Ferramentas computacionais híbridas para a otimização da produção de petróleo em águas profundas

Juliana Martins do Nascimento

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Ferramentas computacionais híbridas para a otimização da produção de petróleo em águas profundas

Juliana Martins do Nascimento[1]

Novembro de 2002

Banca Examinadora:

- Prof. Dr. Arnaldo Vieira Moura
  Instituto de Computação, Unicamp (Orientador)
- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação, Unicamp
- Prof. Dr. Celso K. Morooka
  Faculdade de Engenharia Mecânica, Unicamp
- Prof. Dr. Ricardo Dahab
  Instituto de Computação, Unicamp (Suplente)

[1] **Auxílio financeiro da FAPESP processo 00/14120-8**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Nascimento, Juliana Martins do

N174f Ferramentas computacionais híbridas para a otimização da produção de petróleo em águas profundas / Juliana Martins do Nascimento. -- Campinas, [S.P. :s.n.], 2002.

Orientadores : Arnaldo Vieira Moura; Cid Carvalho de Souza.

Dissertação (Mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. 2. Programação heurística. 3. Programação inteira. I. Moura, Arnaldo Vieira. II. Souza, Cid Carvalho de. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 06 de dezembro de 2002, pela Banca Examinadora composta pelos Professores Doutores:

Prof. Dr. Celso Kazuyuki Morooka
FEM - UNICAMP

Prof. Dr. Flávio Keidi Miyazawa
IC - UNICAMP

Prof. Dr. Arnaldo Vieira Moura
IC - UNICAMP

Ferramentas computacionais híbridas para a otimização da produção de petróleo em águas profundas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Juliana Martins do Nascimento e aprovada pela Banca Examinadora.

Campinas, 6 de dezembro de 2002.

Prof. Dr. Arnaldo Vieira Moura
Instituto de Computação, Unicamp
(Orientador)

Prof. Dr. Cid Carvalho de Souza
Instituto de Computação, Unicamp
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Resumo

Problemas de otimização combinatória são classificados na grande maioria das vezes como NP-difíceis. Para estes problemas, não são conhecidos algoritmos polinomiais capazes de resolvê-los. Logo, é necessário o desenvolvimento de estratégias eficientes para tratá-los. O desenvolvimento de técnicas híbridas para a resolução destes problemas tem por objetivo valorizar os pontos fortes dos métodos que estão sendo empregados, para, desta forma, compensar os pontos mais fracos, criando um procedimento de qualidade superior. Este trabalho propõe um método híbrido que integra técnicas de Programação por Restrições com metaheurísticas de Busca Tabu para atacar o problema de escalonamento de atividades na produção de um campo petrolífero. Como não há resultados anteriores para serem comparados com os resultados obtidos para as instâncias consideradas neste trabalho, modelos de programação matemática foram utilizados para a obtenção de limitantes duais para a solução do problema. Além disso, para determinar quão robusta é a técnica proposta, uma análise de sensibilidade foi realizada sobre as instâncias consideradas.

Abstract

Combinatorial optimization problems are generally NP-hard. As it is not known polinomial time algorithms to solve them, it is necessary to develop efficient strategies to treat them. The aim in developing hybrid techniques to solve combinatorial optimization problems is to strength the good features of the methods that are being combined to compensate for their weakness. In this paper, we propose a hybrid method that combines Constraint Programming techniques and Tabu Search metaheuristics to schedule the activities involved in the production process of an oil field. As there are no previous results to estabilish a comparision with the results obtained with the instances considered in this work, bounds were determined using mathematical programming models. Finally, to estabilish the robusteness of proposed method, a sensibility analysis was performed over the considered instances.

À minha família.

Agradecimentos

A Deus por tudo que sou e por tudo que conquistei;

Aos meus pais, Adão e Mari Léa, e às minhas irmãs, Aline e Ana Cândida, pelo carinho, pelo apoio que sempre me dão e por estarem sempre presentes em todos os momentos da minha vida;

Aos meus orientadores, Arnaldo Vieira Moura e Cid Carvalho de Souza, pela dedicação, paciência e incentivo ao meu trabalho;

Ao meu orientador da graduação, Lúcio Tunes dos Santos, por ter me iniciado no mundo da pesquisa e, principalmente, pelo apoio e amizade dedicados a mim nestes 6 anos de convívio;

Aos meus companheiros de labinho, Glauber, Eduardo (Fofinho), Chenca e Silvana por tornarem as horas de trabalho mais divertidas;

Aos meus amigos do LSC, Bartho, os gordos Rodolfo, Sandro e Borin, e o casal coisinha Guilherme e Desiree por estarem presentes nos melhores e também nos piores momentos;

Às amigas Silvania e Amanda pelo carinho, conversas, jantinhas, idas ao El Rancho, enfim, pela amizade de vocês;

A todos os amigos que eu conquistei nestes dois anos de IC, pelos almoços divertidos, horas do café, emails engraçados, conversas no corredor, festas e, principalmente, pela amizade de vocês;

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), pelo apoio financeiro que viabilizou este trabalho.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Problemas de otimização combinatória [36] são classificados na grande maioria das vezes como NP-difíceis [21]. Para estes problemas, não são conhecidos algoritmos polinomiais capazes de resolvê-los. Logo, é necessário o desenvolvimento de estratégias eficientes para tratá-los. O desenvolvimento de técnicas híbridas para a resolução destes problemas tem por objetivo valorizar os pontos fortes dos métodos que estão sendo empregados, para, desta forma, compensar os pontos mais fracos, criando um procedimento de qualidade superior.

A integração de técnicas de Programação por Restrições com metaheurísticas é objeto de pesquisa atual [28], e que pode trazer resultados promissores na área de otimização combinatória [65, 66, 52, 11, 51, 43, 60, 57]. O estudo de algoritmos de Busca Local que exploram de maneira eficiente vizinhanças *muito grandes* em relação aos dados de entrada (vizinhança de larga escala) também está em foco e tem obtido sucesso no ataque a vários problemas [18, 1, 2]. Estes algoritmos podem percorrer a vizinhança demarcada em tempo polinomial ou heuristicamente. Utiliza-se heurísticas quando a exploração da vizinhança é comprovadamente NP-difícil ou simplesmente não é conhecido um algoritmo polinomial para realizar esta busca. A integração citada anteriormente pode possibilitar a exploração de vizinhanças de larga escala em tempos viáveis. Com isso as chances de obtenção de melhoras realmente significativas nas soluções é muito grande, visto que a eficiência de metaheurísticas como a Busca Local melhora à medida em que aumenta o tamanho da vizinhança de busca.

Além disso, o uso da Programação por Restrições para percorrer as vizinhanças da Busca Local é especialmente indicado para tratar problemas reais. Isto acontece porque estes problemas, normalmente, mudam algumas de suas características freqüentemente e a Programação por Restrições torna a modelagem de problemas bastante flexível, permitindo com que novas restrições possam ser incorporadas sem o dispêndio de muito esforço. O mesmo não ocorreria se a Busca Local fosse resolvida por métodos tradicio-

nais através de algoritmos especialmente desenvolvidos para este fim. Espera-se também que a abordagem híbrida seja mais robusta quando comparada com as técnicas empregadas individualmente. Assim, variações nos dados do problema tratado teriam um menor impacto no desempenho da busca por boas soluções.

Para validar as expectativas acima, estas técnicas serão aplicadas no problema de escalonamento de atividades na produção de um campo petrolífero, mais especificamente no planejamento das atividades realizadas pela Petrobras para construir poços na Bacia Petrolífera de Campos. Este problema foi escolhido por se tratar de um problema combinatório real, de grande porte, e que possui relevância a nível mundial. Além disso, bons resultados experimentais podem provocar um considerável impacto na produtividade da empresa.

Como não há resultados anteriores para serem comparados com os resultados obtidos para as instâncias consideradas neste trabalho, modelos de programação matemática foram utilizados para a obtenção de limitantes duais para a solução do problema. Além disso, para determinar quão robusta é a técnica proposta, uma análise de sensibilidade foi realizada sobre as instâncias consideradas.

As próximas seções descrevem os objetivos deste trabalho e a organização do texto.

1.1 Objetivos do Trabalho

O objetivo principal deste trabalho é estudar a integração da Programação por Restrições com a Busca Tabu para resolver o problema do escalonamento das atividades na produção de petróleo.

Além disso, deseja-se comparar o desempenho da técnica híbrida com o desempenho da Busca Tabu quando esta última é aplicada isoladamente. Serão comparados o tempo necessário para melhorar as soluções iniciais, a qualidade da solução final obtida com cada técnica e a robustez de cada método. Para isso, serão testadas diversas instâncias com diferentes soluções iniciais aplicadas a cada uma delas. Além disso, será apresentada uma análise de sensibilidade sobre cada uma das instâncias.

Também é um dos objetivos desse trabalho estabelecer a qualidade das soluções encontradas. Como não havia resultados anteriores para as instâncias consideradas e como o método empregado não dá garantias sobre a qualidade das soluções obtidas, foram obtidos limitantes para o problema utilizando-se alguns modelos de programação matemática.

1.2 Trabalhos Relacionados

Há vários trabalhos que tratam diferentes problemas de otimização com aplicações na indústria petrolífera. Entre os temas relacionados a este assunto destacam-se: previsão de produção de petróleo; localização e sequenciamento de poços; otimização de sistemas produtores de petróleo e escalonamento de atividades para desenvolvimento de poços. O problema tratado nesta dissertação é referente ao último ítem.

A seguir, serão relacionados alguns trabalhos que discutem estes problemas.

- **Previsão da produção de petróleo**

  A importância de prever a produção de petróleo é inegável. Métodos de previsão variam desde curvas de extrapolação, simulação computacional, e até analogias. Um dos trabalhos mais recentes nesta área é o trabalho de Weiss, Balch e Stubbs [64]. Eles utilizam Ordenação Fuzzy e Redes Neurais para estabelecer correlações usadas para prever a produção de petróleo. Neste trabalho, as redes neurais são treinadas, testadas e então utilizadas para realizar a previsão.

- **Localização de poços**

  A determinação da localização de poços de petróleo é um problema complexo. Depende de propriedades do reservatório, da especificação dos equipamentos de superfície e do poço, e também de critérios econômicos. As variáveis deste problema são as localizações dos poços e a principal função objetivo considerada é a *maximização do fluxo de caixa (net present value - NPV)*. Para o cálculo da função objetivo, em geral, são utilizados simuladores.

  Beckner e Song [8] formularam este problema como o *Problema do Caixeiro Viajante*. Neste trabalho as possíveis localizações dos poços são as cidades, e a ordem que os poços devem ser desenvolvidos é a seqüência da viagem. Além disso, quando há mais possíveis localizações do que poços que devem ser efetivamente desenvolvidos, as localizações excedentes são consideradas poços inativos. Este fato influencia a função objetivo e, como ele é um pouco diferente do Problema do Caixeiro Viajante padrão, a taxa de convergência dos algoritmos utilizados para resolver o problema pode sofrer uma redução. Para uma dada seqüência de poços, um simulador é utilizado para calcular o NPV (distância total). A técnica de *Simulated Annealing* é utilizada para resolver o problema. Este trabalho mostrou a necessidade do desenvolvimento de mais pesquisas para se chegar a um número adequado de iterações que leve a um nível aceitável de qualidade da solução.

  Bittencourt e Horne [10] hibridizaram o Método do Politopo, Algoritmos Genéticos e a abordagem *Proxy* proposta por Pan e Horne [49]. O Método do Politopo é uma

heurística de *hill climbing* primeiramente proposta por Nelder e Mead [47]. Um politopo para n variáveis de decisão consiste de $n+1$ pontos arranjados de forma côncava na superfície determinada pela função objetivo. Como este método não requer o cálculo de gradientes, ele é especialmente adequado para tratar funções não lineares. Métodos *Proxy* são utilizados para aproximar o comportamento de funções, quando o cálculo das mesmas é excessivamente caro. Várias técnicas podem ser usadas como método *Proxy*. Um exemplo são as Redes Neurais.

A intenção do trabalho de Pan e Horne [49] era reduzir o número de simulações necessárias para otimizar a localização dos poços. Baseado no algoritmo proposto por Bittencourt e Horne, Güyagüler e Horne [31] apresentaram um Algoritmo Genético Híbrido (HGA) capaz de evitar soluções sub-ótimas, incorporando o Método do Politopo e a abordagem *Proxy* aos procedimentos de um Algoritmo Genético convencional. Esta abordagem foi aplicada na bacia petrolífera do Golfo do México trazendo bons resultados e diminuindo consideravelmente o número de simulações. Centilmen, Ertekin e Grader [13] utilizaram Redes Neurais em substituição ao simulador para o cálculo da função objetivo.

Já a questão de que incertezas estão presentes nos dados usados para estabelecer os modelos é tratada por Güyagüler e Horne em [30]. Neste trabalho, as incertezas associadas com a localização dos poços são tratadas dentro do escopo da *Utility Theory* [33], a qual lança mão de simulação numérica como a ferramenta para medir a função objetivo. A *Utility Theory* é empregada para transformar o problema com incertezas em um problema determinístico. Em seguida, técnicas convencionais de otimização podem ser utilizadas. Neste trabalho a técnica HGA de [31] foi aplicada como a técnica otimizadora.

- **Otimização de sistemas produtores de petróleo**

  Sistemas produtores de petróleo são complexos e heterogêneos. Freqüentemente eles não são operados em níveis ótimos, mas sim em níveis que simplesmente satisfazem as restrições operacionais. Isso acontece principalmente devido a uma visão local de cada uma das restrições operacionais e também devido às limitações que as técnicas tradicionais de otimização podem demandar tempo e recursos computacionais.

  Normalmente, as variáveis de decisão destes problemas são parâmetros que precisam ser ajustados para determinar a condição operacional de cada poço, como a *freqüência de operação* e a *taxa de injeção*. Estes parâmetros podem variar ao longo do tempo.

  As funções objetivo consideradas para este tipo de problema são: a *maximização da produção total*, a *maximização dos lucros*, a *minimização dos custos* e a *maximização*

do fluxo de dinheiro (NPV). É necessário o uso de simuladores ou de alguma técnica de aproximação para calcular o valor da função objetivo quando o valor das variáveis já é conhecido.

Em 1990, Carrol e Horne [34] aplicaram os Métodos do Gradiente e do Politopo para maximizar a produção de um único poço de petróleo. Neste trabalho, os parâmetros não poderiam variar ao longo do tempo. Em 1992, Ravindra [55] deu continuação a este trabalho e permitiu a variação dos parâmetros. Em 1993, Fujii [20] continuou esta linha de estudo, considerando um conjunto de poços, ao invés de apenas um poço. Nestes três trabalhos cada restrição operacional era tratada como uma caixa preta. Em 1997, Palke e Horne [48] deram continuação ao trabalho de Ravindra, substituindo as caixas pretas por modelos detalhados de cada restrição. Três métodos de otimização foram considerados separadamente neste trabalho, sendo suas potencialidades e fraquezas comparadas. Os métodos considerados foram: Método de Newton, Método do Politopo e Algoritmos Genéticos. A função objetivo considerada foi a maximização do fluxo de dinheiro. Comparando os três métodos, o Algoritmo Genético foi identificado como o mais robusto para tratar o problema.

Stoisits e outros [61] utilizaram Redes Neurais e Algoritmos Genéticos para maximizar a produção de petróleo em grandes reservatórios. Neste trabalho, um Algoritmo Genético é utilizado como o método de otimização e Redes Neurais são utilizadas para substituir o uso de simuladores no cálculo da função objetivo.

Vázquez e outros [62] propõem um algoritmo híbrido que combina Algoritmos Genéticos e Busca Tabu. Nesta abordagem, cada solução individual gerada usando o Algoritmo Genético é melhorada utilizando a Busca Tabu. Bons resultados, tanto computacionais quanto de tempo de execução, foram obtidos. A principal fraqueza deste trabalho é a necessidade de se efetuar o cálculo da função objetivo inúmeras vezes, utilizando-se simuladores.

- **Escalonamento de atividades para desenvolvimento de poços**

  O desenvolvimento de poços de petróleo, antes que estes comecem a produzir, envolve um número substancial de atividades e recursos. A execução das atividades deve ser coordenada para que restrições sejam atendidas e, ao mesmo tempo, algum critério, como a produção total de óleo, seja otimizado. Em geral, é considerado um período de 4 a 5 anos para o escalonamento das atividades e o cálculo da produção.

  As variáveis de decisão deste problema são o instante de início de cada atividade e qual o recurso que será alocado para cada atividade.

  Hasle, Haut, Johansen e Ølberg em [32] tratam este problema utilizando Programação por Restrições. Neste trabalho, são considerados dois tipos principais de

recursos e cinco diferentes restrições, entre elas a restrição de precedência tecnológica. A função objetivo utilizada foi a minimização do *makespan*. Um único conjunto de dados, referente a 17 poços foi utilizado. O resultado obtido para o *makespan* foi 21 dias superior que o *makespan* do escalonamento real. Isto apontou a necessidade de uma substancial melhora, tanto na modelagem quanto na técnica utilizada. Os autores também apontaram a necessidade de se considerar outras funções objetivo.

Nenhum dos trabalhos mencionados acima aborda problemas de otimização envolvendo petróleo utilizando uma técnica híbrida que combine Programação por Restrições com Busca Tabu. Serão então relacionados alguns trabalhos que utilizam esta abordagem híbrida para atacar problemas combinatórios.

Wallace [63] apresenta um estudo onde a integração da Programação com Restrições e a Busca Tabu é feita de uma forma *fraca*: a Programação por Restrições é usada apenas para gerar uma solução inicial para o algoritmo de Busca Tabu.

Outra forma de integrar as duas técnica é utilizar a Programação por Restrições para testar a viabilidade e avaliar o custo de cada vizinho explorado. Esta forma de integração é utilizada em [54] para resolver *Problemas de Transporte*. Já [4, 14] tratam o problema de *Roteamento de Veículos* e [12] trata o problema de *Escalonamento de Tarefas*. Em [3] é considerado o problema de *Otimização de Padrão de Testes em Projetos de Circuitos Integrados*.

Pesant e Gendrau [52] utilizam a Programação por Restrições para controlar todo o processo de exploração das vizinhanças da Busca Tabu. Neste trabalho, é criado um modelo de Programação por Restrições para representar a vizinhança, um modelo para representar o problema original e um modelo que serve de interface entre a vizinhança e o problema original. A intenção é utilizar as restrições e a função objetivo do problema original na redução do domínio das variáveis do modelo de vizinhança. Esta abordagem foi utilizada para resolver o *Problema do Caixeiro Viajante*.

A integração das duas técnicas também pode ser feita da seguinte forma: ao invés de especificar os movimentos que definem como os vizinhos são obtidos a partir de uma solução corrente, como acontece na Busca Tabu pura, define-se apenas um *consenso* entre todas as soluções de uma vizinhança. Feito isso, a Programação por Restrições é utilizada para completar este *fragmento comum* e obter desta maneira os vizinhos. Ou seja, a Programação por Restrições é utilizada para obter uma nova solução (vizinho) a partir de uma solução parcial. Vale lembrar que esta *solução parcial* é um fragmento da solução corrente. Esta é a abordagem que mais se aproxima da utilizada neste trabalho.

Esta técnica foi utilizada em [11] para resolver um problema de *Job Shop*. Neste trabalho, o *consenso* entre as soluções é um conjunto de pares de atividades que são executadas pela mesma máquina. É imposto que a ordem relativa das atividades de cada

par seja a mesma em todos os elementos da vizinhança. A ordem relativa destas atividades é a ordem existente entre as mesmas na solução corrente.

Outros problemas combinatórios também foram atacados utilizando este tipo de integração. O problema da *Montagem de Tabela de Horário* é tratado em [51]. Já [43] considera o *Problema de Designação Quadrático*, enquanto o *Roteamento de Veículos* é abordado em [60, 57].

1.3 Organização do Texto

O texto desta dissertação está organizado da seguinte forma. O capítulo 2 descreve o problema que será tratado neste trabalho, bem como as características da instância real fornecida pela Petrobras e como um gerador de instâncias foi implementado.

O capítulo 3 apresenta, resumidamente, fundamentos teóricos necessários para um entendimento adequado do trabalho descrito nesta dissertação. Este capítulo também descreve como a Busca Tabu será integrada com a Programação por Restrições, dando origem a abordagem híbrida proposta neste trabalho.

O capítulo 4 expõe detalhes do tratamento do problema. Por exemplo, descreve como as soluções iniciais foram geradas, descreve as vizinhanças consideradas e também os parâmetros utilizados na Busca Tabu.

Já o capítulo 5 trata a questão da obtenção de limitantes duais para o problema. Este capítulo descreve os modelos de Programação Linear Inteira utilizados para modelar relaxações do problema original, bem como os resultados obtidos por estes modelos.

No capítulo 6 são apresentados e discutidos os resultados computacionais obtidos na resolução do problema. O capítulo 7 apresenta a análise de sensibilidade realizada sobre as instâncias consideradas neste trabalho.

Finalmente, o capítulo 8 traz as conclusões obtidas nesta pesquisa e também discute trabalhos futuros.

Capítulo 2

Descrição do Problema

Este capítulo descreve o problema combinatório considerado nesta dissertação. Esta descrição inclui os procedimentos necessários para a construção de um poço de petróleo, quais as restrições que se aplicam ao problema, e também as funções objetivo de interesse. Além disso, a Petrobras forneceu apenas uma instância real para que os testes computacionais pudessem ser realizados. Como este número mostrou-se insuficiente para testar quão robustas são as implementações propostas, foi realizado um estudo estatístico na instância real e, baseado neste estudo, foi criado um gerador de instâncias. Este capítulo também descreve as características da instância real e a construção de um gerador de instâncias, baseado nestas características.

2.1 O Problema

A Petrobras é uma das mais eficientes empresas do mundo na exploração de petróleo em águas profundas. A *Bacia de Campos* é uma imensa área no mar onde a Petrobras explora petróleo. Nela, existem centenas de locais que são considerados promissores poços de petróleo. Estes poços de petróleo precisam ser construídos, para que sejam colocados em produção. Existem também diversos poços que estão em diferentes estágios do processo de construção.

Portanto, o problema em questão é: dados um conjunto de poços, as atividades a serem executadas em cada poço e os recursos disponíveis, como barcos e sondas, necessários para a execução destas atividades, determinar um sequenciamento das atividades em um dado horizonte de tempo, de forma a otimizar uma função objetivo. Este sequenciamento é claro, deve satisfazer uma série de restrições.

As principais características do problema são descritas em maiores detalhes nas próximas subseções.

2.1.1 Procedimentos

O construção de um poço compreende várias etapas. Quando um determinado local é considerado um promissor poço de petróleo, barcos e sondas são enviados a este local para realizar as devidas operações de perfuração.

Perfurado o poço, inicia-se o processo de sua preparação para extração de petróleo (completação). Em primeiro lugar, é instalada a "Árvore de Natal Molhada (ANM)" [1] sobre a boca do poço para que o petróleo não extravaze para o mar. Nesta etapa são utilizadas sondas e barcos. Posteriormente, um barco leva uma linha de produção da qual uma extremidade é ligada na ANM e a outra extremidade é encaixada num *manifold* ou então sobe direto para a superfície até a plataforma de produção (interligação). Um *manifold* é uma estrutura para junção das linhas de produção no fundo do mar. É instalado por uma sonda ou um barco e seu uso evita que cada poço necessite de tubulações exclusivas que o liguem desde o fundo do mar até a superfície. Assim, as linhas de produção de vários poços relativamente próximos podem se interligar no *manifold* e, deste, uma única tubulação sobe até a superfície. Na ausência de *manifolds*, e com o metro de linha de produção alcançando cerca de mil dólares, a ligação até a superfície de um único poço isolado a uma profundidade de dois mil metros acarretaria um custo de cerca de dois milhões de dólares.

Completado o processo de interligação, parte-se para a extração do petróleo propriamente dita. Para tal, na superfície do mar são colocadas bases de captura de petróleo, as chamadas UEPs (Unidade Estacionária de Produção), onde se processará e armazenará (quando possuir capacidade de armazenamento) o produto até que navios venham recolhê-lo e levá-lo para terra. A interligação entre o *manifold* e a plataforma UEP é feita por um barco. Se a vazão de petróleo for muito elevada e/ou a UEP não possuir capacidade de armazenamento, pode-se optar pela instalação local de uma plataforma petrolífera para armazenar o petróleo extraído.

2.1.2 Restrições

As principais restrições envolvidas no processo de sequenciamento das atividades necessárias para o desenvolvimento de um poço de petróleo são:

1. **Precedência Tecnológica:** restrição que define a ordem entre as atividades. Se uma atividade A deve ser executada antes de uma atividade B, há precedência tecnológica de A para B.

2. **Restrição de Marcos:** uma atividade deve terminar antes ou iniciar depois de uma determinada data, com ou sem *lag time*.

[1] Imensa estrutura metálica com tubulações onde se encaixam válvulas.

3. **Restrição de Data:** uma atividade deve iniciar e terminar em uma data fixa.

4. **Características das Atividades:** as atividades serão realizadas por recursos que possam executá-las, atendendo às restrições de lâminas d'água, tipos de equipamentos, entre outras.

5. **Indisponibilidade de Poços:** só pode ser realizada uma atividade simultânea em cada poço.

6. **Indisponibilidade de Recursos:** os recursos só podem realizar uma atividade por vez. Além disso, os recursos podem ficar indisponíveis num determinado período de tempo, seja por motivo de manutenção ou por não estarem com seu contrato em dia.

7. **Seqüência de Poços:** as atividades nos poços a serem perfurados para entrar em produção devem obedecer a uma seqüência imposta pela geologia do local.

8. **Restrições de Superfície:** dependendo dos tipos de sondas atuando sobre um poço, deve haver uma área grande o suficiente entre essas sondas para evitar risco de colisão durante manobras.

9. **Fator de Eficiência:** o tempo de duração da tarefa pode variar conforme um fator de eficiência do recurso que irá executá-la.

É importante ressaltar que não é permitida a interrupção de atividades. Neste trabalho, apenas as restrições $1, 4, 5$ e 6 são consideradas. Estas são as principais restrições. Os dados reais fornecidos pela Petrobras cobrem somente estas restrições. Outras restrições podem ser incorporadas na técnica híbrida sem muito esforço, já que uma das características da Programação por Restrições é permitir a inserção de novas restrições de uma maneira fácil e eficiente. Na Busca Tabu pura, há uma conexão mais forte entre as restrições do problema e a vizinhança, o que torna modificações uma tarefa mais complexa.

2.1.3 Objetivos

Deseja-se um sequenciamento das atividades satisfazendo às restrições acima e que atendam alguns dos seguintes critérios:

1. Maximizar a produção de óleo em um determinado horizonte de tempo (maximizar a vazão);

2. Minimizar o instante de tempo em que todas as atividades já foram executadas (minimizar *makespan*);

3. Aumentar o lucro das atividades dos poços no horizonte de tempo definido;

4. Diminuir os custos de desenvolvimento dos poços no horizonte de tempo do sequenciamento;

5. Diminuir o tempo de ociosidade das sondas de perfuração e barcos para uma curva de produção de óleo determinada;

6. Aumentar a produção de determinados tipos de petróleo em detrimento de outros.

Este trabalho vai focar o primeiro objetivo (maximizar a vazão), pois é um dos mais relevante tanto do ponto de vista teórico quanto do ponto de vista prático. Ele é mais interessante teoricamente pois soluções obtidas tendo em vista o objetivo 2, já na primeira versão de um modelo de Programação por Restrições, ficaram a apenas 0.2% de uma cota inferior do problema. Comportamento bem diferente acontece com o objetivo 1, como ficará claro na seção de resultados computacionais. A relevância prática vem do seguinte fato externado pela própria indústria: em grande parte do tempo, o interesse maior é pelo aumento da produção.

A produção de óleo de um poço é calculada da seguinte forma. Cada poço tem uma vazão associada e uma atividade cujo propósito é colocar o poço em produção. Quando esta atividade é concluída, é considerado que o poço está em produção. A produção do poço é obtida multiplicando-se a sua vazão pelo tempo restante desde a entrada do poço em produção até um horizonte de tempo definido. Vale ressaltar que atividades podem ser executadas depois do horizonte de tempo especificado, mas os poços onde estas atividades são executadas não entram para o cálculo da produção.

Pela descrição do problema, pode-se observar que ele não apenas refere-se a uma aplicação real, mas também possui um grande número de particularidades que o tornam bastante diferenciado. Daí a relevância prática e acadêmica da abordagem proposta neste trabalho.

2.2 A Instância Real e o Gerador de Instâncias

Nesta seção serão apresentadas as características da instância real e como foi construído o gerador de instância, baseado nestas características.

2.2.1 A Instância da Petrobras

Dois tipos de recursos são considerados na instância da Petrobras: sondas e barcos. Além disso, sempre que uma atividade requer um tipo de recurso, todos os recursos deste tipo podem executar a atividade. Ou seja, não há nenhuma restrição ou distinção no uso de qualquer um dos barcos disponíveis, quando a atividade requer um barco. O recurso necessário para a execução da atividade é determinado pelo tipo da atividade.

O número de poços, atividades, barcos e sondas presentes na instância real são apresentados na tabela 2.1. Já a tabela 2.2 mostra alguns dados numéricos sobre a duração de todas as atividades.

NumPoços	NumAtividades	NumBarcos	NumSondas
130	482	3	5

Tabela 2.1: Dados da instância real

Média	Mediana	Modo	Mínimo	Máximo	Soma
17	12	1	129	1	8195

Tabela 2.2: Duração das atividades (em dias) - Instância real

Os poços da instância real têm um conjunto de atividades a serem executadas que segue um determinado padrão. O tipo da atividade e a relação de precedência entre as mesmas é o que determina um padrão. Os padrões apresentados pelos poços da instância real são mostrados na figura 2.1, onde os nós representam as atividades e os arcos as relações de precedência entre as mesmas. Nesta figura, percebe-se a existência de padrões distintos associados a um mesmo grafo. Isto significa que as relações de precedência entre as atividades são as mesmas, mas o tipo destas atividades é diferente. Por exemplo, os padrões 17, 19 e 20 apresentam uma ordem total entre as suas atividades, mas seus tipos de atividades são diferentes. A figura 2.1 também mostra que a ordem das atividades nos poços não é sempre total. Este fato será importante para estabelecermos que as vizinhanças consideradas neste trabalho são de larga escala.

A tabela 2.3 apresenta alguns dados numéricos de cada padrão. Pode-se observar que a freqüência de ocorrência de cada padrão não é uniforme. Por exemplo, apenas um poço apresenta o padrão 3. Por outro lado, 42 poços apresentam o padrão 19.

A figura 2.2 apresenta o histograma da vazão e da lâmina dos poços. A unidade de medida para a lâmina dos poços é metros e para a vazão é uma medida interna adotada pela Petrobras.

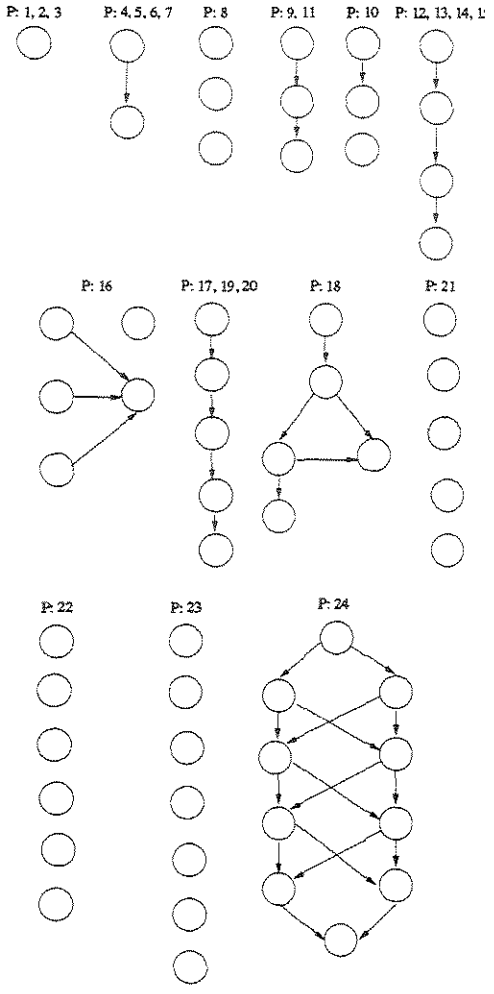


Figura 2.1: Precedência tecnológica entre as atividades

Esta figura mostra que os poços possuem vazões bem variadas. Esta informação eleva a dificuldade para se estabelecer o instante de início de cada atividade de forma a maximizar a vazão, pois existem dois fatores para serem considerados e balanceados: a vazão do poço e qual é o tempo mínimo necessário para executar todas as atividades do poço até que o mesmo entre em produção. Outra informação que esta figura traz é que a lâmina máxima de um poço é 1600 metros. Mas todos os recursos da instância real são capazes de operar em lâminas d'água superiores a este valor, exceto por uma das sondas.

Padrão	Num Poços	% Num Poços	Num Atividades	Padrão	Num Poços	% Num Poços	Num Atividades
1	1	0,87	1	13	8	7,01	4
2	1	0,87	1	14	2	1,75	4
3	1	0,87	1	15	1	0,87	4
4	5	4,38	2	16	1	0,87	5
5	1	0,87	2	17	12	10,52	5
6	1	0,87	2	18	4	3,50	5
7	1	0,87	2	19	42	36,84	5
8	1	0,87	3	20	1	0,87	5
9	21	18,42	3	21	2	1,75	5
10	1	0,87	3	22	1	0,87	6
11	1	0,87	3	23	1	0,87	7
12	3	2,63	4	24	1	0,87	12

Tabela 2.3: Dados dos padrões - Instância real

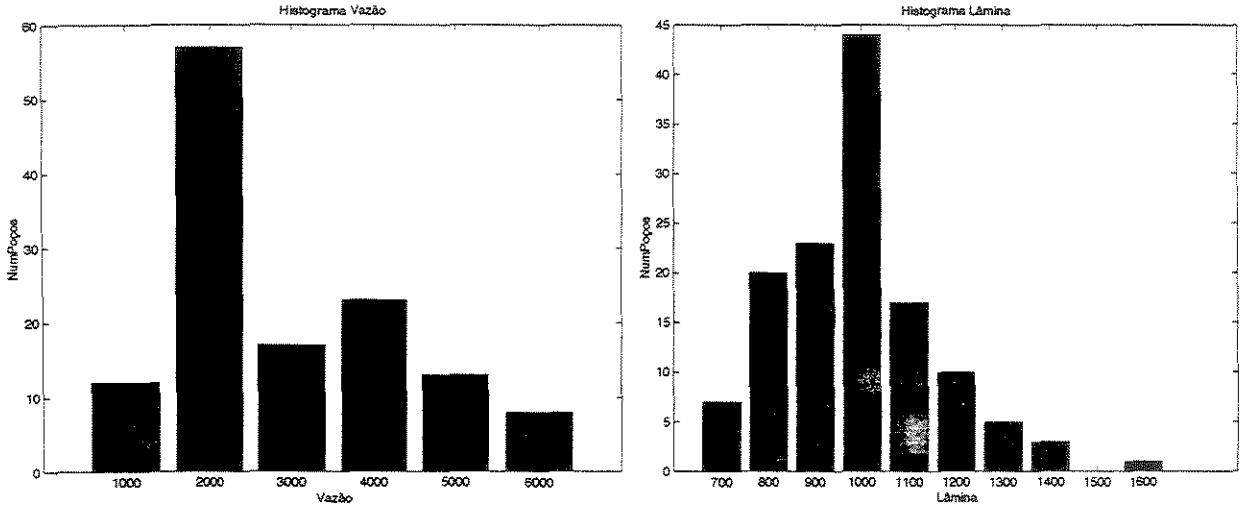
2.2.2 O Gerador de Instâncias

Com base na instância real, foi implementado um gerador de instâncias similares. O gerador constrói novas instâncias aplicando uma pertubação randômica nos dados da instância real.

O padrão dos poços das instâncias geradas foi determinado pela freqüência de ocorrência de cada padrão na instância real fornecida pela Petrobras. A duração de cada atividade gerada foi escolhida aleatoriamente entre as durações das atividades de mesmo tipo da instância real. A vazão e a lâmina de cada poço foram geradas seguindo uma distribuição uniforme entre os valores mínimos e máximos dos valores das vazões e das lâminas, respectivamente, dos poços de mesmo padrão presentes nos dados reais. Como já mencionado anteriormente, os recursos da instância real possuem todas as características possíveis e capacidade de operar em lâminas superiores às lâminas dos poços da instância, exceto por uma sonda. Sendo assim, os recursos gerados também possuem todas as características possíveis e são capazes de operar em lâminas superiores à maior lâmina gerada. O número de poços, de sondas e de barcos são parâmetros de entrada para o programa que gera as instâncias.

A notação seguida para o nome das instâncias está padronizada da seguinte forma: o nome contém um identificador, seguido do número de poços, sondas e barcos da mesma. Por exemplo, o nome 2P112S5B3, indica que a instância 2 possui 112 poços, 5 sondas e 3 barcos.

Os resultados obtidos, quando o gerador de instâncias foi utilizado para gerar uma instância com 112 poços, 5 sondas e 3 barcos, são apresentados nas tabelas 2.4, 2.5 e 2.6 e na figura 2.3. Esta instância recebeu o nome 2P112S5B3. Comparando estes resultados com as tabelas 2.1, 2.2 e 2.3, respectivamente, pode-se observar que os resultados para a instância gerada e a para a instância real são muito similares, exceto por variações



(a) Histograma Vazão

(b) Histograma Lâmina

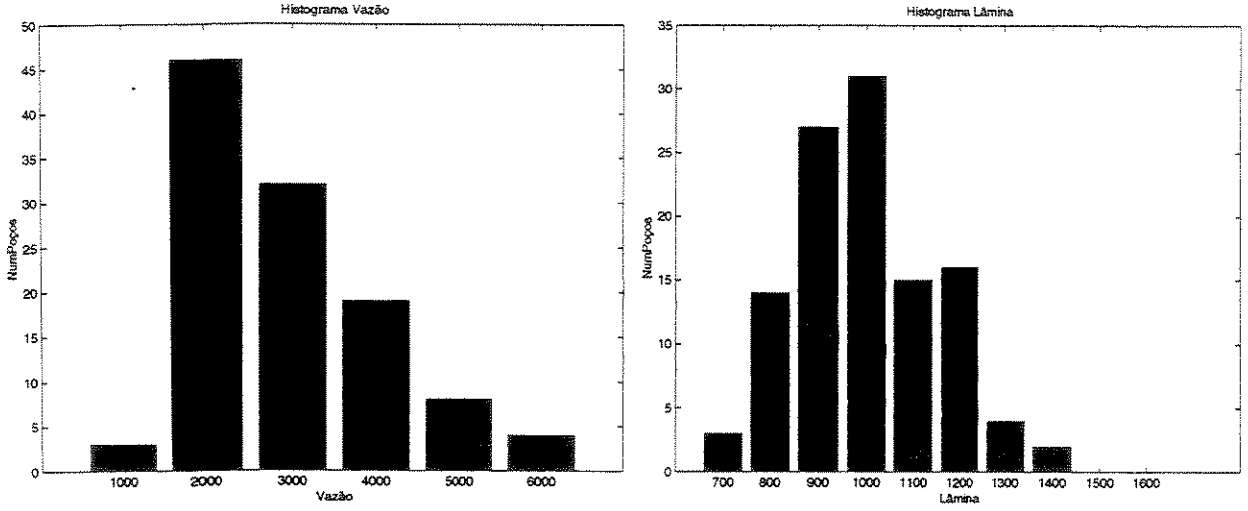
Figura 2.2: Histogramas - Instância real

randômicas inseridas pelo gerador de instância. Nas tabelas 2.3 e 2.6 o padrão 19 é o mais comum, representando cerca de 36% dos poços. As tabelas 2.2 and 2.5 também mostram que o tempo médio de duração das atividades é de 17 dias.

NumPoços	NumAtividades	NumBarcos	NumSondas
112	464	3	5

Tabela 2.4: Dados da instância 2P112S5B3

Neste trabalho, quatro instâncias serão consideradas. A instância real 1P130S5B3 e as instâncias obtidas usando o gerador: 2P112S4B3, 3P95S5B3 e 4P130S5B3. A primeira instância gerada apresenta um número menor de recursos e de poços do que a instância real. A segunda mantém o número de recursos e diminui o número de poços. A terceira foi gerada utilizando-se o mesmo número de poços e recursos que a instância real.



(a) Histograma Vazão (b) Histograma Lâmina

Figura 2.3: Histogramas - Instância 2P112S5B3

Média	Mediana	Modo	Mínimo	Máximo	Soma
16.65	12	1	129	1	7726

Tabela 2.5: Estatísticas sobre a duração das atividades - Instância 2P112S5B3

Padrão	Num Poços	% Num Poços	Num Atividades	Padrão	Num Poços	% Num Poços	Num Atividades
1	1	0,89	1	13	3	2,67	4
2	0	0	1	14	5	4,46	4
3	0	0	1	15	0	0	4
4	5	4,46	2	16	0	0	5
5	0	0	2	17	7	6,25	5
6	4	3,57	2	18	3	2,67	5
7	0	0	2	19	40	35,71	5
8	0	0	3	20	4	3,57	5
9	28	25	3	21	1	0,89	5
10	3	2,67	3	22	2	1,78	6
11	2	1,78	3	23	1	0,89	7
12	2	1,78	4	24	1	0,89	12

Tabela 2.6: Dados dos padrões - Instância 2P112S5B3

Capítulo 3

Técnicas

Este capítulo apresenta técnicas e conceitos básicos necessários para uma melhor compreensão dos capítulos que se seguem. Também descreve como as técnicas de Programação por Restrições e Busca Tabu são integradas para dar origem a abordagem híbrida, foco deste trabalho.

3.1 NP-Completude

Há uma grande variedade de problemas para os quais são conhecidos algoritmos *eficientes* (com complexidade de tempo polinomial no tamanho da instância) para resolvê-los. Exemplos de tais problemas são: ordenação de vetores, obtenção da mediana de um vetor, árvore geradora mínima de um grafo, caminhos mais curtos em grafos e multiplicação de matrizes. Infelizmente, existem inúmeros problemas para os quais não são conhecidos algoritmos eficientes.

Nesta seção será descrita uma classe contendo vários problemas para os quais não sabemos encontrar algoritmos polinomiais para resolvê-los. Sabendo-se a pertinência de um problema a esta classe, saberemos que se trata de um problema difícil e, portanto, dificilmente encontraremos um algoritmo polinomial para o mesmo. Por outro lado, pode-se decidir com maior segurança por outras técnicas apropriadas para a resolução deste problema.

3.1.1 Reduções entre problemas

Uma redução de um problema A para um problema B é um par de transformações τ_I e τ_S tal que, dada uma instância qualquer I_A de A tem-se que:

- τ_I transforma I_A em uma instância I_B de B;

- τ_S transforma a solução S_B de I_B em uma solução S_A de I_A.

Se as transformações τ_I e τ_S são feitas em tempo polinomial, então a redução de A para B é chamada de *redução polinomial*. As reduções são utilizadas em duas situações:

1. Deseja-se encontrar um algoritmo para A e é conhecido um algoritmo para B, ou seja, determina-se uma cota superior para o problema A.

2. Deseja-se encontrar uma cota inferior para o problema B e é conhecida uma cota inferior para o problema A.

3.1.2 Classes de Problemas

Iremos agora catalogar os problemas como estando em pelo menos duas classes:

- A classe dos problemas para os quais se conhece um algoritmo eficiente para resolução.

- A classe dos problemas para os quais se conhece um algoritmo eficente de verificação. Ou seja, para estes problemas é difícil encontrar um algoritmo polinomial que resolve o problema, mas existe um algoritmo polinomial que verifica se uma *proposta de solução* resolve de fato o problema.

Tradicionalmente, o estudo de classes de complexidade é feito para **problemas de decisão**, ou seja, aqueles em que a resposta é da forma *SIM* ou *NÃO*. No entanto, em geral é fácil encontrar uma redução polinomial de problemas de otimização para problemas de decisão.

Define-se a classe P como o conjunto de problemas que podem ser resolvidos por um **algoritmo determinístico** polinomial. Antes de definir a classe NP, será dada uma noção de **não-determinismo**. Dizemos que um algoritmo é não determinístico se, além de todas as regras de um algoritmo determinístico, ele pode fazer escolhas de forma não determinística. A classe NP é o conjunto de todos os problemas que podem ser resolvidos por um **algoritmo não-determinístico** polinomial. Tal algoritmo é dividido em duas partes. A primeira parte corresponde a **escolha** da provável solução (podendo fazer uso de escolhas não determinísticas). A outra parte consiste em **verificar** de forma determinística se a provável solução é de fato uma solução para o problema. Como todo algoritmo determinístico é um caso particular de um algoritmo não determinístico, segue que $\mathrm{P} \subseteq \mathrm{NP}$.

A questão fundamental da Teoria da Computação é se $\mathrm{P} = \mathrm{NP}$, embora a maioria dos pesquisadores acredite que esta igualdade não seja válida. Uma das maiores razões que se

acredita que P $\neq$ NP é a existência da classe de problemas NP-completo. Um problema A é NP-difícil se existe uma redução polinomial de qualquer problema em NP para A. Um problema B é NP-completo se $B \in$ NP e $B \in$ NP-difícil. Portanto, se existir um algorimo polinomial para um problema NP-completo, então todos os problemas de NP podem ser resolvidos polinomialmente. Parece intrigante a eixistência de tal conjunto, mas de fato, Cook em 1971, mostrou que o problema da satisfabilidade (SAT) [15] pertence à classe NP-completo. Para provar que um problema B está em NP-completo, basta mostrar que B é um problema em NP e que dado um certo problema A em NP-difícil, existe uma redução polinomial de A para B.

3.2 Programação Matemática

Esta seção apresenta algumas metodologias conhecidas no campo da Programação Matemática, como a Programação Linear e a Programação Linear Inteira. Assume-se que o leitor possui conhecimentos elementares a respeito de Álgebra Linear. Nesse sentido, recomenda-se a leitura da Seção 1.5 de [9].

