procedures for net present value and resource leveling in project scheduling. **Technical report**, Universität Karlsruhe, 1998.

NEUMANN, K.; ZIMMERMANN, J. **Methods for resource-constrained project scheduling with regular and non-regular objective functions and schedule-dependent time windows**, chapter Handbook on Recent Advances in Project Scheduling, pages 261–288. Kluwer Academic Publishers, 1999.

NEUMANN, K.; ZIMMERMANN, J. Resource leveling for projects with schedule dependent time windows. **Technical report**, Universität Karlsruhe, 1997.

NONOBE, K.; IBARAKI, T. **Formulation and tabu search algorithm for the resource constrained project scheduling problem**, chapter Essays and Surveys in Metaheuristics, page 557588. 2002.

NOVAS, Máquinas. **Especificações**. Disponível em: <http://www.maquinasnovas.com.br/especificacoes?ca=8-retro-escavadeira-case-580m-4x4&cv=B28C0CEE>. Acesso em: 02 nov. 2014.

PALMER, D. S. Sequencing jobs through a multi-stage process in the minimum total time - A quick method of obtaining a near optimum. **Operational Research Quarterly**, London, v.16, n.1, 1965, p.101-107, Mar.

PATTERSON, J. H.; SLOWISKI, R., TALBOT, F. B., and WGLARZ, J. **An algorithm for a general class of precedence and resource constrained scheduling problems**, chapter Advances in Project Scheduling, pages 3–28. Elsevier, Amsterdam, 1989.

PÉREZ. Miguel Angel Fernández. **Um método heurístico para o problema de escalonamento multiobjetivo em vários ambientes de máquinas**. 109 f. 2012. Dissertação (Mestrado). Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, Rio de Janeiro, 2012.

PINEDO, M. L. **Planning and planejament in manufacturing and services**. Springer Series Operation Research: New York, USA, 2005.

PINEDO, M. **Scheduling**: theory, algorithms and systems. Prentice Hall, New Jersey. 2008.

PINEDO, M. **Scheduling**: theory, algorithms, and systems. Englewood Cliffs, NJ, 1995.

RINNOOY KAN, A. H. G. **Machine scheduling problems**: classification, complexity, and computations. The Hahue, Nijhoff. 1976.

RUSSELL, R. A. A comparison of heuristics for scheduling projects with cash flow and resource restrictions. **Management Science**, 32:1291–1300, 1986.

RUTQUIST, P. E., EDVALL, M. M. **Tomlab optimization**. PROPT - Matlab Optimal Control Software, 2010.

SANTOS, Haroldo Gambini, TOFFOLO, Túlio Ângelo Machado, CARVALHO, Marco Antonio Moreira de; SOARES, Janniele Aparecida. Minicurso: modelos e métodos de resolução para problemas de escalonamento de projetos. In **SBPO**. Natal, 2013.

SANTOS, L. S.; SECCHI, A. R.; BISCAIA JR, E. C.; "TOOLBOX MATLAB PARA SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO DINÂMICA", p. 11803-11810. In: **Anais…** XX Congresso Brasileiro de Engenharia Química - COBEQ 2014 [= Blucher Chemical Engineering Proceedings, v.1, n.2]. São Paulo: Blucher, 2015.

SAVIN, D.; ALKASS, S.; FAZIO, P. Construct resource leveling using neural networks. **Canadian Journal of Civil Engineering**, 23: 917–925, 1996.

SILVA, André Renato Villela da. Um método híbrido para um problema de escalonamento de projetos. In **Congresso Latino-Iberoamericano de Investigácion Operativa. Simpósio Brasileiro de Pesquisa Operacional**, Setembro 2012.

SOARES, C. Evolutionary computation for the job shop scheduling problem. 1994. In.: BECK, Felipe Luís. **Escalonamento de tarefas job shop realistas utilizando algoritmos genéticos em Matlab**. Dissertação (Mestrado em Engenharia Elétrica). 2000. 104p. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia Elétrica. Florianópolis, 2000.