3.2.1 Programação Linear

O problema geral de Programação Linear é dado pelo *programa linear* abaixo:

$$z_{PL} = \min\{cx : Ax \leq b, x \in R_+^n\},$$

onde c é um vetor $1 \times n$ de custos, A é a matriz $m \times n$ de coeficientes e b é um vetor $m \times 1$. Todos os componentes de c, A e b são números racionais e R_+^n denota todos os vetores $n \times 1$ cujas componentes assumem valores reais não negativos. Deseja-se, então, atribuir valores às componentes de x (variáveis) de modo a satisfazer às desigualdades $Ax \leq b$ (restrições) e minimizar o valor da *função objetivo* cx. Dá-se o nome de *formulação* de um problema à sua expressão em termos de um conjunto de variáveis, restrições e uma função objetivo. Quando o conjunto $\{Ax \leq b, x \in R_+^n\}$ é vazio, o problema não tem solução e é dito *inviável*. Vale ressaltar, ainda, que $\{cx : Ax \leq b, x \in R_+^n\}$ pode ser infinito.

Todo programa linear pode ser escrito no formato acima, bastando para isso utilizarem-se algumas manipulações algébricas. Por exemplo, uma função objetivo de maximização, $\max cx$, pode ser escrita na forma de minimização como $-\min -cx$. Da mesma forma, multiplicando-se por -1 ambos os lados de inequações do tipo $\geq$ obtêm-se inequações do tipo $\leq$. Restrições de igualdade, por sua vez, podem ser reescritas na forma de duas restrições de desigualdade: uma do tipo $\leq$ e outra do tipo $\geq$.

O algoritmo mais conhecido para resolver problemas de Programação Linear é o algoritmo Simplex de Dantzig [16]. Detalhes sobre este algoritmo podem ser encontrados em

[36]. Embora este algoritmo não seja de tempo polinomial, na prática ele é bem eficiente quando implementado de uma forma adequada. Os Métodos de Ponto Interiores [36] são algoritmos que resolvem problemas de Programação Linear em tempo polinomial.

3.2.2 Programação Linear Inteira

Existem problemas que não podem ser modelados como programas lineares da forma apresentada na subseção anterior. Por exemplo, é possível que as variáveis x representem o número de trabalhadores a serem contratados para determinadas instalações de uma empresa. Nesse caso, não faz sentido dizer que 4.37 trabalhadores devem ser admitidos. Precisa-se de um número *inteiro* como resposta. Portanto, um modelo de Programação Linear não é apropriado pois não garante a integralidade da solução, e arredondando-se os valores fracionários geralmente não se obtêm bons resultados.

Em casos como esse, tem-se em mãos um problema de *Programação Linear Inteira*, cuja formulação é dada por

$$z_{PI} = \min\{cx : Ax \leq b, x \in Z_+^n\}, \qquad \text{(PI)}$$

onde Z_+^n é o conjunto dos vetores de n componentes em que cada uma delas assume valores inteiros não negativos. Dá-se o nome de *relaxação linear* de (PI) ao problema obtido a partir de (PI) substituindo-se a restrição $x \in Z_+^n$ por $x \in R_+^n$. Note que o valor da solução da relaxação linear é um limitante inferior para o valor da solução de (PI). Isto porque $Z_+^n \subset R_+^n$.

Em geral, problemas de *Programação Linear Inteira* são muito mais difíceis de serem resolvidos do que problemas de *Programação Linear*. Isto é um fato esperado, já que muitos problemas aparentemente difíceis podem ser formulados como problemas de *Programação Linear Inteira*. É preciso, portanto, recorrer a algoritmos que, apesar de possuírem complexidade exponencial de tempo, percorrem o espaço de soluções de forma criteriosa. Um desses algoritmos é descrito abaixo, seguindo [50].

Branch-and-Bound

A idéia básica de um algoritmo de *Branch-and-Bound* é enumerar, *implicitamente*, todas as soluções viáveis para o problema de *Programação Linear Inteira*, até que a solução ótima seja encontrada. Esse mecanismo evolui através de partições sucessivas do espaço de soluções. Seja o problema de Programação Linear Inteira abaixo

$$\begin{aligned} \min\ & c(x) = cx \\ \text{sujeito a } & Ax \leq b, \\ & x \geq 0 \text{ e inteiro.} \end{aligned} \tag{PI}$$

Resolvendo-se a relaxação linear de (PI), obtém-se uma solução x^0 que, caso seja inteira, corresponde a uma solução ótima de (PI). Todavia, em geral, x^0 não é inteira e seu custo $c(x^0)$ é apenas um limite inferior para o valor da solução ótima de (PI). O próximo passo do algoritmo consiste em dividir (PI) em dois subproblemas, (PI_1) e (PI_2), com o auxílio de duas restrições mutuamente exclusivas. Seja x_i^0 a i-ésima componente de x^0, e cujo valor não é inteiro. Tem-se portanto

$$\begin{aligned} \min\ & c(x) = cx \\ \text{s.a. } & Ax \leq b, \\ & x \geq 0 \text{ e inteiro}, \\ & x_i \leq \lfloor x_i^0 \rfloor, \end{aligned} \tag{PI_1}$$

e

$$\begin{aligned} \min\ & c(x) = cx \\ \text{s.a. } & Ax \leq b, \\ & x \geq 0 \text{ e inteiro}, \\ & x_i \geq \lfloor x_i^0 \rfloor + 1. \end{aligned} \tag{PI_2}$$

Note que uma solução ótima de (PI), denotada por x^*, tem de ser igual a uma solução ótima de um dos dois subproblemas, pois, necessariamente, ou $x_i^* \leq \lfloor x_i^0 \rfloor$ ou $x_i^* \geq \lfloor x_i^0 \rfloor + 1$. Nesse ponto, escolhe-se um dos subproblemas, e.g. PI_1, e resolve-se a sua relaxação linear. Tem-se agora uma outra solução, x^1, que pode novamente não ser inteira. De maneira semelhante, divide-se PI_1 em outros dois subproblemas e o processo todo se repete, criando a chamada *árvore de enumeração*. O conjunto de todos os subproblemas que ainda não foram divididos corresponde a uma partição das soluções viáveis do problema original. Em determinado nó da árvore, esse processo pode ser interrompido por uma de duas razões. Em primeiro lugar, a solução da relaxação linear do subproblema considerado pode ser inteira. Em segundo lugar, o programa linear correspondente àquele nó da arvore pode ser inviável.

Até este ponto, o algoritmo prosseguiu por meio de ramificações na árvore de enumeração. Essa operação recebe o nome de *branching*. Caso as operações de *branching* se

repitam até que não haja mais como particionar um nó em subproblemas, aquela folha da árvore que eventualmente possua uma solução inteira viável com o menor custo corresponderá a uma solução ótima do problema original. Contudo, existe ainda uma outra alternativa para aumentar a eficiência do algoritmo como um todo. Trata-se da análise dos limitantes, ou *bounds*.

Suponha que, em algum ponto da execução do algoritmo de *Branch-and-Bound*, a melhor solução inteira conhecida tenha um valor igual a z_m. Suponha também que num determinado nó k da árvore, ainda não dividido em subproblemas, o limitante inferior fornecido pela solução da relaxação linear seja igual a z_k. Isso significa que qualquer solução inteira que se possa encontrar a partir de nós descendentes de k terá valor igual ou maior a z_k. Caso $z_k \geq z_m$, não é necessário dar continuidade ao processo de *branching* a partir do nó k, pois já se sabe que, nos nós descendentes de k, não haverá solução inteira melhor que z_m. Daí surge o nome de *enumeração implícita*. Foram eliminados *todos* os nós descendentes de k sem necessidade de criá-los explicitamente. Esta, portanto, é uma terceira maneira de se interromper o avanço do algoritmo a partir de um nó da árvore de enumeração.

Dois tipos de decisões têm de ser tomadas a cada passo do algoritmo. É preciso escolher qual nó da árvore será o próximo a sofrer *branching* e, em seguida, qual variável de valor fracionário irá compor as restrições adicionadas aos subproblemas do nó selecionado. Essas decisões podem afetar sensivelmente o desempenho do algoritmo de *Branch-and-Bound* e, normalmente, a melhor estratégia a usar depende do problema em questão.

3.3 Busca Tabu

Heurísticas são algoritmos que procuram boas soluções para problemas complexos em tempos computacionais viáveis, porém sem oferecer garantias de que uma solução ótima será encontrada. Heurísticas de Busca Local são algoritmos que partem de uma solução factível inicial do problema e, iterativamente, tentam obter uma solução melhor, indo de uma solução para outra através de um *movimento local*. Se um *movimento local* é válido ou não, depende do problema que está sendo resolvido e do movimento em si. O conjunto de todas as soluções que podem ser atingidas a partir de uma solução s através de um *movimento local* é chamado *vizinhança* de s. Existem diferentes estratégias propostas na literatura para guiar um algoritmo de Busca Local. Cada uma destas estratégias dá origem a uma metaheurística, sendo que as mais conhecidas são *Simulated Annealing*, Algoritmos Genéticos e Busca Tabu [45].

A Busca Tabu tem obtido soluções ótimas, ou próximas disso, em uma grande variedade de problemas clássicos e práticos, abrangendo desde escalonamento em telecomunicações e reconhecimento de caracteres até redes neurais [27, 56, 25, 23, 24]. Ela pode ser

integrada com Programação por Restrições, Algoritmos Genéticos, *Simulated Annealing* e *Branch-and-Bound*, entre outras técnicas [26, 52].

Na Busca Tabu, há vários critérios para selecionar qual vizinho será escolhido para ser a próxima solução. Pode-se escolher o primeiro vizinho que melhore a melhor solução, pode-se escolher o melhor vizinho entre os x primeiros vizinhos percorridos, entre outros critérios de seleção. O vizinho escolhido pode deteriorar a solução corrente. Isto é permitido para se escapar de ótimos locais. Para se evitar ciclos (ocasionados por um movimento inverso) é mantida uma lista de movimentos proibidos, ou *tabu*. Cada movimento local selecionado permanece tabu durante um certo número de iterações. Porém, se um movimento considerado tabu levar a uma solução bem melhor do que as obtidas anteriormente, este movimento pode deixar de ser tabu. Isto é chamado de *critério de aspiração*. O número de iterações que um movimento permanece na *lista tabu* é denominado *prazo tabu*. O *critério de parada* usualmente adotado restringe-se a um simples limite sobre o número total de iterações do algoritmo ou então sobre o número de iterações em que a melhor solução encontrada tenha permanecido inalterada. O tempo computacional também pode ser utilizado como critério de parada. Estes parâmetros, seleção do vizinho, movimentos tabu, prazo tabu, critério de aspiração e critério de parada, devem ser ajustados por meio de testes, sendo uma tarefa essencial para o bom desempenho do algoritmo.

Cabe observar que a busca tabu, apesar da tentativa de escapar dos mínimos locais, não oferece qualquer garantia de que irá produzir uma solução ótima.

Um aspecto crucial desta técnica é a escolha da vizinhança. Vizinhanças ambiciosas aumentam as chances de sucesso, mas são complexas de serem exploradas. Vizinhanças pequenas são simples e rápidas de serem exploradas, mas podem requerer que muitas iterações sejam realizadas para se obter uma boa solução, sendo que em uma vizinhança grande esta solução pode ser alcançada em poucas iterações. A forma como as vizinhanças são percorridas também é um aspecto relevante a ser considerado.

A maior vantagem da Busca Tabu é sua eficiência, haja visto a boa qualidade das soluções que são reportadas na literatura para uma grande variedade de problemas combinatórios difíceis. Sua maior fragilidade reside na forte dependência das restrições do problema que está sendo considerado, ou seja, o algoritmo é pouco flexível. Visto que a grande maioria dos problemas reais sofrem constantes mudanças, esta fragilidade deve ser minorada de alguma forma, para que essa técnica se torne robusta e atraente o suficiente para ser empregada na resolução destes problemas.

3.4 Vizinhanças de Larga Escala

Vizinhanças de Larga Escala, denotadas por VLSN (do inglês *Very Large Scale Neighborhood*) são aquelas cujo tamanho aumenta exponencialmente à medida que o tamanho

dos dados de entrada crescem. Também são consideradas VLSN vizinhanças cujo tamanho é muito grande para ser explorado explicitamente em tempo razoável. Por exemplo, se o tamanho da entrada de um dado problema é $O(n)$, uma vizinhança com $O(n^3)$ elementos pode ser excessivamente grande para que ela seja explorada na prática quando n for muito grande[1]. Neste trabalho também consideraremos como VLSN vizinhanças que apresentam um número polinomial de vizinhos, desde que a determinação do melhor vizinho seja um problema NP-difícil. Explorar de maneira eficiente este tipo de vizinhança é importante pois, usualmente, quanto maior a vizinhança explorada por algoritmos de Busca Local, melhor é a qualidade tanto das soluções obtidas localmente quanto da solução final.

Há problemas, como o do caixeiro viajante e o problema da árvore geradora mínima capacitada, em que certas estruturas de vizinhança de larga escala podem ser percorridas em tempo polinomial [1, 2, 18]. Por exemplo: uma vizinhança simples para o problema do caixeiro viajante chamada ASSIGN foi introduzida em 1981 por Sarvanov e Doroshko [59]. Esta vizinhança é obtida fixando-se todas as cidades que estão em posição ímpar no tour, removendo as cidades que estão em posição par e finalmente reinserindo-as de forma arbitrária nas posições vazias. Esta vizinhança tem tamanho exponencial, mas é fácil perceber que para se obter o melhor vizinho basta resolver um problema padrão de designação, de ordem polinomial [18]. Ou seja, este é um caso de VLSN que pode ser percorrida em tempo polinomial. Outra vizinhança que pode ser definida para este problema é aquela em que as cidades nas posições ímpares são permutadas simultaneamente com as cidades das posições pares. Esta vizinhança também é exponencial, e [18] ainda prova que explorar toda esta vizinhança é um problema NP-difícil. Portanto, o mesmo problema pode apresentar várias estruturas de vizinhança de larga escala. Se uma delas possui um algoritmo polinomial para percorrê-la, não significa que todas as VLSN deste problema apresentarão essa facilidade. Isto aponta uma dificuldade deste método: a determinação de uma boa estrutura de vizinhança.

3.5 Programação por Restrições

Muitos problemas em Pesquisa Operacional, tais como escalonamento de tarefas, alocação de horários e outros problemas combinatórios, podem ser representados por um modelo de Programação por Restrições. Estes problemas são denominados *Constraint Satisfaction Problems* (CSPs). A Programação por Restrições vem se afirmando como uma ferramenta poderosa para abordar tais problemas [39, 38, 32, 65, 66].

Um modelo de Programação por Restrições é composto por um conjunto de variáveis X e um conjunto de restrições C. A toda variável de X é associado um domínio (conjunto de valores que a variável pode assumir). O conjunto C serve para limitar os valores que

podem ser atribuídos às variáveis em X. Achar uma solução para o CSP significa atribuir valores às suas variáveis de forma a satisfazer suas restrições. Uma característica bem marcante da Programação por Restrições é que ela permite a modelagem das restrições de problemas de maneira fácil e eficiente.

A idéia básica da Programação por Restrições é utilizar as restrições do CSP para eliminar implicitamente regiões infactíveis. Isto é feito reduzindo o domínio das variáveis através de um mecanismo chamado *propagação de restrições*. Este mecanismo visa também garantir a consistência do sistema como um todo. Ele funciona da seguinte forma: quando uma restrição é imposta, os domínios das variáveis relacionadas por esta restrição são checados em busca de valores inconsistentes. Caso existam estes valores, eles são removidos do domínio das variáveis. Variáveis relacionadas por meio de outras restrições com as variáveis que tiveram domínio reduzido, devem ser (re)checadas, e, possivelmente, também terão seus domínios reduzidos. Este processo é interrompido quando todas as restrições já tiverem sido impostas e todas as possíveis reduções de domínio já tiverem sido realizadas. O processo também pode ser interrompido quando algum domínio resultar no conjunto vazio, indicando a inexistência de soluções.

A propagação por si nem sempre detecta inconsistência entre as restrições [40] e geralmente não instancia (atribui um único valor) às variáveis do problema. Para garantir consistência e instanciar todas as variáveis é necessário combinar propagação e busca. A componente de busca, chamada *labelling*, consiste em atribuir às variáveis valores pertencentes ao seu domínio. Cada elemento no domínio representa uma alternativa de escolha e cria um ramo na árvore de enumeração. Além disso, ao longo da busca, o efeito da atribuição de um valor a uma variável é propagado nos domínios das variáveis que ainda não foram instanciadas, podendo cortar possibilidades que permaneciam em aberto. Caso alguma inconsistência seja detectada, a última atribuição de valor é desfeita, dando lugar a outra alternativa (*backtracking*). O processo todo se repete até que uma solução seja encontrada ou até que se prove a inexistência de soluções viáveis [40, 39].

A seleção das variáveis e dos valores que lhes serão atribuídos é feita heuristicamente. A escolha de boas heurísticas é de fundamental importância para o tempo de resposta do programa.

A busca por uma solução ótima, ao invés de uma simples solução é, em geral, tratada na Programação por Restrições aplicando-se a técnica de *Branch-and-Bound* na componente de busca do algoritmo. Uma função de custo é requerida. À medida que as variáveis são instanciadas durante a busca, valores resultantes para a função de custo se qualificam como cotas superiores, assumindo um problema de minimização. Quando uma solução é encontrada, com valor inferior ao da solução atual, seu custo é colocado como um novo limitante superior para o valor do custo em qualquer solução futura. Quando o custo de um *labelling* parcial excede o limitante corrente ocorre *backtracking*, já que o *labelling*

parcial não pode dar origem a uma solução que melhore a solução atual.

Um grande desafio da Programação por Restrições é a criação de um modelo que represente adequadamente as condições reais de um problema. Um modelo ruim, por exemplo, pode gerar soluções errôneas ou não satisfatórias, e até mesmo pode não apresentar uma solução. O modelo, portanto, é um fator determinante para o tempo de resposta do programa e a qualidade da solução gerada. Aqui, por um modelo entende-se não só as restrições que descrevem o problema, mas também as heurísticas de *labelling* que serão usadas.

Como já foi mencionado, a Programação por Restrições resolve problemas reais de otimização combinatória utilizando a estratégia de *branch-and-bound* para realizar o *labelling* das variáveis. Esta estratégia é um método completo de busca, garantindo que a solução obtida é ótima. Há casos, porém, em que esta busca é excessivamente cara. No entanto, a Programação por Restrições, quando integrada com a Busca Tabu, pode aumentar sua eficiência e anular a falta de flexibilidade desta última. Além disso, essa integração pode permitir que vizinhanças de larga escala sejam exploradas de forma eficiente.

3.6 Conceitos básicos sobre escalonamento

Nesta seção apresentaremos alguns conceitos básicos que são aplicados em problemas de escalonamento e que serão utilizados neste trabalho. É fácil perceber a relação entre o problema tratado neste trabalho e os problemas de *Job Shop (JSP)*. O JSP pode ser formulado da seguinte forma: é dado um conjunto M de máquinas, um conjunto J de *jobs* e a cada *job* $j \in J$ está associada uma coleção ordenada de operações $t_k[j], 1 \leq k \leq n_j$, onde n_j é o número de operações associadas ao *job* j. A cada operação t (t é uma abreviação para $t_k[j]$) estão associadas uma duração $l(t)$ e uma máquina $m(t)$. O problema então é encontrar um escalonamento para as operações t de forma a otimizar uma dada função objetivo. Este escalonamento deve respeitar a ordem das operações nos *jobs* e também não pode permitir que uma mesma máquina execute operações simultaneamente. A relação entre o JSP e o problema deste trabalho pode ser estabelecida observando que os poços de petróleo correspondem aos *jobs* e o conjunto de atividade dos poços são o conjunto de operações de cada *job*. A relação de precedência entre as atividades corresponde à ordem das operações nos *jobs*. No caso do JSP esta ordem é total. No entanto, para o problema tratado neste trabalho, isso pode não ocorrer entre as atividades dos poços. Os barcos e as sondas são as máquinas que executam as operações.

3.6.1 Grafos Disjuntivos

Grafos disjuntivos (DG) foram propostos por Roy e Sussmann [58] para modelar e resolver problemas de Job Shop. Balas [7] foi o primeiro autor a explorar extensivamente as propriedades deste grafo para resolver problemas de escalonamento.

Um exemplo de grafo disjuntivo para o JSP com 4 *jobs* e 3 máquinas é mostrado na figura 3.1. Cada *job* possui 3 operações. Nesta figura, os arcos cheios são chamados de conjuntivos. Eles representam a precedência tecnológica entre as operações. As linhas tracejadas são os arcos disjuntivos. Estes arcos representam a ordem das operações nas máquinas. Se duas operações serão executadas pela mesma máquina, é acrescentado um arco disjuntivo com dupla orientação entre estas operações. Quando a ordem das operações nas máquinas é estabelecida, então estes arcos passam a ter sentido único, indicando qual a relação de precedência entre as operações da mesma máquina.

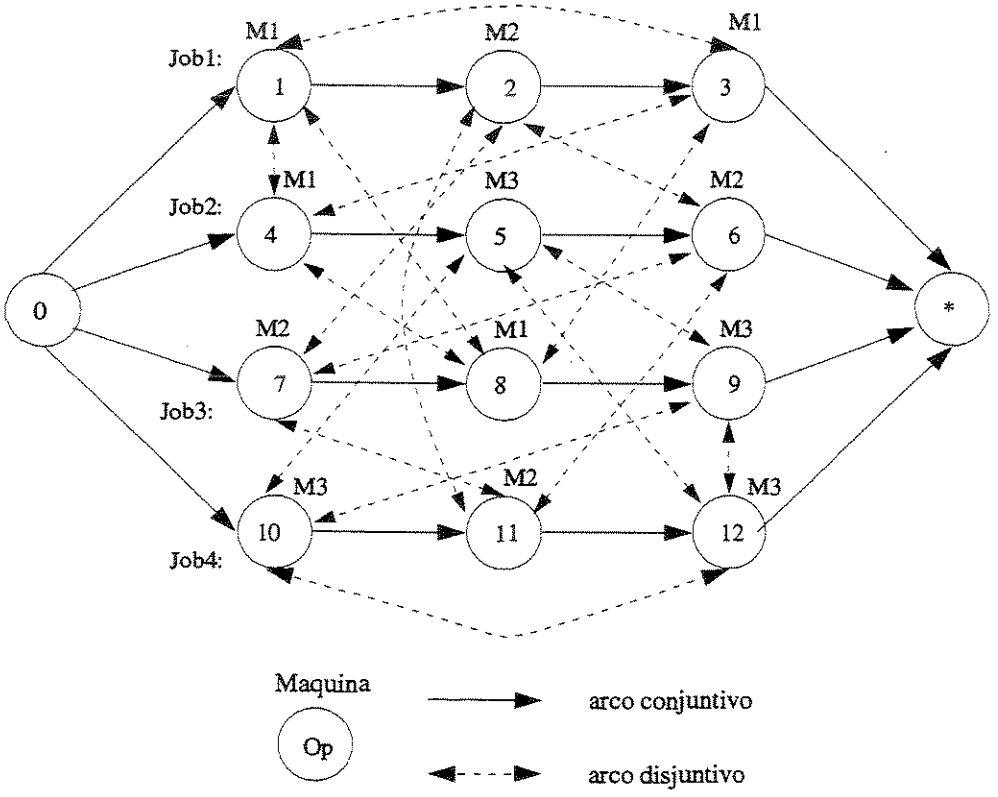


Figura 3.1: Grafo Disjuntivo

Neste trabalho, os grafos disjuntivos são utilizados para representar soluções e vizinhos. As atividades são os nós do grafo disjuntivo, a precedência tecnológica entre as atividades é

representada pelos arcos conjuntivos e a ordem das atividades nos recursos é representada pelos arcos disjuntivos. Se um grafo acíclico é obtido após a adição e a orientação dos arcos disjuntivos, então a correspondente solução ou o correspondente vizinho é viável e é possível determinar o instante de início das atividades. Se os arcos estabelecem uma ordem total entre as atividades, o instante de início pode ser obtido em tempo polinomial, utilizando-se um algoritmo de ordenação topológica [15]. Por outro lado, se uma ordem total não é estabelecida, o problema de atribuir os instantes de início às atividades de forma a maximizar a produção é NP-difícil. O próximo capítulo apresenta um teorema que prova este fato.

3.6.2 Tipos de Escalonamento, Medidas Regulares de Desempenho, Conjuntos Dominantes

Baker [6] define medidas regulares de desempenho e conjuntos dominantes para estas medidas. Ele também define três tipos de escalonamento: *semiativo, ativo* e *sem atraso*. Com estas definições é possível esclarecer porque somente o instante mais cedo possível de início das atividades deve ser considerado para maximizar a vazão, quando há uma ordem total entre as atividades.

Um escalonamento é dito *semiativo* se dada a ordem das operações nas máquinas e a ordem total das operações nos *jobs*, nenhuma operação pode ser adiantada sem violar a ordem estabelecida. Um escalonamento é *ativo* se nenhuma operação pode ser começada mais cedo sem atrasar outras operações. Finalmente, um escalonamento é *sem atraso* se nenhuma máquina é deixada ociosa quando há operações disponíveis para serem executadas naquela máquina.

Para ilustrar o conceito de tipos de escalonamento considere o seguinte exemplo. Seja uma instância do JSP com 3 máquinas, 4 *jobs* e 3 operações associadas a cada *job*. A função objetivo é minimizar o *makespan*. A tabela 3.1 descreve o tempo de processamento e a máquina alocada para cada uma das operações Op_{ij}, onde i representa o *job* i e j é a j-ésima operação do *job* i.

Job_i	Tempo de Processamento			Máquina		
	Op_{i1}	Op_{i2}	Op_{i3}	Op_{i1}	Op_{i2}	Op_{i3}
1	4	3	2	1	2	3
2	1	4	4	2	1	3
3	3	2	3	3	2	1
4	3	3	1	2	3	1

Tabela 3.1: Tempo Processamento e Máquinas Alocadas

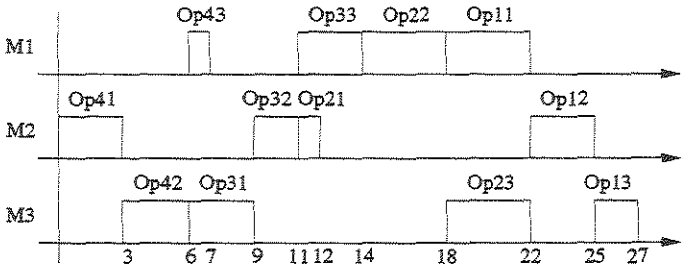
A figura 3.2.a mostra o escalonamento *semiativo* obtido quando a seqüencia de execução dos *jobs* em todas as máquinas é $4-3-2-1$. Note que nenhuma operação pode ser iniciada mais cedo respeitando a ordem estabelecida e sem atrasar nenhuma outra operação. No entanto, é possível obter um escalonamento melhor, adiantando a execução de algumas operações desde que se altere a ordem estabelecida. Mas, este escalonamento deve ser obtido sem o atraso na execução de nenhuma das operações. As figuras 3.2.b, 3.2.c e 3.2.d mostram exemplos de tais escalonamentos. O escalonamento destas figuras é chamado de *ativo*, pois em cada um deles não é possível adiantar a execução de nenhuma operação, mesmo que alterações na ordem de execução sejam permitidas, sem aumentar o tempo de início de alguma operação. Estas figuras mostram que é possível obter mais de um escalonamento *ativo* a partir do mesmo escalonamento *semiativo*. Vale lembrar, entretanto, que para uma dada ordem de execução das operações o escalonamento *semiativo* é único. Observando as figuras 3.2.b e 3.2.c percebe-se que estes escalonamentos não são *sem atraso*. Já no primeiro caso, a máquina 1 fica ociosa quando poderia estar executando a operação Op_{33}. No segundo caso, a máquina 2, fica ociosa quando poderia executar a operação Op_{32}. Já a figura 3.2.d apresenta um escalonamento que é ao mesmo tempo *ativo* e *sem atraso*.

Uma medida de desempenho Z é *regular* se a função objetivo do escalonamento é minimizar (maximizar) Z e Z só pode aumentar (diminuir) se pelo menos uma operação tem seu instante de término (instante de início acrescido da duração) aumentado. De acordo com [6], um conjunto D é dominante para medidas regulares de desempenho, se apenas soluções que estão neste conjunto devem ser consideradas na busca de uma solução ótima. No caso de problemas de Job Shop é mostrado que os escalonamentos *semiativos* e *ativos* dominam o conjunto de todos os escalonamentos se a função objetivo é uma medida regular de desempenho [6].

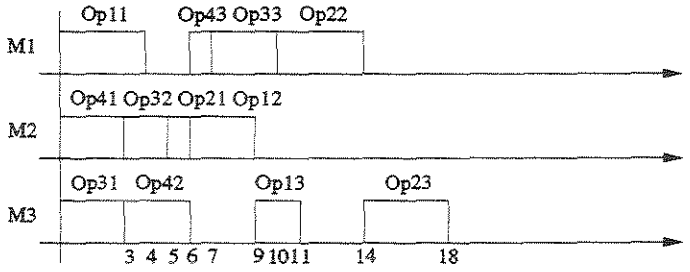
Para o problema que estamos tratando neste trabalho, também é possível mostrar que a vazão é uma medida regular de desempenho e que para se encontrar uma solução que a maximize, podemos nos ater aos escalonamentos *ativos* e *semiativos*. Para tal, seja C_i a variável que indica o instante de término de cada atividade i. Considere também $prod_i$. Esta constante é zero se a atividade i não coloca seu poço e produção. Por outro lado, o valor de $prod_i$ é a vazão do poço associado à atividade i, se esta coloca o poço em produção. Então a função objetivo pode ser escrita como:

$$\max \sum_{i=1}^{n} \max(0, (horizonte - C_i)) \times prod_i,$$

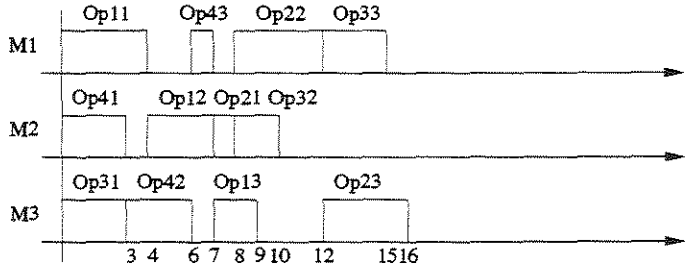
onde n é o número de atividades e *horizonte* é o horizonte de tempo especificado. É fácil perceber que o valor desta função só é diminuído se pelo menos uma das atividades que colocam poços em produção tem seu instante de término aumentado. Logo, pode-se



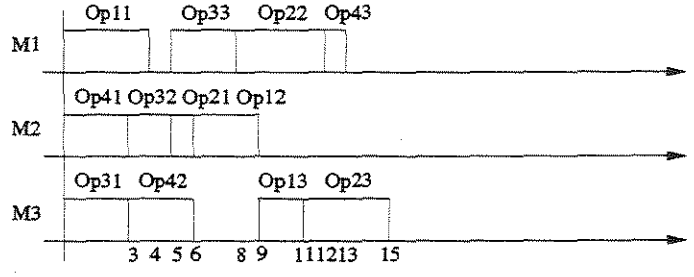
(a) Escalonamento semiativo



(b) Escalonamento ativo



(c) Escalonamento ativo



(d) Escalonamento ativo e sem atraso

Figura 3.2: Exemplo de tipos de escalonamento

afirmar que a maximização da produção de petróleo é uma medida regular de performance.

3.7 Técnica Híbrida - Busca Tabu e Programação por Restrições

Neste trabalho, a Programação por Restrições será integrada com a Busca Tabu para percorrer vizinhanças de larga escala. Como será detalhado no próximo capítulo, determinar o melhor vizinho nas vizinhanças consideradas neste trabalho é um problema NP-difícil. A Programação por Restrições será então a ferramenta utilizada para explorar estas vizinhanças eficientemente.

Em um primeiro momento, foi utilizado um modelo de Programação por Restrições para percorrer vizinhos, verificar vizinhos tabu, testar a factibilidade dos vizinhos e determinar o instante de início das atividades nos poços. Mas esta abordagem mostrou-se pouco eficiente para o problema em questão.

Então a estratégia foi alterada, retirando-se do âmbito da Programação por Restrições as tarefas de percorrer os vizinhos e de verificar se os mesmos são tabu. Além disso, como os vizinhos são representados utilizando-se grafos disjuntivos, um primeiro teste de factibilidade de cada vizinho é executado, verificando se o grafo correspondente é acíclico. Esta última tarefa também é realizada fora do âmbito da Programação por Restrições. Estes passos foram retirados da Programação por Restrições pois, cada vez que a Programação por Restrições é acionada dentro do algoritmo de Busca Tabu, incorre-se um *overhead* considerável. Sem contar que, muitas vezes, a Programação por Restrições poderia estar sendo acionada para vizinhos facilmente detectáveis como sendo tabu ou infactíveis, resultando em código ineficiente. O papel da Programação por Restrições é então, dado o grafo disjuntivo acíclico que representa cada vizinho, determinar o instante de início das atividades, de forma a maximizar a produção total de petróleo.

Capítulo 4

Modelagem

Este capítulo trata detalhes da modelagem do problema descrito no capítulo 2, como a geração de soluções iniciais, as vizinhanças adotadas, a utilização da Programação por Restrições para determinar o instante de início das atividades, e também os parâmetros da Busca Tabu. Por fim, é descrita uma abordagem de Busca Tabu pura, que também será utilizada para solucionar o problema. O objetivo é comparar o desempenho da técnica híbrida com o desempenho da Busca Tabu aplicada isoladamente.

4.1 Solução Inicial

O método de Busca Tabu exige uma solução de partida viável para que a Busca Local seja iniciada. Esta seção descreve quatro heurísticas utilizadas para gerar soluções iniciais para o problema.

As duas primeiras heurísticas são baseadas em Programação por Restrições e foram implementadas usando os *softwares ILOG Scheduler* e *ILOG Solver*[1]. Estas heurísticas serão chamadas de **CP1** e **CP2**, respectivamente.

As variáveis e as restrições consideradas em **CP1** e **CP2** são as mesmas. Um conjunto de variáveis é associado aos recursos que irão executar as atividades e um outro conjunto de variáveis refere-se ao instante de início das atividades. Como descrito na seção 2.1, somente as restrições $1, 4, 5$ e 6 são consideradas. Não foi dispendido muito esforço para refinar os modelos de Programação por Restrições, uma vez que o objetivo deste trabalho estava centrado nos algoritmos de solução dos modelos principais.

As outras duas heurísticas são baseadas em estratégias gulosas e serão chamadas **H1** e **H2**. Os algoritmos que implementam **H1** e **H2** foram muito mais rápidos do que aqueles usados para resolver **CP1** e **CP2**. Além disso, como será discutido no capítulo 6, em

[1]Softwares da suíte ILOG. http://www.ilog.com

todas as instâncias consideradas, a produção de óleo obtida utilizando **H1** foi maior do que aquela obtida pelas outras heurísticas.

4.1.1 Heurística CP1

A questão relevante desta heurística é o modo utilizado para instanciar as variáveis. Primeiramente, todas as variáveis que representam os recursos são instanciadas e depois é determinado o valor das variáveis que representam o instante de início de cada atividade.

A ordem utilizada para selecionar a próxima variável que será instanciada é a mesma para os dois conjuntos de variáveis (recurso e instante de início). Esta ordenação é baseada na seguinte regra: é escolhida prioritariamente a variável que representa a atividade com o maior número de sucessores diretos. Em caso de empate são escolhidas as variáveis que representam as atividades com o maior número total de sucessores e a maior duração, nesta ordem. Se o empate persistir, a escolha é feita de forma arbitrária.

O valor atribuído às variáveis é escolhido seguindo algoritmos internos da ferramenta *ILOG Scheduler*. Devido a limitações deste software, neste modelo, todas as atividades devem ser executadas antes do horizonte especificado na entrada.

4.1.2 Heurística CP2

Como na heurística anterior, a informação relevante é o modo como é feita a instanciação das variáveis. Novamente, as variáveis que representam os recursos são instanciadas antes das variáveis que representam o instante de início. Além disso, a ordem de escolha das variáveis é a mesma para os dois conjuntos.

A regra de prioridade para selecionar as variáveis é baseada na produção estimada do poço correspondente da atividade e também no número total de sucessores da atividade. Estes dois valores são somados e a variável que representa a atividade com a maior soma é escolhida prioritariamente. A produção de cada poço é estimada considerando que todas as atividades do poço são executadas seqüencialmente e sem interrupções. Além disso, considera-se que o poço começa a produzir quando a última atividade da seqüência é concluída. Como os valores obtidos para a produção são iguais ou diferem por um fator muito grande, o número total de sucessores é utilizado para desfazer empates. Vale ressaltar que esta ordenação é decidida no início da instanciação, e não é alterada dinamicamente.

O valor escolhido depende do tipo da variável. Para variáveis que representam recursos, o valor atribuído é o recurso compatível com o tipo da atividade e que, até o momento da instanciação, tem o menor número de atividades designadas. Para variáveis que representam o instante de início das atividades, é atribuído o menor valor do domínio corrente da variável.

4.1.3 Heurística H1

Esta heurística procura finalizar primeiramente os poços que apresentam maior vazão ou que apresentam o menor tempo remanescente (tempo total de processamento de todas as atividades remanescentes do poço) até a sua entrada em produção. Para atingir este objetivo, a cada momento a heurística verifica se há recursos disponíveis e se há atividades que podem ser executadas por estes recursos. Se isto ocorre, um recurso disponível é alocado para uma atividade viável e o instante de início desta atividade é definido como o instante corrente. Se há mais que um recurso disponível, é selecionado primeiramente um recurso capaz de executar menos tipos de atividades. Se há mais que uma atividade viável de ser executada pelo recurso escolhido, é selecionada:

1. A atividade que tem o maior valor para

$$(horizonte - (tempoAtual + tempoRestante)) \times vazao$$

   onde *horizonte* é o limite de tempo especificado, *tempoAtual* é o instante que o recurso é liberado e que a atividade será iniciada, ou seja, o tempo corrente, *tempoRestante* é o tempo total de processamento de todas as atividades remanescentes do correspondente poço e *vazao* é a vazão associada ao poço;

2. A atividade com o maior número total de sucessores;

3. A atividade com maior duração.

A segunda e a terceira regra são utilizadas para desfazer empates.

Inicialmente, o algoritmo segue estas regras e aloca atividades a recursos até que todos os recursos estejam ocupados. São consideradas atividades viáveis no início do algoritmo as atividades sem predecessores. Sempre que uma atividade é concluída, seu recurso é liberado e todas as suas atividades sucessoras diretas tornam-se disponíveis para serem alocadas, desde que estas atividades não sejam sucessoras de outras atividades que ainda não foram alocadas. Então, o recurso liberado é alocado a uma nova atividade e o ciclo é repetido. O algoritmo termina quando todas as atividades estão alocadas.

4.1.4 Heurística H2

Esta heurística, seguindo *CP1* e *CP2*, primeiro aloca os recursos e depois define o instante de início das atividades. As atividades que pertencem a poços com maior vazão são consideradas prioritariamente tanto na alocação dos recursos, quanto na definição do instante de início. Empates são desfeitos arbitrariamente.

Para alocar os recursos, a heurística verifica, para cada atividade, se há atividades do mesmo poço que já foram alocadas para recursos. Se isto é verdadeiro e se entre os recursos alocados para atividades do referido poço, há recursos capazes de executar a atividade em questão, então a heurística escolhe um destes recursos arbitrariamente. Se uma das duas condições é falsa, a heurística escolhe aleatoriamente um recurso qualquer entre todos os recursos viáveis.

Depois que os recursos são alocados é necessário determinar o instante de início das atividades. Para isso, é utilizada uma estrutura de dados para armazenar todas as atividades disponíveis, isto é, todas as atividades cujas predecessoras já foram escalonadas. Inicialmente, esta estrutura contém todas as atividades sem predecessores. A cada iteração da heurística, uma atividade é retirada da estrutura de dados seguindo a mesma regra de prioridade da heurística **H1**. Então as sucessoras diretas da atividade selecionada são adicionadas à estrutura, desde que estas atividades não sejam sucessoras de outras atividades que ainda não foram escalonadas. O *tempoAtual* considerado para cada atividade é o instante mais cedo possível de iniciar a atividade, ou seja, é o instante de início que será atribuído à atividade. Este tempo depende do recurso alocado para executar a atividade e das outras atividades do mesmo poço que estejam sendo executadas. O algoritmo termina quando todas as atividades estiverem escalonadas. Isto acontece quando a estrutura de dados torna-se vazia.

4.2 Vizinhanças

Nesta seção, descreveremos três vizinhanças que serão consideradas na resolução do problema. Também será demonstrado um teorema que estabelece que o problema de determinar o instante de início das atividades em cada vizinho de forma a maximizar a produção total é NP-difícil.

4.2.1 Vizinhança1 - Inserção

Esta vizinhança é obtida considerando-se cada atividade e inserindo-se a mesma em todas as posições possíveis de todos os recursos capazes de executá-la. Esta atividade é chamada de *atividade principal*. A viabilidade de cada vizinho é testada verificando-se se o grafo disjuntivo que representa o vizinho é acíclico.

A figura 4.1.a mostra o grafo disjuntivo para a solução corrente. Na figura 4.1, os nós representam as atividades $(1, 2, 3, \ldots, 12)$, os arcos tracejados são os arcos disjuntivos, que representam a ordem das atividades nos recursos (R1, R2 e R3) e os arcos cheios são os arcos conjuntivos, que representam a precedência tecnológica entre as atividades. Suponha que a atividade 1 é a atividade principal. Quando esta atividade é inserida na

segunda posição do recurso 3, obtém-se o vizinho mostrado na figura 4.1.b. Este vizinho é viável e o instante de início das atividades pode ser determinado. Já quando a atividade principal é inserida n terceira posição do recurso 3, resulta no vizinho representado na figura 4.1.c. Ente vizinho é inviável, pois seu grafo disjuntivo apresenta um ciclo. Note que os arcos conjuntivos são os mesmos para todos os vizinhos, devido ao fato que eles representam a precedência tecnológica entre as atividades. Os arcos disjuntivos são os que mudam para cada vizinho, pois eles representam a posição das atividades nos recursos. Na figura, arcos disjuntivos redundantes não são mostrados. Claramente, esta vizinhança tem $O(n^2)$ vizinhos, onde n é o número de atividades.

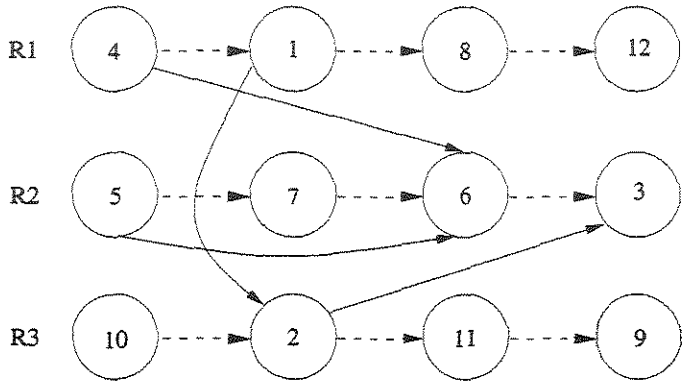
É importante observar que, em cada vizinho, os arcos disjuntivos determinam uma ordem total das atividades em cada recurso. Porém, a ordem total das atividades nos recursos não é suficiente para determinar um escalonamento ótimo para todas as atividades. Isto ocorre porque a ordem total das atividades nos poços pode não ser conhecida, mesmo levando-se em consideração os arcos conjuntivos, uma vez que podem existir atividades de um mesmo poço alocadas em recursos diferentes. Como será demonstrado a seguir, este fato torna a atribuição ótima do instante de início das atividades em cada vizinho um problema NP-difícil. Portanto, esta vizinhança é considerada de larga escala, porque, apesar do número polinomial de vizinhos, a exploração de cada vizinho é um problema NP-difícil. Por outro lado, se os arcos conjuntivos definem uma ordem total da atividades em cada poço, um escalonamento ótimo pode se encontrado em tempo $O(n)$, já que os escalonamentos *semiativos* formam um conjunto dominante para o problema, como discutido na seção 3.6.

4.2.2 A Redução

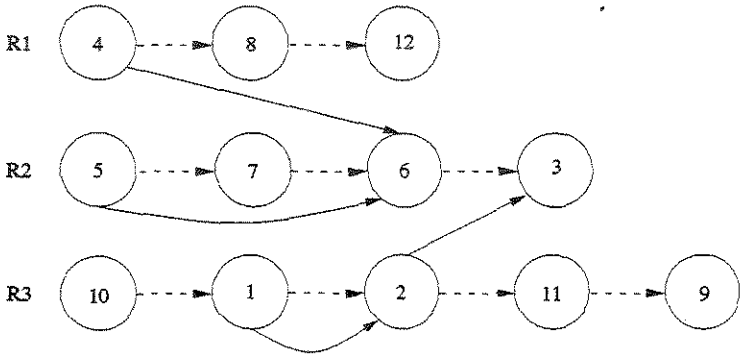
Dois problemas são considerados:

1. **Determinação ótima do instante de início das atividades (Vizinhança Inserção)**

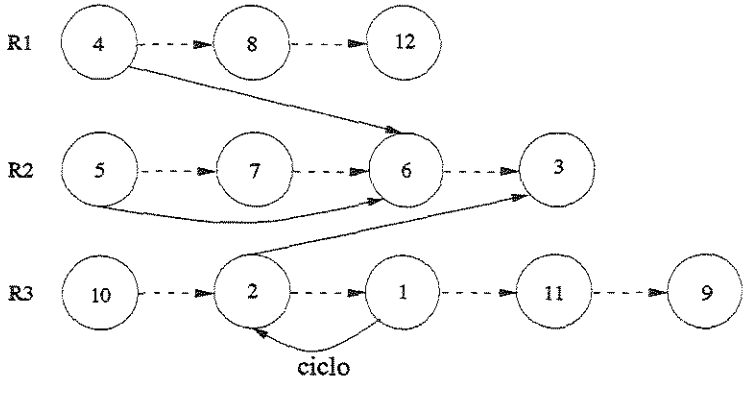
   - *Instância:* Produção total de petróleo $V \in Z^+$, horizonte de tempo $H \in Z^+$, conjunto P de poços, conjunto R de recursos. Para cada recurso $r \in R$ está associada uma coleção ordenada de atividades $a_k[r]$, $1 \leq k \leq n_k$, onde n_k é o número de atividades executadas pelo recurso r. A k-ésima atividade executada pelo recurso r é a atividade $a_k[r]$. Para cada atividade a (a é uma abreviação para $a_k[r]$) estão associados uma duração $l(a) \in Z_0^+$, um poço $p(a) \in P$, um tipo $t(a)$ e uma vazão $v(a)$. O tipo das atividades define uma ordem parcial entre as atividades do mesmo poço.



(a) Solução corrente



(b) Vizinho viável



(c) Vizinho inviável

Figura 4.1: Exemplo da vizinhança1 - Inserção

- *Questão:* Existe uma instanciação de tempo $\sigma(a) \in Z^+$ para as atividades de forma que a produção total de petróleo até o horizonte H é maior ou igual a V? A instanciação deve respeitar tanto a ordem parcial das atividades nos poços quanto a ordem das atividades nos recursos. Além disso, atividades do mesmo poço não podem ser executadas simultaneamente. A produção de petróleo é dada por:

$$\sum_{a \in A}(H - \sigma(a) - l(a)) \times v(a),$$

onde A é o conjunto que reúne todas as atividades.

2. **Job Shop Scheduling**

- *Instância:* Horizonte de tempo H, número $m \in Z^+$ de processadores, conjunto J de *jobs*, cada *job* $j \in J$ consistindo de uma coleção ordenada de operações $t_k[j], 1 \leq k \leq n_j$, onde n_j é o número de operações do *job* j. Para cada operação t (t é uma abreviação para $t_k[j]$) estão associados uma duração $l(t)$ e um processador $p(t)$, onde $p(t_k[j]) \neq p(t_{k+1}[j])$ para todo $j \in J$ e $1 \leq k < n_j$.
- *Questão:* Existe uma instanciação de tempo $\sigma(t) \in Z^+$ tal que: a ordem das operações nos *jobs* é respeitada, $\sigma(t) + l(t) \leq H$ para toda operação t, e não existem duas operações sendo processadas simultaneamente em um mesmo processador?

Teorema 4.1 *O problema da determinação ótima dos instantes de início das atividades (Vizinhança Inserção) é NP-difícil.*

Prova:

Claramente, o problema 1 pertence à classe NP. Como o problema 2 é NP-difícil [21], uma redução que transforma o problema 2 no problema 1, em tempo polinomial, prova que o problema 1 é NP-difícil.

Tome uma instância arbitrária do problema 2, representada pelo horizonte de tempo H, m processadores, pelo conjunto de *jobs* J e pela seqüência de operações associadas com cada *job*. Mapeando este problema no problema 1, cada *job* $j \in J$ passa a ser um recurso $r \in R$. O conjunto de operações associadas a cada *job*, passa a ser a coleção ordenada de atividades de cada recurso. O tempo de execução das operações $l(t)$ é mapeado no tempo de execução $l(a)$ de cada atividade e o processador associado a cada operação $p(t)$ passa a ser o poço associado a cada atividade $p(a)$. A cada atividade a é atribuído um tipo $t(a)$ que não possui relação de precedência com nenhum outro tipo. O horizonte do problema 1 é tomado como $H + 1$ e a produção total é definida por $V = 1$.

Além disso, serão criados um poço, um recurso e $|J|+1$ atividades *dummies*. O tempo de processamento destas atividades será zero e o poço associado a cada uma destas atividades é o poço *dummy*. O recurso alocado para a primeira atividade *dummy*, será o recurso 1, o recurso alocado para a segunda atividade será o recurso 2, e assim sucessivamente. Cada atividade *dummy* será colocada na última posição de cada recurso. O recurso *dummy* será alocado para a atividade *dummy* $|J|+1$. Note-se que é possível associar a cada atividade um recurso e uma posição neste recurso pois uma instância para o problema 1 fornece uma coleção ordenada de atividades para cada recurso. O tipo das primeiras $|J|$ atividades é **t1**, um novo tipo. O tipo da última atividade *dummy* é **produção**, também um novo tipo. A relação de precedência entre estes tipos é **t1** $\rightarrow$ **produção** . A vazão associada a cada atividade a será é zero, exceto pela última atividade *dummy*, cuja vazão é 1.

Portanto, a instância do problema 1, criada a partir de uma instância do problema 2, tem $|J|+1$ recursos, $m+1$ poços de petróleo e $(\sum_{j\in J} n_j)+|J|+1$ atividades.

A figura 4.2 mostra uma instância do problema 2 com 2 *jobs* e 3 processadores. Esta figura também mostra a instância obtida para o problema 1 usando a transformação descrita acima.

Como a atividade *dummy* associada a cada recurso é a última a ser executada pelo recurso, e dada a relação de precedência entre estas atividades e a atividade *dummy* $|J|+1$, pode ser concluído que a atividade *dummy* $|J|+1$ será a última atividade a ser executada considerando-se todas as atividades da instância. Portanto, por construção, a expressão que fornece a produção total de óleo é reduzida para

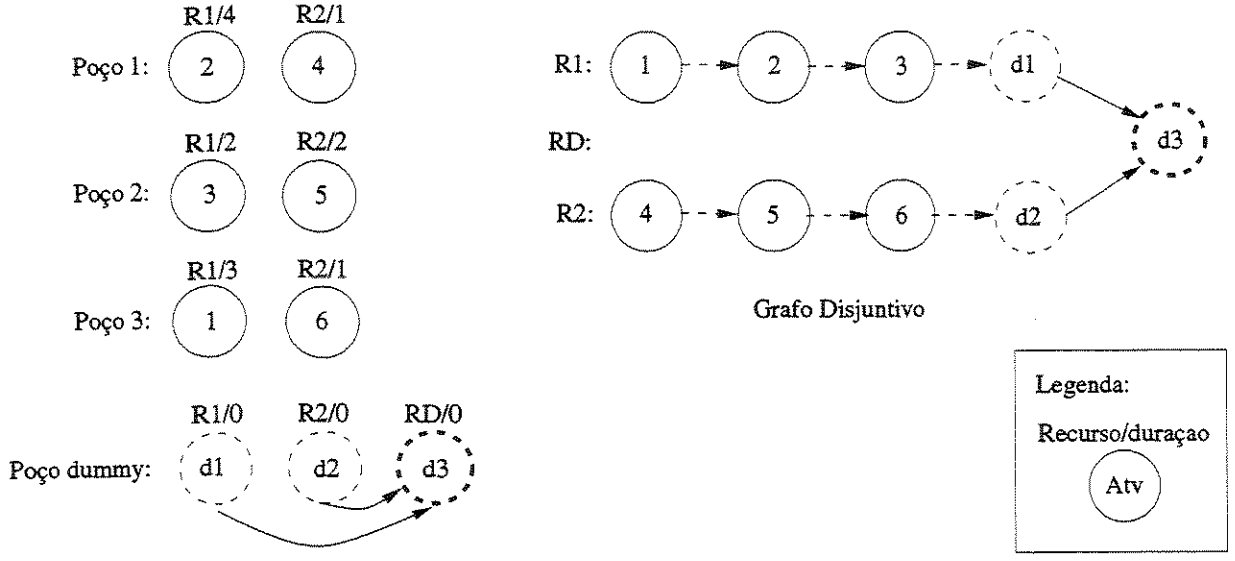
$$\max(0, H+1-\sigma(d)),$$

onde d é a atividade *dummy* $|J|+1$.

Como $V=1$, a produção total de óleo é igual ou maior que V se e somente se $\sigma(d) \leq H$. Suponha que a resposta para o problema 1 seja *sim*. Logo, $\sigma(d) \leq H$. Como $\sigma(a)+l(a) \leq \sigma(d)$ para toda a atividade a, então $\sigma(a)+l(a) \leq H$. Como as atividades do problema 1 correspondem às operações do problema 2, segue que $\sigma(t)+l(t) \leq H$ para todas as operações t. Logo, a resposta para a questão do problema 2 também será *sim*.

Suponha agora que a resposta para o problema 1 é *não*, ou seja, a produção total é sempre menor do que V. Neste caso, sempre vale $\sigma(d) > H$. Isto significa que não é possível terminar a execução de todas as atividades antes do horizonte. Como o tempo de execução atribuído a todas as atividades dummy é zero, existe uma atividade a', que não é uma atividade *dummy*, e que satisfaz $\sigma(a')+l(a') > H$. Mas, devido ao mapeamento entre os dois problemas, existe uma operação t' correspondente à atividade a' que também é terminada depois do horizonte do problema 2. Logo, a resposta para a questão do

(a) Instância do problema 2



(b) Instância correspondente do problema 1

Figura 4.2: Exemplo de redução

problema 2 também será *não*.

Claramente, o mapeamento entre os dois problemas pode ser realizado em tempo polinomial. Portanto, fica demonstrado que o problema 1 é NP-difícil.

□

4.2.3 Vizinhança2 - Janela

Esta vizinhança é uma generalização da vizinhança anterior. A motivação é gerar vizinhos capazes de realizar mais modificações na solução corrente do que os vizinhos da vizinhança anterior. Com isso, espera-se obter uma melhora mais significativa na produção total de petróleo. Para alcançar este objetivo, esta vizinhança considera dois conjuntos de atividades, chamados *janelas*. Também é considerada uma *atividade principal*. As janelas são escolhidas de acordo com esta atividade. As atividades que compõem as janelas serão liberadas das posições relativas nos seus recursos, mas o recurso permanece o mesmo. As novas posições destas atividades nos recursos serão determinadas implicitamente, quando o instante de início de cada atividade for definido.

A idéia principal não é apenas inserir uma única atividade em todas as possíveis posições de cada recurso capaz de executar a atividade, mas, além disso, liberar a posição corrente das atividades que estão em uma janela. A atividade que tem seu recurso alterado é a *atividade principal*. Esta atividade será alocada para todos os recursos capazes de executá-la, mas a sua posição no novo recurso, chamado *recurso destino* não será especificada. Além disso, esta atividade determina o centro de uma janela. Um parâmetro é utilizado para estabelecer o tamanho da mesma. Este parâmetro indica quantas atividades consecutivas serão liberadas de cada lado da atividade principal. Se o parâmetro é zero, somente a atividade principal será considerada na janela. Atividades alocadas no recurso de origem da atividade principal e que são escalonadas anteriormente a ela, são consideradas atividades que estão à esquerda da atividade principal. Da mesma forma, as atividades que estão à direita da atividade principal são as atividades alocadas no recurso original da atividade principal e que são escalonadas posteriormente à mesma. Se há menos atividades de um lado da atividade principal do que o parâmetro fixado para a janela, somente estas atividades serão consideradas para formar a janela. A ordem parcial entre as atividades que não pertencem à janela é preservada. Em outras palavras, os arcos disjuntivos das atividades que compõem a janela são não orientados no grafo que representa o vizinho. Além disso, os arcos disjuntivos relacionados à atividade principal na janela de origem são removidos, e novos arcos são inseridos entre a atividade principal e as atividades do recurso destino da atividade principal.

Uma janela também é considerada no recurso destino da atividade principal. Da

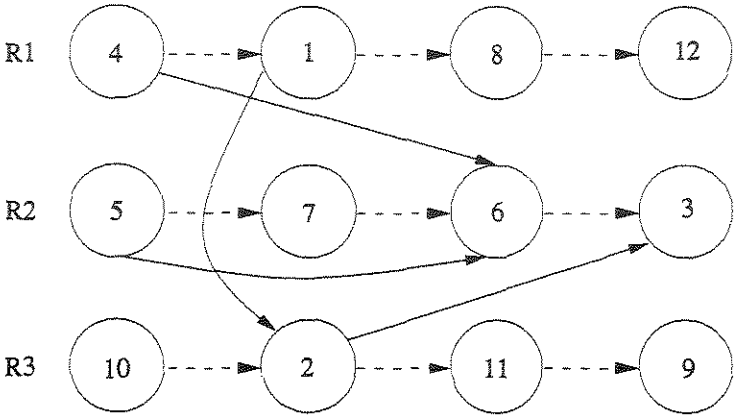
mesma forma que na janela descrita anteriormente, um parâmetro determina o tamanho desta nova janela. O centro desta janela é a atividade do recurso destino que está na mesma posição ocupada pela atividade principal no seu recurso de origem na solução corrente. Se a posição da atividade principal no seu recurso de origem é superior à maior posição do recurso destino, então a atividade que ocupa a maior posição deste recurso será considerada como o centro da janela. Neste caso, o lado direito da janela será nulo. As atividades que compõem esta janela, como na janela anterior, serão liberadas das suas posições correntes. Isto significa que os arcos disjuntivos correspondentes a estas atividades serão transformados de orientados para bi-direcionados. Se o parâmetro que determina o tamanho desta janela é zero, então esta janela não será considerada.

A figura 4.3.a ilustra o grafo disjuntivo de uma solução corrente. A figura 4.3.b ilustra as janelas obtidas quando a atividade principal é a atividade 1, no recurso R1. O recurso destino é o recurso R2. Ambos os parâmetros de janela estão fixados em 1. A figura 4.3.c mostra o grafo disjuntivo resultante do vizinho obtido indicado na figura 4.3.b.

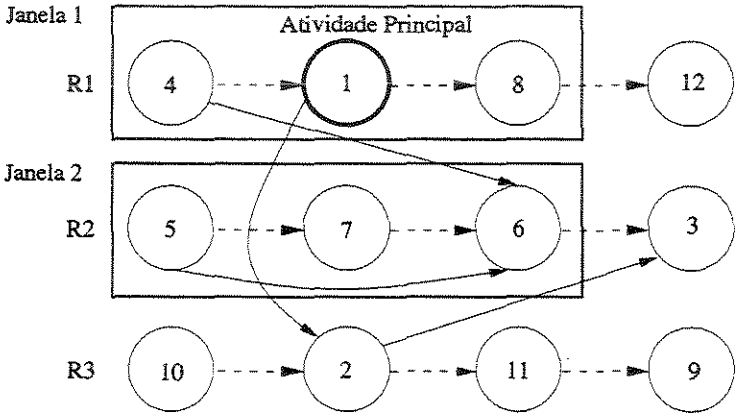
Note que, como na vizinhança *Inserção*, cada atividade será designada para todos os possíveis recursos capazes de executá-la, mas uma posição fixa não será especificada. Isto será estabelecido implicitamente quando os instantes de início das atividades forem instanciados. A outra diferença é que a posição das atividades que fazem parte das janelas nos seus respectivos recursos também não é conhecida. Obviamente, se o parâmetro das duas janelas é zero, a diferença entre esta vizinhança e a anterior é que, na anterior, a posição da atividade principal é fixada no recurso destino. Nesta vizinhança, a atividade principal pode ocupar qualquer posição neste recurso, ou seja, um vizinho desta vizinhança representa vários vizinhos da vizinhança *Inserção* simultaneamente.

É fácil ver que os vizinhos são definidos pela atividade principal e pelo recurso destino da mesma. Isto acontece pois a escolha das janelas depende apenas da atividade principal e do seu recurso destino, já que o tamanho das janelas é definido por parâmetros que não variam. Portanto, esta vizinhança tem $O(nm)$ vizinhos, onde n é o número de atividades e m é o número de recursos.

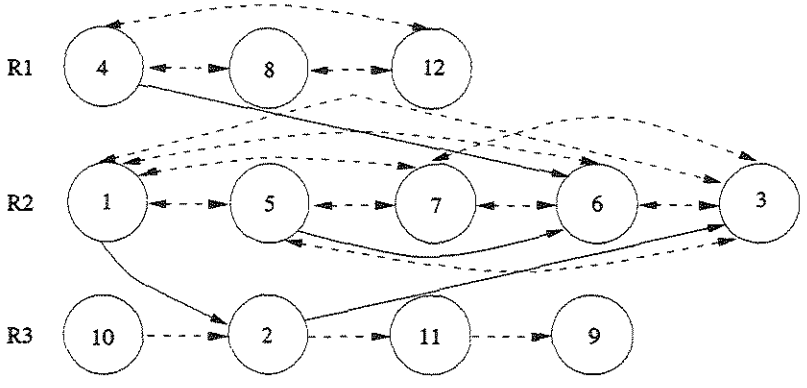
Na vizinhança anterior, a posição das atividades em cada recurso já era conhecida antes que os instantes de início fossem determinados. Entretanto, mesmo considerando este fato, é um problema NP-difícil determinar o escalonamento ótimo das atividades para cada vizinho da vizinhança *Inserção*. Portanto, determinar o escalonamento ótimo das atividades em cada vizinho desta nova vizinhança é um problema no mínimo tão difícil quanto o problema correspondente da vizinhança anterior.



(a) Solução corrente



(b) Atividade principal e janelas quando o parâmetro das duas janelas é 1



(c) Grafo disjuntivo representando o vizinho

Figura 4.3: Exemplo da vizinhança2 - Janela

4.2.4 Vizinhança3 - Poço

A idéia desta vizinhança vem da intuição de que as atividades de um mesmo poço devem ser escalonadas próximas uma das outras nas soluções que apresentam uma alta produção total de petróleo.

Nesta vizinhança, também é considerada uma *atividade principal*, que é a atividade que muda do seu recurso original para todos os recursos capazes de executá-la. Mas, ao invés de liberar as posições de atividades que fazem parte de janelas, esta vizinhança libera a posição das atividades que estão no mesmo poço que a atividade principal, preservando contudo a alocação dos recursos.

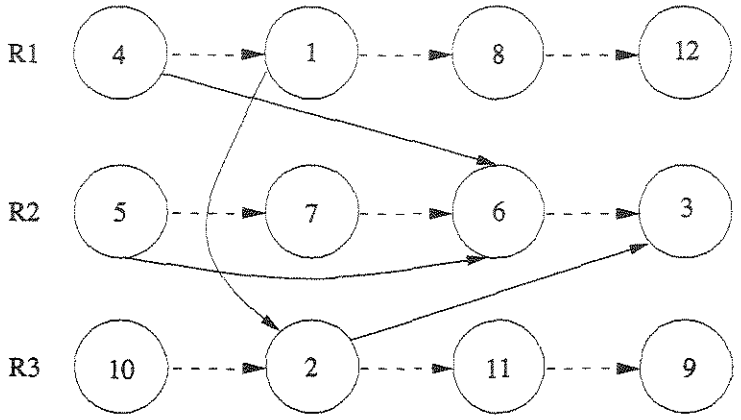
As atividades de outros poços também podem ser liberadas das suas respectivas posições. Um parâmetro é utilizado para estabelecer o número de poços que terão as suas atividades liberadas. Quando este parâmetro é 1, apenas as atividades do poço da atividade principal são liberadas. Se o parâmetro é maior do que 1, a regra de prioridade para liberar os poços é: considerando a solução corrente, para cada poço, determina-se a maior distância (diferença de instante de início) entre duas atividades consecutivas do mesmo poço. Então, multiplica-se esta distância pela vazão do poço correspondente. Quanto maior este valor, maior a prioridade do poço. Empates são desfeitos arbitrariamente. Utilizando esta regra, serão selecionados prioritariamente os poços cujo escalonamento das atividades é mais disperso e que têm uma vazão alta. Portanto, prioriza-se concentrar a execução das atividades destes poços com o intuito de aumentar a produção total.

A figura 4.4.a mostra o grafo disjuntivo de uma solução corrente. A figura 4.4.b indica a atividade 1 como a principal, no recurso de origem R1. As atividades 2 e 3 fazem parte do mesmo poço que a atividade principal. O grafo disjuntivo obtido quando o recurso de destino atividade principal é o recurso 2 é ilustrado na figura 4.4.c. O parâmetro que estabelece o número de poços liberados é 1. Como as atividades 2 e 3 pertencem ao mesmo poço que a atividade principal, estas atividades são liberadas das suas posições atuais.

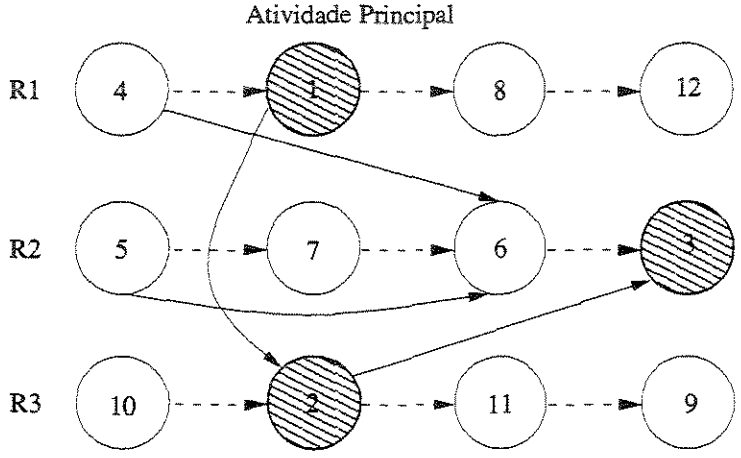
Como na vizinhança *Janela*, esta vizinhança tem $O(nm)$ vizinhos. Novamente, determinar o escalonamento ótimo das atividades para cada vizinho é um problema NP-difícil. Isto ocorre porque este problema é pelo menos tão difícil quanto o problema correspondente da vizinhança *Inserção*, já que, aqui, a posição nos recursos de algumas atividades não é conhecida antes da instanciação do tempo de início das atividades.

4.3 Instanciação de tempo nos vizinhos

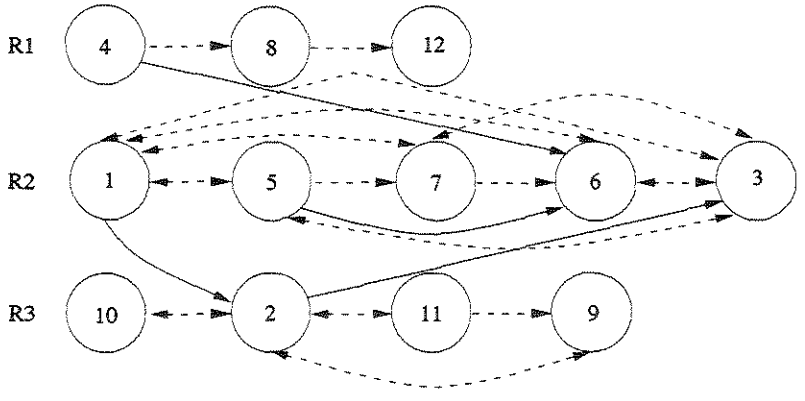
Duas técnicas foram utilizadas para determinar o instante de início das atividades em cada vizinho. Umas das técnicas é chamada *Gulosa* e a outra é chamada *Otimizada*. A primeira técnica é uma estratégia gulosa enquanto a segunda utiliza a Programação por



(a) Solução corrente



(b) Atividade principal e demais atividades que serão liberadas



(c) Grafo disjuntivo representando o vizinho

Figura 4.4: Exemplo da vizinhança 3 - Poço

Restrições para realizar a instanciação de tempo.

4.3.1 Estratégia gulosa

Esta técnica foi implementada em C++ sem utilizar a Programação por Restrições. Um *heap* S é utilizado para armazenar as atividades que podem ser escalonadas, ou seja, armazenar as atividades cujas predecessoras já foram escalonadas. A ordem de prioridades em S é baseada no instante mais cedo possível de início das atividades. Quanto mais cedo o instante de início, maior a prioridade. É utilizado um algoritmo que gera escalonamentos *sem atraso* [6] e a cada iteração deste algoritmo é selecionada a atividade que está no topo do heap. Vale lembrar que quando a instanciação de tempo é realizada, os recursos já estão alocados às atividades.

Inicialmente, S contém todas as atividades sem predecessores. A cada iteração, a atividade que está no topo de S é removida e é atribuído como seu instante de início o tempo mais cedo permitido pelo seu recurso e pelas atividades que pertencem ao mesmo poço que a atividade em questão. Em seguida, todas as sucessoras da atividades são inseridas em S, desde que não sejam sucessoras de atividades que ainda não tiveram o instante de início atribuído. O algoritmo é encerrado quando S é esvaziado. Isto acontece quando todas as atividades já estão escalonadas.

4.3.2 Estratégia Otimizada

Esta técnica é baseada em Programação por Restrições e foi implementada utilizando o *Ilog Solver*. As restrições do modelo são:

1. restrições de precedência tecnológica;
2. ordem das atividades nos recursos;
3. restrições que estabelecem que atividades do mesmo poço ou que estejam alocadas para o mesmo recurso não podem ser executadas simultaneamente.

Uma das estratégias mais comuns de *branching* para problemas de escalonamento que têm o *makespan* como função objetivo é chamado *Settling Essential Conflicts (SEC)* [37, 5]. Esta estratégia determina uma ordem de escalonamento para todas as atividades, ou seja, ela determina a orientação dos arcos disjuntivos até que todos estejam direcionados. Em seguida, dada a ordem total que foi construída orientando os arcos, um algoritmo de ordenação topológica determina o instante de início das atividades, gerando um escalonamento *semiativo*.

Esta estratégia é aplicada com sucesso quando a função objetivo do escalonamento é o *makespan*, em razão de propriedades do grafo disjuntivo [35] e também devido ao cálculo eficiente de limitantes inferiores para o problema [35, 42, 41]. Usando esta estratégia, em cada nó da árvore de enumeração vários arcos podem ser orientados ao mesmo tempo. Como a produção total de petróleo é a função objetivo aqui considerada e não há uma relação direta da vazão com o *makespan*, esta estratégia de *branching* não é adequada ao problema em estudo. Neste trabalho, as variáveis do modelo representam o instante de início das atividades e não a ordem entre duas atividades como ocorre na estratégia SEC. As restrições 1 e 2 são impostas antes da fase de *labelling*. Estas restrições são dadas pelos arcos disjuntivos que já estão orientados e pelo conjunto de arcos conjuntivos do grafo que representa o vizinho. Como é natural em qualquer modelo de Programação por Restrições, a imposição destas restrições ajuda a reduzir o domínio das variáveis.

Primeiramente, o mecanismo padrão de *labelling* do *ILOG Solver* foi utilizado. Como requerido pelo resolvedor, foram fornecidos parâmetros que estabelecem a função objetivo e o tempo máximo de execução do resolvedor. Além disso, também foi especificada uma regra de prioridade para definir a ordem de escolha das variáveis. A regra especificada foi: entre todas as variáveis ainda não instanciadas, escolha a variável que representa a atividade que tem o menor instante de início possível. Empates são desfeitos arbitrariamente. O valor atribuído à variável é o menor do seu domínio. Note que este procedimento padrão não leva em consideração o conhecimento que os escalonamentos *ativos* formam um conjunto dominante para este problema, como explicado na seção 3.6.

Mas há um problema ainda mais sério com este procedimento. O mecanismo padrão de *labelling* atua da seguinte forma: se uma variável a é instanciada antes que uma variável b, então todo o domínio de b deve ser explorado, antes que um novo valor seja atribuído a a. Um exemplo simples ilustra como esta característica pode ser danosa para a performance do procedimento padrão. A instância do exemplo é: dois poços, um conjunto de 5 atividades e 2 recursos. As atividades $1, 2$ e 3 pertencem ao poço 1, enquanto as atividade 4 e 5 pertencem ao poço 2. Todos os recursos são capazes de executar todas as atividades. Todas as atividades têm duração 5 e a relação de precedência entre elas é mostrada na figura 4.5. As atividades 3 e 5 são responsáveis por colocar os respectivos poços em produção. A vazão associada a ambos os poços é 10 e o horizonte é 50. Agora, suponha que em um vizinho, a ordem das atividades no recurso 1 seja $1, 4$ e 5 e no recurso 2 seja 2 e 3. Se a ordem de instanciação, seguindo a regra do menor instante inicial possível, é $2 - 1 - 4 - 3 - 5$, o instante de início das atividades é mostrado na tabela 4.1 e a produção total obtida é 650. Mas, se o instante de início das atividades fosse como ilustrado na tabela 4.2, a produção total seria 700. Mas para esta instanciação ser obtida, o algoritmo de *labelling* teria que instanciar a variável que representa a atividade 2 com os valores $1, 2, 3, 4, 5$. Pior, como a atividade 2 é a primeira na ordem de instanciação,

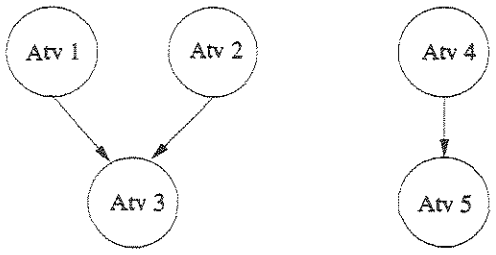


Figura 4.5: Restrição de precedência tecnológica

o domínio de todas as outras variáveis teria que ser completamente percorrido antes que um novo valor fosse atribuído à variável 2.

Atividade	Instante de início
1	5
2	0
3	10
4	10
5	15

Tabela 4.1: Primeira instanciação das variáveis

Atividade	Instante de Início
1	0
2	5
3	10
4	5
5	10

Tabela 4.2: Instanciação final das variáveis

Por causa deste comportamento, o mecanismo padrão de *labelling* foi substituído por um mecanismo especializado. Este novo mecanismo muda a ordem de instanciação das variáveis que representam o início das atividades, considerando apenas escalonamentos ativos. Para conseguir este comportamento, a instanciação das variáveis é realizada em duas fases. Em cada nível da árvore de enumeração, uma nova variável é criada para representar as atividades que podem ser instanciadas naquele nível da árvore. Estas

variáveis são chamadas de *variáveis de nível*. As atividades viáveis de cada nível (aquelas que estão no domínio da variável de nível), são as atividades com grau de entrada zero e que satisfazem a regra do escalonamento ativo. O grau de entrada de uma atividade é o número de atividades predecessoras da mesma, no grafo disjuntivo, e que ainda não foram escalonadas. O grau de entrada é dinamicamente atualizado durante todo o processo de busca. O grau de entrada inicial é dado pelos arcos orientados do grafo disjuntivo que representa cada vizinho. Cada passo do processo de busca consiste em escolher um valor do domínio da variável de nível em questão e atribuir o menor instante de tempo possível para a variável que representa o instante de início da atividade apontada pela variável de nível. Portanto, quando uma solução é encontrada ou quando uma falha ocorre, o *backtracking* é realizado sobre a variável de nível, permitindo que a ordem de instanciação das variáveis que representam o tempo de início seja alterada.

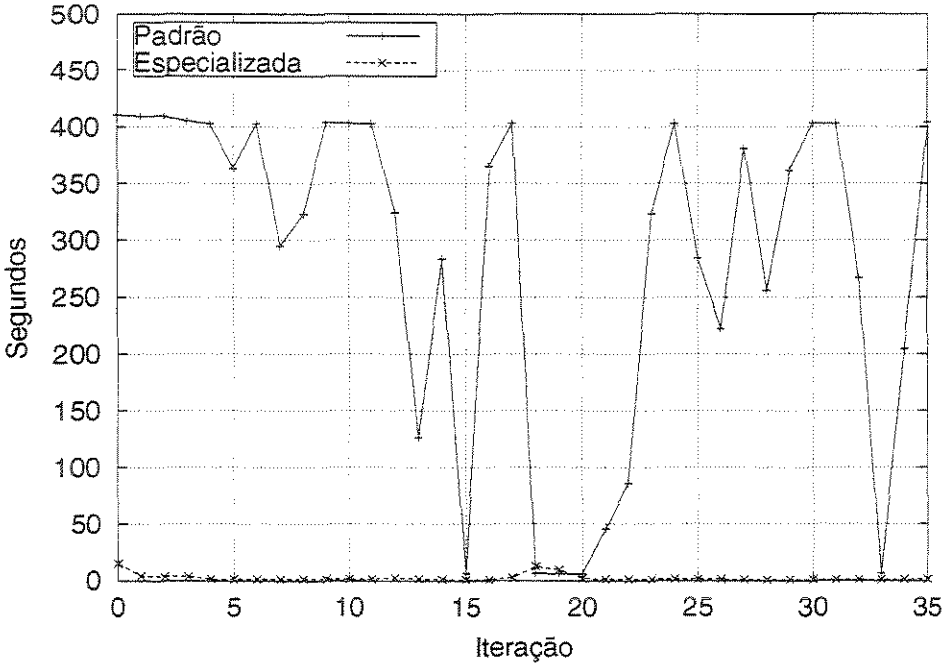
A figura 4.6 mostra o tempo de computação e a produção total obtida por um algoritmo de Busca Tabu que utiliza a vizinhança *Inserção*, aplicando os mecanismos de *labelling* padrão e especializado para uma mesma instância e solução inicial. O tempo máximo de execução permitido por iteração foi 410 segundos. Maiores detalhes sobre a Busca Tabu serão tratados na próxima seção. Analisando esta figura, é fácil perceber que o mecanismo especializado é aproximadamente 100 vezes mais rápido que o mecanismo padrão. Além disso, o mecanismo especializado foi capaz de encontrar a solução ótima em cada vizinho (fato não indicado na figura). Já o mecanismo padrão, na maioria das iterações, foi terminado por atingir o tempo máximo de execução permitido, sem achar uma solução ótima. Em vista desses resultados experimentais, somente a versão especializada será considerada quando a estratégia *Otimizada* for utilizada para determinar o instante de início das atividades.

4.4 Parâmetros da Busca Tabu

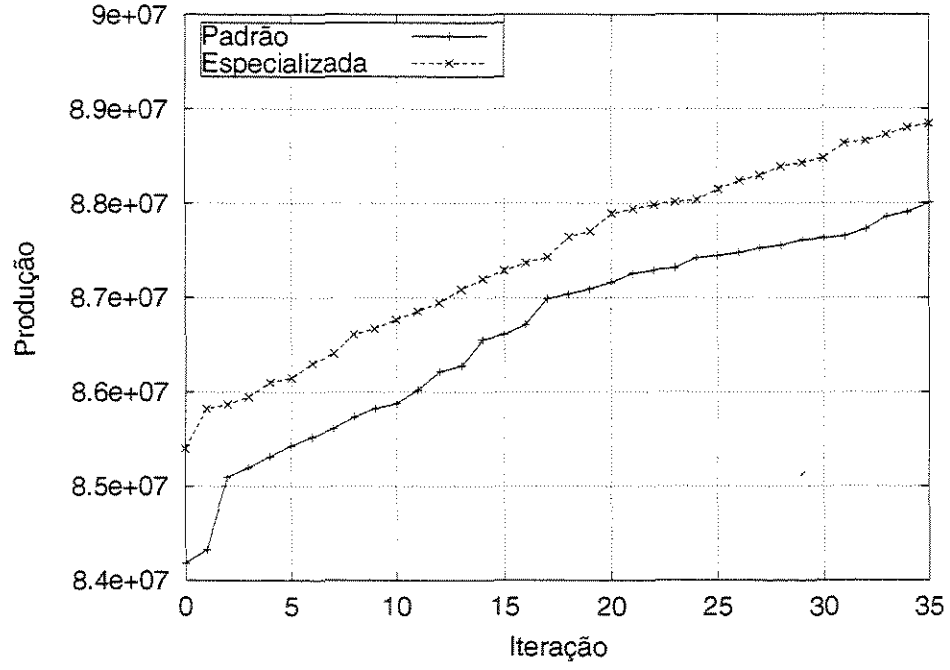
A seção 4.2 descreve a estrutura das vizinhanças utilizadas neste trabalho, e a seção 4.3 explica como o instante de início das atividades é determinado. Esta seção descreve, para cada vizinhança, os parâmetros da Busca Tabu. Estes parâmetros incluem o *critério de parada*, a *lista tabu*, o *critério de aspiração* e a *seleção dos vizinhos*.

4.4.1 Critério de Parada

O critério de parada para todas as vizinhanças é o tempo de execução. Isto torna mais justa a comparação entre o desempenho de cada uma da vizinhanças, pois não importa quantas iterações ou vizinhos cada uma delas explorou, depois da mesma quantidade de tempo, todas são finalizadas e os resultados comparados. Neste trabalho, o tempo máximo



(a) Tempo × Iteração



(b) Produção × Iteração

Figura 4.6: Comparação entre os mecanismos padrão e especializado

de execução é 3600 segundos.

4.4.2 Lista Tabu

Para as vizinhanças *Inserção* e *Janela* a atividade principal do vizinho selecionado é considerada tabu. Isto significa que esta atividade não pode ser a atividade principal enquanto a mesma é considerada tabu, a menos que o critério de aspiração seja satisfeito. Um vizinho é considerado tabu, se a sua atividade principal é tabu. Neste trabalho, uma atividade é considerada tabu por 50 iterações.

Para a vizinhança *Poço*, o poço da atividade principal é considerado tabu. Isto significa que as atividades deste poço não podem ser liberadas enquanto o poço é considerado tabu. Nesta vizinhança, não é permitido que um poço tabu faça parte do conjunto poços que compõem um vizinho, ou seja, do conjunto de poços que terão as suas atividades liberadas no vizinho. Nenhum critério de aspiração foi adotado. No capítulo 6, sempre que o número de vizinhos tabu é mencionado, este número significa quantas vezes foi proibida a entrada de um poço tabu no conjunto de poços que formam um vizinho. Neste trabalho, um poço é considerado tabu por 10 iterações.

4.4.3 Critério de Aspiração

O critério de aspiração para as vizinhanças *Inserção* e *Janela* diz que se a produção total de um vizinho considerado tabu for melhor do que a melhor produção já encontrada, então o vizinho é considerado, mesmo sendo tabu.

4.4.4 Seleção do Vizinho

A seleção do vizinho é diferente para a vizinhança *Inserção*, sendo comum para as outras duas vizinhanças. O critério de seleção será explicado separadamente para cada grupo de vizinhanças.

- *Vizinhança Inserção*

  Várias abordagens foram testadas para selecionar o vizinho. Primeiramente, a vizinhança inteira era explorada utilizando a estratégia *Otimizada* para determinar o instante de início das atividades. Nesta abordagem o melhor vizinho era selecionado. Como o tempo requerido para explorar cada vizinho era da ordem de 0.2 segundos, esta abordagem mostrou-se impraticável, pois, em uma instância real do porte daquela descrita na seção 2.2, há aproximadamente 250.000 vizinhos. Na segunda abordagem testada, toda a vizinhança era explorada utilizando-se a estratégia *Gulosa* para instanciar os tempos de início. Esta exploração levava em torno de 15

segundos, no total. Em seguida, os melhores x vizinhos eram explorados novamente utilizando a estratégia *Otimizada* e então o melhor vizinho era selecionado. Porém, testes computacionais mostraram que explorar apenas os y primeiros vizinhos que melhorassem a melhor solução era mais eficiente do que explorar toda a vizinhança [46].

Portanto, na terceira abordagem, a abordagem efetivamente utilizada neste trabalho, a vizinhança é explorada utilizando a estratégia *Gulosa* até que y vizinhos que melhorem a melhor solução encontrada até o momento sejam encontrados. Em seguida, dentre estes y vizinhos, são escolhidos x vizinhos, onde $x \leq y$, para serem explorados utilizando a estratégia *Otimizada*. O melhor vizinho obtido com a estratégia *Otimizada* é, então, selecionado. Duas estratégias distintas foram consideradas para escolher os x vizinhos que serão explorados pela técnica *Otimizada*. A primeira escolhe os x melhores vizinhos deterministicamente. Esta abordagem foi chamada de *ID*. A segunda escolhe os x vizinhos de forma aleatória ente os y vizinhos. Esta abordagem foi chamada de *IA*. Os valores escolhidos para y e x foram $y = 10$ e $x = 10$ na abordagem *ID* e $y = 20, x = 5$ na abordagem *IA*. A escolha destes valores foi baseada em testes computacionais preliminares.

- *Vizinhanças Janela e Poço*

  Para estas duas vizinhanças, a regra para selecionar o vizinho é a mesma. Em ambas as vizinhanças, não foi eficiente utilizar a estratégia *Gulosa* para selecionar vizinhos. Também não foi eficiente explorar toda a vizinhança, nem mesmo os y primeiros melhores vizinhos. Possivelmente a obtenção de boas soluções nestas vizinhanças mais complexas é um problema mais difícil do que o problema correspondente da vizinhança *Inserção*. Note que a técnica *Gulosa* nestas vizinhanças está operando com um número maior de arcos disjuntivos não orientados do na vizinhança *Inserção*. Isto pode ter causado a obtenção de soluções inferiores quando comparadas às soluções que poderiam ter sido obtidas se uma ordenação diferente das atividades fosse considerada.

  Por outro lado, o uso da estratégia *Otimizada* para achar a instanciação de tempo ótima também mostrou-se impraticável. Esta estratégia leva da ordem de 1 segundo para achar uma primeira solução para cada vizinho e um tempo bem maior para provar a otimalidade da solução encontrada.

  Foi decidido então adotar uma estratégia que usa a técnica *Otimizada* para explorar a vizinhança até que um vizinho que melhore a melhor solução seja encontrado, ou então até que o tempo de execução do resolvedor na vizinhança alcance 20 segundos. Além disso, a técnica *Otimizada* deve obedecer a um tempo limite de execução de 1 segundo em cada vizinho da vizinhança *Janela* e 3 segundos na vizinhança *Poço*.

Isto significa que a otimalidade de cada vizinho pode não ser comprovada. O vizinho selecionado é o melhor encontrado. Nos próximos capítulos, será chamada de *WI* a vizinhança *Janela* quando esta abordagem for utilizada. De forma similar, quando a vizinhança *Poço* for utilizada com esta abordagem, a mesma será chamada de *WE*.

Após a realização de alguns testes computacionais, o valor escolhido para o parâmetro que determina o tamanho da *Janela* foi 3 para a janela do recurso origem e 4 para o recurso destino. Já o valor escolhido para o parâmetro que determina o número de poços que compõem um vizinho da vizinhança *Poço* foi 90.

Como as vizinhanças não são inteiramente exploradas, pois a exploração é limitada tanto pelo número de vizinhos que melhoram a melhor solução quanto pelo tempo de execução, é essencial definir em que ordem os vizinhos serão percorridos. Para todas as vizinhanças, a ordem é a mesma. Considerando a solução corrente, para cada poço, determina-se a maior distância (diferença de instante de início) entre duas atividades consecutivas do mesmo poço. Então, multiplica-se esta distância pela vazão do poço correspondente. Em seguida, para cada atividade é somado a este valor, o número total de sucessores diretos da atividade. São explorados prioritariamente os vizinhos cuja atividade principal tem o maior valor para esta soma. Empates são desfeitos arbitrariamente.

4.5 Busca Tabu: Abordagem Pura

Nesta seção, será descrita uma abordagem pura da Busca Tabu. O objetivo é estabelecer uma comparação entre esta abordagem e as abordagens híbridas. A abordagem pura foi implementada por Vinicius Fortuna em seu projeto de iniciação científica [19].

4.5.1 Representação das Soluções e Vizinhos

Nesta abordagem soluções e vizinhos são representados por uma lista ordenada de atividades e por uma estrutura de dados que indica o recurso alocado a cada atividade.

Além disso, considera-se o grafo orientado G cujos nós representam as atividades e os arcos representam as precedências tecnológicas entre as atividades. Com o auxílio do grafo G, calcula-se o escalonamento representado por uma solução da seguinte forma:

1. Adiciona-se ao grafo G arestas (a_i, a_j) se a atividade a_i vem antes da atividade a_j na lista ordenada das atividades, e as atividades pertencem ao mesmo poço ou são executadas pelo mesmo recurso.

2. O instante de início das atividades é obtido aplicando-se um algoritmo de ordenação topológica em G.

Note que as arestas adicionadas ao grafo G garantem uma ordem total entre as atividades de um mesmo poço e entre as atividades alocadas ao mesmo recurso.

4.5.2 Vizinhança

Há dois movimentos que definem a vizinhança. O primeiro movimento consiste em mudar o recurso atribuído a uma determinada atividade para todos os recursos capazes de executar a atividade em questão. O segundo movimento consiste em retirar uma atividade da sua atual posição na lista ordenada de atividades e inserí-la em todas as possíveis posições da lista, preservando o recurso alocado a ela. Tanto a viabilidade quanto o escalonamento das atividades de cada vizinho é obtido através de uma ordenação topológica no grafo G que representa o vizinho. Esta vizinhança tem $O(nm + n^2)$ vizinhos, onde n é o número de atividades e m o número de recursos.

4.5.3 Parâmetros da Busca Tabu

Os critérios de parada e de aspiração utilizados por esta abordagem são os mesmos utilizados pelas abordagens híbridas, ou seja, a Busca Tabu é encerrada após 3600 segundos e um vizinho tabu que melhore a melhor solução encontrada, deixa de ser tabu.

O vizinho selecionado em cada iteração é o primeiro vizinho que melhora a melhor solução encontrada até o momento.

Duas estratégias foram adotadas para identificar movimentos tabu . A primeira considera tabu apenas o movimento utilizado para gerar vizinhos selecionados em iterações anteriores. Por exemplo, se o vizinho selecionado foi obtido mudando o recurso da atividade a_1 de r_1 para r_2, então é tabu o movimento que aloca r_1 para a atividade a_1. Esta estratégia é chamada *TabuPF*. A segunda estratégia considera grupos de movimentos tabu. Por exemplo, se o movimento que gerou vizinhos selecionados em iterações anteriores alterou o recurso de uma atividade, então todos os movimentos que alterem o recurso desta atividade são considerados tabu. Da mesma forma, se o movimento alterou a posição da atividade na lista ordenada, estão todos os movimentos que alterem a posição desta atividade são considerados tabu. Esta estratégia é chamada *TabuPR*. Um movimento ou um grupo de movimentos é considerado tabu por 25 iterações.