SOUZA, Eduardo Cordeiro. **Programação de tarefas em um *flow shop***. Tese (Doutorado). 124 p. 2009. Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Naval e Oceânica. São Paulo, 2009.

SPRECHER, A. Resource-constrained project scheduling - exact methods for the multi-mode case. **Lecture Notes in Economics and Mathematical Systems**, 409, 1994.

SPRECHER, A., HARTMANN, S., and DREXL, A. An exact algorithm for project scheduling with multiple modes. **OR Spektrum**, 19:195–203, 1997.

SPRECHER, A.; DREXL, A. Minimal delaying alternatives and semi-active timetabling in resource-constrained project scheduling. **Manuskripte aus den Instituten für Betriebswirtschaftslehre**, 426, 1996.

TAKAMOTO, M.; YAMADA, N.; KOBAYASHI, Y.; NONAKA, H. Zeroone quadratic programming algorithm for resource leveling of manufacturing process schedule. **Systems and Computers in Japan**, 26:68–76, 1995.

TALBOT, F. B. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. **Management Science**, 28:1197–1210, 1982.

TAVARES NETO, Roberto Fernandes. **Proposta de solução de problemas de scheduling considerando possibilidade de terceirização usando a técnica de otimização por colônia de formigas**. Dissertação (Mestrado em Engenharia de Produção). 2010, 129p. Universidade Federal de São Carlos. Departamento de Engenharia de Produção. São Carlos, 2010.

TAVARES NETO, Roberto Fernandes; GODINHO FILHO, Moacir. Otimização por colônia de formigas para o problema de sequenciamento de tarefas em uma única máquina com terceirização permitida. **Gest. Prod.**, São Carlos, v. 20, n. 1, p. 76-86, 2013.

VALLS, V.; BALLESTÍN, F.; QUINTANILLA, S. A hybrid genetic algorithm for the resource-constrained project scheduling problem. **European Journal of Operational Research**, 185:495–508, 2008.

WOODWORTH, B. M.; WILLIE, Ch. J. A heuristic algorithm for resource leveling in multi-project, multi-resource scheduling. **Decision Sciences**, 6:525–540, 1975.

YANG, B.; GEUNES, J.; OBRIAN, W. J. Resource-constrained project scheduling: past work and new directions. **Research Report**, 6, 2001.

YANG, K. K.; TALBOT, F. B.; PATTERSON, J. H. Scheduling a project to maximize its net present value: an integer programming approach. **European Journal of Operational Research**, 64:188–198, 1993.

YANG, K. K.; TAY, L. C.; SUM, C. C. A comparison of stochastic priority rules for maximizing project net present value. **European Journal of Operational Research**, 85:327–339, 1995.

YOUNIS, M. A.; SAAD, B. Optimal resource leveling of multi-resource projects. **Computers and Industrial Engineering**, 31: 1–4, 1996.

ZHANG, H.; LI, H. Particle swarm optimization for resource-constrained project scheduling. **International Journal of Project Management**, 24, 2006.

ZHANG, H.; LI, X. Particle swarm optimization-based schemes for resource constrained project scheduling. **Automation in Construction**, 14, 2005.

ZIMMERMANN, J. Heuristics for resource leveling problem in project scheduling with minimum and maximum time lags. **Technical report, Universität Karlsruhe**, 1997.

APENDICE A – RESULTADO CPLEX
20 ASP´S

===== * * * ========== * * *
TOMLAB - Tomlab user Demo license 999100. Valid to 2015-08-18
==========

A sequencia ótima é: ASP 15 → ASP 20 → ASP 10 → ASP 19 → ASP 16 → ASP 1 → ASP 9 → ASP 17 → ASP 18 → ASP 2 → ASP 12 → ASP 4 → ASP 14 → ASP 8 → ASP 11 → ASP 6 → ASP 13 → ASP 7 → ASP 3 → ASP 5

Escalonamento das ASP´s

Sequencia da ASP 15
- Máquina 1: 0-4
- Máquina 2: 9-11
- Máquina 3: 16-20
- Máquina 4: 25-27
- Máquina 5: 37-41