Capítulo 5

Limitantes Duais

Como não há resultados computacionais prévios para as instâncias do problema consideradas neste trabalho, é extremamente importante a obtenção de limitantes duais. Com estes limitantes é possível determinar a qualidade das soluções obtidas pelas diversas estratégias empregadas neste trabalho.

5.1 Abordagens

Esta seção descreve como foram calculados quatro limitantes duais para o problema. O primeiro limitante é obtido a partir de um argumento combinatório. A estratégia utilizada para obter este limitante é chamada *Upper0*. Já as outras três abordagens, chamadas *Upper1*, *Upper2* e *Upper3*, resolvem relaxações do problema original, utilizando para isso a Programação Linear Inteira.

5.1.1 Upper0

Este limitante é dado pela seguinte expressão:

$$\sum_{i=1}^{p}(H - \sum_{j \in Atv_i} d_j) \times v_i,$$

onde p é o número de poços, H é o horizonte, Atv_i é o conjunto de atividades do poço i, d_j é o tempo de processamento da atividade j e v_i é a vazão do poço i.

Esta expressão assume que há um número ilimitado de recursos capazes de executar qualquer uma das atividades. Como o número de recursos não está sendo considerado, e este número é muito limitado nas instâncias utilizadas neste trabalho, o limitante obtido com esta abordagem pode ser melhorado.

5.1.2 Upper1

Nesta abordagem, um modelo de Programação Linear Inteira é formulado para a obtenção do limitante. Este modelo considera o fato que há um número limitado de recursos. Por outro lado, supõe que todos os recursos disponíveis são capazes de executar todas as atividades. Ou seja, tem-se uma situação semelhante ao *ambiente de máquinas paralelas* [53].

O modelo considera que cada poço apresenta apenas uma única atividade, chamada de *super-atividade*. O tempo de processamento da super-atividade, denotado por Δ, é a soma do tempo de processamento de todas as atividades originais do poço. Após a execução da super-atividade, o poço entra em produção. Como a super-atividade representa todas as atividades do poço, e como esta é uma versão relaxada do problema original, é permitida a interrupção desta atividade. A interrupção pode ocorrer a qualquer instante durante o processamento da super-atividade, por um número ilimitado de ocorrências. A função objetivo é a mesma que a do problema original, ou seja, maximizar a produção total de petróleo. Esta versão relaxada do problema será chamada de $P||Production$.

No modelo desta abordagem, o valor de Δ será arredondado para o maior múltiplo de 5 que seja menor ou igual a Δ. Isto é feito porque o tempo neste modelo é discretizado agrupando-se blocos de 5 unidades de tempo do problema original. Ou seja, cada unidade de tempo no modelo representa 5 unidades de tempo no problema original. Testes computacionais mostraram que não é viável considerar modelos onde o tempo se mantenha discretizado nas unidades originais.

As variáveis do modelo são dadas por x_{it}, onde i representa um poço e t é um instante de tempo anterior ao horizonte. Cada x_{it} assume valor 1 se o poço i é finalizado no instante de tempo t e 0 caso contrário.

A função objetivo é:

$$\max \sum_{i=1}^{p} \sum_{t=0}^{H/5} 5(H/5 - t) v_i x_{it},$$

onde p, H e v_i têm o mesmo significado que na abordagem anterior, e assumimos que H é múltiplo de 5.

As restrições do modelo são:

1. $\sum_{t=0}^{H/5} x_{it} \leq 1, \quad i = 1 \ldots p;$

2. $\sum_{k=0}^{t} \sum_{i=1}^{p} x_{ik} \Delta_i / 5 \leq t \times m, \quad t = 0 \ldots H/5$, onde m é o número de recursos;

3. $x_{it} = 0$, para todo i e t tal que $\Delta_i > t$.

A restrição 1 estabelece que cada poço terá no máximo um instante de término. Como essa restrição está na forma de uma desigualdade, um poço pode não terminar antes do horizonte de tempo H. Se isto acontece, este poço simplesmente não é contabilizado na produção total de petróleo. Os termos execução/término do poço são utilizados com o mesmo significado que execução/término da super-atividade. Já a restrição 2 garante que há tempo suficiente nos recursos disponíveis para executar todos os poços que terminam até o instante t. Vale observar que é considerada a soma de tempo disponível em todos os recursos. Por isso, esta restrição supõe que uma super-atividade pode ser executada por mais de um recurso ao mesmo tempo. Um aspecto relevante é que, como a produção total está sendo maximizada, então todos os poços terminarão antes do horizonte, caso exista um escalonamento que o permita. A restrição 3 reduz o número de variáveis do modelo e o número de vezes que a super-atividade de cada poço é executada em paralelo por mais de um recurso. Isto é alcançado forçando que um poço não seja terminade antes do seu tempo de execução. Por exemplo, se esta restrição não for imposta, um poço com tempo de execução 6 pode terminar no instante 2, se houverem 3 recursos na instância em questão. No entanto, pode-se perceber que esta restrição só causa impacto nos poços que terminam no início do escalonamento, porque para valores suficientemente grandes de t, o mesmo sempre será maior que o tempo de execução dos poços.

5.1.3 Upper2

Esta abordagem é uma versão mais refinada da abordagem anterior. Em particular, não permite que um poço seja executado simultaneamente por mais de um recurso. A versão relaxada do problema considerada nesta abordagem é a mesma versão utilizada em *Upper1*. O modelo desta abordagem e da anterior só diferem na restrição 2. A modificação nesta restrição reflete o fato que, segundo o teorema 2 abaixo, interrupções são redundantes para o problema $P||Production$, isto é, o valor ótimo da função objetivo para o caso sem interrupção não pode ser melhorado se interrupções forem permitidas.

Teorema 5.1 *No problema $P||Production$, interrupções são redundantes.*

Prova: A prova é dividida em dois casos. O caso 1 ocorre quando o horizonte de tempo é longo o suficiente para que todos os poços terminem antes do mesmo. O caso 2 ocorre quando este fato não é verdadeiro.

- *Caso 1:* Neste caso, resolver o problema $P||Production$ é equivalente a resolver o problema $P||\sum w_j C_j$ (a notação $\alpha|\beta|\gamma$ de [29] é utilizada para representar o

problema de máquinas paralelas com o objetivo de minimizar a soma ponderada dos instantes de término). Um resultado clássico de McNaughton [44] mostra que para o problema $P||\sum w_j C_j$ interrupções são redundantes. Logo, como os problemas são equivalentes, interrupções também são redundantes para o problema $P||Production$.

- *Caso 2:* Considere uma solução ótima para $P||Production$ quando interrupções são permitidas. Seja S o conjunto de poços que terminam antes do horizonte de tempo. Então, de acordo com o *Caso 1*, os poços em S também podem ser otimamente escalonados sem que haja interrupções. Como os poços que não estão em S não contribuem para a função objetivo, uma solução ótima para $P||Production$, quando interrupções não são permitidas, também é uma solução ótima quando interrupções são permitidas. O teorema está demonstrado.

□

Considerando este fato, a restrição 2 do modelo anterior é substituída por

$$2.\ \sum_{k=t}^{H/5} \sum_{i:k<t+\Delta_i} x_{ik} \leq m,\quad t = 0 \ldots H/5,\ \text{onde } m \text{ é o número de recursos.}$$

Note que, uma vez que não há interrupções, se um poço i termina no instante k, então um recurso é considerado *ocupado* durante os instantes $k-1, k-2, \ldots, k-(\Delta_i - 1)$. Assim, se $k < t + \Delta_i$, o poço i estará consumindo um recurso no instante t. Portanto, a restrição acima estabelece que no máximo m recursos podem estar ocupados em cada instante t.

5.1.4 Upper3

Os modelos anteriores não levaram em consideração informações relevantes presentes nas instâncias consideradas neste trabalho. Uma destas informações é que só existem dois grupos principais de recursos - os barcos e as sondas, que há mais sondas do que barcos, e que muito mais atividades requerem sondas do que barcos. Além disso, o padrão das atividades presente nos poços exige que a alocação dos recursos seja feita na seguinte seqüência:

$$\text{Sonda} \rightarrow \text{Barco} \rightarrow \text{Sonda}.$$

O número de atividades executadas em cada fase da seqüência pode ser diferente para cada poço, inclusive zero.

Portanto, a versão relaxada do problema considerada nesta abordagem assume que cada poço consiste de três atividades, chamadas de *super-atividades*. A primeira super-atividade representa todas as primeiras atividades do poço que são executadas por sonda. O tempo de processamento desta super-atividade é a soma do tempo de processamento de todas as atividades representadas por ela, ou então é zero, se não há atividades no poço que devam ser executadas pela primeira sonda. O mesmo mecanismo é aplicado para as outras duas super-atividades. A j-ésima super-atividade do poço i é denotada por A_i^j e seu tempo de processamento é Δ_i^j, para $j = 1, 2, 3$. As super-atividades A_i^1 e A_i^3 devem ser executadas por sondas e a super-atividade A_i^2 deve ser executada por barcos. Além disso, a relação de precedência $A_i^1 \rightarrow A_i^2 \rightarrow A_i^3$ deve ser respeitada. A produção do poço i é iniciada quando a super-atividade A_i^3 é terminada. Novamente, a função objetivo é a mesma do problema original.

Infelizmente, interrupção não é redundante para esta versão do problema. Suponha que a instância do problema é dada por dois poços, duas sondas ($S1$, $S2$) e um barco ($B1$). O horizonte considerado é 25. A vazão de cada poço e o tempo de processamento das super-atividades é dado na tabela 5.1.

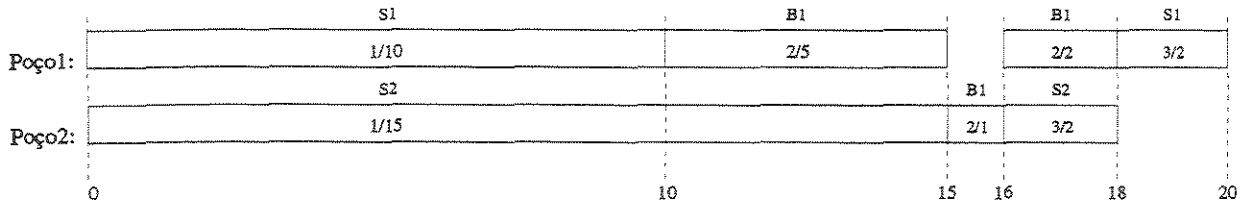
Poço	Δ_i^1	Δ_i^2	Δ_i^3	Vazão
1	10	7	2	1
2	15	1	2	2

Tabela 5.1: Durações e vazões

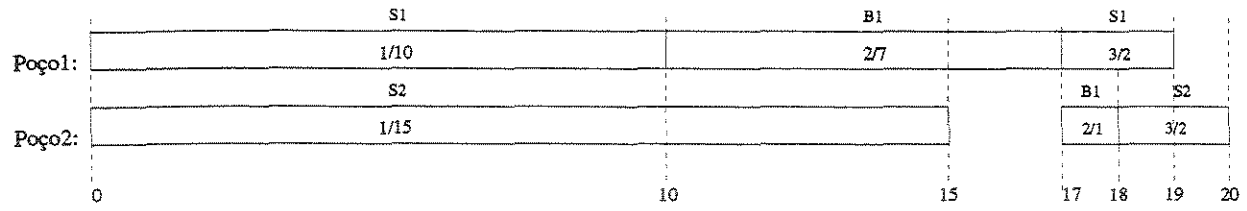
A figura 5.1 mostra as soluções ótimas quando interrupções são permitidas e quando estas são proibidas. Nesta figura, os valores dentro do retângulo indicam a super-atividade que está sendo executada e o respectivo tempo de execução. Os valores acima dos retângulos mostram que recurso está sendo alocado. A produção para o primeiro caso é 19 e para o segundo é 16 mostrando que para este exemplo a interrupção é relevante.

A idéia utilizada para modelar o problema relaxado proposto nesta abordagem é bastante similar à idéia utilizada em *Upper1*. Mas, no modelo desta abordagem, devem ser inseridas restrições para representar a precedência tecnológica entre as atividades. Também deve ser feita uma distinção entre sondas e barcos. Como nos modelos anteriores, os valores de Δ_i^1, Δ_i^2 e Δ_i^3 serão arredondados para o maior múltiplo de 5 que são iguais ou menores que Δ_i^1, Δ_i^2 e Δ_i^3, respectivamente.

O modelo apresenta três grupos de variáveis binárias x_{it}, y_{it} e z_{it}, onde i representa um poço e t um instante de término. A variável x_{it} assume valor 1 se a super-atividade A_i^1 termina no instante t, caso contrário o valor atribuído é zero. A atribuição é feita



(a) Escalonamento ótimo com interrupções - Produção: 19



(b) Escalonamento ótimo sem interrupções - Produção: 16

Figura 5.1: Interrupção não é redundante

da mesma forma para as variáveis y_{it} e z_{it}, considerando-se o instante de término das super-atividades A_i^2 e A_i^3, respectivamente.

A função objetivo é:

$$\max \sum_{i=1}^{p} \sum_{t=0}^{H/5} 5(H/5 - t) v_i z_{it}$$

onde o valor das constantes p, H and v_i é o mesmo que nas últimas três abordagens.

As restrições para este modelo são:

1. (a) $\sum_{t=0}^{H/5} x_{it} \leq 1, \quad i = 1 \ldots p;$

   (b) $\sum_{t=0}^{H/5} y_{it} \leq 1, \quad i = 1 \ldots p;$

   (c) $\sum_{t=0}^{H/5} z_{it} \leq 1, \quad i = 1 \ldots p;$

   (d) $\sum_{t=0}^{H/5} x_{it} - \sum_{t=0}^{H/5} y_{it} = 0, \quad i = 1 \ldots p;$

(e) $\sum_{t=0}^{H/5} x_{it} - \sum_{t=0}^{H/5} z_{it} = 0, \quad i = 1 \ldots p;$

2. (a) $\sum_{k=0}^{t} \sum_{i=1}^{p} (x_{ik}\Delta_i^1/5 + z_{ik}\Delta_i^3/5) \leq t \times m_1, \quad t = 0 \ldots H/5$, onde m_1 é o número de sondas ;

   (b) $\sum_{k=0}^{t} \sum_{i=1}^{p} y_{ik}\Delta_i^2/5 \leq t \times m_2, \quad t = 0 \ldots H/5$, onde m_2 é o número de barcos;

3. (a) $x_{it} = 0, \quad \forall i, t$ tal que $t < \Delta_i^1/5$;

   (b) $y_{it} = 0, \quad \forall i, t$ tal que $t < (\Delta_i^1 + \Delta_i^2)/5$;

   (c) $z_{it} = 0, \quad \forall i, t$ tal que $t < (\Delta_i^1 + \Delta_i^2 + \Delta_i^3)/5$.

4. (a) $\sum_{t=0}^{H/5} t(y_{it} - x_{it}) \geq \Delta_i^2/5, \quad i = 1 \ldots p;$

   (b) $\sum_{t=0}^{H/5} t(z_{it} - y_{it}) \geq \Delta_i^3/5, \quad i = 1 \ldots p;$

A restrição 1 estabelece que ou todas as fases de um poço terminam exatamente uma vez antes do horizonte de tempo ou nenhuma delas é executada, uma vez que se o poço não é completamente executado, então ele não contribui na função objetivo. Portanto, não é necessário executar alguma das fases se as outras não forem também executadas. Como no modelo utilizado em *Upper1*, a restrição 2 garante que há tempo disponível suficiente nos recursos para executar todas as atividades que terminam até o instante t. Além disso, nesse modelo, uma distinção é feita entre sondas e barcos. A restrição 3 diminui o número de variáveis do modelo enquanto a restrição 4 estabelece as precedências tecnológicas.

5.2 Resultados

A tabela 5.2 mostra dados quantitativos para os modelos de Programação Linear Inteira das últimas três abordagens, usando a instância real. Nesta tabela, a coluna *Restrições* representa o número de restrições, a coluna *Variáveis* representa o número de variáveis binárias, a coluna *RestNZ* é o número de coeficientes não nulos nas restrições e a coluna *ObjNZ* é o número de coeficientes não nulos na função objetivo.

Pode ser observado a partir desta tabela que o modelo para a abordagem *Upper3* é muito maior que os outros dois modelos. Por outro lado, os modelos de *Upper1* e *Upper2* têm o mesmo número de variáveis e restrições. A diferença entre eles está no número de

Abordagem	Restrições			Variáveis	RestNZ	ObjNZ
	$\leq$	$\geq$	$=$			
Upper1	407	-	1470	31906	4819470	31694
Upper2	407	-	1470	31906	491189	31694
Upper3	920	212	3925	95718	12326123	31694

Tabela 5.2: Dados quantitativo dos modelos - Instância Real

coeficientes não nulos. O número de restrições, variáveis e coeficientes não nulos para os modelos que representam as instâncias 2P112S4B3, 3P95S5B3 e 4P130S5B3 é da mesma ordem de magnitude que estes números para a instância real. Portanto, os dados da tabela 5.2 são representativos para todas as instâncias consideradas neste trabalho.

A tabela 5.3 mostra os melhores limitantes duais obtidos para a produção em cada uma das abordagens. O valor entre parênteses indica a porcentagem que as abordagens *Upper1*, *Upper2* e *Upper3* melhoraram *Upper0*. O valor do limitante dual está em milhões de unidades. Os modelos foram implementados e resolvidos utilizando a ferramenta *ILOG CPLEX*, versão 7.0, limitando-se o tempo máximo de execução a 3600 segundos. Estes testes foram realizados em uma máquina dotada de um processador Alpha21264 com 4Gb de memória RAM e rodando sob o sistema operacional True64, da HP. A tabela 5.4 mostra o número de nós percorridos na árvore de enumeração por cada uma das abordagens e o intervalo entre os melhores limitantes dual e primal dos programas lineares inteiros.

Instância	Upper0	Upper1	Upper2	Upper3
1P130S5B3(real)	378.6	310.2 (17.3%)	300.6 (20.4%)	287.4 (23.9%)
2P112S4B3	363.8	310.4 (14.6%)	302.4 (16.8%)	270.9 (25.5%)
3P95S5B3	317.8	280.3 (11.8%)	274.9 (13.5%)	257.2 (19.0%)
4P130S5B3	420.7	371.6 (11.6%)	369.6 (12.1%)	334.7 (20.4%)

Tabela 5.3: Limitantes Duais

Estas duas tabelas trazem informações bastante relevantes. A única abordagem limitada pelo tempo máximo de execução foi *Upper1*. O refinamento feito no modelo da abordagem *Upper1* para a obtenção do modelo da abordagem *Upper2* permitiu a obtenção da solução ótima para o problema relaxado representado por *Upper2*. A solução ótima foi encontrada explorando-se apenas um nó da árvore de enumeração. Por outro lado, a abordagem *Upper3* foi a que produziu os melhores limitantes duais, melhorando o limitante obtido por *Upper0* em mais de 19% em todas as instâncias. No entanto, não foi

Instância	Upper1		Upper2		Upper3	
	Nós	Intervalo	Nós	Intervalo	Nós	Intervalo
1P130S5B3	1600	1.88%	1	0	1	?
2P112S4B3	1200	1.87%	1	0	1	?
3P95S5B3	2580	0.55%	1	0	1	?
4P130S5B3	800	2.27%	1	0	1	?

Tabela 5.4: Dados quantitativos computacionais - Limitante Dual

possível obter uma solução inteira viável utilizando esta abordagem. O *software* utilizado na resolução dos modelos apresentou o erro "*1001: out of memory*" após a exploração do primeiro nó. O mesmo ocorreu após alguns parâmetros do *ILOG CPLEX* terem sido modificados para evitar este comportamento. Isto pode ser explicado pelo tamanho do modelo.

Como a abordagem *Upper1* foi a única que explorou mais que um nó da árvore de enumeração, a figura 5.2 mostra a evolução dos limitantes primais e duais em cada nó. Os limitantes estão em milhões de unidades. A instância em questão é a instância real. Pode ser observado, nesta figura, que o limitante dual permanece o mesmo em todos os nós. O valor do limitante primal é alterado apenas em alguns nós. Um comportamento similar foi observado nas outras instâncias.

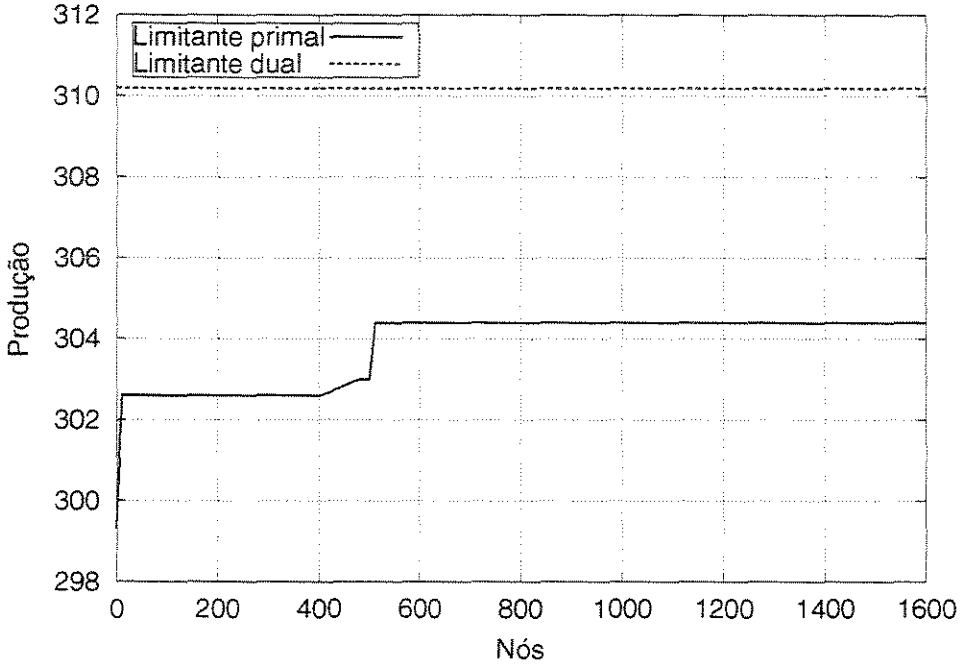


Figura 5.2: Limitantes primais e duais em cada nó da árvore de enumeração - Instância Real

Capítulo 6

Resultados Computacionais

Neste capítulo serão apresentados os resultados computacionais obtidos para as instâncias 1P130S5B3 (instância real), 2P112S4B3, 3P95S5B3 e 4P130S5B3. Para todas estas instâncias, o horizonte de tempo considerado é 1500 dias. Todos os tempos computacionais referem-se a uma máquina AMD Atlhon de 1.2 GHz e memória de 1 Gb.

A tabela 6.1 mostra a produção total obtida, em milhões de unidades, para a solução inicial da instância real e das demais instâncias. O tempo de execução para as heurísticas *CP1* e *CP2* foi menos do que 200 segundos e menos do que 60 segundos, respectivamente, para todas as instâncias. O tempo de execução para ambas as heurísticas *H1* e *H2* foi inferior a 1 segundo. Células sem valor nesta tabela significam que não foi encontrada uma solução em menos de 200 segundos.

Instância	Heurística			
	CP1	CP2	H1	H2
1P130S5B3(real)	163.6	207.4	246.2	206.5
2P112S4B3	-	184.3	220.3	173.3
3P95S5B3	155.6	187.3	225.4	185.7
4P130S5B3	-	218.9	276.5	218.3

Tabela 6.1: Soluções Iniciais

Como *H1* obteve os melhores resultados para todas as instâncias, esta heurística será empregada para gerar a solução inicial utilizada pelas técnicas híbridas e também pela Busca Tabu pura, descritas no Capítulo 4. Para testar o comportamento das técnicas quando uma solução inicial de qualidade inferior é utilizada, a heurística *H2* também será empregada para fornecer as soluções iniciais, pois esta é muito mais rápida que as heurísticas *CP1* e *CP2*. Em cada caso, será claramente indicada qual heurística foi utilizada

para gerar a solução inicial.

A tabela 6.2 resume os resultados computacionais para todas as instâncias quando as abordagens híbridas, *ID*, *IA*, *WE*, *WI* e as abordagens de Busca Tabu pura, *TabuPF* e *TabuPR*; descritas no Capítulo 4 nas páginas 55, 56 e 57, foram utilizadas para resolver o problema. A solução inicial foi obtida usando a heurística *H1*.

As colunas na tabela 6.2 têm os seguintes significados:

- Instância: identificação da instância;
- Abordagem: identificação da abordagem utilizada;
- It: número total de iterações na Busca Tabu;
- Vizinhos: número total de vizinhos viáveis explorados;
- Tabu: número total de vizinhos tabu;
- AC: número total de vizinhos que satisfizeram o critério de aspiração;
- Produção: melhor valor encontrado pela abordagem para a produção total de petróleo;
- Tempo: tempo total de execução, em segundos.

Para cada instância, o melhor valor obtido para a produção, entre todas as abordagens, está em negrito.

Para o caso particular da instância real 1P130S5B3, há dados mais detalhados descrevendo o comportamento de cada uma das colunas *Vizinhos*, *Tabu*, *AC*, *Produção* e *Tempo* ao longo das iterações. A figura 6.1 descreve o comportamento do número total de vizinhos explorados por iteração por cada uma das estratégias tabu. A figura 6.2 faz o mesmo para o número total de vizinhos tabu. A figura 6.3 trata o número total de vizinhos que satisfizeram o critério de aspiração. A figura 6.4 mostra a produção total de óleo e a figura 6.5 traz o tempo total de cada iteração.

A figura 6.6 mostra a variação da produção total de óleo ao longo do tempo de processamento para cada uma das abordagens quando utilizadas na instância real. As figuras 6.7, 6.8 e 6.9 fazem o mesmo para as instâncias geradas 2P112S4B3, 3P95S5B3 e 4P130S5B3.

A figura 6.1 indica que o número de vizinhos viáveis percorridos em cada iteração é muito variável, pois cada iteração é terminada após se obter os primeiros y melhores vizinhos. A única abordagem que não seguiu este padrão foi a *WE*, pois explorou um número fixo de vizinhos após as primeiras iterações. Nas abordagens *ID* e *IA*, pode ser observada uma tendência geral onde o número de vizinhos viáveis aumenta à medida que o

Instância	Abordagem	It	Vizinhos	Tabu	AC	Produção	Tempo
1P130S5B3	TabuPF	348	5404175	1762	9	260480500	3624
	TabuPR	368	6078448	390383	112	260955920	3619
	ID	216	5097314	482245	65	261797010	3621
	IA	234	5549670	569485	74	**262968780**	3616
	WE	169	1172	1645	0	247388220	3618
	WI	188	1882	174	4	250410270	3606
2P112S4B3	TabuPF	485	6437783	2446	15	250305872	3625
	TabuPR	494	6772949	562549	139	**250448195**	3631
	ID	185	1955786	345472	116	246309923	3639
	IA	153	1462175	262204	198	248792249	3616
	WE	176	708	1715	0	220362248	3609
	WI	200	3361	425	21	224523265	3618
3P95S5B3	TabuPF	307	5453469	1648	2	**238685695**	3644
	TabuPR	335	6999181	309804	68	**238685695**	3604
	ID	128	2247780	165676	109	233766103	3630
	IA	155	2891257	224596	65	238258503	3628
	WE	166	1164	1615	0	225755389	3614
	WI	184	1789	262	3	227267290	3605
4P130S5B3	TabuPF	331	5669485	1760	11	296412436	3622
	TabuPR	307	5232493	441899	98	294203512	3603
	ID	154	1568798	198769	156	297504129	3656
	IA	163	1919470	236697	425	**298814360**	3667
	WE	175	704	1705	0	276423927	3600
	WI	193	1752	316	15	280450182	3607

Tabela 6.2: Dados Quantitativos - H1

programa é executado. Isto indica que é mais fácil melhorar a melhor solução corrente no início da execução das abordagens, já que são explorados os primeiros y melhores vizinhos em cada iteração. Comparando as figuras 6.1 e 6.5, é fácil perceber que o tempo gasto por iteração é proporcional ao número de vizinhos explorados.

Com relação ao número total de vizinhos, a tabela 6.2 mostra que as abordagens *TabuPF*, *TabuPR*, *ID* e *IA* podem explorar um número de vizinhos três ordens de magnitude superior que as abordagens *WE* e *WI*. Este comportamento já era esperado, pois estas últimas utilizam vizinhanças de larga escala. As abordagens *ID* e *IA*, apesar de também utilizarem vizinhanças de larga escala, adotam uma heurística gulosa para que mais vizinhos possam ser explorados. Na abordagem *WE* é imposto um tempo máximo de 20 segundos por iteração. Na figura 6.1.f pode-se perceber que este tempo máximo de

processamento está sendo totalmente consumido por esta abordagem, o que lhe permite explorar apenas 7 vizinhos a cada iteração.

A figura 6.2 mostra que o número de vizinhos tabu tende a ser maior nas iterações finais, quando as abordagens *ID* e *IA* são utilizadas. Isto ocorre porque mais vizinhos são explorados nas últimas iterações e também porque a lista tabu contém menos elementos nas iterações iniciais.

A figura 6.3 indica que o número de vizinhos tabu que atendem ao critério de aspiração não segue um padrão facilmente identificável. Note que a estratégia *WE* não utiliza um critério de aspiração. A tabela 6.2 mostra que o número total de vizinhos tabu é alto, considerando-se o número total de vizinhos explorados, com exceção da abordagem *TabuPF*. Isto pode ser explicado pelo fato desta abordagem possuir a lista tabu menos restritiva. É importante ressaltar que para a abordagem *WE*, a coluna *Tabu* indica quantas vezes um poço tabu foi impedido de integrar o conjunto de poços que formam um vizinho. Isto explica porque o número de poços tabu é maior que o número de vizinhos explorados.

Os resultados mais interessantes são aqueles relacionados com a evolução das curvas de produção, apresentadas nas figuras 6.4, 6.6, 6.7, 6.8 e 6.9. A abordagem *IA* forneceu o melhor resultado para as instâncias 1P130S5B3 (instância real) e 4P130S5B3, enquanto a abordagem *TabuPR* forneceu os melhores resultados para as instâncias 2P112S4B3 e 3P95S5B3. As abordagens *WE* e *WI*, para todas as instâncias, entraram em um patamar inferior de produção e não conseguiram sair do mesmo. Uma explicação para este comportamento pode ser o fato que estas abordagens não conseguiram explorar um número grande de vizinhos e, além disso, os vizinhos explorados não foram capazes de promover ganhos significativos na produção de óleo. Estas figuras mostram um comportamento desejado quando vizinhanças de larga escala são utilizadas. As estratégias *ID* e *IA* apresentaram, para todas as instâncias, com exceção da estratégia *ID* para a instância 3P95S5B3, uma melhora bastante significativa da produção já nas primeiras iterações. Este comportamento foi particularmente intenso para instância real, como pode ser observado na figura 6.6. Acreditamos que a qualidade dos vizinhos obtidos com as estratégias *ID* e *IA* é superior, enquanto são mantidos tempos computacionais bastante competitivos. Apesar da estratégia de Busca Tabu pura, *TabuPR*, ter fornecido a melhor produção para duas instâncias, os resultados obtidos com a estratégia *IA* foram muito próximos das melhores produções obtidas.

A figura 6.4 mostra que um patamar não foi atingido quando as abordagens *TabuPF*, *TabuPR* e *IA* foram utilizadas. Para verificar se a produção poderia ser melhorada, as estratégias foram executadas por um tempo adicional de 3 horas. Um patamar de 263.5 milhões de unidades foi alcançado e não foi possível melhorá-lo.

Para observar o comportamento das abordagens quando uma solução inicial de qua-

lidade inferior fosse utilizada, todas as abordagens foram empregadas quando a solução inicial considerada é dada pela heurística *H2*. A tabela 6.3 e as figuras 6.10.b, 6.11.b, 6.12.b e 6.13.b são as correspondentes da tabela 6.2 e das figuras 6.6, 6.7, 6.8, 6.9, respectivamente, quando *H2* foi utilizada ao invés de *H1*. As figuras 6.10.a, 6.11.a, 6.12.a e 6.13.a detalham os 360 segundos iniciais de execução de cada abordagem.

Instância	Abordagem	It	Vizinhos	Tabu	AC	Produção	Tempo
1P130S5B3	TabuPF	435	4096832	1726	13	243419100	3641
	TabuPR	458	5377802	344077	121	243639100	3614
	ID	171	5222193	452063	0	242923630	3603
	IA	181	5549303	452396	0	**244595390**	3602
	WE	172	1166	1675	0	241947730	3602
	WI	218	3618	424	6	243724840	3600
2P112S4B3	TabuPF	475	5272845	2137	11	223398783	3623
	TabuPR	486	6057373	447554	129	223389183	3657
	ID	212	1444141	191602	235	232061191	3651
	IA	207	2066677	306046	217	**230394281**	3622
	WE	174	1164	1695	0	216710680	3615
	WI	250	3544	661	34	225733988	3618
3P95S5B3	TabuPF	428	4776629	2393	22	227393876	3621
	TabuPR	466	6717009	198437	119	227536172	3613
	ID	222	2319646	320605	119	**229981247**	3634
	IA	202	2727995	370660	266	229520098	3634
	WE	181	1172	1765	0	223389186	3612
	WI	222	3941	452	8	227100961	3602
4P130S5B3	TabuPF	445	4133240	1694	21	272924096	3629
	TabuPR	457	4280146	271755	141	274236257	3608
	ID	203	1184770	145761	220	**283119592**	3663
	IA	209	1475233	178982	220	**283119592**	3647
	WE	180	708	1755	0	273515507	3616
	WI	224	3624	343	10	281009822	3609

Tabela 6.3: Dados Quantitativos - H2

A tabela 6.3 mostra que os valores obtidos, quando a heurística *H2* é utilizada para gerar as soluções iniciais, o número de vizinhos viáveis, de vizinhos tabu e de vizinhos que satisfazem o critério de aspiração é aproximadamente da mesma ordem de grandeza que os valores obtidos quando a heurística *H1* é considerada. Já as curvas de produção apresentam alterações consideráveis que são analisadas a seguir. A abordagem *IA* forneceu as melhores produções para as instâncias 1P130S5B3, 2P112S4B3 e 4P130S5B3, enquanto

a abordagem *ID* forneceu a melhor produção para a instância 3P95S5B3. Note que as abordagens de Busca Tabu pura, *TabuPF* e *TabuPR*, não forneceram a melhor produção para nenhuma das instâncias consideradas. Além disso, quando *H1* foi utilizada, a diferença entre a melhor produção obtida com a abordagem de Busca Tabu pura *TabuPR* e a abordagem híbrida *IA* foi menor que 1%. Quando *H2* foi utilizada para gerar as soluções iniciais, esta diferença ficou em torno de 4% para as instâncias 2P112S4B3 and 4P130S5B3 em favor da abordagem híbrida, como pode ser visto analisando as tabelas 6.2 e 6.3 e as figuras 6.11.b e 6.13.b. Outro resultado interessante é que a produção obtida com a abordagem *WI* foi maior que a produção obtida com as abordagens puras *TabuPF* e *TabuPR*, exceto para a instância 3P95S5B3, como pode ser observado nas figuras 6.10.b, 6.11.b, 6.12.b e 6.13.b e na tabela 6.3.

As figuras 6.10.a, 6.11.a, 6.12.a e 6.13.a detalham os primeiros 360 segundos de execução para cada uma das instâncias. Desses gráficos pode ser observado quão rápido cada uma das abordagens consegue melhorar uma solução inicial ruim. No caso anterior, quando *H1* foi utilizada, *IA* melhorou a solução inicial rapidamente. Neste caso, utilizando *H2*, a estratégia *WE* foi a que melhorou a solução inicial de forma mais rápida. As abordagens puras, *TabuPF* e *TabuPR*, só conseguiram chegar ao nível obtido inicialmente por *WE* após mais de 360 segundos de computação, como pode ser observado nas figuras 6.10.a, 6.11.a, 6.12.a e 6.13.a. Infelizmente, a abordagem *WE* entrou em um patamar inferior de produção e não conseguiu melhorar ainda mais a produção.

Outro padrão sistemático é que todas as estratégias, quando empregadas utilizando soluções iniciais dadas por *H2*, ficaram estacionadas em um patamar inferior quando comparado ao patamar atingido quando a heurística *H1* foi utilizada para gerar as soluções iniciais. Isto indica que a qualidade das soluções produzidas pelas diferentes estratégias está relacionada à qualidade da solução inicial, ou seja, uma solução inicial ruim pode levar a soluções finais de qualidade inferior. Entretanto, o aspecto relevante que deve ser considerado quando *H2* foi utilizada é que as abordagens híbridas melhoraram uma solução inicial ruim mais rapidamente que as abordagens puras e, além disso, produziram os melhores resultados finais.

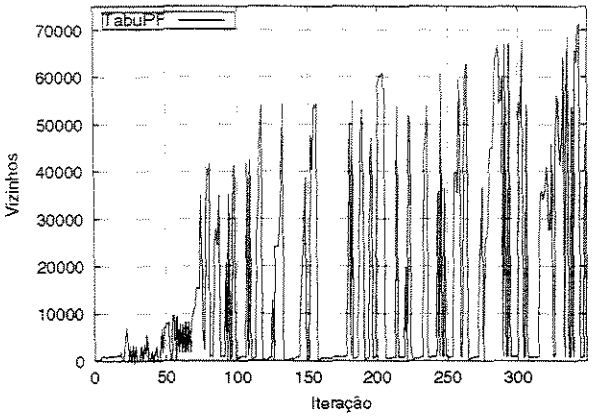
As figuras 6.14.a e 6.14.b mostram, para todas as instâncias, a solução inicial, a melhor produção obtida entre todas as estratégias e o melhor limitante dual, quando as heurísticas *H1* e *H2* são utilizadas, respectivamente, para gerar as soluções iniciais. A tabela 6.4 mostra a porcentagem que a melhor solução obtida melhorou a solução inicial e a porcentagem que a melhor solução está abaixo do melhor limitante dual dado pela tabela 5.3. Para todas as instâncias, o melhor limitante foi o obtido com a abordagem *Upper3*.

Obviamente, como as soluções iniciais fornecidas por *H2* eram piores do as soluções dadas por *H1*, as soluções produzidas por *H2* foram as que tiveram maior acréscimo. A

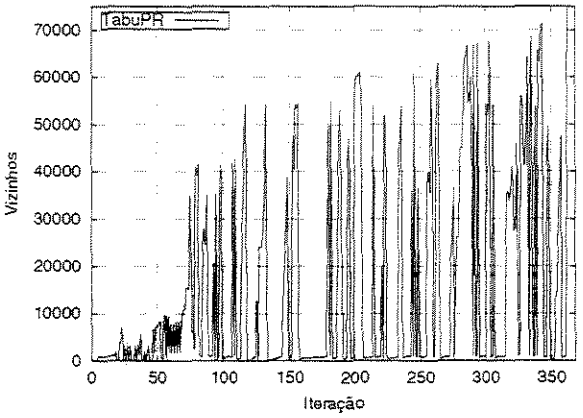
Instância	H1		H2	
	Solução Inicial	Limitante Dual	Solução Inicial	Limitante Dual
1P130S5B3	6.8%	8.5%	18.4%	14.9%
2P112S4B3	13.9%	7.55%	28.9%	17.5%
3P95S5B3	5.89%	7.2%	23.84%	10.6%
4P130S5B3	8.06%	10.7%	29.7%	15.4%

Tabela 6.4: Distância da solução inicial e do limitante dual para a melhor solução

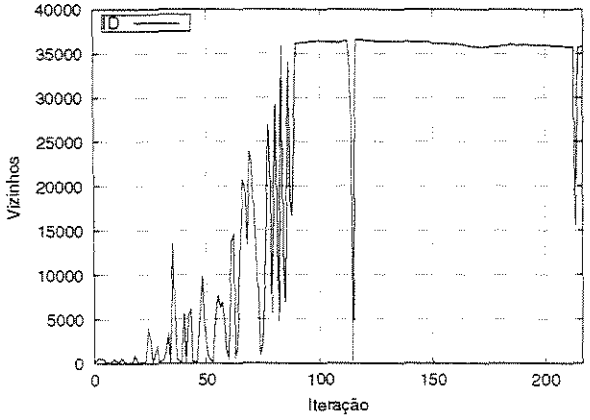
distância entre a melhor solução e o melhor limitante, também foi maior para este caso. Entretanto, a informação mais importante trazida pela tabela 6.4 é que a melhor solução está a menos de 9% do melhor limitante para as instâncias 1P130S5B3, 2P112S4B3 e 3P95S5B3 e a 10.7% para a instância 4P130S5B3. Estes dados mostram que as melhores soluções são de boa qualidade.



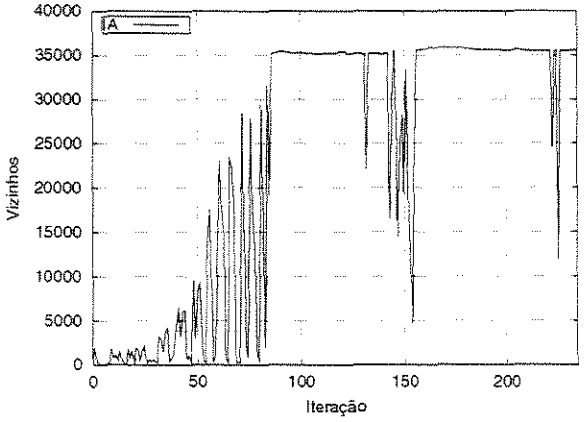
(a) TabuPF



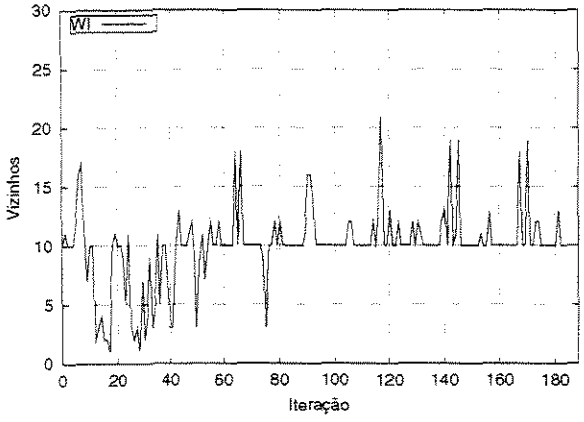
(b) TabuPR



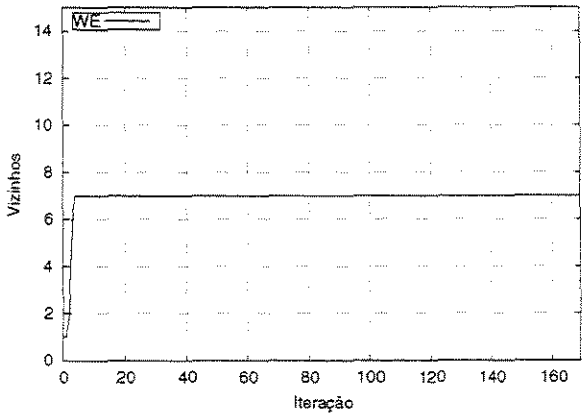
(c) ID



(d) IA

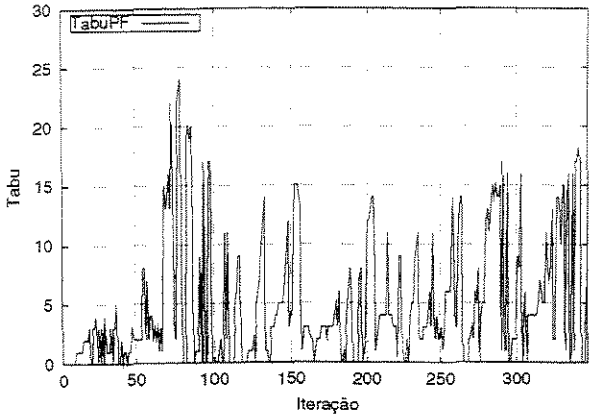


(e) WI

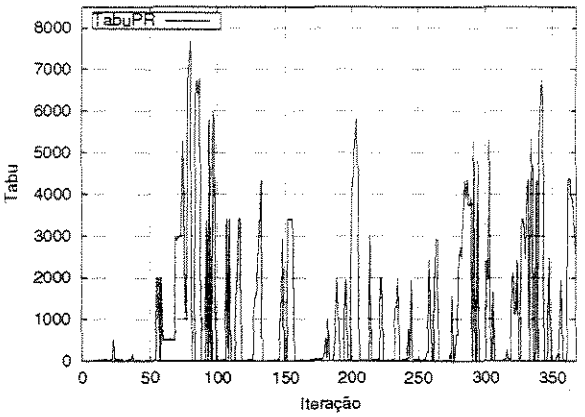


(f) WE

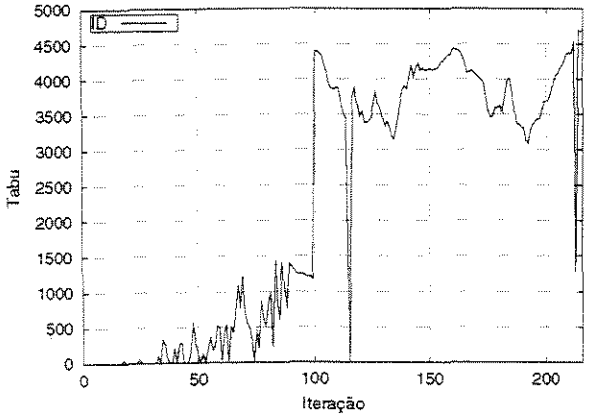
Figura 6.1: Vizinhos viáveis × Iteração - Instância Real



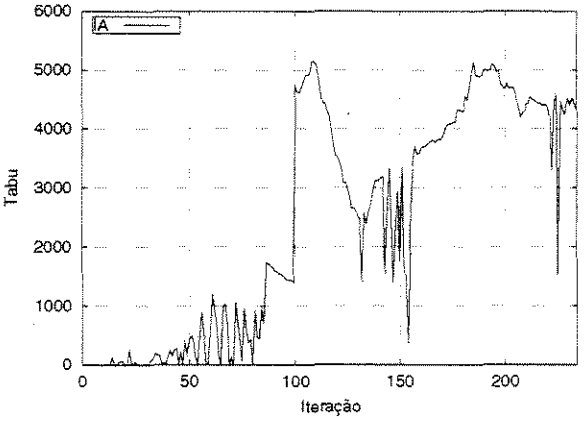
(a) TabuPF



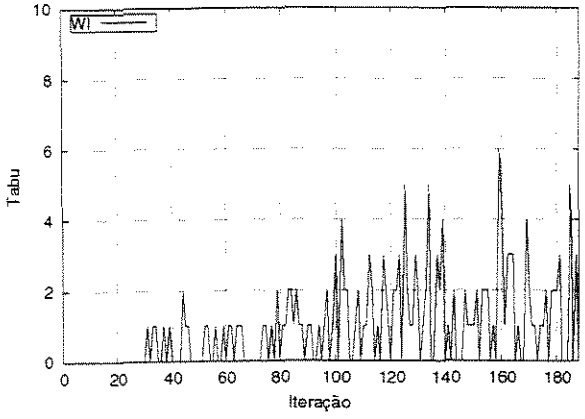
(b) TabuPR



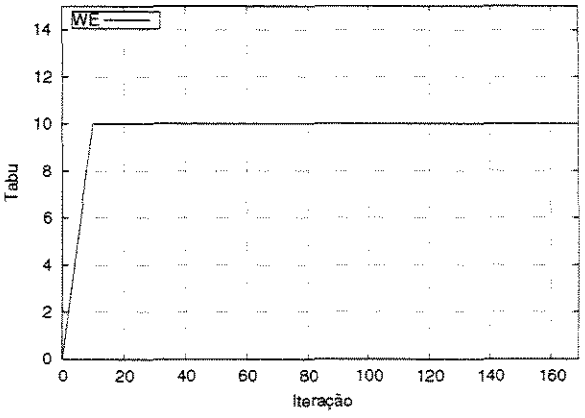
(c) ID



(d) IA

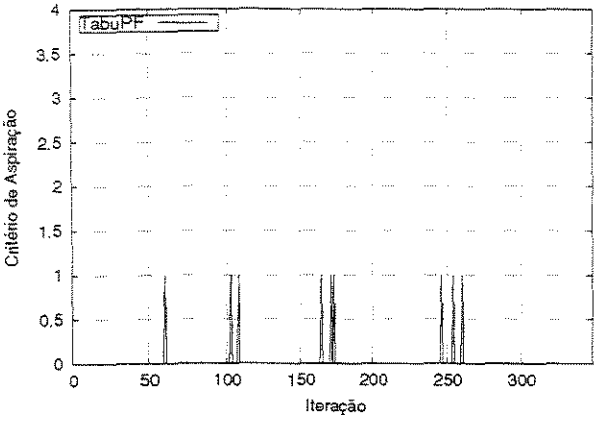


(e) WI

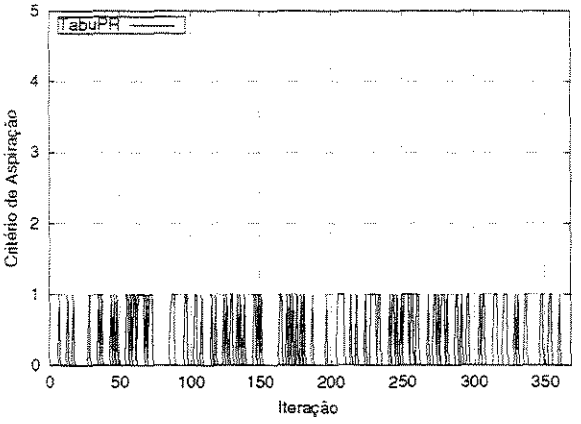


(f) WE

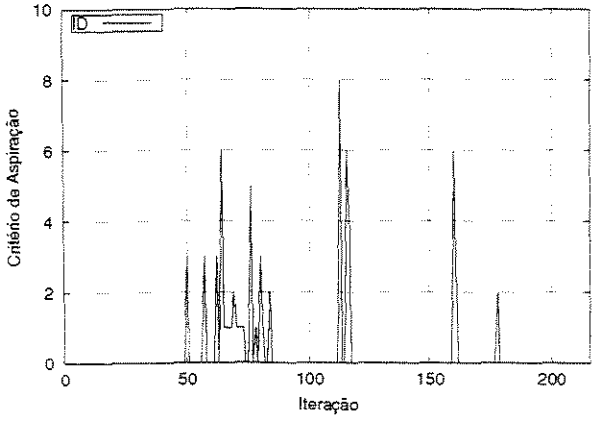
Figura 6.2: Vizinhos tabu × Iteração - Instância Real



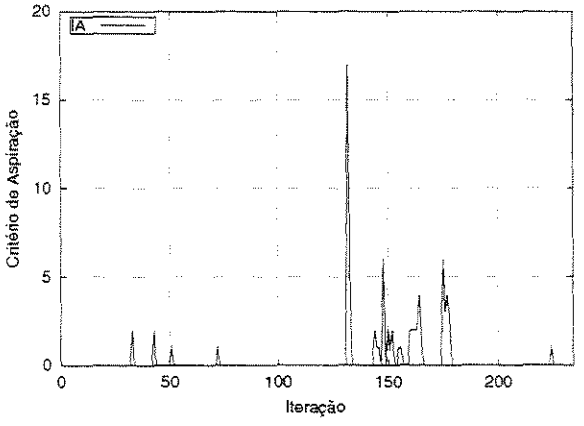
(a) TabuPF



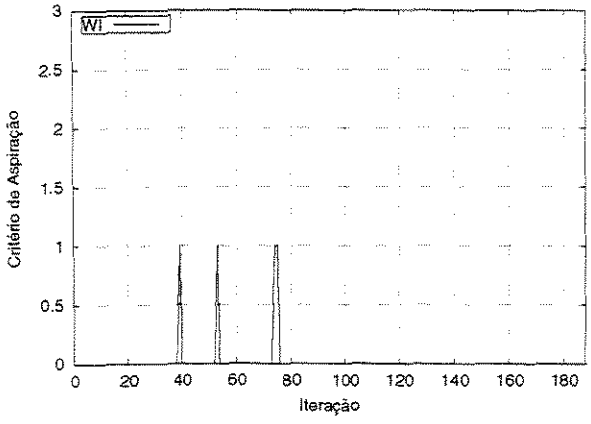
(b) TabuPR



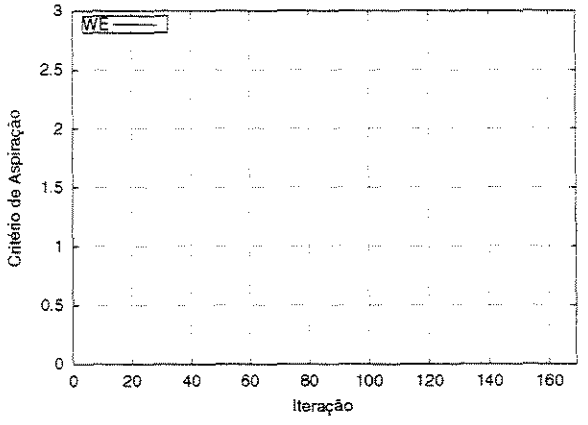
(c) ID



(d) IA

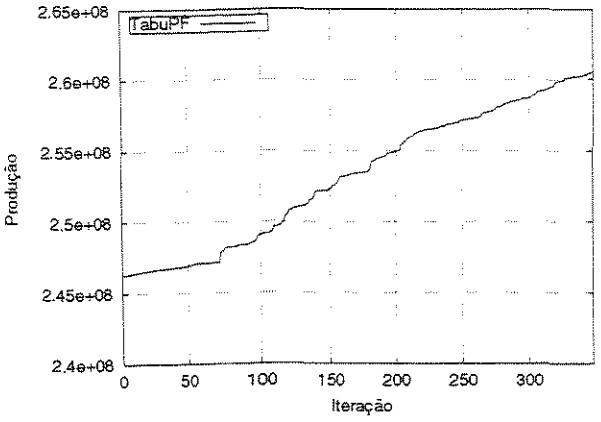


(e) WI

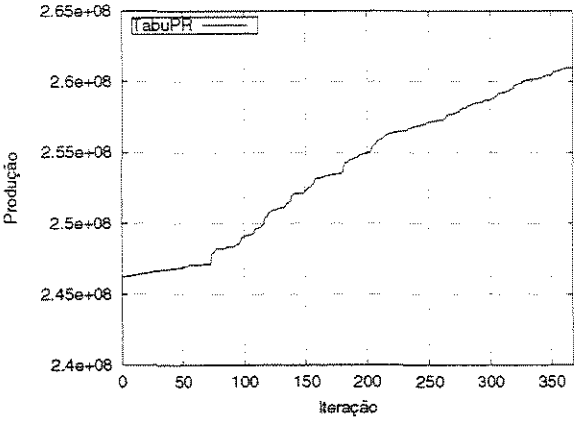


(f) WE

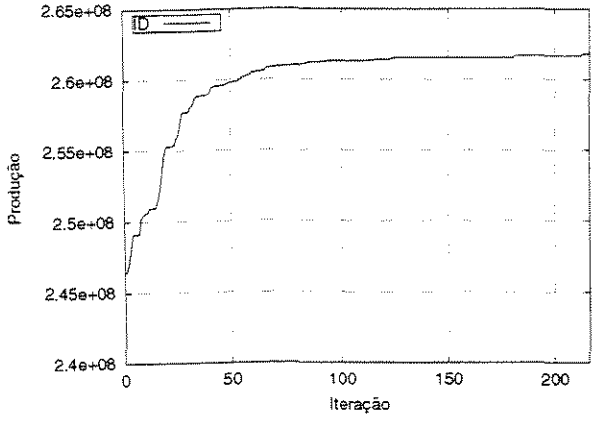
Figura 6.3: Vizinhos satisfazendo critério aspiração × Iteração - Instância Real



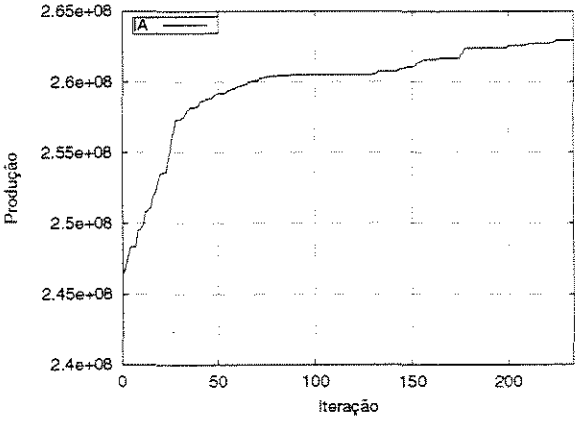
(a) TabuPF



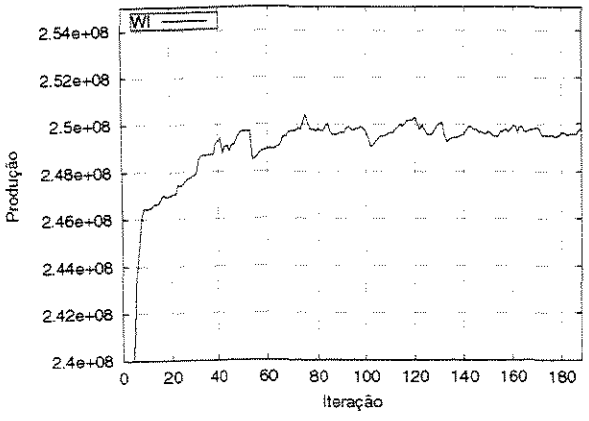
(b) TabuPR



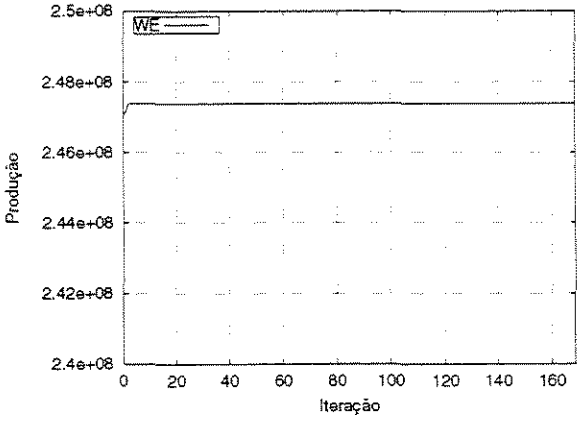
(c) ID



(d) IA

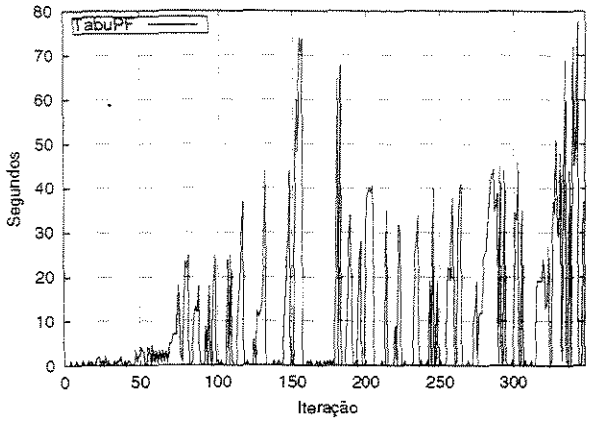


(e) WI

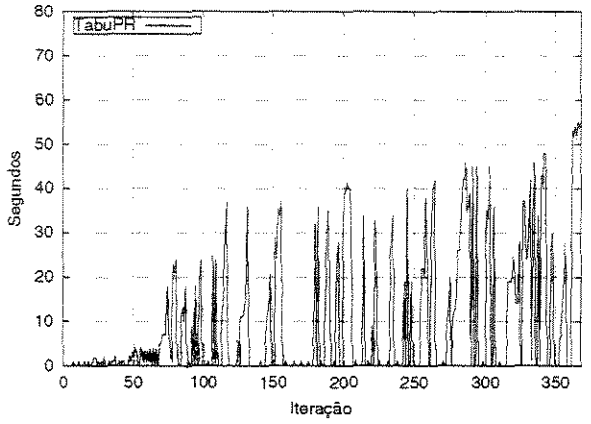


(f) WE

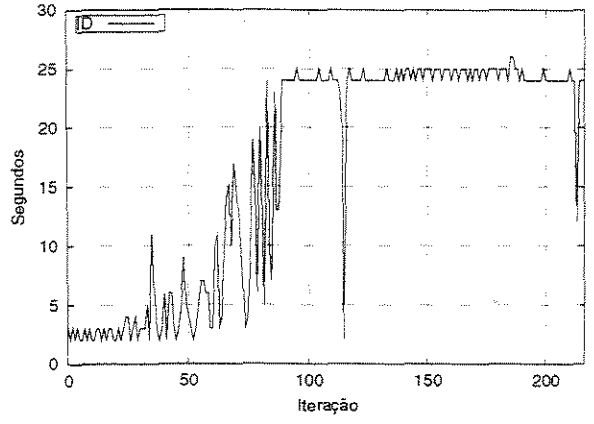
Figura 6.4: Produção × Iteração - Instância Real



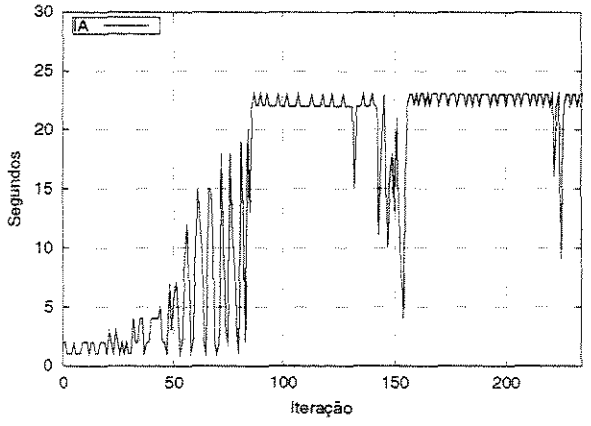
(a) TabuPF



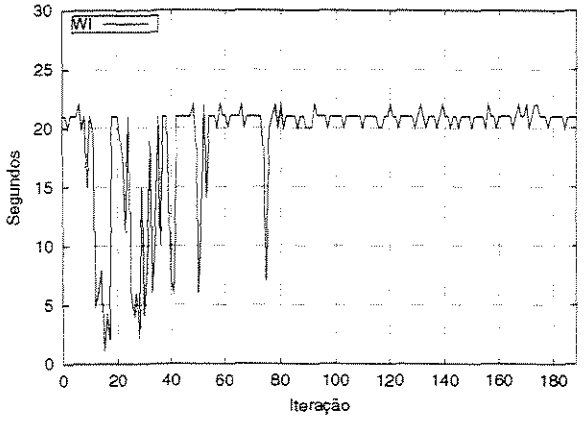
(b) TabuPR



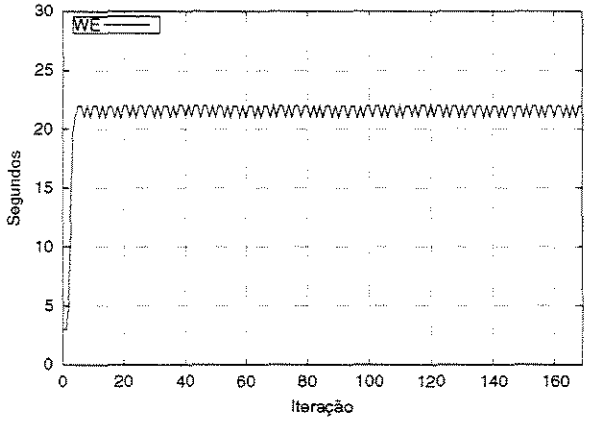
(c) ID



(d) IA



(e) WI



(f) WE

Figura 6.5: Tempo × Iteração - Instância Real

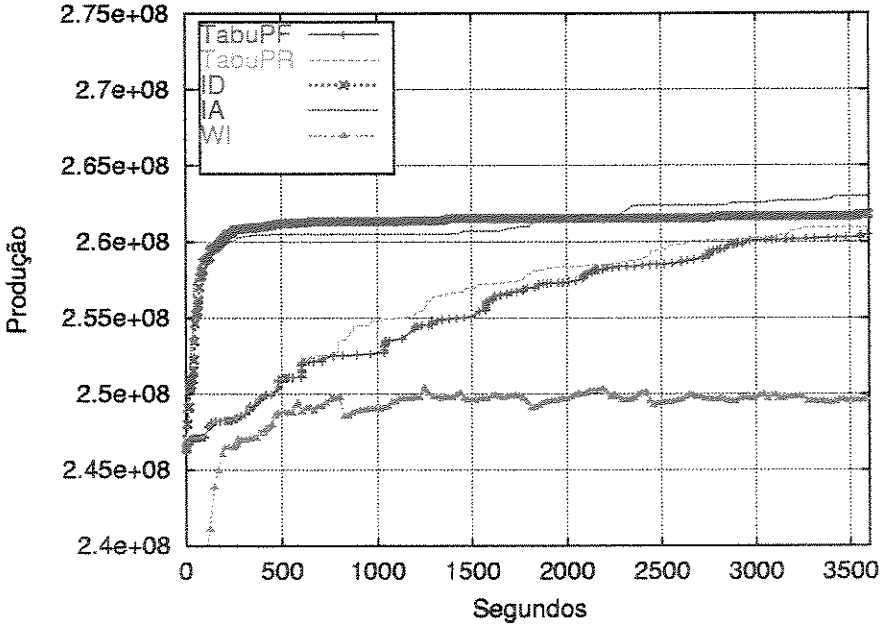


Figura 6.6: Produção × Tempo - 1P130S5B3 (Instância Real) - H1

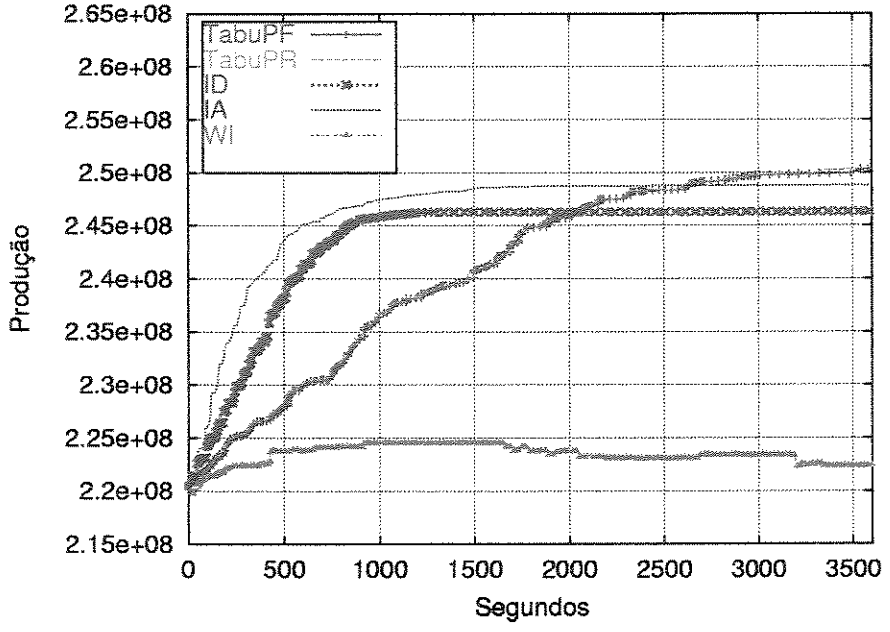


Figura 6.7: Produção × Tempo - 2P112S4B3 - H1

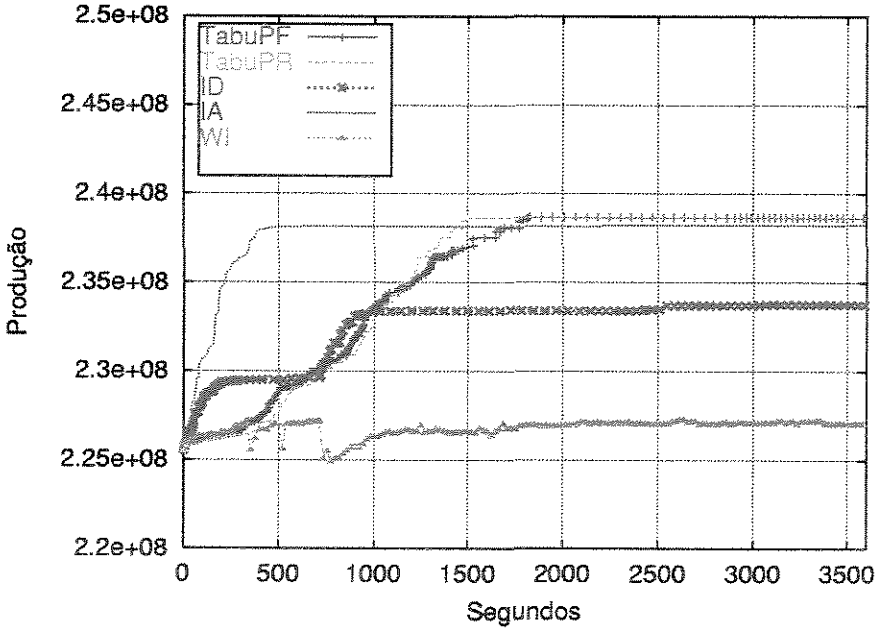


Figura 6.8: Produção × Tempo - 3P95S5B3 - H1

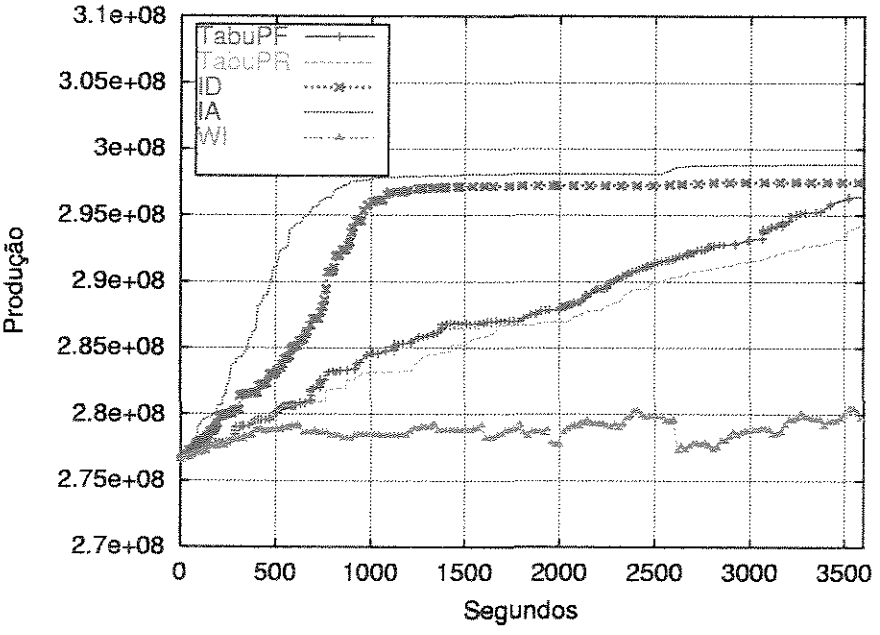
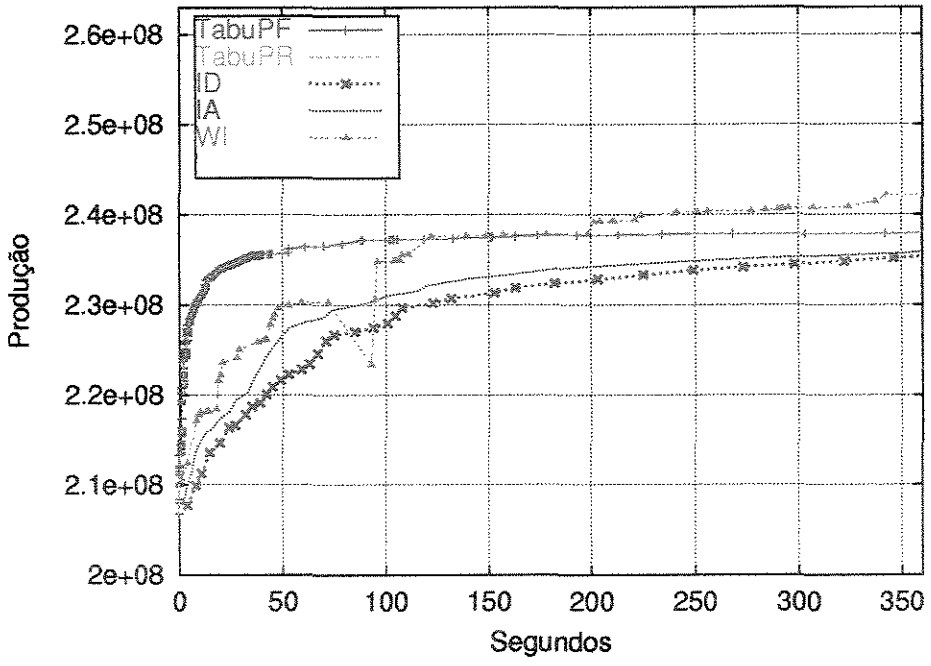
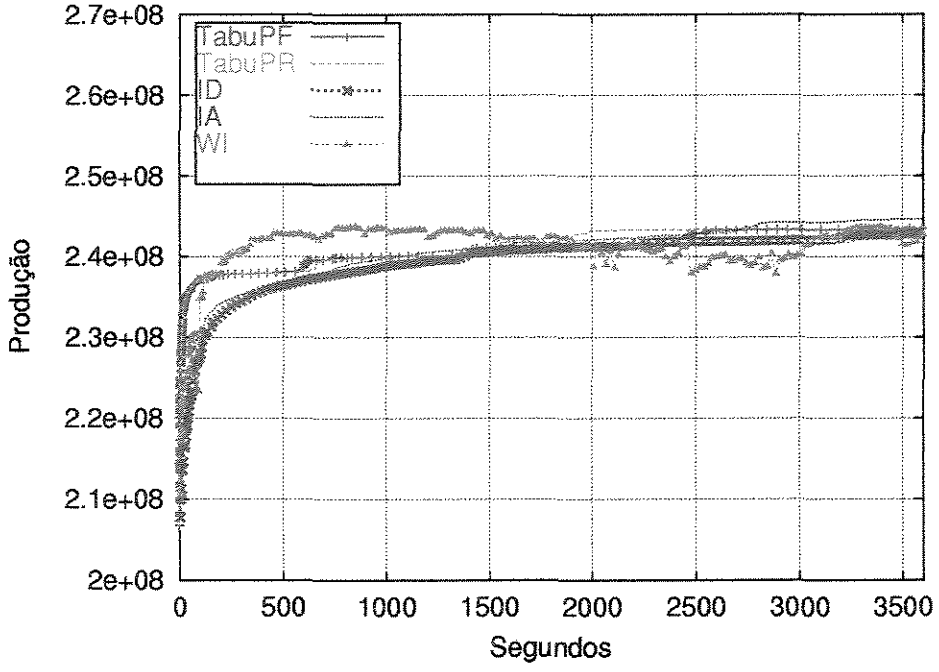


Figura 6.9: Produção × Tempo - 4P130S5B3 - H1

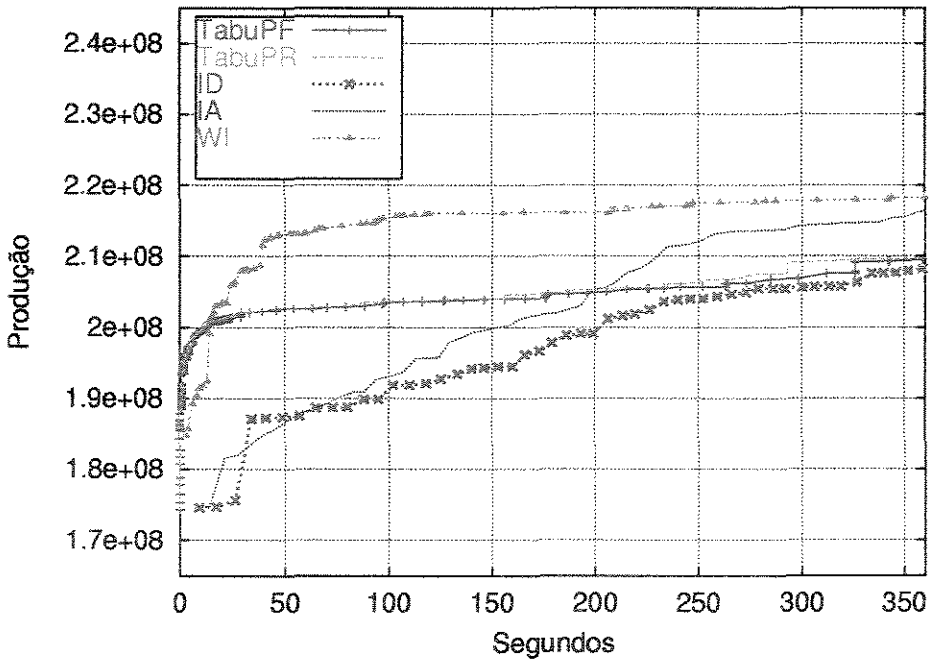


(a) Tempo=0...360

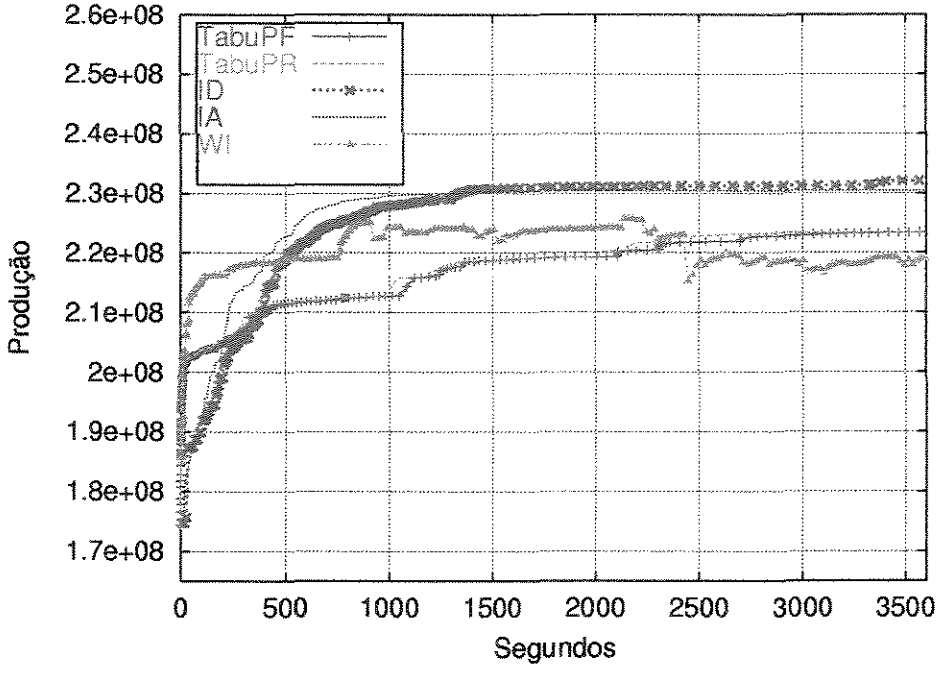


(b) Tempo=0...3600

Figura 6.10: Produção × Tempo - 1P130S5B3 (Instância Real) - H2

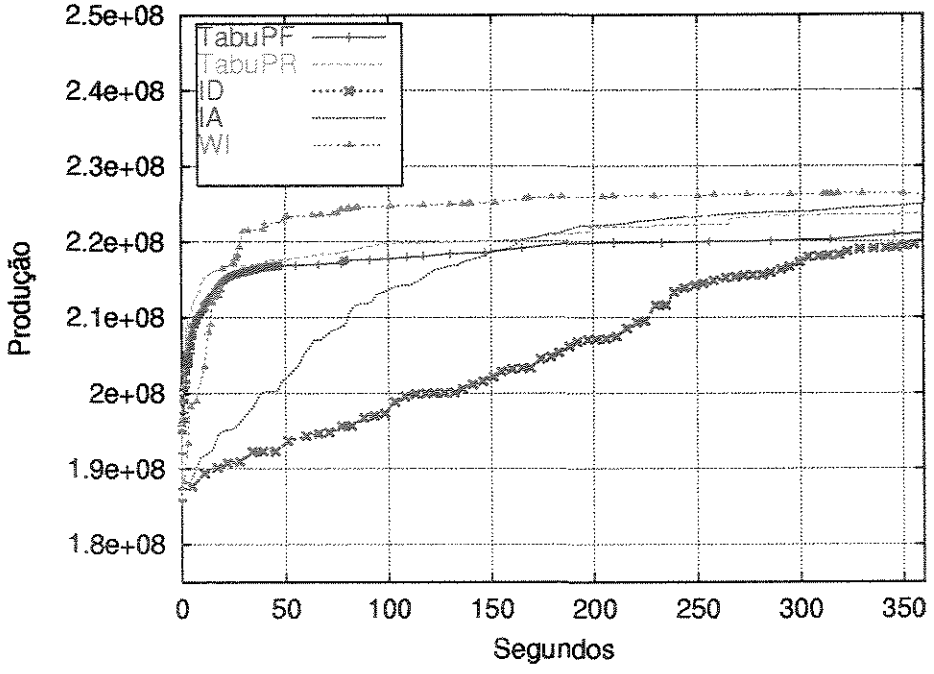


(a) Tempo=0...360

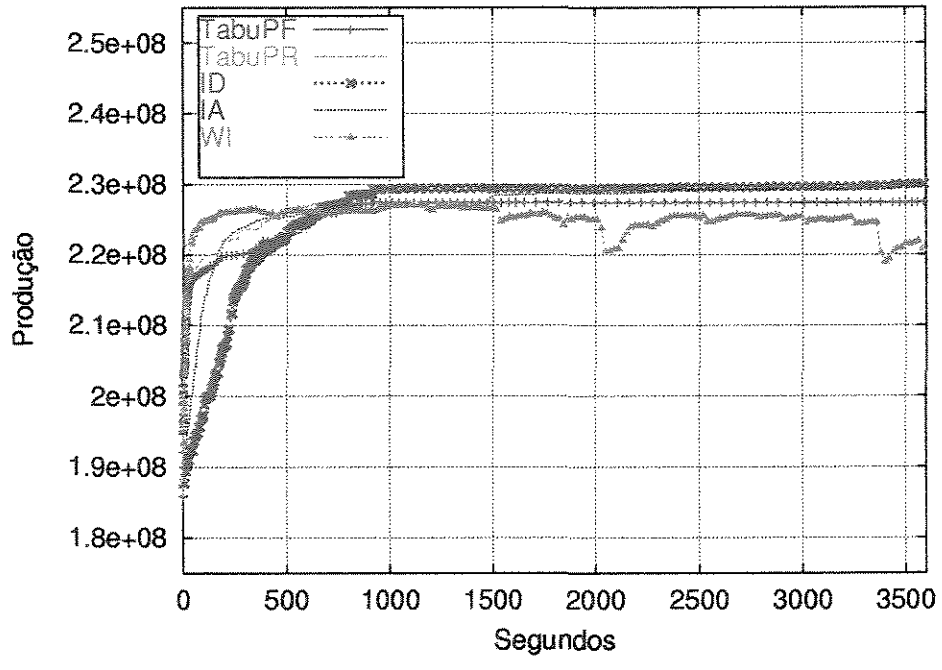


(b) Tempo=0...3600

Figura 6.11: Produção × Tempo - 2P112S4B3 - H2

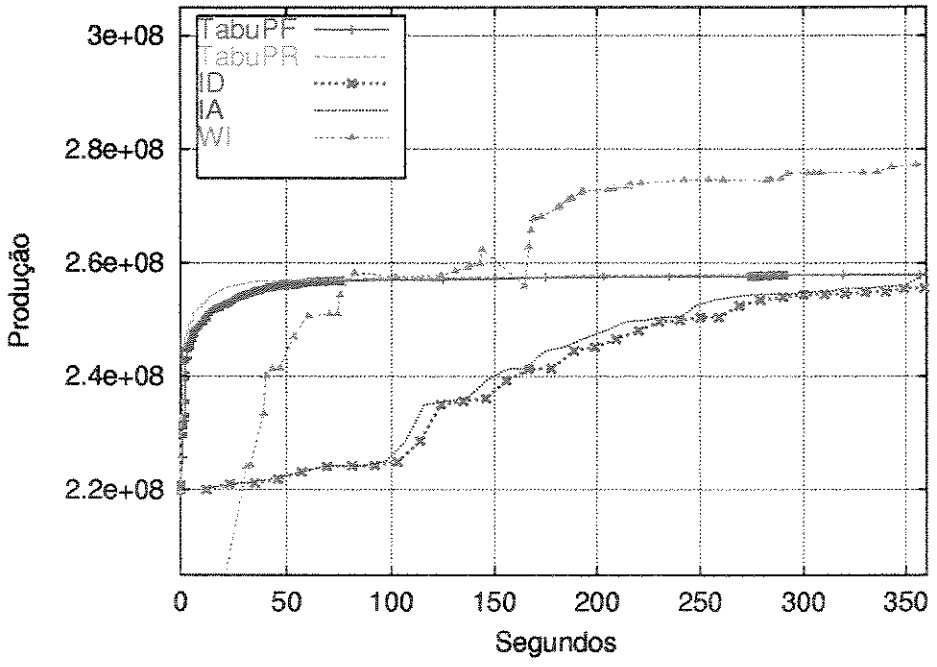


(a) Tempo=0...360

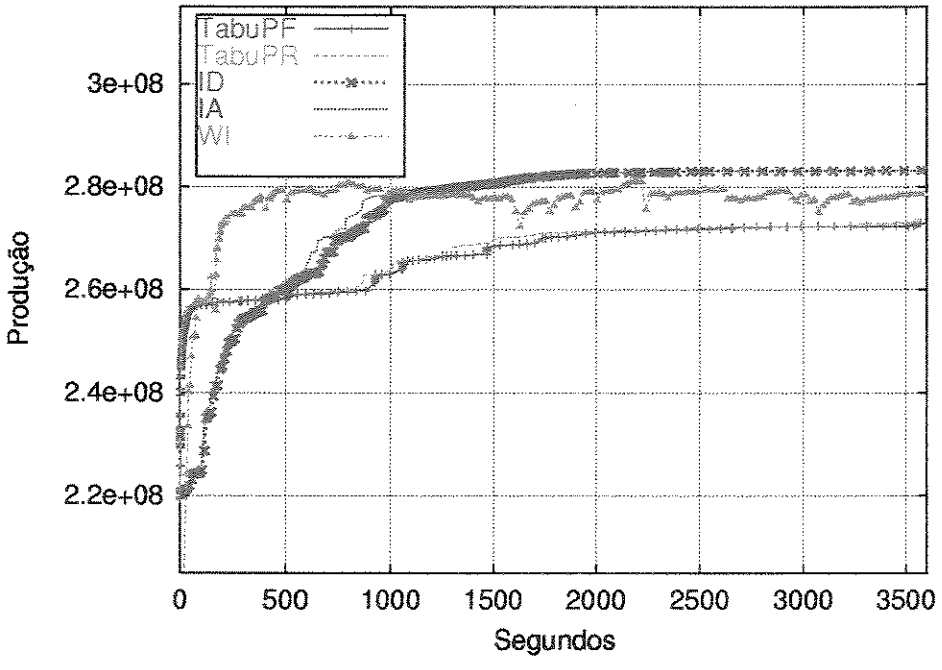


(b) Tempo=0...3600

Figura 6.12: Produção × Tempo - 3P95S5B3 - H2

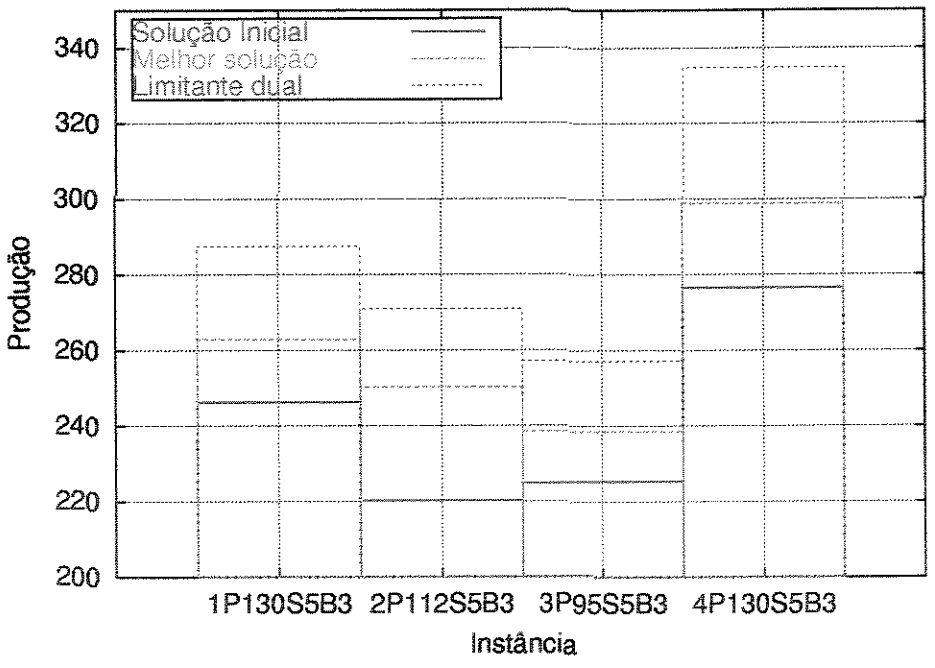


(a) Tempo=0...360

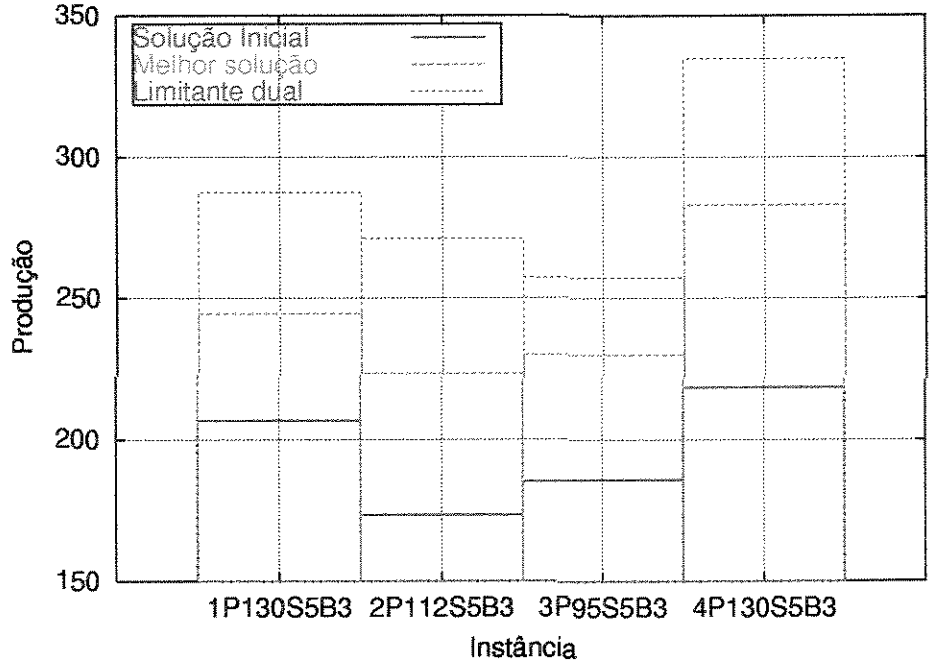


(b) Tempo=0...3600

Figura 6.13: Produção × Tempo - 4P130S5B3 - H2



(a) H1



(b) H2

Figura 6.14: Solução inicial, melhor solução e limitante dual

Capítulo 7

Análise de Sensibilidade

Neste capítulo, é apresentada uma análise de sensibilidade com o objetivo de se verificar como as técnicas aqui propostas se comportam quando as instâncias de entrada são perturbadas.