Sequencia da ASP 20
- Máquina 1: 4-7
- Máquina 2: 13-14
- Máquina 3: 22-25
- Máquina 4: 29-30
- Máquina 5: 41-44

Sequencia da ASP 10
- Máquina 1: 7-12
- Máquina 2: 16-18
- Máquina 3: 27-32
- Máquina 4: 32-34
- Máquina 5: 44-49

Sequencia da ASP 19
- Máquina 1: 16-21
- Máquina 2: 24-26
- Máquina 3: 38-43
- Máquina 4: 40-42
- Máquina 5: 54-59

Sequencia da ASP 16
- Máquina 1: 21-23
- Máquina 2: 26-27
- Máquina 3: 43-45
- Máquina 4: 42-43
- Máquina 5: 59-61

Sequencia da ASP 1
- Máquina 1: 23-31
- Máquina 2: 30-33
- Máquina 3: 48-56
- Máquina 4: 46-49
- Máquina 5: 68-76

Sequencia da ASP 9
- Máquina 1: 32-42
- Máquina 2: 43-47
- Máquina 3: 66-76
- Máquina 4: 59-63
- Máquina 5: 77-87

Sequencia da ASP 17
- Máquina 1: 49-50
- Máquina 2: 55-56
- Máquina 3: 84-85
- Máquina 4: 71-72
- Máquina 5: 92-93

Sequencia da ASP 18
- Máquina 1: 53-61
- Máquina 2: 56-59
- Máquina 3: 85-93
- Máquina 4: 72-75
- Máquina 5: 93-101

Sequencia da ASP 2
- Máquina 1: 65-73
- Máquina 2: 59-62
- Máquina 3: 93-101
- Máquina 4: 75-78
- Máquina 5: 101-109

Sequencia da ASP 12
- Máquina 1: 76-84
- Máquina 2: 64-67
- Máquina 3: 103-111
- Máquina 4: 80-83
- Máquina 5: 109-117

Sequencia da ASP 4
- Máquina 1: 89-96
- Máquina 2: 70-73
- Máquina 3: 114-121
- Máquina 4: 86-89
- Máquina 5: 121-128

Sequencia da ASP 14
- Máquina 1: 99-100
- Máquina 2: 73-74
- Máquina 3: 121-122
- Máquina 4: 89-90
- Máquina 5: 128-129

Sequencia da ASP 8
- Máquina 1: 100-106
- Máquina 2: 74-77
- Máquina 3: 122-128
- Máquina 4: 90-93
- Máquina 5: 136-142

Sequencia da ASP 11
- Máquina 1: 106-110
- Máquina 2: 87-89
- Máquina 3: 138-142
- Máquina 4: 103-105
- Máquina 5: 143-147

Sequencia da ASP 6
- Máquina 1: 110-115
- Máquina 2: 89-91
- Máquina 3: 142-147
- Máquina 4: 105-107
- Máquina 5: 147-152

Sequencia da ASP 13
- Máquina 1: 115-120
- Máquina 2: 96-98
- Máquina 3: 152-157
- Máquina 4: 112-114
- Máquina 5: 162-167

Sequencia da ASP 7
- Máquina 1: 121-130
- Máquina 2: 102-106
- Máquina 3: 161-170
- Máquina 4: 118-122
- Máquina 5: 167-176

Sequencia da ASP 3
- Máquina 1: 131-134
- Máquina 2: 106-107
- Máquina 3: 170-173
- Máquina 4: 122-123
- Máquina 5: 179-182

Sequencia da ASP 5
- Máquina 1: 136-142
- Máquina 2: 111-114
- Máquina 3: 177-183
- Máquina 4: 127-130
- Máquina 5: 189-192

Escalonamento das Máquinas

Sequencia para a máquina 1
- ASP 15: 0-4
- ASP 20: 4-7
- ASP 10: 7-12
- ASP 19: 16-21
- ASP 16: 21-23
- ASP 1: 23-31
- ASP 9: 32-42
- ASP 17: 49-50
- ASP 18: 53-61
- ASP 2: 65-73
- ASP 12: 76-84
- ASP 4: 89-96
- ASP 14: 99-100
- ASP 8: 100-106
- ASP 11: 106-110
- ASP 6: 110-115
- ASP 13: 115-120
- ASP 7: 121-130
- ASP 3: 131-134
- ASP 5: 136-142