São considerados dois tipos de perturbação. Como discutido na seção 5.1, nas instâncias tratadas neste trabalho, há mais sondas do que barcos e muito mais atividades que requerem o primeiro tipo de recurso. Além disso, o padrão das atividades presentes nos poços exige que a alocação dos recursos seja feita na seguinte seqüência:

$$\text{Sonda} \rightarrow \text{Barco} \rightarrow \text{Sonda}.$$

Assim, devido à sua menor quantidade, os barcos podem ser considerados o gargalo no processo de desenvolvimento dos poços. Nas instâncias tratadas nos capítulos anteriores, os tempos de processamento das atividades que requerem um barco estão no intervalo de $[5, 17]$ dias. Estas instâncias serão perturbadas, substituindo-se o tempo de processamento original de cada atividade que requer barco, por um tempo de processamento gerado aleatoriamente seguindo uma distribuição uniforme nos intervalos $[1, 30]$, $[1, 60]$ e $[1, 90]$. Este procedimento deu origem a variação aqui chamada de *Atv*, gerando instâncias perturbadas a partir das instâncias originais 1P130S5B3, 2P112S4B3, 3P95S5B3 e 4P130S5B3. Para cada uma das instâncias originais, a variação *Atv* apresenta três subcasos, *Atv30*, *Atv60* e *Atv90*, que correspondem às situações em que o intervalo de tempo considerado para a duração das atividades nos barcos é $[1, 30]$, $[1, 60]$ e $[1, 90]$, respectivamente.

Por outro lado, nas instâncias consideradas nos capítulos anteriores, todos os barcos podem executar todas as atividades que requerem barco. O mesmo é válido para as sondas, com exceção de uma delas. Mas é natural considerar que podem existir instâncias onde o conjunto de recursos capaz de executar as atividades é mais restrito. Portanto, em um segundo tipo de perturbação analisado neste trabalho, considerar-se-á que a cada atividade está associado um subconjunto mais restrito de recursos, o qual será escolhido

entre todos os recursos que são capazes de executar a atividade na instância original. Assim, nas instâncias perturbadas, a cada atividade são atribuídos aleatoriamente x recursos específicos, entre aqueles que têm o tipo compatível com a atividade. Este procedimento deu origem à variação chamada de *Rec*. Esta variação, como a anterior, também subentende três subcasos, *Rec1*, *Rec2* e *Rec3*, que correspondem aos casos que x assume os valores 1, 2 e 3, respectivamente.

Como as instâncias perturbadas são geradas com um componente aleatório, para cada variação de cada instância original, foram geradas 10 instâncias perturbadas. Dessa forma é possível obter resultados *médios* para cada variação, atenuando possíveis distorções.

Neste capítulo, somente a heurística *H1* será considerada para gerar as soluções iniciais. Além disso, somente as abordagens *TabuPR*, *IA*, *WE* e *WI* serão testadas, já que na grande maioria dos casos testados no capítulo 6 as abordagens *TabuPF* e *ID* ou tiveram um desempenho inferior quando comparadas às abordagens *TabuPR* e *IA*, ou o desempenho foi bem semelhante. Para a obtenção dos limitantes duais, somente as abordagens *Upper2* e *Upper3* serão consideradas, já que estas abordagens produziram resultados muitos melhores que as abordagens *Upper0* e *Upper1*, como pode ser visto no capítulo 5.

A tabela 7.1 (página 97) apresenta a solução inicial, em milhões de unidades, para cada uma das 10 instâncias perturbadas (colunas *I1-I10*) obtidas aplicando-se cada subcaso da variação *Atv* sobre as instâncias originais. Esta tabela também apresenta, na coluna *Med*, o valor médio da solução inicial para cada subcaso da variação *Atv*. A tabela 7.2 (página 98) faz o mesmo quando a variação *Rec* é considerada.

Analisando a tabela 7.1, pode-se perceber que para a variação *Atv30* a diferença entre as menores e maiores soluções iniciais obtidas para as instâncias perturbadas de uma mesma instância original foi em torno de 3 milhões de unidades. Para a variação *Atv60* esta diferença foi em torno de 13 milhões de unidades, enquanto que para a variação *Atv90*, o intervalo entre a menor e a maior solução foi em torno de 30 milhões de unidades. Este padrão já era esperado, já que quanto maior o intervalo de possíveis valores para o tempo de processamento das atividades, maior é a chance de que sejam geradas discrepâncias entre os valores das soluções iniciais das instâncias perturbadas. Já a tabela 7.2 mostra que para a variação *Rec1*, a diferença foi em torno de 7 milhões de unidades, enquanto que para as variações *Rec2* e *Rec3*, a diferença foi em torno de 2 milhões de unidades. Esta tabela também mostra que o valor obtido para as soluções iniciais da variação *Rec2* são bem próximos aos valores obtidos para a variação *Rec3*. Como esperado, por ser a variação mais restritiva entre as três variações *Rec*, a variação *Rec1* foi a que apresentou os menores valores para a solução inicial.

Os valores dos limitantes duais para as instâncias perturbadas que seguem a variação *Rec* são os mesmos valores obtidos para as instâncias originais, apresentados na seção 5.2 na tabela 5.3. Note que a informação relevante para os modelos dos limitantes duais é

o tempo de processamento das atividades e o número de sondas e barcos. Todos estes valores permaneceram os mesmos nas instâncias perturbadas da variação *Rec*.

Por outro lado, como o tempo de processamento das atividades foi alterado nas instâncias perturbadas da variação *Atv*, novos limitantes duais foram calculados. A tabela 7.3 (página 99) mostra os valores dos limitantes para os modelos *Upper2* e *Upper3*, além do valor médio para cada subcaso da variação *Atv* (coluna *Med*).

Embora não mostrado nesta tabela, a abordagem *Upper2* obteve a solução ótima para o problema relaxado em questão, enquanto que a abordagem *Upper3* teve sua execução interrompida por problemas de falta de memória antes de conseguir encontrar uma solução inteira viável para a relaxação do problema, a exemplo do que acontecera com as instâncias originais. O comportamento inesperado, como pode ser observado na tabela 7.3, é que o melhor limitante para a variação *Atv90* foi o limitante obtido com a abordagem *Upper2*, ao contrário do que aconteceu com as instâncias originais e com as instâncias das variações *Atv30* e *Atv60* (na grande maioria dos casos), onde o melhor limitante foi dado pela abordagem *Upper3*. Há pelo menos duas explicações para este fato. Isto pode ter ocorrido porque a abordagem *Upper3* apresentou problemas na sua execução e foi incapaz de melhorar o limitante inicial, enquanto que a abordagem *Upper2* achou a solução ótima. Outra provável explicação para este fato é que o modelo da abordagem *Upper2* não permite interrupções e execuções paralelas de uma mesma atividade, o que não é verdade para o modelo da abordagem *Upper 3*. Como estas permissões não foram um problema quando as instâncias originais e as variações *Atv30* e *Atv60* foram consideradas, o problema deve ter surgido devido a grande variabilidade dos tempos de processamento das atividades das instâncias perturbadas da variação *Atv90*.

A tabela 7.4 (página 100) é similar à tabela 6.2 (página 71) e mostra alguns dados numéricos obtidos para todas as variações *Atv* quando cada uma das abordagens *TabuPR*, *IA*, *WE* e *WI* foi utilizada. Os dados apresentados nesta tabela são os dados médios obtidos a partir das 10 instâncias perturbadas de cada subcaso da variação *Atv*. As colunas desta tabela exibem as seguintes informações: as identificações da instância, da variação e da abordagem; o número total médio de iterações; os números totais médios de vizinhos viáveis explorados, de vizinhos tabu e de vizinhos que satisfizeram o critério de aspiração e, finalmente, as médias das melhores produções obtidas e dos tempos de computação requeridos para cada uma das abordagens.

Esta tabela mostra que o número de iterações percorridas por cada abordagem foi similar ao valor obtido quando as instâncias originais foram consideradas, com exceção das abordagens *TabuPR* e *IA* quando a variação *Atv90* é considerada. Neste caso os valores praticamente triplicaram. O número de vizinhos viáveis, bem como o número de vizinhos tabu, são da mesma ordem de magnitude para todos os subcasos, analisando separadamente cada uma das abordagens. Entretanto, comparando estes valores com

os valores obtidos para as instâncias originais, não foi possível estabelecer um padrão de comparação, a não ser no caso da abordagem *WI*, que teve estes valores aproximadamente multiplicados por 2 nas instâncias 1P130S5B3, 3P95S5B3 e 4P130S5B3. Já o número de vizinhos que satisfizeram ao critério de aspiração também foi bem semelhante para cada subcaso quando a mesma abordagem é analisada. No entanto, pode ser observado que para a abordagem *IA*, os valores foram bem superiores, especialmente no subcaso *Atv90*, mesmo levando-se em consideração que o número de vizinhos tabu desta abordagem é superior ao das outras abordagens. Os valores para o critério de aspiração para *IA* também foram bem superiores aos obtidos para as instâncias originais. Como esperado, as produções obtidas para as instâncias perturbadas são inferiores às produções obtidas para as instâncias originais correspondentes, já que o tempo de processamento das atividades tende a ser maior nas instâncias perturbadas.

A seguir, propõe-se uma análise detalhada da produção total obtida pelas diferentes abordagens. A tabela 7.5 (página 101) mostra a melhor produção obtida para cada uma das instâncias perturbadas. Já as figuras 7.1, 7.2 e 7.3.b (páginas 102 e 103) são similares à figura 6.6 (página 81) para a instância real. Elas mostram a curva de produção quando as variações *Atv30*, *Atv60* e *Atv90* foram exercitadas. Cada figura mostra a curva de produção quando cada uma das abordagens *TabuPR*, *IA*, *WE* e *WI* foi utilizada. Em cada figura, foi escolhida uma instância representativa entre as 10 instâncias perturbadas de cada subcaso da variação *Atv*. As figuras 7.4, 7.5 e 7.6.b (páginas 104 e 105) fazem o mesmo para a instância 2P112S4B3. A figura correspondente para o caso sem perturbações é a figura 6.7 (página 81). As figuras 7.7, 7.8 e 7.9.b (páginas 106 e 107) correspondem à instância 3P95S5B3 e o caso original é mostrado na figura 6.8 (página 82). Finalmente, as figuras 7.10, 7.11 e 7.12.b (páginas 108 e 109) correspondem à instância 4P130S5B3 e são similares à figura 6.9 (página 82).

A tabela 7.5 mostra que a abordagem *IA* produziu os melhores resultados para todas as instâncias e variações, com exceção de apenas 3 instâncias perturbadas da instância original 2P112S4B3.

O comportamento da curva de produção de cada abordagem quando aplicadas à variação *Atv30* foi similar aos seus respectivos comportamentos quando as instâncias originais foram consideradas, como pode ser observado nas figuras 7.1, 7.4, 7.7 e 7.10. Para esta variação, as abordagens híbridas *WI* e *WE* não conseguiram escapar de patamares inferiores de produção, mesmo considerando-se que as curvas de produção da abordagem *WI* apresentaram, em algumas iterações, decréscimo na produção total. Outra semelhança é que a abordagem de Busca Tabu pura *TabuPR* produziu melhores soluções que *WI* e *WE*, sendo as soluções produzidas pela abordagem *TabuPR* um pouco inferiores às obtidas pela abordagem híbrida *IA*. Também como no caso das instâncias originais, a abordagem híbrida *IA* foi a que apresentou os melhores resultados e teve a maior taxa de convergência.

Quando a variação *Atv60* é considerada, o comportamento das abordagens já começa a ser modificado, como pode ser observado nas figuras 7.2, 7.5, 7.8 e 7.11. Nesta variação a abordagem híbrida *WI* produziu melhores resultados que a abordagem de Busca Tabu pura *TabuPR*, com exceção da instância 2P112S4B3, e a taxa de convergência da abordagem *WI* também foi superior.

Entretanto, quando a variação *Atv90* é considerada, o comportamento das abordagens foi bastante diferente, como pode ser observado nas figuras 7.3.b, 7.6.b, 7.9.b e 7.12.b. Para que mais detalhes das primeiras iterações fossem analisados, as figuras 7.3.a, 7.6.a, 7.9.a e 7.12.a (páginas 103, 105, 107 e 109) mostram os primeiros 360 segundos de execução para as mesmas instâncias apresentadas nas figuras 7.3.b, 7.6.b, 7.9.b e 7.12.b, respectivamente.

Para a variação *Atv90*, a abordagem *WI* produziu, consistentemente para todas as instâncias perturbadas das instâncias originais 1P130S5B3 e 4P130S5B3, resultados melhores do que os produzidos pela abordagem de Busca Tabu pura, *TabuPR*, além de possuir uma taxa maior de convergência. O mesmo pode ser observado para a maioria das instâncias perturbadas das instâncias originais 2P112S4B3 e 3P95S5B3.

Além disso, a produção obtida utilizando a abordagem *TabuPR* foi consideravelmente inferior à melhor produção encontrada, especialmente nas instâncias originadas a partir de 1P130S5B3 e 4P130S5B3. Este fato deixa claro que a abordagem de Busca Tabu pura não é robusta em relação a variações na duração das atividades nos barcos.

Já as figuras 7.3.a, 7.6.a, 7.9.a e 7.12.a mostram que, como ocorreu para as instâncias originais quando a heurística *H2* gerou as soluções iniciais, a abordagem *WE* foi capaz de melhorar a solução inicial rapidamente, mas de novo não foi capaz de escapar de um patamar inferior de produção. Também pode ser observado, a partir destas figuras, que a taxa de convergência das abordagens híbridas *IA* e *WI* é bem maior que a da abordagem pura *TabuPR*. Isto mostra que, além da melhor qualidade das soluções obtidas por estas abordagens elas foram alcançadas em um tempo de execução inferior que as soluções obtidas pela abordagem pura *TabuPR*.

A tabela 7.6 (página 110) mostra o percentual que a melhor solução obtida melhorou a solução inicial da respectiva instância. A coluna *Med* mostra o percentual médio para todas as instâncias de um mesmo subcaso das variações *Atv*. Já a tabela 7.7 (página 111) mostra quanto, percentualmente, a melhor solução ficou abaixo dos limitantes duais. Vale lembrar que a melhor solução foi obtida utilizando a abordagem híbrida *IA*.

Pode ser inferido da tabela 7.6 que a heurística de solução inicial não foi capaz de gerar boas soluções iniciais para a variação *Atv90*, já que a abordagem híbrida *IA* foi capaz de melhorar estas soluções em torno de 41%. Por outro lado, a tabela 7.7 mostra que a estratégia híbrida conseguiu produzir boas soluções, já que a diferença entre as melhores soluções e os melhores limitantes duais ficou em torno de 10.5%, para a variação *Atv90*. Já para as variações *Atv30* e *Atv60* pode ser concluído que as soluções iniciais já

eram de boa qualidade. Novamente a diferença percentual entre as melhores soluções e os melhores limitantes duais ficou em torno de 11%.

A tabela 7.8 (página 112) é similar à tabela 7.4 (página 100) utilizando neste caso os subcasos da variação *Rec*. Esta tabela mostra que o número de iterações percorridas por cada abordagem foi similar ao valor obtido quando as instâncias originais foram consideradas, com exceção da abordagem *TabuPR* que teve os valores aproximadamente reduzidos pela metade para todos os subcasos, e da abordagem *IA* quando a variação *Rec1* é considerada, quando os valores aproximadamente dobraram. O número de vizinhos viáveis explorados, bem como o número de vizinhos tabu, são da mesma ordem de magnitude para todos os subcasos, analisando separadamente cada uma das abordagens, com exceção da abordagem *TabuPR* para o subcaso *Rec1*, onde o número de vizinhos tabus foi aproximadamente o dobro do valor obtido para os outros subcasos. O número de vizinhos tabu, para a abordagem *TabuPR* quando a variação *Rec1* é considerada, também é aproximadamente o dobro dos valores obtidos para as instâncias originais correspondentes. Já o número de vizinhos viáveis explorados, ao contrário do esperado, já que o número de vizinhos tabu aumentou, é aproximadamente a metade dos valores obtidos para as instâncias originais 1P130S5B3 e 4P130S5B3. Como ocorreu com as variações *Atv*, a abordagem *WI* teve o número de vizinhos viáveis e tabus aproximadamente multiplicados por 2 nas instâncias 1P130S5B3, 3P95S5B3 e 4P130S5B3, quando comparados com os números originais. Também como nas variações *Atv*, o número de vizinhos que satisfizeram ao critério de aspiração também foi bem semelhante para cada subcaso quando a mesma abordagem é analisada. No entanto, pode ser observado que para a abordagem *IA*, os valores foram bem superiores, especialmente no subcaso *Rec1*. Os valores para o critério de aspiração quando a abordagem *IA* é analisada, também foram bem superiores aos obtidos para as instâncias originais. Como esperado, as produções obtidas para as instâncias perturbadas são inferiores às produções obtidas para as instâncias originais correspondentes, já que o número de recursos capazes de realizar cada atividade foi reduzido.

Para observar melhor o comportamento da produção obtida por cada abordagem, a tabela 7.9 (página 113) mostra a produção final obtida por cada uma das abordagens *TabuPR*, *IA*, *WI* e *WE* para todas as instâncias perturbadas da variação *Rec*. A curva de produção obtida para uma instância perturbada gerada a partir da instância real é mostrada nas figuras 7.13.b, 7.14 e 7.15 (páginas 114 e 115) quando as variações *Rec1*, *Rec2* e *Rec3* são aplicadas, respectivamente. Para instâncias perturbadas geradas a partir da instância 2P112S4B3, as correspondentes curvas de produção são mostradas nas figuras 7.16.b, 7.17 e 7.18 (páginas 116 e 7.18). As figuras 7.19.b, 7.20 e 7.21 (páginas 118 e 119) mostram as curvas de produção para instâncias perturbadas da instância original 3P95S5B3, e finalmente, as figuras 7.22.b, 7.23 e 7.24 (páginas 120 e 121) tratam as instâncias perturbadas relativas à instância original 4P130S5B3.

A tabela 7.9 mostra que a abordagem *IA* produziu os melhores resultados para 99.3% das instâncias perturbadas.

O comportamento da curva de produção de cada abordagem quando aplicadas às variações *Rec2* e *Rec3* foi similar aos seus respectivos comportamentos quando as instâncias originais foram consideradas. Como pode ser visto nas figuras 7.14, 7.15, 7.17, 7.18, 7.20, 7.21, 7.23 e 7.24, as abordagens híbridas *WI* e *WE* não conseguiram escapar de patamares inferiores de produção, mesmo considerando-se que as curvas de produção da abordagem *WI* apresentaram, em algumas iterações, variação na produção total. Outra semelhança é que a abordagem de Busca Tabu pura *TabuPR* produziu melhores soluções que *WI* e *WE*, sendo as soluções produzidas pela abordagem *TabuPR* um pouco inferiores às obtidas pela abordagem híbrida *IA*. Também como no caso das instâncias originais, a abordagem híbrida *IA* foi a que apresentou os melhores resultados e teve a maior taxa de convergência.

Entretanto, quando a variação *Rec1* é considerada, o comportamento das abordagens foi diferente, como pode ser observado nas figuras 7.13.b, 7.16.b, 7.19.b, 7.22.b e 7.24. Para que mais detalhes do princípio da execução das abordagens pudessem ser observados, as figuras 7.13.a, 7.16.a, 7.19.a e 7.22.a (páginas 114, 116, 118 e 120) mostram os primeiros 360 segundos de execução para as mesmas instâncias apresentadas nas figuras 7.13.b, 7.16.b, 7.19.b e 7.22.b, respectivamente.

Quando a variação *Rec1* é considerada, a abordagem *WI* produziu, consistentemente para todas as instâncias originadas das instâncias 1P130S5B3 e 4P130S5B3, resultados melhores do que os produzidos pela abordagem de Busca Tabu pura *TabuPR*.

Além disso, a produção obtida utilizando a abordagem *TabuPR* foi consideravelmente inferior à melhor produção encontrada, principalmente para a variação *Rec1*. Assim, fica evidenciado também que a abordagem de Busca Tabu pura não é robusta com relação a variações na quantidade de recursos capacitados a executar as atividades.

Já as figuras 7.3.a, 7.6.a, 7.9.a e 7.12.a mostram que, ao contrário do que aconteceu com a variação *Atv90*, a abordagem *WE* não teve um bom desempenho, mesmo considerando-se apenas os instantes iniciais de execução. Por outro lado, a taxa de convergência da abordagem híbrida *IA* é bem superior que a da abordagem pura *TabuPR*, além de produzir os melhores resultados finais.

As tabelas 7.10 e 7.11 (páginas 122 e 123) são similares às tabelas 7.6 e 7.7 quando a variação *Rec* é considerada. Como no caso da variação *Atv*, a melhor solução foi obtida utilizando a abordagem híbrida *IA*.

Ao contrário da variação *Atv*, foi possível obter boas soluções iniciais para todas as instâncias perturbadas da variação *Rec*, já que as soluções iniciais foram melhoradas em torno de 8.3%. Como na variação *Atv*, as soluções obtidas para as instâncias perturbadas da variação *Rec* foram de boa qualidade, pois a diferença entre as melhores soluções e os melhores limitantes foi em torno de 10.4%, mesmo considerando que estes limitantes

foram calculados para as instâncias originais, ou seja, instâncias menos restritivas.

A conclusão que pode ser inferida dos resultados apresentados neste capítulo é que as abordagens híbridas mostraram-se menos sensíveis às mudanças nos dados de entrada, mantendo a qualidade de suas soluções praticamente inalteradas, enquanto que a abordagem de Busca Tabu pura, *TabuPR*, gera resultados de qualidade bastante variável com os dados de entrada. Além disso, esta última técnica, além de ser dependente dos dados de entrada, também é muito dependente das restrições do problema.

Instância	Var	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Atv30	244.0	242.0	240.3	242.4	241.3	238.5	242.9	243.7	242.0	242.0	241.9
	Atv60	227.1	219.7	222.0	219.2	229.8	227.5	230.0	215.2	227.8	225.2	224.3
	Atv90	168.8	181.2	168.6	161.2	183.0	182.0	170.0	178.4	170.5	170.2	173.4
2P112S4B3	Atv30	216.5	216.5	215.6	216.8	215.7	216.1	214.8	216.8	216.2	217.4	216.3
	Atv60	205.2	202.4	203.6	204.2	209.2	198.0	209.8	204.2	197.2	206.5	204.0
	Atv90	163.3	159.7	128.9	147.7	146.1	156.9	155.6	150.9	164.5	163.6	153.7
3P95S5B3	Atv30	220.7	221.1	221.3	220.1	221.7	221.9	219.2	222.6	221.7	222.8	221.3
	Atv60	206.0	201.5	200.5	208.8	198.9	199.5	195.8	192.2	194.2	199.1	199.7
	Atv90	133.6	164.4	165.6	166.5	137.4	157.3	139.5	157.2	153.9	160.2	153.6
4P130S5B3	Atv30	270.9	272.2	272.5	273.1	272.7	271.7	271.6	270.4	271.4	272.3	271.9
	Atv60	238.9	252.3	252.8	240.2	247.9	235.9	245.2	254.6	248.7	246.3	246.3
	Atv90	186.9	205.4	188.0	175.2	171.4	206.4	180.9	204.4	178.5	195.9	189.3

Tabela 7.1: Solução Inicial - Variação Atv

Instância	Var	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Rec1	229.8	231.2	223.2	226.9	227.2	234.2	230.0	228.8	224.6	229.5	228.5
	Rec2	240.3	241.7	240.1	242.2	240.7	240.4	240.9	241.9	241.6	240.4	241.0
	Rec3	243.5	243.2	243.5	243.2	242.1	242.8	244.0	243.4	243.5	242.9	243.2
2P112S4B3	Rec1	206.2	202.5	206.5	210.1	208.6	207.9	205.5	208.5	209.1	194.5	205.9
	Rec2	216.8	213.6	215.9	215.9	214.7	214.8	214.2	215.4	214.6	214.3	215.0
	Rec3	216.2	216.5	216.4	216.1	216.2	216.0	216.2	216.7	216.4	215.4	216.2
3P95S5B3	Rec1	218.2	217.1	217.6	210.0	210.6	210.2	210.0	214.6	213.3	209.2	213.1
	Rec2	222.8	222.7	223.7	222.6	222.8	223.1	223.2	222.9	223.2	222.7	223.0
	Rec3	224.0	223.5	223.6	224.2	224.3	224.0	224.1	224.1	223.9	224.1	224.0
4P130S5B3	Rec1	262.2	254.7	260.9	260.5	246.3	260.2	260.3	249.8	263.6	256.1	257.5
	Rec2	272.7	272.2	270.8	271.9	272.9	271.2	272.0	271.5	270.6	271.4	271.7
	Rec3	273.3	273.3	273.6	273.9	273.6	273.7	273.8	273.5	273.0	273.7	273.6

Tabela 7.2: Solução Inicial - Variação Rec

Instância	Var	Lim	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Atv30	Upper2	297.4	298.2	297.5	298.6	296.6	297.8	298.7	299.3	297.6	299.0	298.1
		Upper3	284.0	284.3	284.0	284.1	283.9	284.0	284.2	284.2	283.8	284.3	284.1
	Atv60	Upper2	282.5	279.1	277.8	278.2	281.0	278.0	280.8	281.0	282.0	278.6	279.9
		Upper3	280.6	279.0	278.9	279.2	279.6	278.9	279.6	280.1	280.1	279.1	279.5
	Atv90	Upper2	250.3	262.3	257.3	256.4	263.6	264.9	262.6	268.1	260.4	264.3	261.0
		Upper3	253.8	268.2	262.6	261.5	269.9	271.4	269.1	274.1	266.8	270.9	266.8
2P112S4B3	Atv30	Upper2	287.3	289.5	288.9	289.0	290.7	291.4	286.8	288.9	290.6	290.9	289.4
		Upper3	267.9	268.6	268.5	268.7	269.1	269.3	268.2	268.6	269.1	269.2	268.7
	Atv60	Upper2	273.9	262.8	268.2	264.3	275.6	264.1	273.0	264.4	266.0	265.9	267.8
		Upper3	265.7	260.4	262.2	261.4	265.0	261.6	264.8	260.0	261.2	261.9	262.4
	Atv90	Upper2	243.6	248.7	236.4	239.0	237.9	248.7	251.1	247.5	247.0	245.1	244.5
		Upper3	248.5	253.7	241.8	243.7	242.5	253.8	256.3	252.4	252.5	249.6	249.5
3P95S5B3	Atv30	Upper2	268.0	268.6	267.9	267.4	267.8	269.4	265.2	267.5	267.0	267.1	267.6
		Upper3	255.8	256.0	255.9	255.8	255.8	256.2	255.5	258.3	255.9	255.8	256.1
	Atv60	Upper2	258.2	254.6	255.6	257.6	260.1	253.2	256.4	251.4	255.2	256.3	255.9
		Upper3	253.9	252.8	253.1	253.5	254.0	252.3	253.4	251.8	253.1	253.2	253.1
	Atv90	Upper2	237.2	245.4	247.1	244.3	237.7	239.6	237.7	241.8	241.4	247.6	242.0
		Upper3	241.4	249.5	250.5	248.7	242.6	244.8	243.1	247.1	246.5	250.0	246.4
4P130S5B3	Atv30	Upper2	352.3	357.8	355.2	357.3	355.5	352.9	354.5	353.5	356.4	357.4	355.3
		Upper3	332.0	333.1	332.4	333.0	332.7	331.8	332.6	332.2	332.8	332.9	332.5
	Atv60	Upper2	352.3	337.9	326.8	325.8	336.0	327.1	333.3	334.9	328.2	327.9	333.0
		Upper3	333.2	304.7	324.7	325.1	327.3	325.7	327.5	328.2	325.6	325.7	324.8
	Atv90	Upper2	297.8	305.9	315.1	304.3	295.2	304.3	310.1	311.6	306.6	306.7	305.8
		Upper3	302.9	313.0	321.1	311.6	298.1	311.8	317.5	319.0	314.2	314.2	312.3

Tabela 7.3: Limitantes Duais - Variação Atv

Instância	Var	Abordagem	It	Vizinhos	Tabu	AC	Produção	Tempo
1P130S5B3	Atv30	TabuPR	225.6	2996123.2	242413.6	57.5	251.2	3625.4
		IA	236.6	4641681.6	652327.6	388.1	**256.8**	3591.0
		WE	176.8	880.5	1723.0	0.0	244.1	3608.2
		WI	205.5	3386.7	390.1	15.4	248.1	3615.1
	Atv60	TabuPR	257.5	2809663.6	330153.7	60.6	239.3	3636.8
		IA	255.8	4512984.8	607569.6	516.6	**246.2**	3591.4
		WE	177.8	881.2	1733.0	0.0	234.3	3609.6
		WI	231.6	3366.6	421.5	35.6	240.8	3609.9
	Atv90	TabuPR	901.3	2819540.8	241760.3	241.9	219.8	3626.7
		IA	658.3	4665215.6	661911.3	2767.5	**233.6**	3585.7
		WE	177.4	880.9	1729.0	0.0	218.3	3611.9
		WI	277.2	3366.3	642.0	69.1	227.1	3609.1
2P112S4B3	Atv30	TabuPR	475.8	5832065.6	502098.3	130.7	240.0	3616.9
		IA	242.3	3235973.2	512748.1	533.0	**241.5**	3594.2
		WE	177.0	881.9	1725.0	0.0	217.7	3609.6
		WI	152.0	2348.0	306.3	22.6	221.4	3614.3
	Atv60	TabuPR	393.9	4634970.4	397117.4	100.1	224.1	3628.3
		IA	227.8	3196366.8	475673.6	528.8	**228.0**	3613.1
		WE	122.9	610.5	1184.0	0.0	210.1	3607.7
		WI	174.3	2351.9	362.5	41.7	218.1	3617.0
	Atv90	TabuPR	968.4	4568142.8	443120.2	283.7	207.4	3622.3
		IA	531.3	3183979.2	475261.8	2292.8	**218.4**	3603.3
		WE	124.5	617.3	1200.0	0.0	198.1	3616.0
		WI	232.7	2364.2	579.3	88.5	206.7	3611.4
3P95S5B3	Atv30	TabuPR	300.9	5817150.0	310523.0	63.7	231.8	3626.8
		IA	289.5	6633105.6	908268.5	340.2	**235.2**	3606.4
		WE	173.4	866.7	1689.0	0.0	222.5	3607.0
		WI	195.8	3441.4	391.8	13.9	225.9	3610.1
	Atv60	TabuPR	423.0	6042788.8	503045.2	92.4	220.3	3622.2
		IA	337.0	6102475.2	907960.1	717.1	**228.6**	3606.6
		WE	178.7	888.0	1742.0	0.0	216.5	3608.8
		WI	241.7	3538.6	480.7	39.4	222.2	3611.2
	Atv90	TabuPR	1172.9	5664760.8	438345.9	341.0	211.4	3612.1
		IA	526.3	5885021.6	922882.8	1470.8	**220.3**	3608.2
		WE	178.8	888.8	1743.0	0.0	201.7	3609.9
		WI	260.0	3288.8	636.1	66.4	212.6	3610.5
4P130S5B3	Atv30	TabuPR	205.4	2742140.4	262723.5	47.7	283.0	3635.6
		IA	212.3	4075359.2	540468.3	443.5	**293.3**	3614.0
		WE	176.6	879.7	1721.0	0.0	274.1	3611.9
		WI	208.7	3362.9	381.8	22.8	279.9	3610.5
	Atv60	TabuPR	276.8	2650768.4	309114.3	67.1	267.1	3642.1
		IA	239.1	3646393.6	465870.0	493.5	**277.7**	3332.4
		WE	177.4	879.9	1729.0	0.0	262.4	3611.7
		WI	244.2	3372.7	447.3	45.7	273.0	3609.9
	Atv90	TabuPR	893.4	2637739.0	200932.5	255.2	241.5	3640.5
		IA	597.0	4085759.2	517132.4	2458.1	**268.6**	3608.1
		WE	176.4	878.8	1719.0	0.0	244.3	3611.3
		WI	292.3	3388.1	642.9	85.9	258.8	3606.5

Tabela 7.4: Dados Quantitativos (Média) - Variação Atv - H1

Instância	Var	Abordagem	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Atv30	TabuPR	251.5	251.6	251.4	250.7	250.6	249.4	252.1	252.5	249.7	252.9	251.2
		IA	254.3	257.2	256.1	257.6	256.0	257.0	257.5	256.8	257.1	258.1	256.8
		WE	243.8	243.6	242.8	244.2	244.0	242.4	245.5	245.7	245.0	244.1	244.1
		WI	248.7	248.7	246.2	248.7	246.6	246.0	250.1	248.4	249.3	247.9	248.1
	Atv60	TabuPR	244.4	233.8	239.1	238.6	240.9	236.8	243.2	235.6	240.6	240.1	239.3
		IA	249.4	243.6	243.0	245.3	247.7	246.7	247.2	244.0	248.5	246.6	246.2
		WE	233.2	234.4	234.5	235.1	236.1	231.9	234.9	233.4	233.9	235.7	234.3
		WI	240.6	239.4	239.6	242.4	242.6	241.0	238.1	241.2	241.6	241.1	240.8
	Atv90	TabuPR	213.6	222.6	216.2	218.6	215.7	224.0	219.4	227.4	221.7	219.0	219.8
		IA	223.9	232.2	232.1	229.4	235.9	234.2	237.2	241.1	229.4	240.7	233.6
		WE	199.5	222.5	218.2	212.9	223.4	227.0	222.2	223.3	222.2	211.6	218.3
		WI	213.8	230.0	223.0	221.5	231.2	235.7	229.8	230.2	228.3	227.4	227.1
2P112S4B3	Atv30	TabuPR	240.9	237.9	239.4	239.6	239.4	236.4	241.0	243.3	238.6	243.3	240.0
		IA	242.8	243.5	239.6	240.7	241.0	244.6	240.1	242.7	238.8	240.7	241.5
		WE	217.4	217.4	217.2	217.2	217.9	217.6	217.4	218.8	218.0	217.8	217.7
		WI	224.4	221.8	217.8	219.2	220.7	221.4	221.8	222.6	221.7	222.4	221.4
	Atv60	TabuPR	227.2	219.2	224.1	226.8	225.5	219.4	226.3	221.6	224.6	226.1	224.1
		IA	228.3	225.2	229.6	225.7	231.1	225.1	232.0	225.6	229.2	228.4	228.0
		WE	213.0	209.2	209.0	205.5	213.0	209.6	214.8	209.0	207.8	209.6	210.1
		WI	223.3	215.1	217.8	214.6	221.5	217.6	218.8	217.4	215.7	219.7	218.1
	Atv90	TabuPR	207.0	220.0	202.9	202.9	200.1	215.4	211.1	207.3	206.0	201.4	207.4
		IA	219.9	221.7	206.2	215.0	208.7	221.5	227.8	219.8	223.9	219.8	218.4
		WE	197.0	201.5	191.8	196.3	192.1	200.5	199.6	200.9	201.1	200.5	198.1
		WI	207.7	209.8	200.8	204.2	201.9	210.7	209.2	209.2	208.4	205.5	206.7
3P95S5B3	Atv30	TabuPR	229.8	233.8	232.2	232.1	234.1	232.0	228.5	230.7	232.7	231.6	231.8
		IA	236.1	234.9	236.7	234.1	234.8	232.8	235.7	236.2	235.5	235.0	235.2
		WE	223.0	222.0	223.0	222.4	222.3	222.2	221.6	222.7	222.6	223.5	222.5
		WI	224.9	227.6	224.8	225.3	225.6	225.0	226.7	227.5	226.6	225.2	225.9
	Atv60	TabuPR	223.3	219.2	219.0	224.2	218.0	218.6	218.6	220.4	216.8	225.1	220.3
		IA	231.3	228.9	228.0	231.5	226.5	229.3	228.6	225.2	228.1	228.3	228.6
		WE	217.1	216.2	216.7	216.9	218.2	215.5	215.7	213.9	217.0	218.2	216.5
		WI	223.6	221.5	221.7	221.7	223.6	221.9	222.9	219.7	221.5	223.8	222.2
	Atv90	TabuPR	204.2	212.8	218.5	214.9	208.7	209.7	209.1	211.8	207.4	217.3	211.4
		IA	215.5	222.9	224.5	223.0	216.3	218.6	218.6	222.6	218.6	222.7	220.3
		WE	194.9	208.4	207.8	210.5	196.1	198.6	196.4	200.8	203.8	200.1	201.7
		WI	206.6	216.1	218.1	215.3	210.5	211.3	207.2	212.8	211.3	216.5	212.6
4P130S5B3	Atv30	TabuPR	287.7	282.7	281.3	283.7	281.1	283.0	279.7	281.4	283.9	285.2	283.0
		IA	290.5	294.9	294.4	296.1	293.6	292.3	292.0	291.8	293.5	293.8	293.3
		WE	273.2	274.3	274.7	275.5	275.0	273.9	273.0	272.4	275.0	274.2	274.1
		WI	279.8	277.5	280.4	280.2	282.1	279.1	279.2	280.0	280.9	279.9	279.9
	Atv60	TabuPR	267.4	271.2	271.1	261.3	268.8	259.3	266.9	272.3	266.4	266.2	267.1
		IA	274.7	288.1	277.9	276.8	275.8	277.2	273.3	279.9	279.1	274.1	277.7
		WE	261.1	264.6	263.0	262.5	264.3	262.1	261.8	262.5	261.0	261.5	262.4
		WI	274.0	274.7	272.0	273.3	273.7	270.0	272.9	274.7	272.1	272.8	273.0
	Atv90	TabuPR	237.8	240.0	246.8	237.5	229.3	242.6	246.4	246.7	239.2	249.1	241.5
		IA	265.9	271.6	275.5	266.9	254.2	265.5	268.7	275.2	271.7	270.4	268.6
		WE	241.7	244.1	252.0	240.2	237.0	244.2	249.7	247.4	243.3	243.1	244.3
		WI	254.8	258.0	265.1	256.4	246.5	259.2	263.7	263.3	261.3	259.4	258.8

Tabela 7.5: Solução Final - Variação Atv

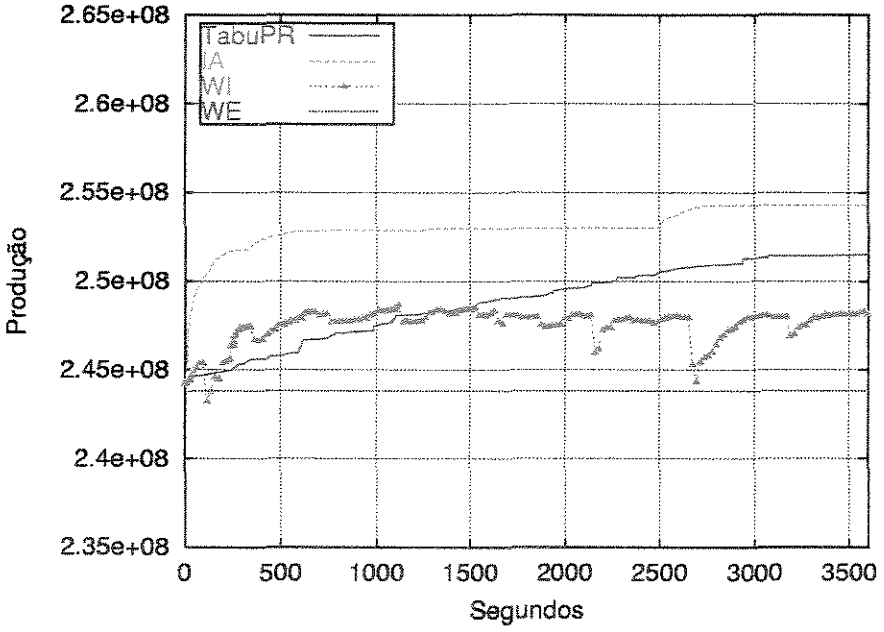


Figura 7.1: Produção × Tempo - 1P130S5B3 (Instância Real) - Variação Atv30

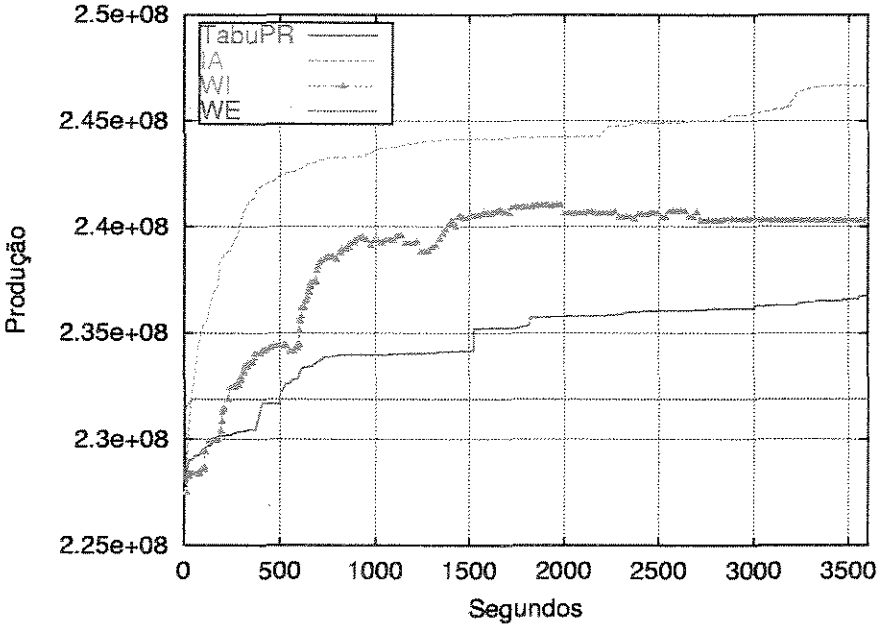
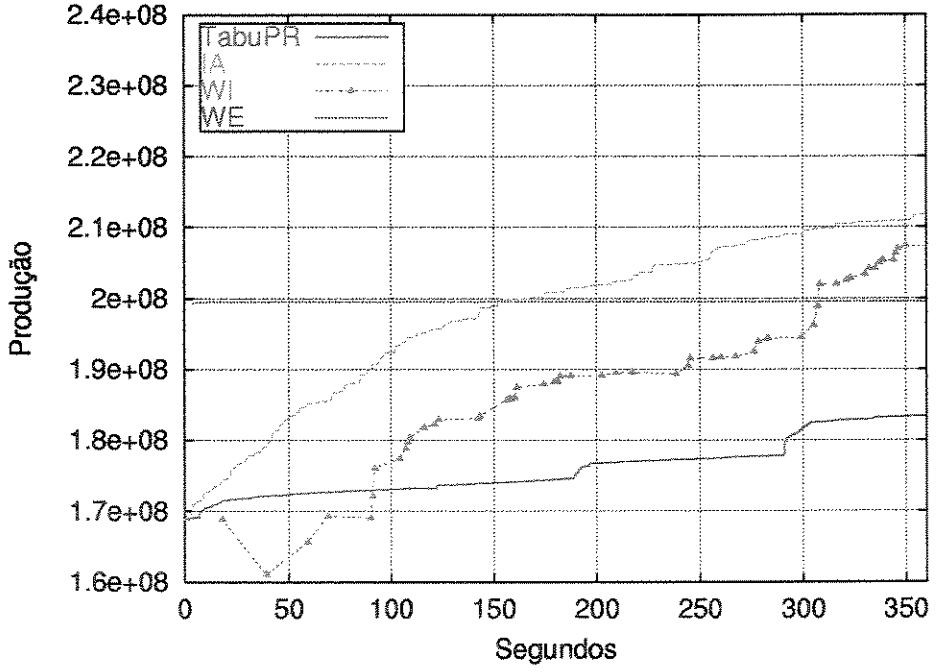
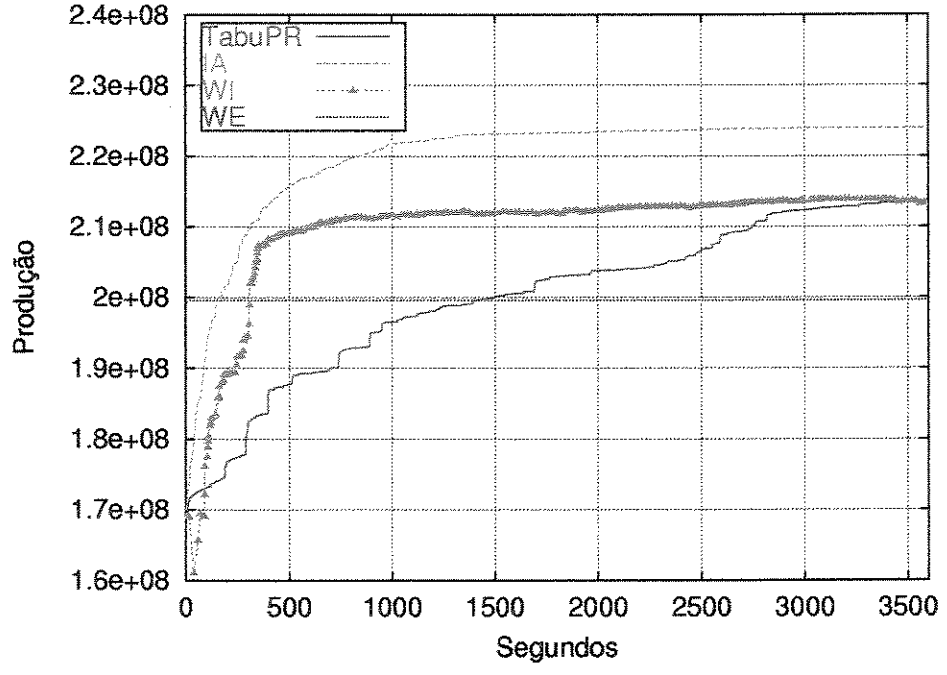


Figura 7.2: Produção × Tempo - 1P130S5B3 (Instância Real) - Variação Atv60



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.3: Produção × Tempo - 1P130S5B3 (Instância Real) - Variação Atv90

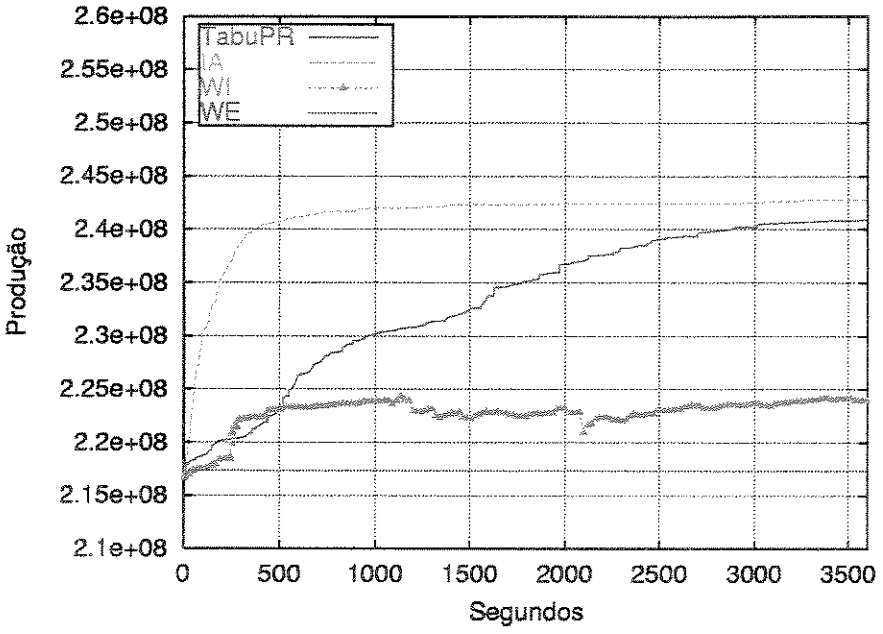


Figura 7.4: Produção × Tempo - 2P112S4B3 - Variação Atv30

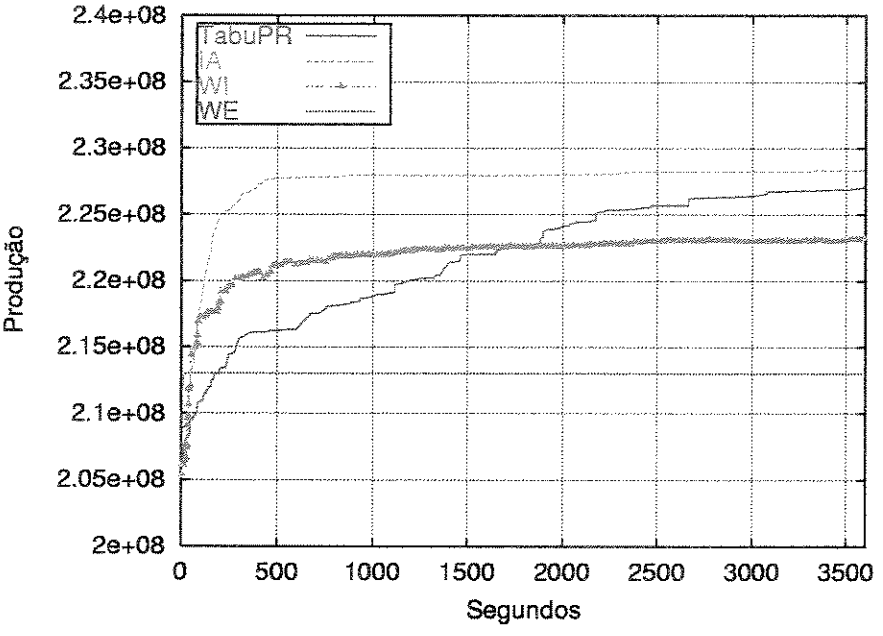
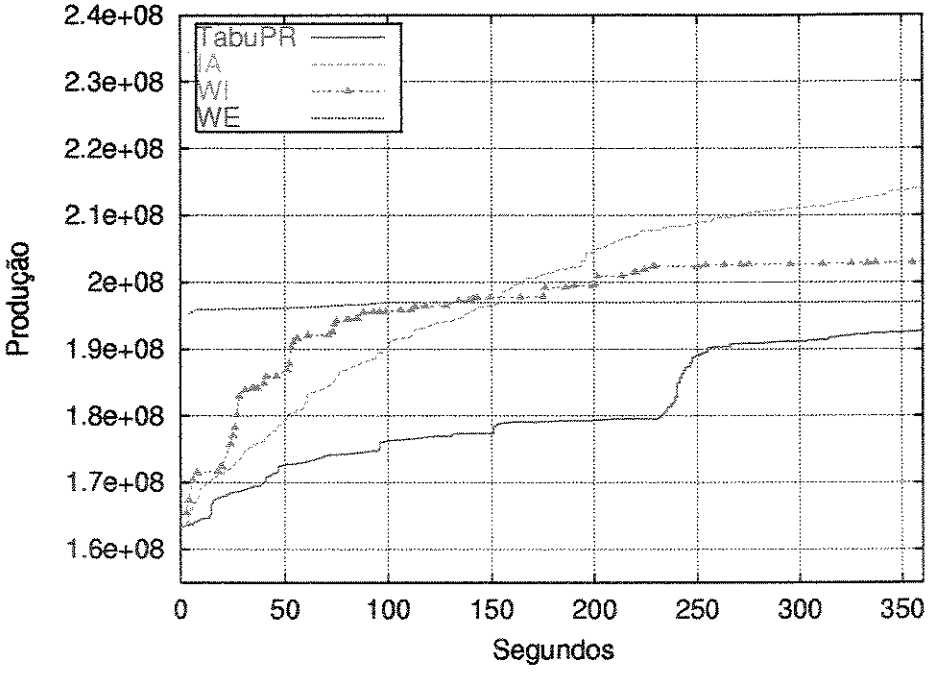
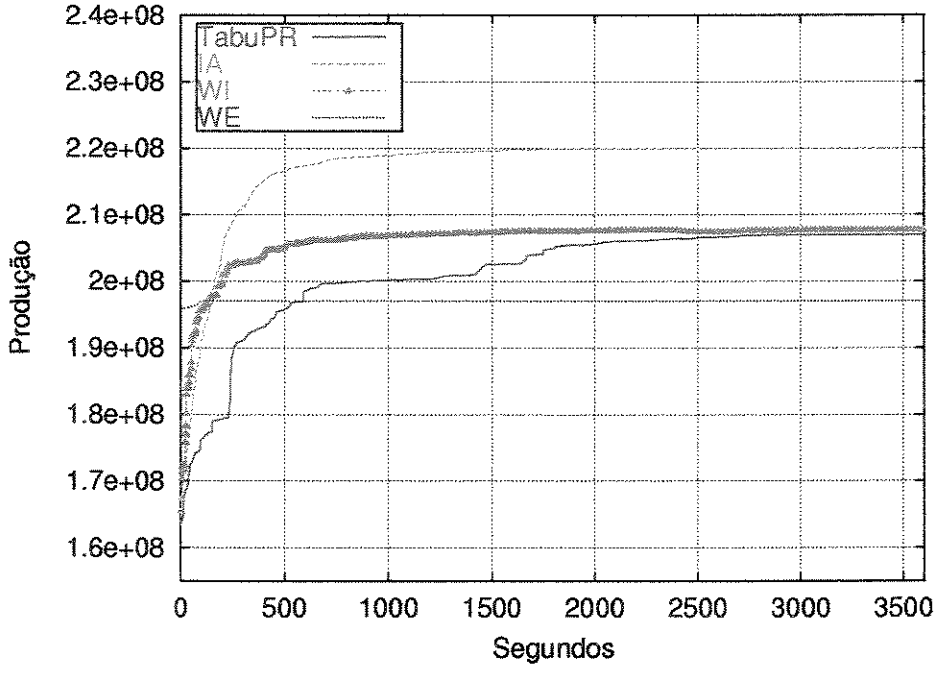


Figura 7.5: Produção × Tempo - 2P112S4B3 - Variação Atv60



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.6: Produção × Tempo - 2P112S4B3 - Variação Atv90

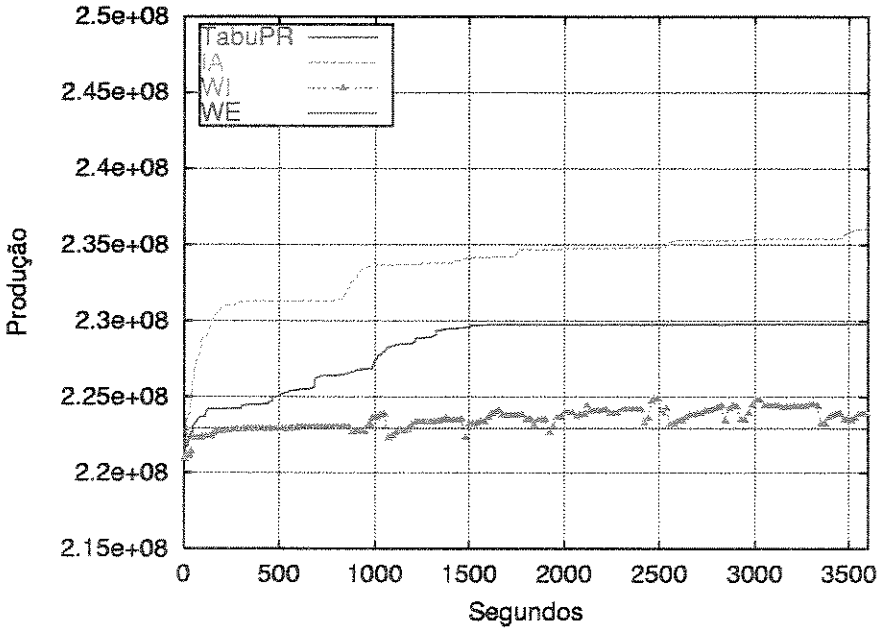


Figura 7.7: Produção × Tempo - 3P95S5B3 - Variação Atv30

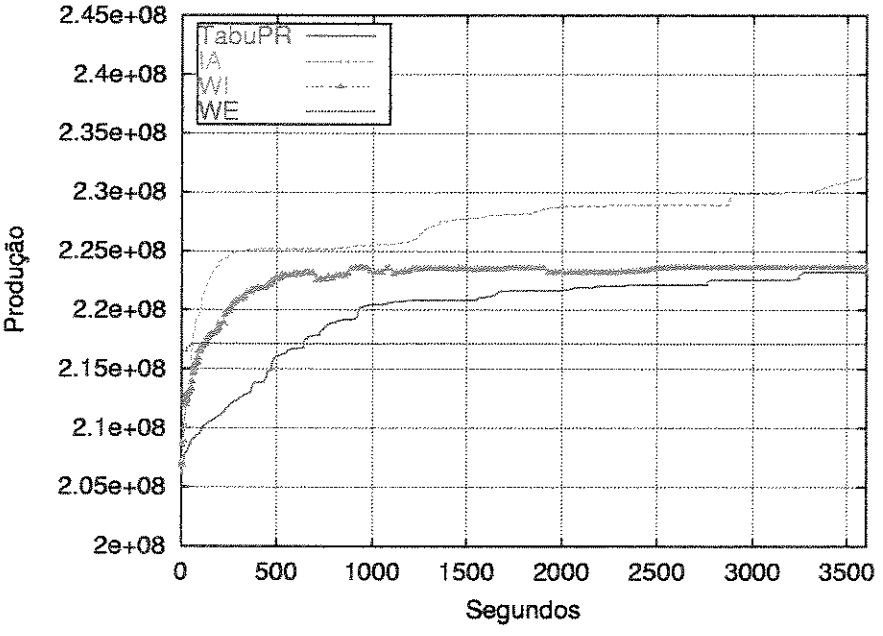
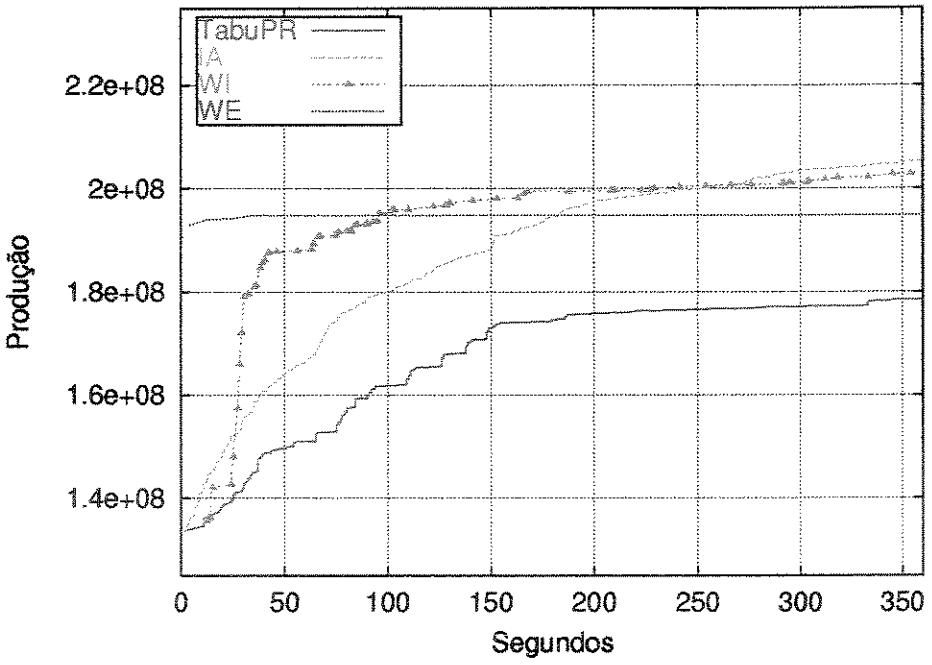
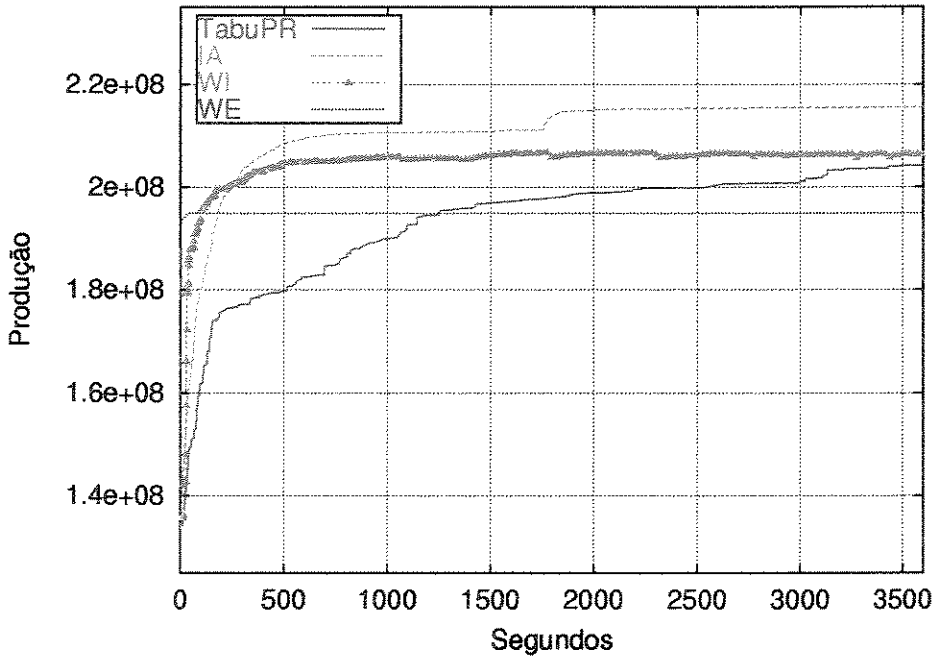


Figura 7.8: Produção × Tempo - 3P95S5B3 - Variação Atv60



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.9: Produção × Tempo - 3P95S5B3 - Variação Atv90

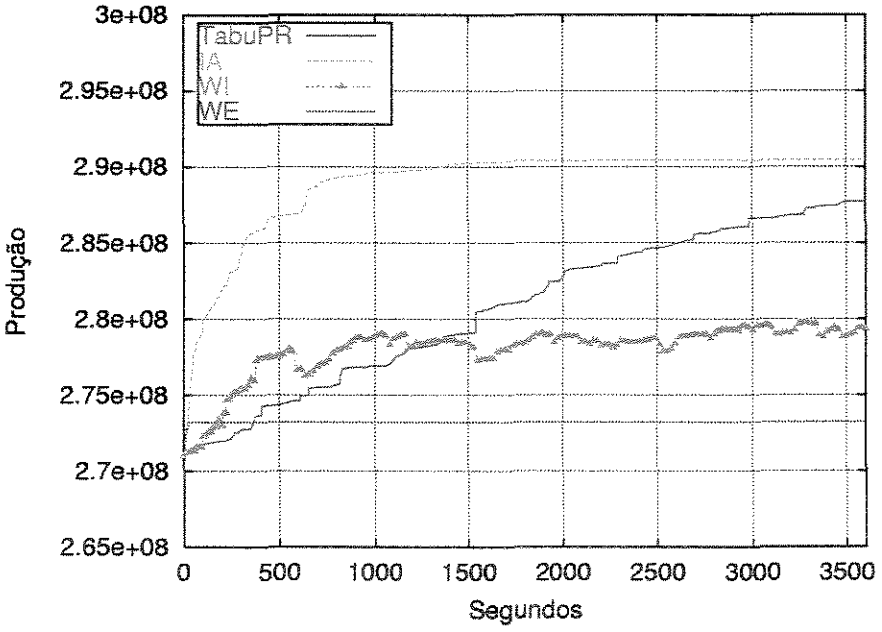


Figura 7.10: Produção × Tempo - 4P130S5B3 - Variação Atv30

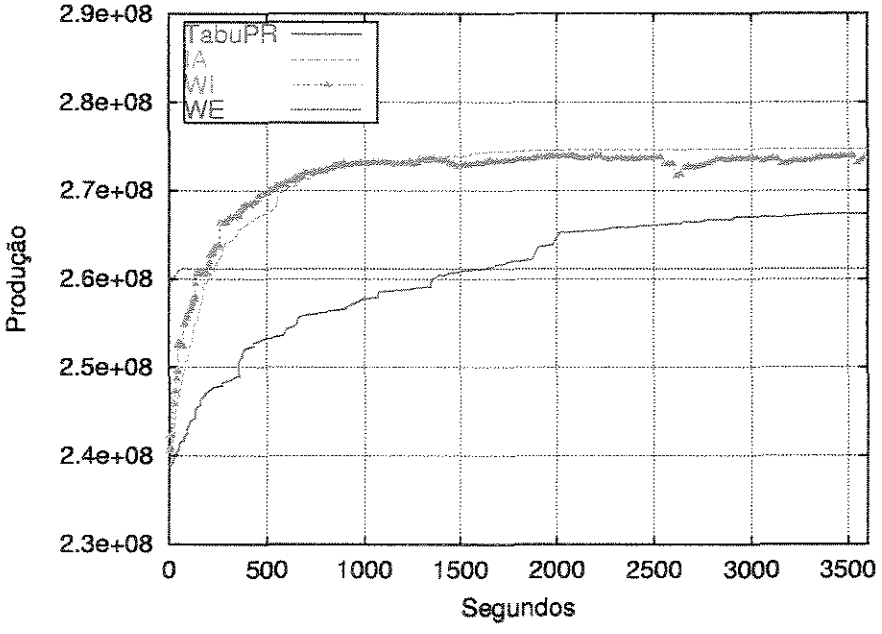
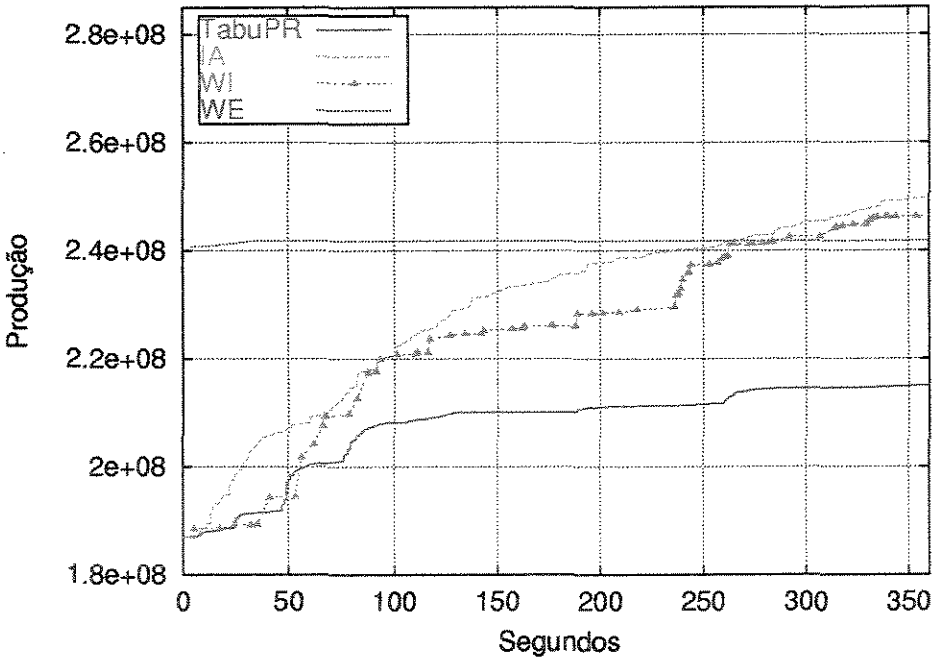
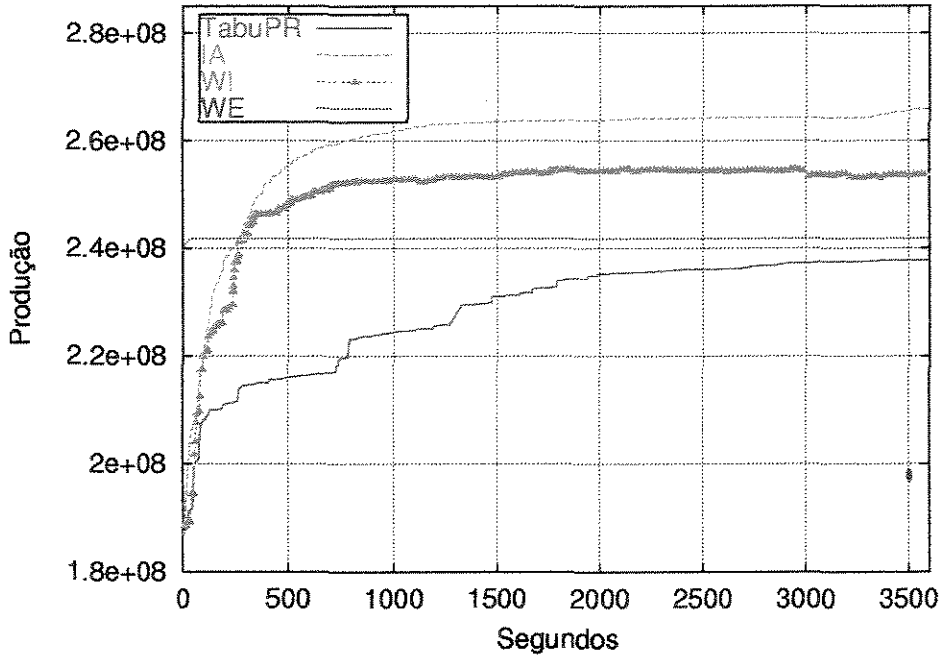


Figura 7.11: Produção × Tempo - 4P130S5B3 - Variação Atv60



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.12: Produção × Tempo - 4P130S5B3 - Variação Atv90

Instância	Var	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Atv30	4.2	6.3	6.5	6.3	6.1	7.8	6.0	5.4	6.2	6.7	6.1
	Atv60	9.8	10.9	9.5	11.9	7.8	8.4	7.5	13.3	9.1	9.5	9.8
	Atv90	32.7	28.1	37.7	42.4	28.9	29.5	39.5	35.1	34.5	41.4	35.0
2P112S4B3	Atv30	12.1	12.5	11.1	11.0	11.8	13.2	12.2	12.2	10.5	11.9	11.8
	Atv60	11.3	11.3	12.8	11.1	10.5	13.7	10.6	10.5	16.2	10.6	11.9
	Atv90	34.7	38.8	60.0	45.6	42.8	41.2	46.4	45.7	36.1	34.4	42.6
3P95S5B3	Atv30	7.0	6.3	6.9	6.3	5.9	4.9	7.6	6.1	6.2	5.5	6.3
	Atv60	12.3	13.6	13.7	10.8	13.9	14.9	16.7	17.1	17.5	14.7	14.5
	Atv90	61.3	35.6	35.6	33.9	57.4	39.0	56.7	41.5	42.1	39.0	44.2
4P130S5B3	Atv30	7.2	8.4	8.1	8.4	7.7	7.6	7.5	7.9	8.1	7.9	7.9
	Atv60	15.0	14.2	9.9	15.3	11.2	17.5	11.4	9.9	12.2	11.3	12.8
	Atv90	42.3	32.2	46.5	52.3	48.3	28.6	48.5	34.6	52.2	38.0	42.4

Tabela 7.6: Percentual de melhora da solução inicial - Variação Atv

Instância	Var	Lim	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Atv30	Upper2	14.5	13.7	13.9	13.7	13.7	13.7	13.8	14.2	13.6	13.7	13.9
		Upper3	10.5	9.5	9.8	9.4	9.8	9.5	9.4	9.7	9.4	9.2	9.6
	Atv60	Upper2	11.7	12.7	12.5	11.8	11.8	11.3	12.0	13.2	11.9	11.5	12.0
		Upper3	11.1	12.7	12.9	12.1	11.4	11.6	11.6	12.9	11.3	11.6	11.9
	Atv90	Upper2	10.5	11.5	9.8	10.5	10.5	11.0	9.7	10.1	11.9	8.9	10.4
		Upper3	11.8	13.4	11.6	12.3	12.6	13.2	11.9	12.0	14.0	11.1	12.4
2P112S4B3	Atv30	Upper2	15.5	15.9	17.1	16.7	17.1	16.1	16.0	15.8	17.8	16.4	16.4
		Upper3	9.4	9.3	10.8	10.4	10.4	9.2	10.2	9.4	11.3	9.6	10.0
	Atv60	Upper2	16.6	14.3	14.4	14.2	16.1	14.8	15.0	14.7	13.9	14.1	14.8
		Upper3	14.1	13.5	12.5	13.2	12.8	13.9	12.4	13.2	12.3	12.8	13.1
	Atv90	Upper2	9.7	10.9	12.7	10.0	12.3	10.9	9.3	11.2	9.4	10.3	10.7
		Upper3	11.5	12.6	14.7	11.8	13.9	12.7	11.1	12.9	11.3	11.9	12.5
3P95S5B3	Atv30	Upper2	11.9	12.5	11.6	12.5	12.3	13.6	11.1	11.7	11.8	12.0	12.1
		Upper3	7.7	8.2	7.5	8.5	8.2	9.1	7.7	8.6	8.0	8.1	8.2
	Atv60	Upper2	10.4	10.1	10.8	10.1	12.9	9.5	10.9	10.4	10.6	10.9	10.7
		Upper3	8.9	9.5	9.9	8.7	10.8	9.1	9.8	10.6	9.8	9.8	9.7
	Atv90	Upper2	9.2	9.2	9.2	8.7	9.0	8.8	8.0	8.0	9.4	10.1	8.9
		Upper3	10.7	10.6	10.4	10.3	10.8	10.7	10.1	9.9	11.3	10.9	10.6
4P130S5B3	Atv30	Upper2	17.5	17.6	17.1	17.1	17.4	17.2	17.6	17.4	17.7	17.8	17.4
		Upper3	12.5	11.5	11.4	11.1	11.7	11.9	12.2	12.2	11.8	11.7	11.8
	Atv60	Upper2	22.0	14.7	15.0	15.0	17.9	15.3	18.0	16.4	14.9	16.4	16.6
		Upper3	17.6	5.4	14.4	14.8	15.7	14.9	16.6	14.7	14.3	15.8	14.4
	Atv90	Upper2	10.7	11.2	12.6	12.3	13.9	12.7	13.4	11.7	11.4	11.8	12.2
		Upper3	12.2	13.2	14.2	14.4	14.7	14.9	15.4	13.7	13.5	14.0	14.0

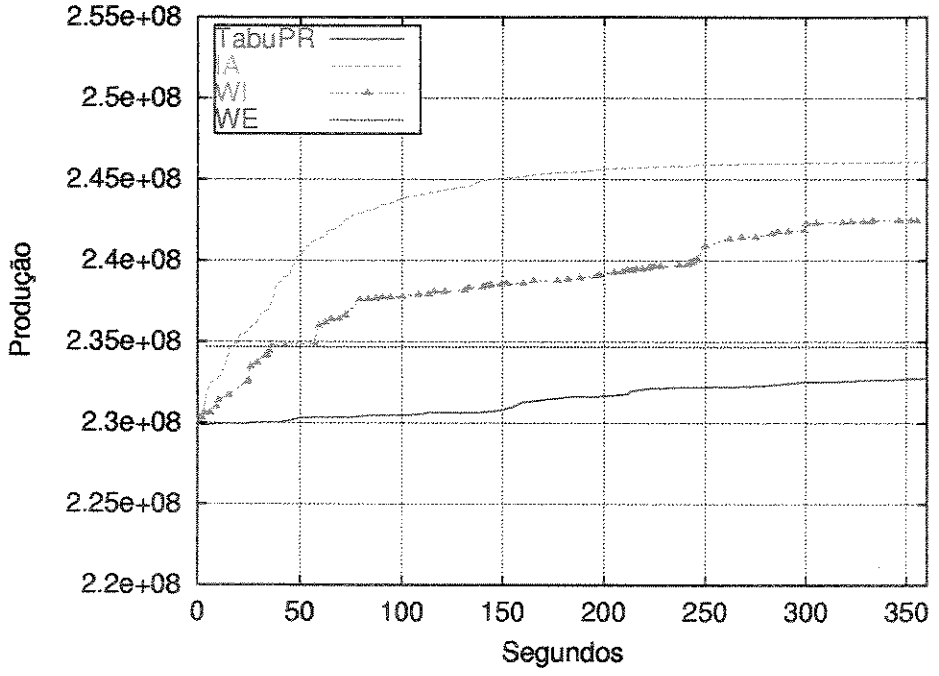
Tabela 7.7: Percentual que a melhor solução está abaixo dos limitantes duais - Variação Atv

Instância	Var	Abordagem	It	Vizinhos	Tabu	AC	Produção	Tempo
1P130S5B3	Rec1	TabuPR	159.9	3414412.8	677937.9	48.5	238.2	3625.1
		IA	554.1	3770679.6	579824.0	584.8	**250.6**	3573.7
		WE	176.2	881.4	871.0	0.0	234.4	3606.3
		WI	240.9	3970.7	287.3	29.9	242.3	3609.4
	Rec2	TabuPR	191.6	3021953.8	385791.9	40.5	251.2	3622.3
		IA	342.2	4224825.6	649083.4	390.7	**258.1**	3571.2
		WE	177.0	882.1	875.0	0.0	242.1	3607.7
		WI	214.3	3635.7	266.6	17.9	247.5	3607.5
	Rec3	TabuPR	222.1	3411922.0	308191.3	57.2	253.1	3623.5
		IA	262.7	4363241.2	672869.1	359.5	**258.3**	3569.6
		WE	177.0	882.2	875.0	0.0	243.9	3610.4
		WI	207.8	3496.4	268.0	14.9	249.0	3605.5
2P112S4B3	Rec1	TabuPR	270.1	5369573.6	1057102.5	100.3	227.4	3616.4
		IA	431.1	2659797.6	433889.2	903.8	**240.0**	3591.4
		WE	122.8	613.5	604.0	0.0	208.5	3614.0
		WI	185.5	2999.7	237.7	35.3	220.4	3615.1
	Rec2	TabuPR	356.8	5027097.6	590410.1	98.5	239.1	3621.0
		IA	287.0	3010248.0	487981.9	649.0	**245.1**	3600.5
		WE	122.5	611.5	602.5	0.0	215.2	3611.5
		WI	161.5	2499.4	241.6	22.8	221.7	3609.4
	Rec3	TabuPR	402.3	5359570.4	518635.0	113.5	242.2	3615.6
		IA	239.9	3182668.4	514843.3	489.2	**245.7**	3612.5
		WE	121.2	608.5	596.0	0.0	216.2	3607.6
		WI	156.2	2418.7	254.2	21.9	221.9	3610.8
3P95S5B3	Rec1	TabuPR	238.5	6876081.6	1095061.8	68.4	228.5	3617.7
		IA	869.2	4966820.8	947461.1	516.7	**230.8**	3600.0
		WE	173.7	868.4	858.5	0.0	214.6	3614.3
		WI	218.1	4514.2	258.0	24.6	221.5	3610.8
	Rec2	TabuPR	194.5	3342531.2	367985.2	52.0	234.7	3632.2
		IA	535.5	6102645.6	1079376.5	383.8	**237.4**	3601.7
		WE	177.3	888.0	876.5	0.0	223.1	3610.3
		WI	207.1	4153.5	265.8	15.8	227.4	3613.6
	Rec3	TabuPR	250.9	6272734.4	365911.4	57.8	234.2	3620.2
		IA	379.8	6206974.4	1095840.4	378.2	**237.0**	3604.8
		WE	176.9	885.9	874.5	0.0	224.2	3606.1
		WI	201.2	3956.3	267.4	14.0	227.3	3610.6
4P130S5B3	Rec1	TabuPR	151.7	2864549.2	671382.5	44.3	266.5	3629.5
		IA	461.1	3252159.6	461842.4	661.9	**289.9**	3578.4
		WE	176.6	880.9	873.0	0.0	262.7	3613.9
		WI	246.0	3922.3	286.0	36.3	273.8	3607.7
	Rec2	TabuPR	167.5	2786902.8	337914.7	36.6	283.1	3626.2
		IA	292.6	3750419.2	543486.1	464.5	**297.5**	3583.5
		WE	176.8	879.7	874.0	0.0	273.2	3609.4
		WI	220.1	3506.3	264.9	22.7	280.7	3609.6
	Rec3	TabuPR	176.6	2804587.6	275992.9	46.0	286.0	3619.5
		IA	237.8	4010254.0	547506.2	458.1	**297.3**	3606.0
		WE	175.9	880.6	869.5	0.0	274.2	3614.4
		WI	206.8	3495.2	260.1	17.3	280.1	3607.7

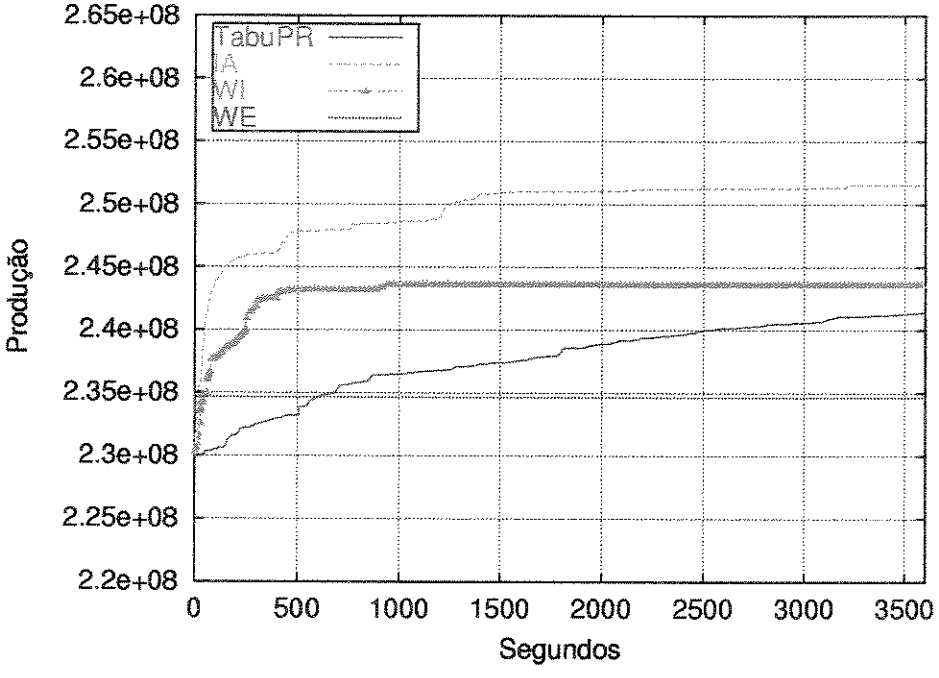
Tabela 7.8: Dados Quantitativos (Média) - Variação Rec - H1

Instância	Var	Abordagem	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Rec1	TabuPR	241.4	241.4	233.3	237.7	237.2	241.4	240.3	236.6	236.5	236.5	238.2
		IA	251.5	255.0	249.8	250.5	247.2	255.0	249.5	251.0	245.3	251.0	250.6
		WE	234.7	236.6	231.9	231.6	234.0	240.1	236.2	233.8	230.4	235.1	234.4
		WI	243.6	243.9	241.6	239.3	241.7	247.6	245.5	241.3	236.5	241.7	242.3
	Rec2	TabuPR	254.7	250.8	250.7	250.8	251.0	250.7	250.1	250.1	251.3	252.0	251.2
		IA	255.8	259.5	258.6	257.9	259.2	258.0	259.2	256.5	259.2	257.5	258.1
		WE	241.9	243.1	239.9	243.2	242.3	242.1	241.6	242.4	242.7	241.6	242.1
		WI	246.2	247.7	247.7	249.4	246.5	247.8	247.1	248.4	247.9	246.0	247.5
	Rec3	TabuPR	257.4	254.0	250.2	255.5	251.4	250.9	254.8	252.2	252.3	252.7	253.1
		IA	259.2	260.4	256.7	258.7	258.4	257.8	258.2	257.4	258.1	257.8	258.3
		WE	244.0	243.1	244.1	243.6	243.6	243.6	245.4	244.3	243.9	243.8	243.9
		WI	248.6	248.8	248.5	249.0	248.3	248.5	250.4	249.4	248.8	249.9	249.0
2P112S4B3	Rec1	TabuPR	235.1	220.4	223.9	231.7	231.5	226.4	226.6	236.6	226.5	214.8	227.4
		IA	239.5	238.3	238.1	241.7	245.0	242.7	239.6	245.0	242.1	228.2	240.0
		WE	209.5	206.1	208.3	212.2	210.1	212.2	209.4	210.4	211.0	195.4	208.5
		WI	225.0	216.4	222.6	224.4	221.4	220.0	222.7	223.1	220.7	207.9	220.4
	Rec2	TabuPR	243.7	242.3	238.5	241.2	238.1	233.9	240.2	236.6	237.6	239.4	239.1
		IA	244.8	245.5	244.2	245.7	245.5	246.9	244.0	244.9	243.3	246.0	245.1
		WE	216.5	214.7	216.9	214.8	213.6	214.8	214.4	215.6	215.5	214.9	215.2
		WI	220.5	219.3	224.1	224.0	221.2	221.4	222.0	222.1	218.7	223.8	221.7
	Rec3	TabuPR	246.2	241.7	244.0	243.7	242.2	243.4	242.5	237.2	237.8	243.6	242.2
		IA	247.1	246.1	243.8	243.3	246.6	246.9	246.8	244.8	244.9	246.3	245.7
		WE	216.3	216.6	215.2	216.1	216.1	216.9	216.9	215.8	216.7	215.9	216.2
		WI	228.1	223.1	221.0	219.7	221.6	219.9	222.0	221.8	220.9	220.8	221.9
3P95S5B3	Rec1	TabuPR	231.9	232.6	232.1	226.5	226.2	226.5	225.9	226.7	229.6	227.5	228.5
		IA	233.0	234.7	233.4	230.0	229.9	226.9	229.2	228.4	231.8	230.5	230.8
		WE	218.5	218.0	219.3	212.2	212.0	212.4	211.8	216.4	214.8	211.1	214.6
		WI	225.3	226.0	224.9	220.2	216.9	221.6	219.5	220.6	221.4	218.2	221.5
	Rec2	TabuPR	237.5	234.9	235.2	234.0	234.8	235.5	235.3	231.1	233.2	235.5	234.7
		IA	238.8	237.0	237.4	239.0	238.4	235.6	236.5	235.8	238.1	237.5	237.4
		WE	222.8	223.0	223.9	222.6	222.9	223.3	223.0	223.1	223.6	223.1	223.1
		WI	228.4	226.6	228.4	227.4	225.9	228.3	227.3	227.1	228.9	227.5	227.4
	Rec3	TabuPR	235.6	234.8	233.8	229.4	226.3	238.3	234.3	236.5	236.8	236.0	234.2
		IA	237.1	237.5	234.9	237.8	237.3	238.1	238.0	238.2	237.3	234.2	237.0
		WE	223.4	224.0	223.7	224.7	224.7	224.2	224.4	224.2	224.0	224.3	224.2
		WI	228.5	228.6	226.4	226.8	227.6	228.0	226.9	228.0	225.6	226.8	227.3
4P130S5B3	Rec1	TabuPR	274.7	263.1	272.7	268.8	254.8	268.0	267.0	260.0	271.3	264.9	266.5
		IA	291.1	290.4	292.8	296.2	285.1	291.3	291.5	279.5	293.0	288.6	289.9
		WE	264.8	260.6	263.9	266.5	254.7	263.7	264.7	259.0	268.6	260.4	262.7
		WI	277.0	272.8	276.9	276.5	268.5	275.6	273.7	267.4	278.5	270.7	273.8
	Rec2	TabuPR	288.9	282.4	281.2	285.7	282.2	281.3	284.3	281.8	282.0	281.3	283.1
		IA	293.2	300.2	293.9	297.7	299.0	297.6	297.8	298.9	296.3	300.0	297.5
		WE	274.4	273.4	272.0	273.8	273.3	273.2	273.2	272.5	272.3	273.3	273.2
		WI	279.9	281.4	278.9	280.6	281.8	281.9	281.3	282.9	278.8	279.3	280.7
	Rec3	TabuPR	292.7	286.1	284.9	283.6	287.2	287.0	286.9	285.1	282.5	283.6	286.0
		IA	295.4	300.7	293.2	300.0	299.2	298.2	295.8	298.6	296.2	295.7	297.3
		WE	274.3	272.9	274.0	274.3	274.6	274.7	274.3	274.8	273.7	274.2	274.2
		WI	279.5	279.4	282.5	278.9	280.3	281.4	280.5	278.4	279.2	280.3	280.1

Tabela 7.9: Solução Final - Variação Rec



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.13: Produção × Tempo - 1P130S5B3 (Instância Real) - Variação Rec1