Sequencia para a máquina 2
- ASP 15: 9-11
- ASP 20: 13-14
- ASP 10: 16-18
- ASP 19: 24-26
- ASP 16: 26-27
- ASP 1: 30-33
- ASP 9: 43-47
- ASP 17: 55-56
- ASP 18: 56-59
- ASP 2: 59-62
- ASP 12: 64-67
- ASP 4: 70-73
- ASP 14: 73-74
- ASP 8: 74-77
- ASP 11: 87-89
- ASP 6: 89-91
- ASP 13: 96-98
- ASP 7: 102-106
- ASP 3: 106-107
- ASP 5: 111-114

Sequencia para a máquina 3
- ASP 15: 16-20
- ASP 20: 22-25
- ASP 10: 27-32
- ASP 19: 38-43
- ASP 16: 43-45
- ASP 1: 48-56
- ASP 9: 66-76
- ASP 17: 84-85
- ASP 18: 85-93
- ASP 2: 93-101
- ASP 12: 103-111
- ASP 4: 114-121
- ASP 14: 121-122
- ASP 8: 122-128
- ASP 11: 138-142
- ASP 6: 142-147
- ASP 13: 152-157
- ASP 7: 161-170
- ASP 3: 170-173
- ASP 5: 177-183

Sequencia para a máquina 4
- ASP 15: 25-27
- ASP 20: 29-30
- ASP 10: 32-34
- ASP 19: 40-42
- ASP 16: 42-43
- ASP 1: 46-49
- ASP 9: 59-63
- ASP 17: 71-72
- ASP 18: 72-75
- ASP 2: 75-78
- ASP 12: 80-83
- ASP 4: 86-89
- ASP 14: 89-90
- ASP 8: 90-93
- ASP 11: 103-105
- ASP 6: 105-107
- ASP 13: 112-114
- ASP 7: 118-122
- ASP 3: 122-123
- ASP 5: 127-130

Sequencia para a máquina 5
- ASP 15: 37-41
- ASP 20: 41-44
- ASP 10: 44-49
- ASP 19: 54-59
- ASP 16: 59-61
- ASP 1: 68-76
- ASP 9: 77-87
- ASP 17: 92-93
- ASP 18: 93-101
- ASP 2: 101-109
- ASP 12: 109-117
- ASP 4: 121-128
- ASP 14: 128-129
- ASP 8: 136-142
- ASP 11: 143-147
- ASP 6: 147-152
- ASP 13: 162-167
- ASP 7: 167-176
- ASP 3: 179-182
- ASP 5: 189-192

RESULTADO CPLEX – 30 ASP´S

===== * * * ========== * * *=====
TOMLAB - Tomlab user Demo license 999100. Valid to 2015-08-18
==========

A sequência ótima é: ASP 3 → ASP 7 → ASP 10 → ASP 23 → ASP 24 → ASP 15 → ASP 6 → ASP 30 → ASP 19 → ASP 12 → ASP 4 → ASP 11 → ASP 17 → ASP 25 → ASP 20 → ASP 1 → ASP 29 → ASP 28 → ASP 8 → ASP 16 → ASP 22 → ASP → ASP 21 → ASP 13 → ASP 27 → ASP 9 → ASP 14 → ASP 2 → ASP 26 → ASP 18 → ASP 5

RESULTADO CPLEX – 50 ASP´S

===== * * * ========== * * *
TOMLAB - Tomlab user Demo license 999100. Valid to 2015-08-18
==========