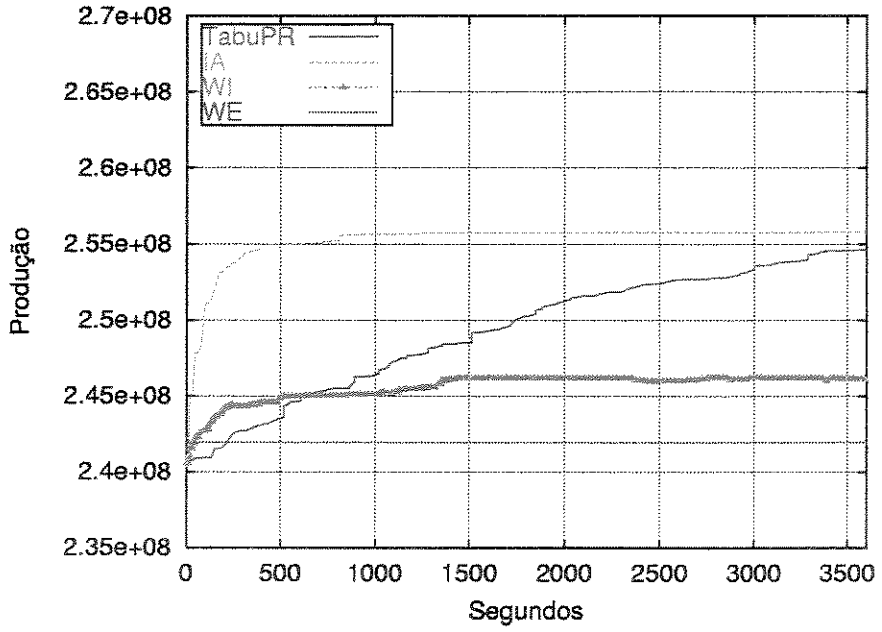


Figura 7.14: Produção × Tempo - 1P130S5B3 (Instância Real) - Variação Rec2

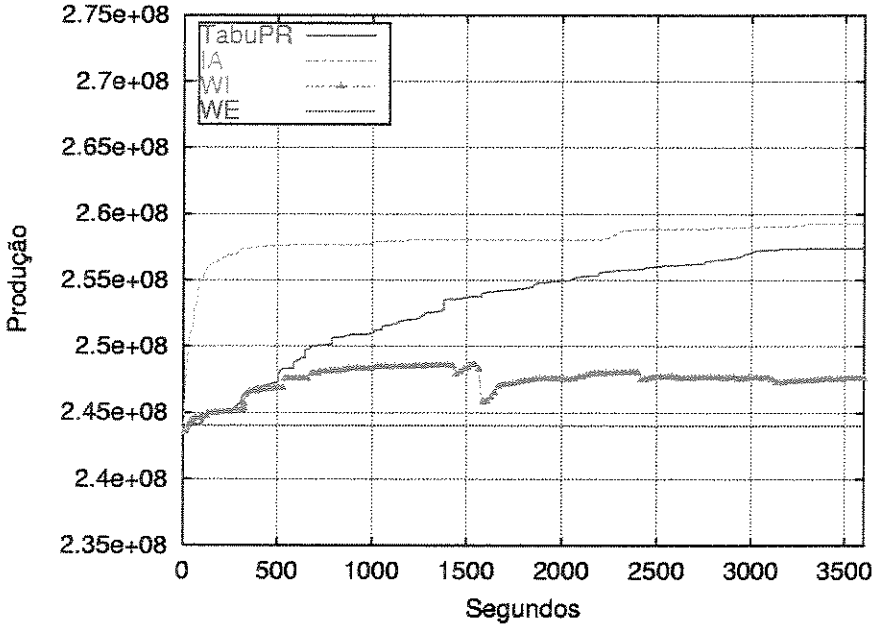
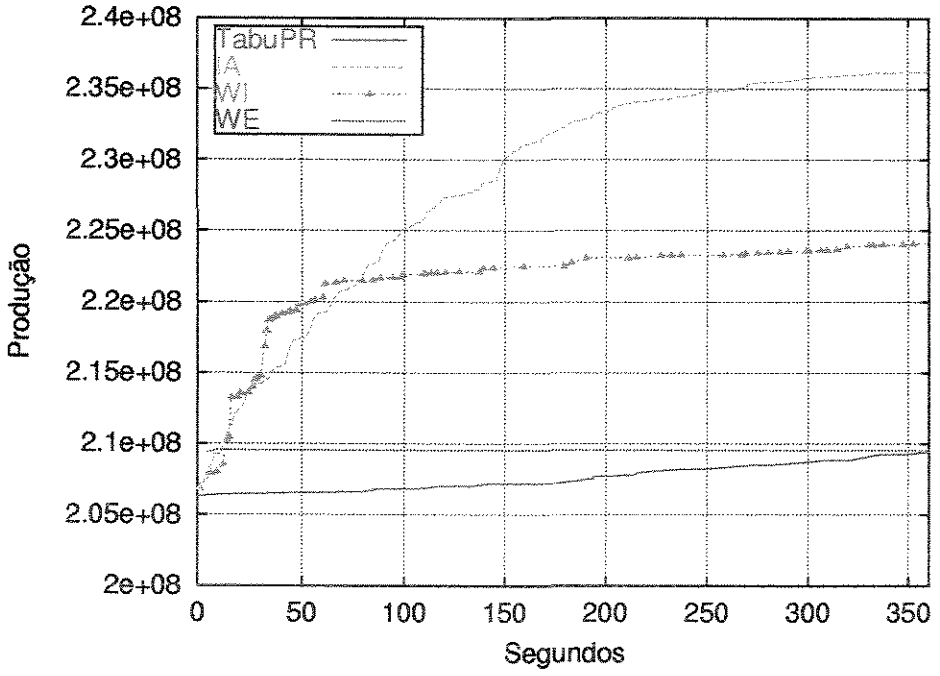
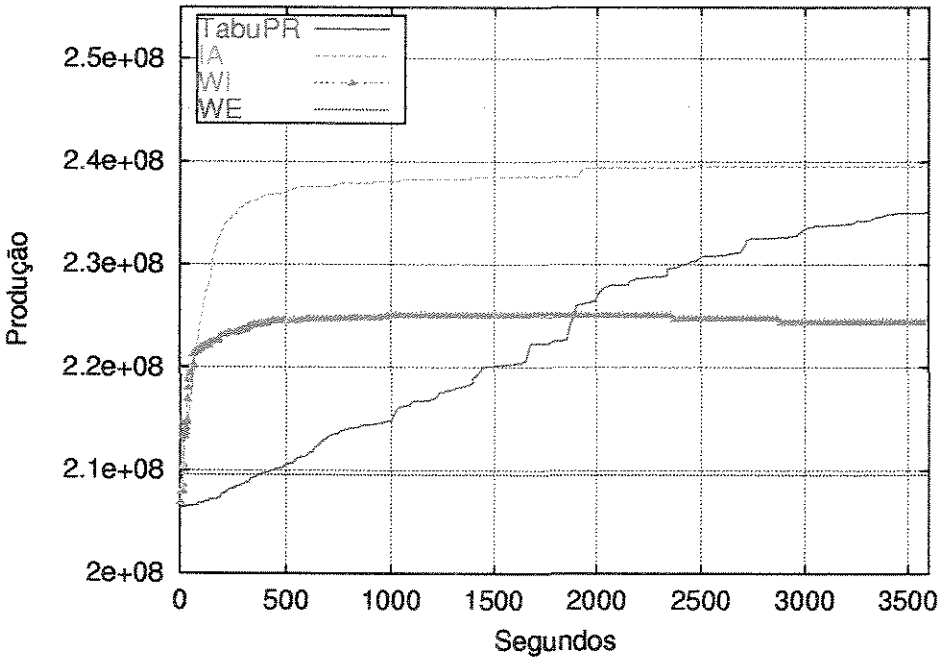


Figura 7.15: Produção × Tempo - 1P130S5B3 (Instância Real) - Variação Rec3



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.16: Produção × Tempo - 2P112S4B3 - Variação Rec1

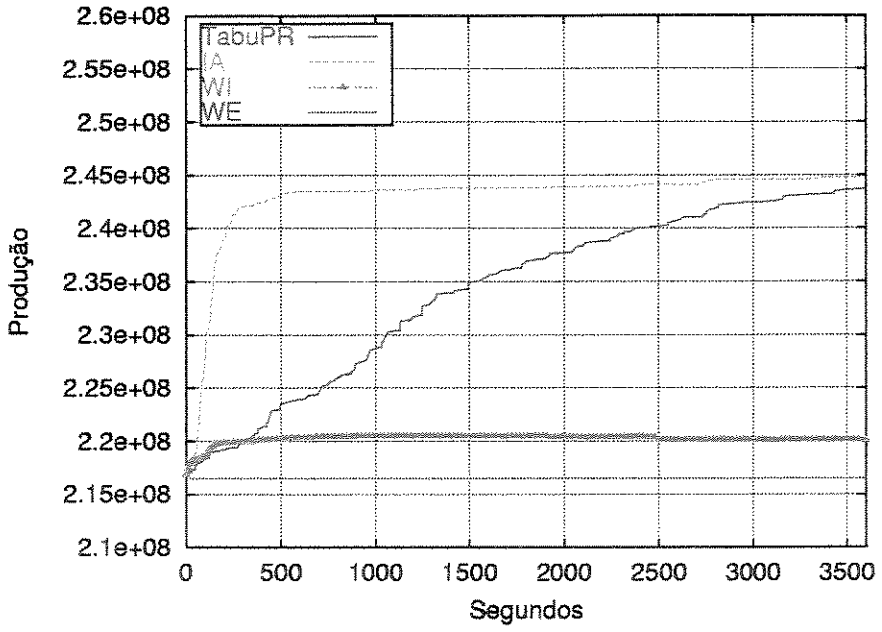


Figura 7.17: Produção × Tempo - 2P112S4B3 - Variação Rec2

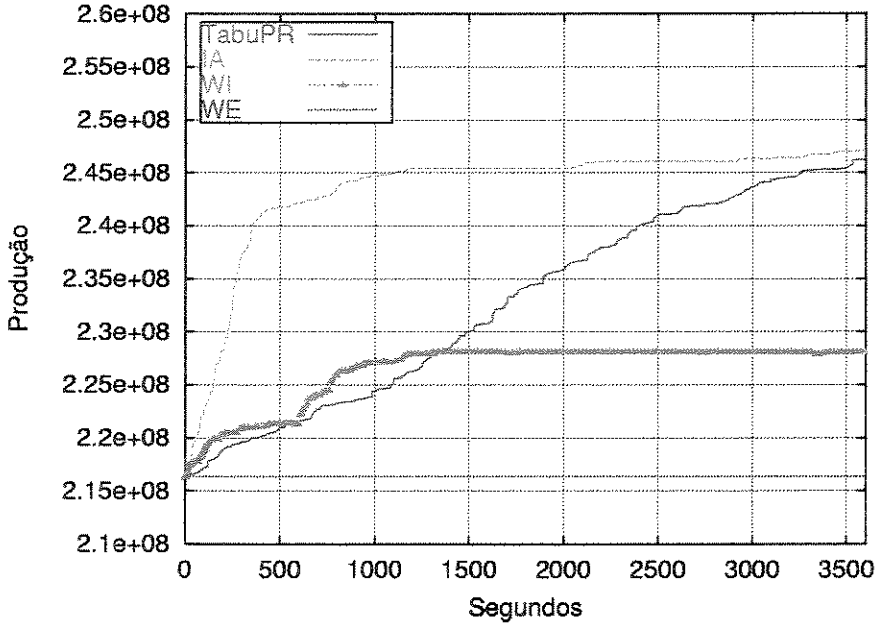
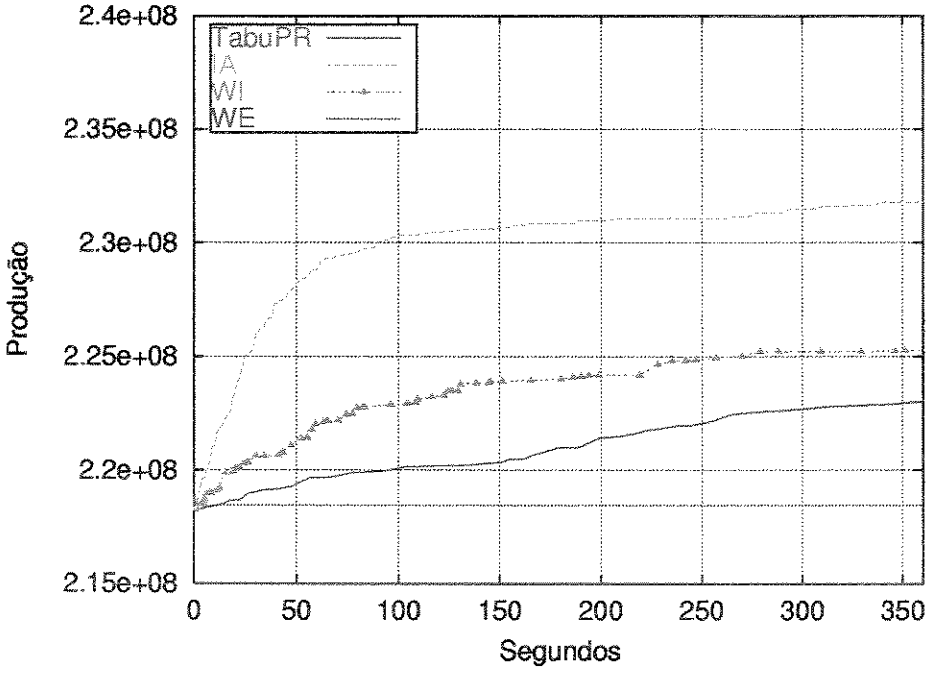
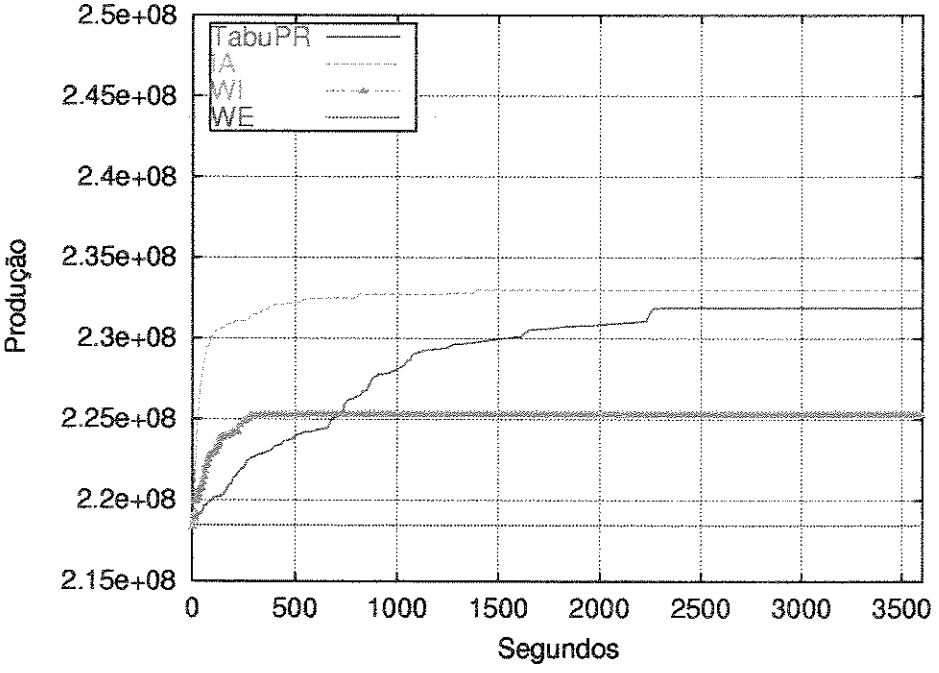


Figura 7.18: Produção × Tempo - 2P112S4B3 - Variação Rec3



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.19: Produção × Tempo - 3P95S5B3 - Variação Rec1

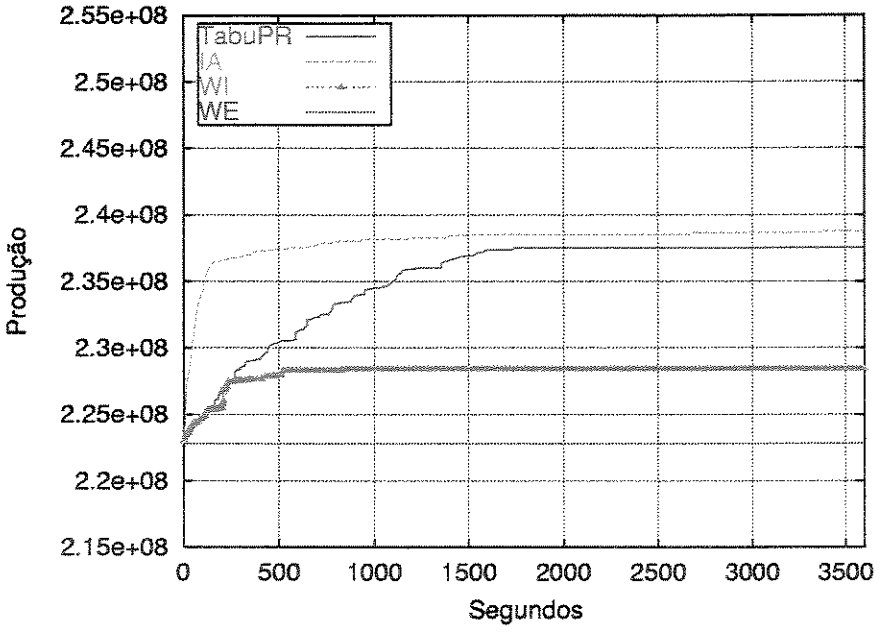


Figura 7.20: Produção × Tempo - 3P95S5B3 - Variação Rec2

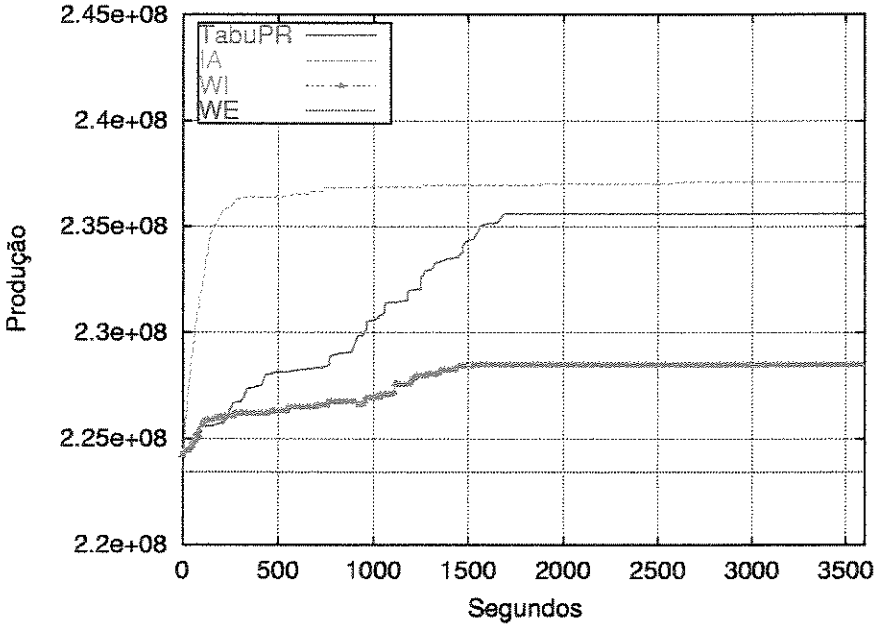
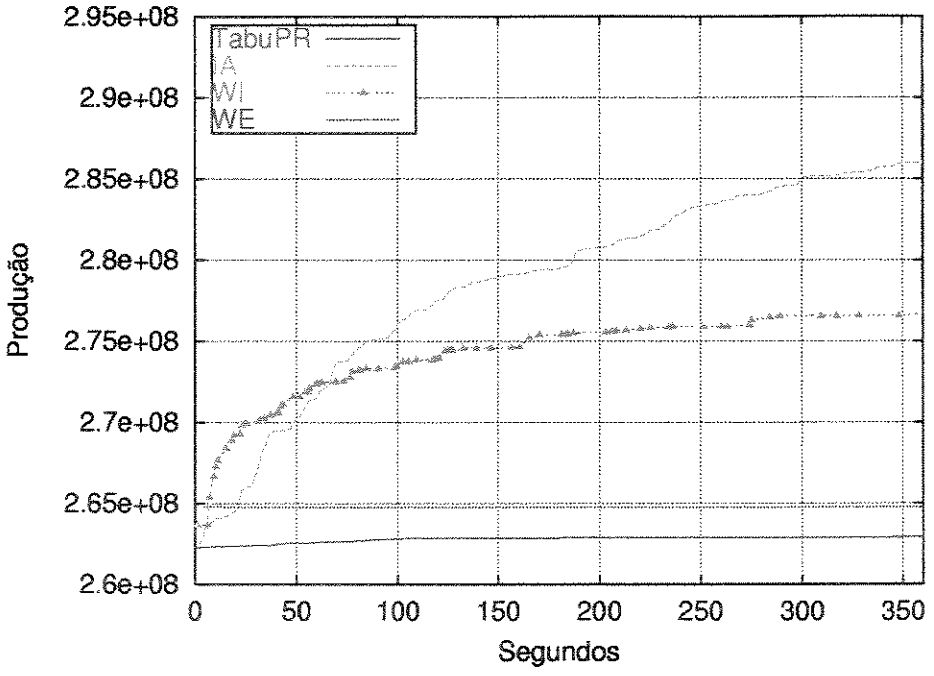
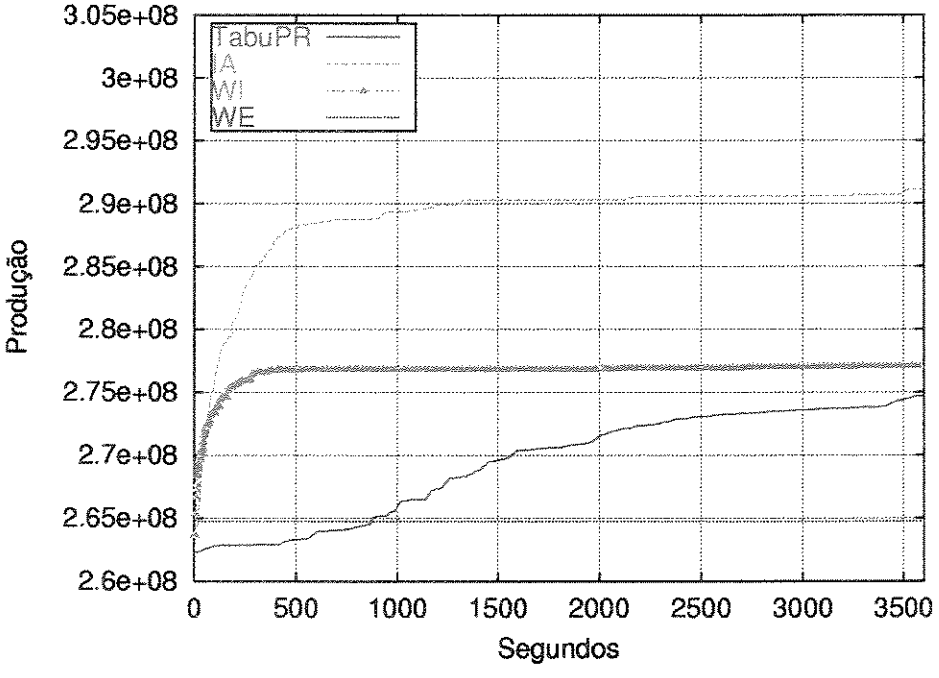


Figura 7.21: Produção × Tempo - 3P95S5B3 - Variação Rec3



(a) Tempo=0...360



(b) Tempo=0...3600

Figura 7.22: Produção × Tempo - 4P130S5B3 - Variação Rec1

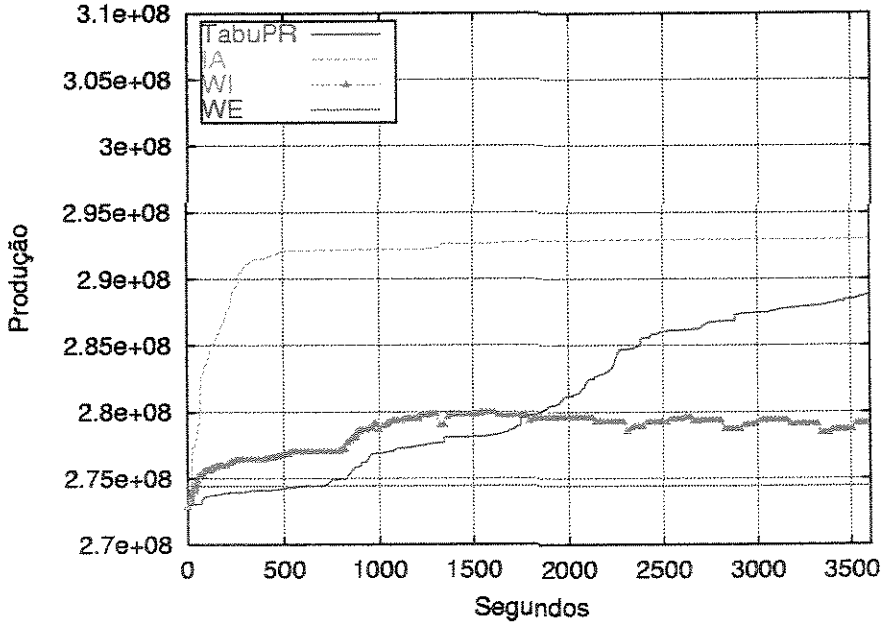


Figura 7.23: Produção × Tempo - 4P130S5B3 - Variação Rec2

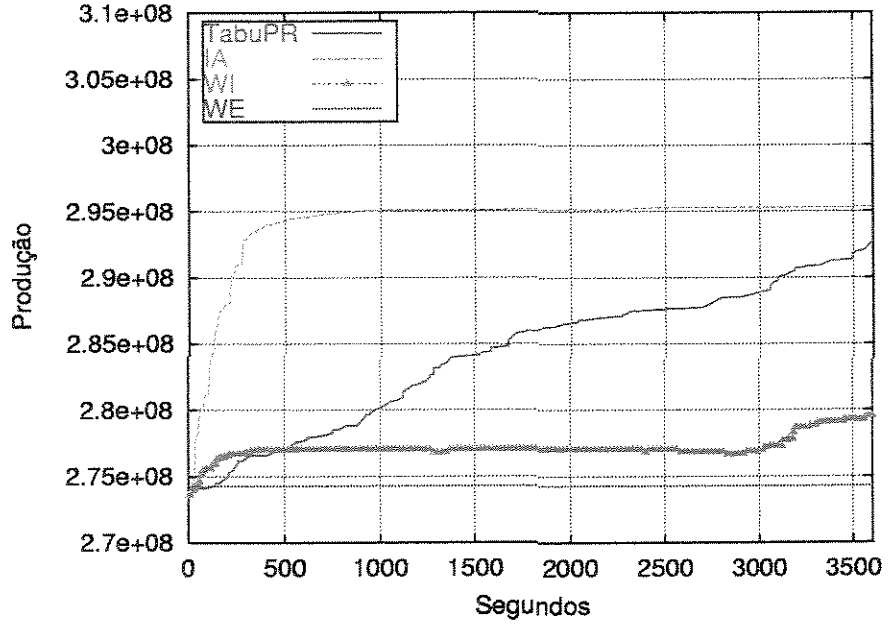


Figura 7.24: Produção × Tempo - 4P130S5B3 - Variação Rec3

Instância	Var	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Rec1	5.0	4.4	14.2	12.4	10.5	1.0	4.4	6.6	10.3	7.0	7.6
	Rec2	6.0	3.8	8.1	7.0	7.7	1.2	0.6	2.4	3.2	3.3	4.3
	Rec3	5.7	4.8	6.9	6.4	6.6	0.5	0.6	2.2	2.3	3.1	3.9
2P112S4B3	Rec1	16.1	17.7	15.3	15.0	17.5	16.7	16.6	17.5	15.8	17.3	16.6
	Rec2	12.9	15.0	13.1	13.8	14.3	15.0	14.0	13.7	13.4	14.8	14.0
	Rec3	14.3	13.7	12.8	12.8	14.0	14.3	14.2	13.0	13.2	14.3	13.6
3P95S5B3	Rec1	6.8	8.1	7.2	9.5	9.2	8.0	9.1	6.4	8.7	10.2	8.3
	Rec2	7.2	6.4	6.1	7.4	7.0	5.6	6.0	5.8	6.7	6.6	6.5
	Rec3	5.8	6.3	5.0	6.0	5.8	6.4	6.2	6.3	6.0	5.3	5.9
4P130S5B3	Rec1	4.8	5.2	11.6	13.7	19.0	2.4	1.7	10.9	5.0	8.8	8.3
	Rec2	5.9	4.5	10.8	9.9	9.9	1.2	0.5	3.6	4.2	4.2	5.5
	Rec3	7.1	5.1	9.9	9.5	9.1	0.2	0.4	2.2	3.5	2.5	4.9

Tabela 7.10: Percentual de melhora da solução inicial - Variação Rec

Instância	Var	Lim	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	Med
1P130S5B3	Rec1	Upper2	16.3	15.2	16.9	16.7	17.8	15.2	17.0	16.5	18.4	16.5	16.6
		Upper3	12.5	11.3	13.1	12.8	14.0	11.3	13.2	12.7	14.7	12.7	12.8
	Rec2	Upper2	14.9	13.7	14.0	14.2	13.8	14.2	13.8	14.7	13.8	14.3	14.1
		Upper3	11.0	9.7	10.0	10.3	9.8	10.2	9.8	10.8	9.8	10.4	10.2
	Rec3	Upper2	13.8	13.4	14.6	13.9	14.0	14.2	14.1	14.4	14.1	14.2	14.1
		Upper3	9.8	9.4	10.7	10.0	10.1	10.3	10.2	10.4	10.2	10.3	10.1
2P112S4B3	Rec1	Upper2	20.8	21.2	21.3	20.1	19.0	19.7	20.8	19.0	19.9	24.5	20.6
		Upper3	11.6	12.0	12.1	10.8	9.5	10.4	11.5	9.6	10.6	15.8	11.4
	Rec2	Upper2	19.0	18.8	19.3	18.7	18.8	18.4	19.3	19.0	19.5	18.7	19.0
		Upper3	9.6	9.4	9.9	9.3	9.4	8.9	9.9	9.6	10.2	9.2	9.5
	Rec3	Upper2	18.3	18.6	19.3	19.4	18.5	18.3	18.4	19.0	19.0	18.6	18.7
		Upper3	8.8	9.2	9.9	10.0	9.0	8.8	8.9	9.6	9.6	9.1	9.3
3P95S5B3	Rec1	Upper2	15.2	14.6	15.1	16.3	16.4	17.5	16.6	16.9	15.7	16.1	16.1
		Upper3	9.4	8.7	9.3	10.6	10.6	11.8	10.9	11.2	9.9	10.4	10.3
	Rec2	Upper2	13.1	13.8	13.6	13.0	13.3	14.3	14.0	14.2	13.4	13.6	13.6
		Upper3	7.2	7.9	7.7	7.1	7.3	8.4	8.0	8.3	7.4	7.7	7.7
	Rec3	Upper2	13.7	13.6	14.6	13.5	13.7	13.3	13.4	13.3	13.7	14.1	13.7
		Upper3	7.8	7.7	8.7	7.6	7.7	7.4	7.5	7.4	7.7	8.2	7.8
4P130S5B3	Rec1	Upper2	21.2	21.4	20.8	19.9	22.9	21.2	21.1	24.4	20.7	21.9	21.6
		Upper3	13.0	13.2	12.5	11.5	14.8	13.0	12.9	16.5	12.5	13.8	13.4
	Rec2	Upper2	20.7	18.8	20.5	19.5	19.1	19.5	19.4	19.1	19.8	18.8	19.5
		Upper3	12.4	10.3	12.2	11.1	10.7	11.1	11.0	10.7	11.5	10.4	11.1
	Rec3	Upper2	20.1	18.6	20.7	18.8	19.0	19.3	20.0	19.2	19.9	20.0	19.6
		Upper3	11.8	10.2	12.4	10.4	10.6	10.9	11.6	10.8	11.5	11.7	11.2

Tabela 7.11: Percentual que a melhor solução está abaixo dos limitantes duais - Variação Rec

Capítulo 8

Conclusão e Trabalhos Futuros

Este trabalho propôs um método híbrido que combina técnicas de Programação por Restrições com metaheurísticas de Busca Tabu para tratar o problema de escalonamento de atividades na produção de um campo petrolífero. Além disso, comparou o desempenho da técnica híbrida com o da Busca Tabu quando esta última é aplicada isoladamente. Também analisou o tempo necessário para melhorar as soluções iniciais, a qualidade da solução final obtida com cada técnica, e a robustez de cada uma delas. Para isso, foram testadas diversas instâncias com diferentes técnicas de obtenção de soluções iniciais aplicadas a cada uma delas. Também foi realizada uma análise de sensibilidade sobre cada uma das instâncias aqui consideradas. Como as técnicas empregadas não dão garantias sobre a qualidade das soluções obtidas, e como não havia resultados anteriores para as instâncias consideradas, foram calculados limitantes duais para o problema utilizando-se modelos de programação matemática desenvolvidos nesta dissertação.

Várias conclusões podem ser inferidas dos resultados obtidos neste trabalho. Como esperado, o método híbrido mostrou-se o mais eficiente, o mais robusto e o mais ágil para melhorar soluções iniciais de qualidade inferior. Isso pode ser constatado pelo fato de que a abordagem híbrida *IA* foi a que produziu os melhores resultados para a grande maioria de instâncias e variações consideradas neste trabalho. Além disso, a taxa de convergência dos métodos híbridos mostrou ser bem superior àquela dos métodos puros *TabuPF* e *TabuPR*. Deve-se destacar a eficiência da abordagem híbrida *WE* para melhorar soluções iniciais de qualidade inferior e também a eficiência da abordagem híbrida *IA*, que sempre atingiu o patamar de melhor produção já no início da sua execução. Também deve ser destacado que, embora a abordagem pura *TabuPR* tenha obtido a melhor produção para duas instâncias, seu desempenho foi fraco quando soluções iniciais de qualidade inferior foram consideradas e, especialmente, quando as variações *Atv90* e *Rec1* foram tratadas.

Por outro lado, os resultados também mostraram que a escolha de uma boa estrutura de vizinhança é fundamental para um desempenho adequado da técnica híbrida. Embora

a abordagem híbrida *WE* tenha mostrado um bom desempenho no início da sua execução, quando soluções iniciais de qualidade inferior eram consideradas, esta não foi capaz de escapar de patamares inferiores de produção. Já abordagem híbrida *WI*, que também apresentou uma rápida taxa de convergência no início da sua execução, especialmente para as variações *Atv90* e *Rec1*, produzindo melhores resultados que a abordagem pura *TabuPR* em algumas instâncias, também não foi capaz de escapar de patamares inferiores de produção na maioria das instâncias testadas. Por outro lado, a abordagem híbrida *IA* produziu bons resultados e teve rápida taxa de convergência para todas as instâncias testadas.

Um resultado teórico importante foi a demonstração que a determinação do melhor escalonamento em cada vizinho é um problema NP-difícil.

O esforço para a obtenção de bons limitantes duais foi crítico para a comprovação da qualidade das soluções obtidas neste trabalho pois, utilizando-se os resultados obtidos pelos modelos de programação linear inteira para versões relaxadas do problema original, pôde-se inferir que as melhores soluções obtidas para cada abordagem são de boa qualidade, já que a diferença entre estas últimas e os melhores limitantes duais ficou em torno de 10%. Além disso, o resultado do teorema 5.1 mostrou-se fundamental para a obtenção do modelo utilizado pelo método *Upper2* e, conseqüentemente, para a obtenção de bons limitantes para a variação *Atv90*.

Em relação à análise de sensibilidade, as variações *Atv30*, *Atv60*, *Rec2* e *Rec3* não impactaram de forma significativa o comportamento das abordagens testadas. Por outro lado, o comportamento das abordagens *TabuPR*, *WE* e *WI* foi diferente, quando as variações *Atv90* e *Rec1* foram consideradas. A abordagem pura *TabuPR* não apresentou um bom de desempenho, enquanto o desempenho de *WE* e *WI* foi melhorado. A variação *Atv90* também causou impacto no cálculo dos limitantes, já que somente para esta variação o limitante dado por *Upper2* foi melhor do que o dado por *Upper3*. Estes resultados mostram que as abordagens híbridas são bem robustas quanto aos dados de entrada. Também, por utilizarem Programação por Restrições para explorar suas vizinhanças, são bem robustas quanto ao conjunto de restrições que está sendo considerado, já que restrições podem ser inseridas ou retiradas facilmente. O mesmo não é verdade para as abordagens puras, que mostraram-se bastante dependentes dos dados de entrada e, além disso, inserir alterações no conjunto de restrições dessa abordagens é bastante dispendioso.

De uma maneira geral, os objetivos dessa dissertação foram alcançados satisfatoriamente. Foi resolvida com muito boa aproximação uma instância real do problema. Também foram resolvidas de forma adequada instâncias obtidas perturbando-se alguns dados de entrada mais sensíveis, o que veio a comprovar a robustez das soluções propostas neste trabalho. Deste modo, o trabalho contribuiu positivamente para o sucesso de futuros

empreendimentos nas áreas estudadas.

O trabalho desenvolvido nesta dissertação pode ser estendido de diversas maneiras:

- As abordagens híbridas *WI* e *WE* mostraram que podem ter seu desempenho melhorado. Para isso, duas características ortogonais destas abordagens devem ser consideradas. Primeiramente, a exploração das respectivas vizinhanças pode ser ajustada, eliminando-se simetrias intrínsecas do problema. Desta forma, mais vizinhos promissores poderiam ser encontrados. Outra característica que carece de mais estudo é a lista tabu adotada nestas abordagens. Esta deve ser melhor ajustada para evitar que estas abordagens fiquem *estacionadas* em um patamar inferior de produção.

- Neste trabalho não foi considerado o problema do roteamento de recursos. Foi assumido que os recursos podiam locomover-se de um poço ao outro sem dispêndio de tempo. Para tornar o trabalho ainda mais realista, uma expansão interessante seria considerar a distância entre os poços e a velocidade de locomoção de cada recurso quando estes fossem alocados às atividades.

- Além disso, também não foi considerado que a vazão de cada poço poderia variar ao longo do tempo. Como este tipo de variação ocorre freqüentemente em ambientes reais, seria interessante acrescentar esta característica ao escopo do problema tratado.

- Outra extensão interessante seria considerar mais restrições na modelagem do problema, como as restrições de marco, de data, de seqüência de poços, de superfície e de eficiência, todas descritas no capítulo 2.

- No que diz respeito à Programação por Restrições, existem diversos estudos sobre outras formas de *backtracking* e sobre o uso de informações obtidas durante a procura de soluções de modo a permitir uma redução mais eficiente do espaço de busca [17, 22]. É possível que a implementação dessas idéias traga melhorias na exploração das vizinhanças das abordagens híbridas já que estas se utilizam de Programação por Restrições.

- Por fim, os algoritmos aqui apresentados não dispõem de interfaces de operação adequadas para um usuário final não especializado. O desenvolvimento de interfaces gráficas mais amigáveis certamente aproximaria este trabalho de sua utilização efetiva num ambiente industrial.

Referências Bibliográficas

[1] R. K. Ahuja, O. Ergun, J. B. Orlin, e A. P. Punnen. A survey of very large-scale neighborhood search techniques. Obtido em: `web.mit.edu/jorlin/www`, diretório `working papers`, Julho 1999.

[2] R. K. Ahuja, J. Orlin, e D. Sharma. Multi-exchange neiborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91:71–97, 2001.

[3] F. Azevedo e P. Barahona. Interaction of constraint programming and local search for optimization problems. Em *7th Conference on Principles and Practice of Constraint Programming*, 2001.

[4] B. Backer, P. Kilby, P. Prosser, e P. Shaw. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, páginas 1–16, 1997.

[5] J.R. Baker e G.B. McMahon. Scheduling the general job-shop. *Management Science*, 5(31):594–598, Maio 1985.

[6] K. R. Baker. *Introduction to sequencing and scheduling.* John Wiley and Sons, 1974.

[7] E. Balas. Machine scheduling via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, 17:941–957, 1969.

[8] B. L. Beckner e X. Song. Field development planning using simulated annealing - Optimal economic well scheduling and placement. Em *SPE Annual Technical Conference & Exhibition*, Dallas, 1995.

[9] D. Bertsimas e J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[10] A. C. Bittencourt e R. N. Horne. Resevoir development and design optimization. Em *SPE Annual Technical Conference & Exhibition*, San Antonio, Texas, Outubro 1997.

[11] Y. Caseau, F. Laburthe, C. L. Pape, e B. Rottmbourg. Combining local and global search in a constraint programming environment. *The Knowledge Engineering Review*, 16(1):41–68, 2001.

[12] Y. Caseau e L. Laburthe. Disjunctive scheduling with task intervals. Relatório Técnico 95-25, Laboratoire d'informatique de l'École Normale Supérieure, 1995.

[13] A. Cetilmen, T. Ertekin, e A. S. Grader. Applications of neural networks in multiwell field development. Em *SPE Annual Technical Conference & Exhibition*, Houston, Texas, Outubro 1999.

[14] N. Christodoulou, E. Stefanitis, E. Kaltsas, e V. Assimakopoulos. A constraint logic programming approach to the vehicle-fleet scheduling problem. Em *Proceedings of Practical Applications of Prolog*, páginas 137–149, 1994.

[15] T. H. Cormen, C. E. Leiserson, e R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[16] G. B. Dantzig. Maximization of linear function of variables subject to linear inequalities. Em *Activity Analysis of Production and Allocation*, páginas 359–373, 1951.

[17] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

[18] V. G. Deineko e G. J. Woeginger. A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming*, páginas 255–279, Fevereiro 2000.

[19] V. J. Fortuna, C. C. de Souza, e A. V. Moura. Relatório FAPESP. Relatório 2, Unicamp, 2002.

[20] H. Fugii. Multivariate production systems optimization in pipeline networks. Tese de Mestrado, Stanford University, 1993.

[21] M. R. Garey e D. S. Johnson. *Computers and intractability: A guide to the theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, California, 1979.

[22] M. L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[23] F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[24] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[25] F. Glover. Tabu search: a tutorial. *Interfaces*, 20(4):74– 94, Julho-Agosto 1990.

[26] F. Glover, J. P. Kelly, e M. Laguna. Genetic algorithms and tabu search: hybrids for optimization. *Computers and Operations Research*, 22(1):111–134, 1995.

[27] F. Glover e M. Laguna. Tabu search. Obtido em: `www-bus.colorado.edu`, diretório `faculty/laguna/Papers/ts.pdf`, Agosto 2000.

[28] C. P. Gomes. On the intersection of AI and OR. *The Knowledge Engineering Review*, 16(1):1–4, 2001.

[29] R.L. Graham, E.L. Lawer, J.K. Lenstra, e A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and squeduling: A survey. Em *Annals on Discrete Mathematics*, volume 5, páginas 287–326, 1979.

[30] B. Güyagüler e R. N. Horne. Uncertainty assessment of well placement optimization. Em *SPE Annual Technical Conference & Exhibition*, New Orleans, Louisiana, Setembro 2001.

[31] B. Güyagüler, R. N. Horne, L. Rogers, e J. J. Rosenzweig. Optimization of well placement in a Gulf of Mexico waterflooding project. Em *SPE Annual Technical Conference & Exhibition*, Dallas, Texas, Outubro 2000.

[32] G. Hasle, R.C. Haut, B.S. Johansen, e T.S. Ølberg. Well activity scheduling – an application of constraint reasoning. Em *Proceedings of PACT'97 (The 1997 International Conference on Parallel Architectures and Compilation Techniques)*, San Francisco, California, Novembro 1997.

[33] C. A. Holloway. *Decision Making Under Uncertainty, Models and Choices.* Prentice-Hall, New Jersey, 1979.

[34] J. Carroll III e R. N. Horne. Multivariate production system optimization. *J. Petroleum Tech.*, páginas 782–789, Julho 1992.

[35] B. Jurish. *Scheduling Jobs in Shops with Multi-purpose Machines.* Tese de Doutorado, Fachbereich Mathematik/Informatik, Universitat Osnabruck, 1992.

[36] B. Korte e J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer-Verlag, Segunda Edição, 2001.

[37] B.J. Lageweg, K. Lenstra, e A.H.G. Rinnoy Kan. Job-shop scheduling by implicit enumeration. *Management Science*, 4(24):441–450, 1977.

[38] C. Le Pape. Implementation of resource constraints in ILOG SCHEDULE: a library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, 3(2), 1994.

[39] J. Lever, M. Wallace, e B. Richards. Constraint logic programming for scheduling and planning. *BT Technology Journal*, 13(1), 1995.

[40] K. Marriott e P. J. Stuckey. *Programming with Constraints: An Introduction.* MIT Press, Cambridge, Massachusetts, 1998.

[41] P. Martin e D. B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. Em *Proceedings of the 5th International IPCO Conference*, páginas 389–403, 1996.

[42] M. Mastrolilli e L.M. Gambardella. Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, 3:3–20, 2000.

[43] T. Mautor e P. Michelon. MIMAUSA: A new hybrid method combining exact solution and local search. Em *2nd International Conference on Meta-Heuristics*, 1997.

[44] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6:1–12, 1959.

[45] Z. Michalewicz e D. B. Fogel. *How to Solve It: Modern Heuristics.* Springer-Verlag, 1999.

[46] J. M. Nascimento, A. V. Moura, e C. C. de Souza. Relatório FAPESP. Relatório 1, Unicamp, 2002.

[47] J. A. Nelder e R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[48] M. Palke e R. N. Horne. Nonlinear optimization of well production considering gas lift and phase behavior. Em *SPE Production Operations Symposium*, Oklahoma City, Oklahoma, Março 1997.

[49] Y. Pan e N. Horne. Improved methods for multivariate optimization of field development scheduling and well placement design. Em *SPE Annual Technical Conference & Exhibition*, New Orleans, Louisiana, Setembro 1998.

[50] C. H. Papadimitriou e K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, 1982.

[51] G. Pesant e M. Gendreau. A view of local search in constraint programming. Em *2nd Conference on Principles and Practice of Constraint Programming*, 1996.

[52] G. Pesant e M. Gendreau. A constraint programming framework for local search methods. *Journal of Heuristics*, 5:255–279, 1999.

[53] M. Pinedo. *Scheduling: Theory, Algorithms and Systems.* Prentice Hall International Series in Industrial and Systems Engineering. Prentice Hall, 1995.

[54] J.-F. Puget. Object-oriented constraint programming for transportation problems. Em *Proceedings of Advanced Software Technology in Air Transportation (ASTAIR)*, 1992.

[55] N. Ravindran. Multivariate optimization of production systems - the time dimension. Tese de Mestrado, Stanford University, 1992.

[56] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.* McGraw-Hill, 1995.

[57] L. M. Rousseau, M. Gendreau, e G. Pesant. Using constraint-based operators with variable neighborhood search to solve the vehicle routing problem with time windows. Em *CP-AR-OR'99*, 1999.

[58] B. Roy e B. Sussmann. Les problèmes d'ordonnancement avec contraints disjonctives. Note DS 9 bis, SEMA, Paris, 1964.

[59] V.I. Sarvanov e N.N. Doroshko. The approximate solution of the travelling salesman problem by a local algorithm that searches neighborhoods of exponencial cardinality in quadratic time. Mathematical Institute of the Belorussian Academy of Sciences, Julho 1981.

[60] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. Em *4th Conference on Principles and Practice of Constraint Programming*, 1998.

[61] R. F. Stoisits, K. D. Crawford, D. J. MacAllister, M. D. McCormack, A. S. Lawal, e D. O. Ogbe. Production optimization at the Kuparuk River Field utilizing neural networks and genetic algorithms. Em *Mid-Continent Operations Symposium*, Oklahoma City, Oklahoma, Março 1999.

[62] M. Vázquez, A. Suárez, H. Aponte, L. Ocanto, e J. Fernandes. Global optimization of oil production systems, A unified operational view. Em *SPE Annual Technical Conference & Exhibition*, New Orleans, Louisiana, Setembro 2001.

[63] M. Wallace. Practical applications of constraint programming. *Constraints*, 1:139–168, 1996.

[64] W. W. Weiss, R. S. Balch, e B. A. Stubbs. How artificial intelligence methods can forecast oil production. Em *SPE/DOE Improved Oil Recovery Symposium*, Tulsa, Oklahoma, Abril 2002.

[65] T. H. Yunes, A. V. Moura, e C. C. de Souza. A hybrid approach for solving large scale crew scheduling problems. Em *Lecture Notes in Computer Science*, vol. 1753, páginas 293–307, Boston, MA, EUA, Janeiro 2000. Anais do *Second International Workshop on Practical Aspects of Declarative Languages (PADL'00)*.

[66] T. H. Yunes, A. V. Moura, e C. C. de Souza. Solving very large crew scheduling problems to optimality. Em *14th ACM Symposium on Applied Computing (SAC'00)*, páginas 446–451, Como, Itália, Março 2000.