A sequencia ótima é: ASP 43 → ASP 32 → ASP 17 → ASP 35 → ASP 48 → ASP 47 → ASP 13 → ASP 31 → ASP 41 → ASP 50 → ASP 49 → ASP 22 → ASP 10 → ASP 4 → ASP 25 → ASP 36 → ASP 45 → ASP 24 → ASP 16 → ASP 28 → ASP 37 → ASP 21 → ASP 46 → ASP 20 → ASP 40 → ASP 42 → ASP 26 → ASP 8 → ASP 34 → ASP 38 → ASP 9 → ASP 33 → ASP 7 → ASP 27 → ASP 29 → ASP 15 → ASP 23 → ASP 2 → ASP 30 → ASP 44 → ASP 18 → ASP 6 → ASP 5 → ASP 19 → ASP 39 → ASP 12 → ASP 3 → ASP 11 → ASP 14 → ASP 1

RESULTADO CPLEX – 80 ASP´S

===== * * * ========== * * *
TOMLAB - Tomlab user Demo license 999100. Valid to 2015-08-18
==========

A sequencia ótima é: ASP 4 → ASP 18 → ASP 77 → ASP 73 → ASP 37 → ASP 23 → ASP 7 → ASP 22 → ASP 74 → ASP 25 → ASP 8 → ASP 58 → ASP 27 → ASP 36 → ASP 45 → ASP 38 → ASP 14 → ASP 40 → ASP 57 → ASP 28 → ASP 39 → ASP 48 → ASP 79 → ASP 46 → ASP 1 → ASP 50 → ASP 51 → ASP 31 → ASP 54 → ASP 19 → ASP 35 → ASP 56 → ASP 26 → ASP 6 → ASP 20 → ASP 64 → ASP 63 → ASP 9 → ASP 43 → ASP 21 → ASP 71 → ASP 33 → ASP 65 → ASP 49 → ASP 59 → ASP 70 → ASP 69 → ASP 62 → ASP 42 → ASP 75 → ASP 78 → ASP 12 → ASP 80 → ASP 10 → ASP 11 → ASP 67 → ASP 17 → ASP 16 → ASP 68 → ASP 2 → ASP 29 → ASP 24 → ASP 61 → ASP 13 → ASP 44 → ASP 41 → ASP 32 → ASP 47 → ASP 34 → ASP 3 → ASP 60 → ASP 53 → ASP 55 → ASP 15 → ASP 30 → ASP 66 → ASP 72 → ASP 76 → ASP 52 → ASP 5

RESULTADO CPLEX – 100 ASP´S

===== * * * ========== * * *
TOMLAB - Tomlab user Demo license 999100. Valid to 2015-08-18
==========

A sequencia ótima é: ASP 14 → ASP 13 → ASP 79 → ASP 50 → ASP 41 → ASP 74 → ASP 66 → ASP 69 → ASP 45 → ASP 35 → ASP 47 → ASP 11 → ASP 77 → ASP 39 → ASP 24 → ASP 54 → ASP 75 → ASP 8 → ASP 29 → ASP 94 → ASP 82 → ASP 99 → ASP 61 → ASP 18 → ASP 16 → ASP 28 → ASP 84 → ASP 2 → ASP 56 → ASP 21 → ASP 32 → ASP 92 → ASP 25 → ASP 96 → ASP 58 → ASP 42 → ASP 100 → ASP 15 → ASP 59 → ASP 27 → ASP 51 → ASP 22 → ASP 52 → ASP 70 → ASP 7 → ASP 65 → ASP 43 → ASP 62 → ASP 17 → ASP 6 → ASP 31 → ASP 72 → ASP 10 → ASP 81 → ASP 95 → ASP 90 → ASP 30 → ASP 57 → ASP 89 → ASP 71 → ASP 49 → ASP 76 → ASP 91 → ASP 55 → ASP 80 → ASP 46 → ASP 78 → ASP 34 → ASP 53 → ASP 67 → ASP 26 → ASP 40 → ASP 19 → ASP 88 → ASP 86 → ASP 97 → ASP 98 → ASP 36 → ASP 3 → ASP 60 → ASP 12 → ASP 23 → ASP 37 → ASP 33 → ASP 73 → ASP 87 → ASP 63 → ASP 83 → ASP 1 → ASP 9 → ASP 48 → ASP 85 → ASP 64 → ASP 20 → ASP 44 → ASP 4 → ASP 68 → ASP 93 → ASP 38 → ASP 5